THE DEVELOPMENT OF A MULTI-FUNCTION COMPUTER-ASSISTED INSTRUCTION SYSTEM USING A HIERARCHICAL DATABASE STRUCTURE AS THE LESSON COMPENDIUM.

by: MICHAEL JAMES LITTLE
Compower Limited

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

June 1982.

CONTENTS

FIGURES

FIGURES

FIGURES

PHOTOGRAPHS

ACKNOWLEDGEMENTS

I would like to proffer my very sincere thanks to the following people:

Mr. J. M. Doubleday (University of Aston Computer Centre)

- for his invaluable help, guidance, supervision and gentle pressure;

Mr. E. Wille (formerly Head of Training, Compower Limited)

- for lighting the fuse in the first place;

Compower Ltd. (computer subidiary of the National Coal Board)

- for their sponsorship and the use of their incomparable computing facilities;

Mrs. M. Little (my wife)

- for making me work on this project when I didn't really want to.

Inevitably, in the cause of brevity, I am forced to omit a vast number of people who have assisted me in a multitude of ways. To all of them - thank you.

Title: The development of a multi-function Computer-Assisted Instruction system using a hierarchical database structure as the lesson compendium.

By: Michael James Little

Submitted for the Degree of Doctor of Philosophy of the University of Aston in Birmingham. 1982.

SUMMARY

Computer-Based Learning systems of one sort or another have been in existence for almost 20 years, but they have yet to achieve real credibility within Commerce, Industry or Education. A variety of reasons could be postulated for this, typically:

- cost
- complexity
- inefficiency
- inflexibility
- tedium

Obviously different systems deserve different levels and types of criticism, but it still remains true that Computer-Based Learning (CBL) is falling significantly short of its potential.

Experience of a small, but highly successful CBL system within a large, geographically distributed industry (the National Coal Board) prompted an investigation into currently available packages, the original intention being to purchase the most suitable software and run it on existing computer hardware, alongside existing software systems. It became apparent that none of the available CBL packages were suitable, and a decision was taken to develop an in-house Computer-Assisted Instruction system according to the following criteria:

- cheap to run;
- easy to author course material;
- easy to use;
- requires no computing knowledge to use (as either an author or student);
- efficient in the use of computer resources;
- has a comprehensive range of facilities at all levels.

This thesis describes the initial investigation, resultant observations and the design, development and implementation of the SCHOOL system. One of the principal characteristics of SCHOOL is that it uses a hierarchical database structure for the storage of course material - thereby providing inherently a great deal of the power, flexibility and efficiency originally required.

Trials using the SCHOOL system on IBM 303X series equipment are also detailed, along with proposed and current development work on what is essentially an operational CBL system within a large-scale Industrial environment.

Keywords: Computer-Assisted Instruction; National Coal Board; Computer-Based Learning; Database.

chapter one

- PROJECT OVERVIEW

1.1 INTRODUCTION

It is important to define at the outset the environment within which this project developed simply because these 'environmental considerations' have had a fundamental effect on the design philosophy and direction of the system. Typical of these constraints were:

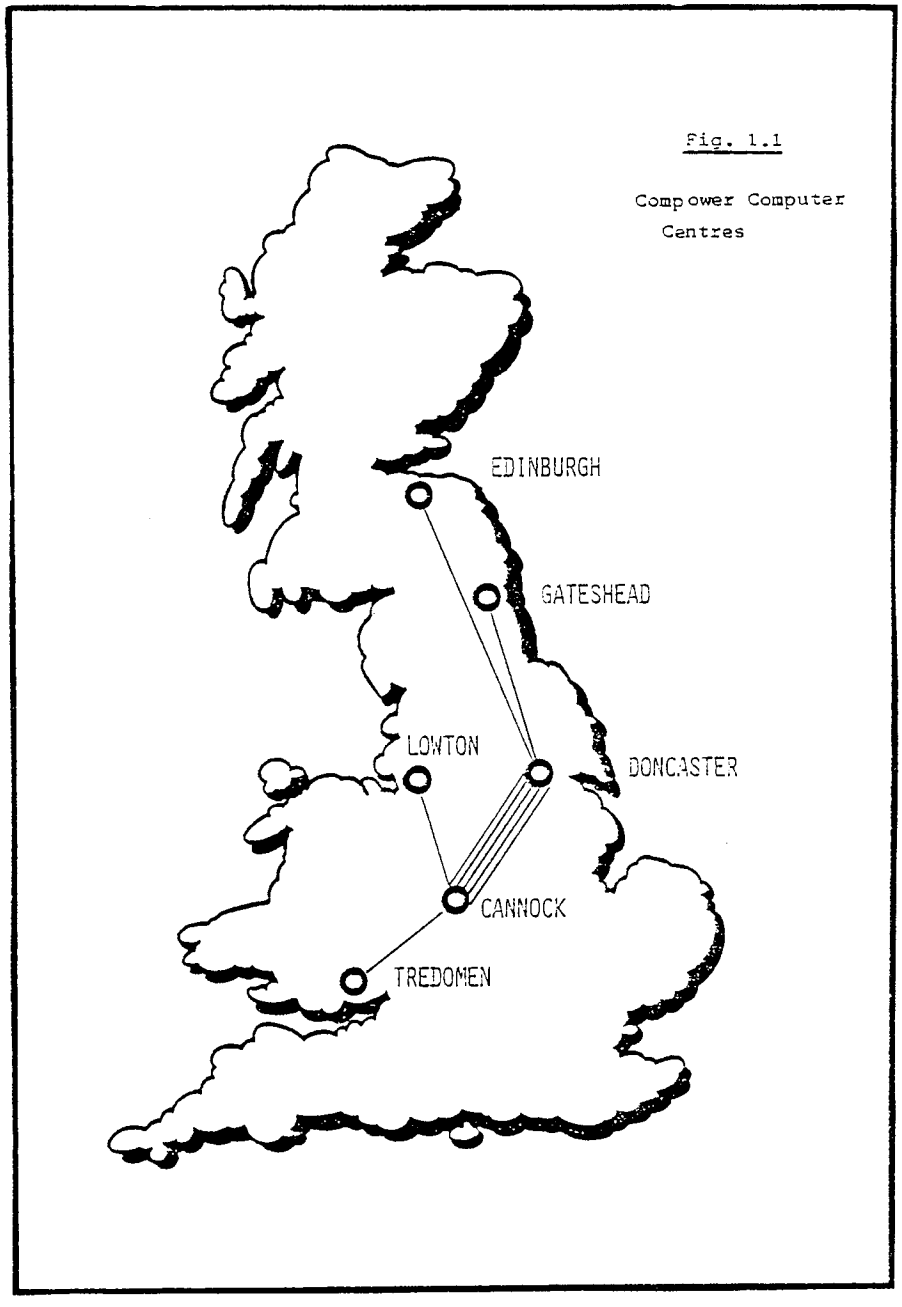
- the system should be entirely commercially viable;
- it should be capable of being used by both authors and students from a wide range of backgrounds, without the requirement of DP experience;
- it should be capable of using existing hardware, and being extended in line with likely/expected hardware developments;
- it will not have dedicated mainframe resources, and consequently should have minimum impact on its cohabitant subsystems;

These factors are in some respects unusual, in that many Computer Based Learning (CBL) systems have concentrated solely on sophisticated teaching mechanisms, often at the expense of viability.

1.2 COMPOWER

This project was developed at, and sponsored by Compower Limited, Cannock, Staffordshire. Compower is the wholly-owned computer subsidiary of the National Coal Board, and is one of the largest computer users in the U.K.

Processing is carried out on a range of equipment primarily from IBM, ICL and DEC, installed across 6 Regional Computer Centres - as Fig 1.1 below illustrates.



The main processing centres are Doncaster and Cannock, the latter being the company Headquarters. Whilst all centres process batch-orientated work, only Cannock and Doncaster have significant on-line systems, based primarily on IBM equipment as the following table indicates:

CENTRE	MACHINE	ON-LINE SYSTEM(S)
CANNOCK	IBM 3081	IMS (Information Management System) IBM (1) - database enquiry/update STAIRS (Storage and information Retrieval) IBM (2) - text enquiry VSPC (Virtual Storage Personal Computing) IBM (3) - program development - conversational programming
	IBM 3033	CMS (Conversational Monitoring System) IBM (4) - program development - conversational programming
DONCASTER	IBM 370/158 (two)	IMS - database enquiry/update

Table 1.1

(Number in brackets refers to Reference section entry number)

Around 70% of all processing work is for the National Coal Board, the remainder being the 'External Market'.

1.3 TELEPROCESSING NETWORK AND TERMINALS

The systems listed in Table 1.1 represent an installed base in excess of 400 terminals, with a considerable increase planned over the next few years. With the advent of IBM's Systems Network Architecture IBM (5), (6), (7)) and compatible releases of systems software, it will become possible to access any on-line system from any terminal on the network.

The effect of this technology can be seen by reference to Fig 1.2:

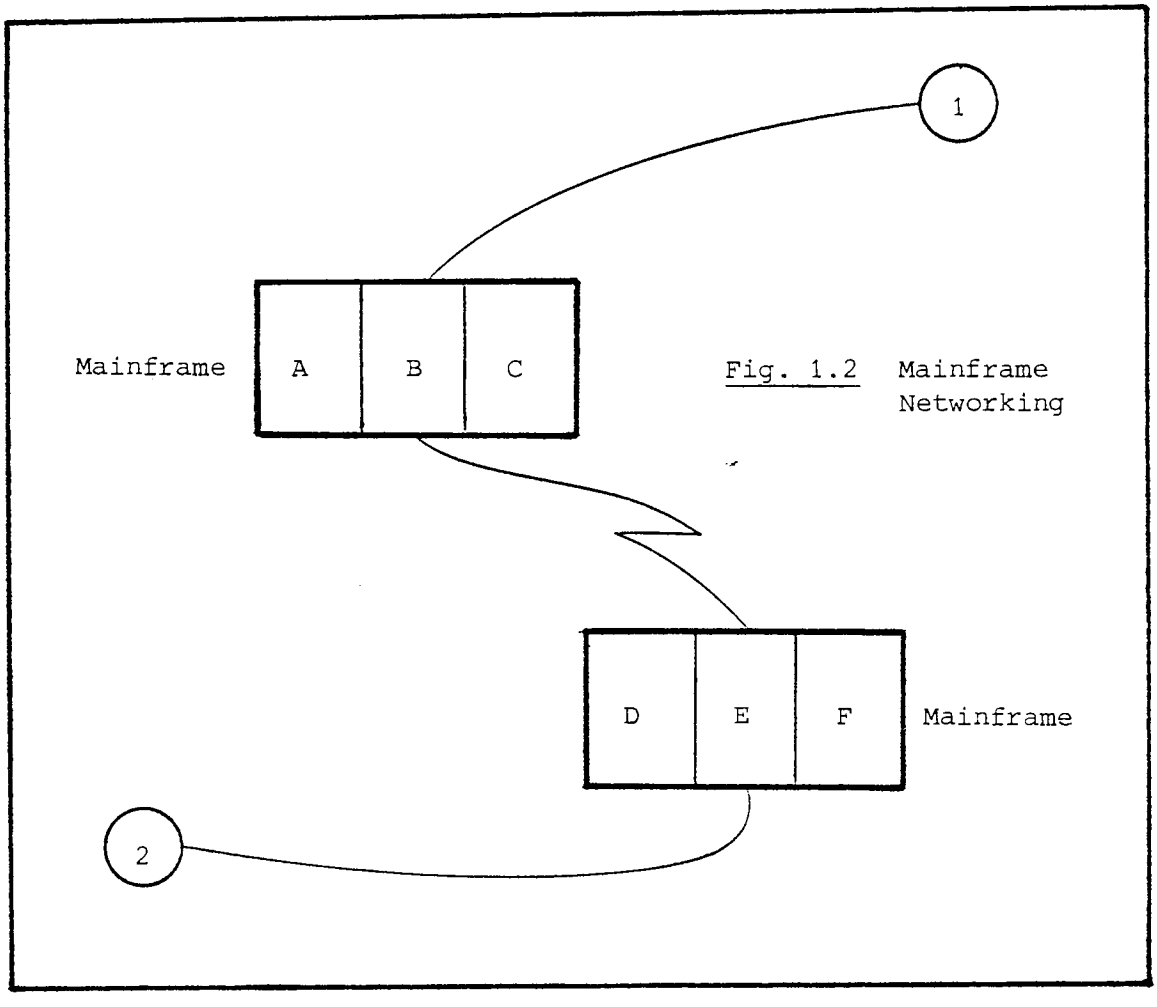


Fig. 1.2 Mainframe Networking

Both terminals 1 and 2 when connected to an SNA Network can access all of the teleprocessing subsystems A, B, C, D, E or F, any of which could represent a Computer-Based Learning system. This has enormous implications for the design of any such system:

(i) the potential CBL audience is massive;

(ii) the training requirements are bound to be diverse;

(iii) implementation can be done at marginal cost (assuming terminals installed for other higher priority purposes);

(iv) processing resources are being shared, therefore software and storage utilisation must be efficient;

Point (iii) above is of paramount importance - the major barrier to the expansion of CBL, whether in industry or education, is cost (Fielden, 9). Usually the most significant cost factor (for the user) is the purchase/rental of the necessary terminal and telecommunications equipment - this has been evaluated as being capable of approaching 56% (Bitzer & Skaperdas, 10) of the cost of implementing a large CBL system. Whilst recent findings show a shift in emphasis towards a predominance of people-cost, the existence of a terminal which is:

(a) already installed

(b) less than 100% employed

(c) capable of accessing a parallel CBL system

represents an enormous re-adjustment to the CBL cost equation, and it is only the advent of truly multi-functional terminals (e.g. via SNA) that have made this possible.

1.4 HISTORY OF COMPUTER BASED LEARNING AT COMPOWER

During the latter half of the 1960's, Compower (at that time National Coal Board Computer Services) created at its Cannock Headquarters a purpose-built Training School with permanent lecturing and tutorial facilities. The function of this new venture was to provide computer-related training for:

(a) NCB Computer Services DP staff;

(b) NCB personnel as appropriate;

(c) the external (i.e. non-NCB) market,

Initially, this training was biased almost totally towards the 'batch' type of Data Processing including systems analysis, programming fundamentals, languages etc. During 1969, the first significant evolution took place, with the installation of IBM's Conversational Programming System (CPS) on the Cannock mainframe, designed to provide relatively simple on-line programming for remote users - the majority non-computing departments within the NCB, with little DP experience.

Within a year, the installed terminal base exceeded 50 (each used by a number of people/departments) and with it came a major training requirement - only partially solved via regular one-week CPS Courses. It became obvious that this could not keep pace with demand, and alternative solutions were considered (film, PI text, audio cassette, terminal-based training).

Terminal-based training was eventually adopted in Summer 1971 following a trip to USA by the then Head of Training, Mr. Edgar Wille. One of the establishments visited was Dartmouth College, New Hampshire, where Dr. T. E. Kurtz, Head of the Computer Facility, demonstrated several operational terminal-based teaching systems, including BASIC programming, Climatology and

State geography. Whilst crude by current standards, these systems were the first indication of Computer-Based Training's potential, and a major project was commissioned immediately afterwards.

This task was given to the author and the design remit was fairly wide, in that any suitable CBL technique could be employed to replace the 1-week CPS Course. The first version of this (system CPS/CAI 1.1) became operational in March 1972, 8 on-line lessons supplemented by a Programmed Instruction Text. Among the various characteristics of CPS/CAI 1.1 were:

- automatic student registration;
- full control of student progress;
- a range of performance analysis facilities;
- various tutorial techniques;
- random message generation (i.e. selecting from a list of similar messages) to avoid student tedium;

Further alterations were made to CPS/CAI as follows:-

- September 1972 - version 1.1 (revisions to lesson control, extra lesson added);
  December 1973 - version 2.1 (more elegant control software, revised lesson material, better answer checking).

This system has been fully operational since this date, and almost 400 people have used it.

1.5 THE CONVERSATIONAL MONITORING SYSTEM

In June 1974, after intensive investigation, Compower obtained from IBM a new teleprocessing system - CMS (Conversational Monitoring System). With this came a new host operating system - VM/370 (Virtual Machine facility/System 370) . This system was installed on a dedicated IBM 370/145 mainframe (since upgraded to an IBM 3031) and was intended to provide a powerful program development facility for Compower staff, as well as a replacement for CPS.

The author was involved in the investigation of CMS, and its potential as a vehicle for Computer-Based Learning soon became apparent. Towards the end of 1974 therefore, a preliminary paper was produced outlining the desired characteristics of a Compower CMS/CBL system, heavily influenced by experience of CPS/CAI, and investigations into other contemporary CBL packages.

1.6 PLANNED CBL SYSTEM - PROPOSED CHARACTERISTICS

When the preliminary definition of the planned CMS/CBL system was produced, the following characteristics were listed (even though the exact software mechanisms required were unknown):

(a) language-free Course Authoring:

- i.e. no need for Authors to learn any form of computer language

(b) subject structuring:

- i.e. the ability to subdivide a course into several (possibly independent) component levels;

(c) no restriction on subject matter:

- i.e. courses of any subject type (not necessarily related to Data Processing).

(d) comprehensive student facilities:

- i.e. students should have a wide range of available facilities including:-

  (i) hint/answer request

  (ii) suspend/resume session

  (iii) request information (e.g. current score, lesson details, range of available subjects).

  (iv) message/comment entry

  (vi) glossary word/phrase explanation

(e) minimal system management:

- i.e. as far as possible, the system should be self-supporting (automatic student sign-on, space management, performance monitoring etc).

(f) full performance monitoring:

- i.e. student and system:

  (i) Student - name, dates, marks obtained, durations, etc.

  (ii) Lessons - no. times used, completion types, individual frames details (times used, number of attempts, times incorrect/hint requested etc).

  (iii) Space - Database space utilisation/ fragmentation.

(g) host system independence:

- i.e. as far as possible, the CBL system software should be written in such a way as to have no independence on its host Operating System;

(h) efficiency:

- i.e. impact on cohabitant systems should be minimal:

  (i) Processing - software is written to be very efficient;

  (ii) Storage - all course text should be compressed to use minimum on-line storage;

(i) device independence/optimisation:

- i.e. the CBL system will optimise output format and control to use each type of user terminal to best effect. This should be transparent to all users.

(j) expansion potential:

- i.e. implementation limits (number of subjects, lessons, etc) should not be permanent, and continuing expansion should be possible;

These preliminary characteristics may be compared to those of other CBL projects, see Chapter 2.

chapter two

- COMPUTERS IN EDUCATION

2.1 COMPUTERS IN EDUCATION

Computers have been used in teaching for over 20 years - essentially for as long as they have represented viable information processing devices. Diversification has been considerable, and virtually all aspects of education, training, instruction, course administration and curriculum management have been tackled. Various attempts have been made to summarise these (Silvern & Silvern (14), Lippert (11), Hooper (12), Beech (13)) but no standard range of definitions currently exists.

For the purposes of this thesis, an attempt has been made by the author to rationalise and standardise the terminology applied to the use of Computers in Education.

2.1.1 COMPUTERS IN EDUCATION - TERMINOLOGY

(i) CBL/CBE - COMPUTER BASED LEARNING/EDUCATION:
- teaching with a computer;
the machine is used as an educating mechanism, may (once set-up) be entirely self-sufficient, and will in all instances be the major component of the education;

(ii) CBT - COMPUTER BASED TRAINING:
- as for CBL & CBE but generally more applicable to industry (i.e. training);

(iii) CAL - COMPUTER ASSISTED LEARNING:
- a main component of CBL (the other being CML);

- the computer is used to teach in either tutorial sense (information, question/answer etc) or laboratory sense (simulation, problem-solving, experimentation);
- the computer will possibly be augmented by other media, and some external form of control will be required (e.g. course administrator, CML system);

(iv) CAT
- COMPUTER ASSISTED TRAINING:
- as for CAL but generally more applicable to industrial applications;
- a main component of CBT;

(v) CAI
- COMPUTER ASSISTED INSTRUCTION:
- essentially the tutorial component of CAL or CAT, i.e. a software structure designed to inform, guide and possibly test students to a prescribed level of competence;

(vi) CML
- COMPUTER-MANAGED LEARNING:
- the administration and management of a teaching function, (typically test marking, progress analysis/control, recording keeping, scheduling,etc);

(vii) CMT
- COMPUTER-MANAGED TRAINING:
- as for CML, but related to the management and administration of teaching within Industry;

A hierarchy becomes apparent, see Fig. 2.1:

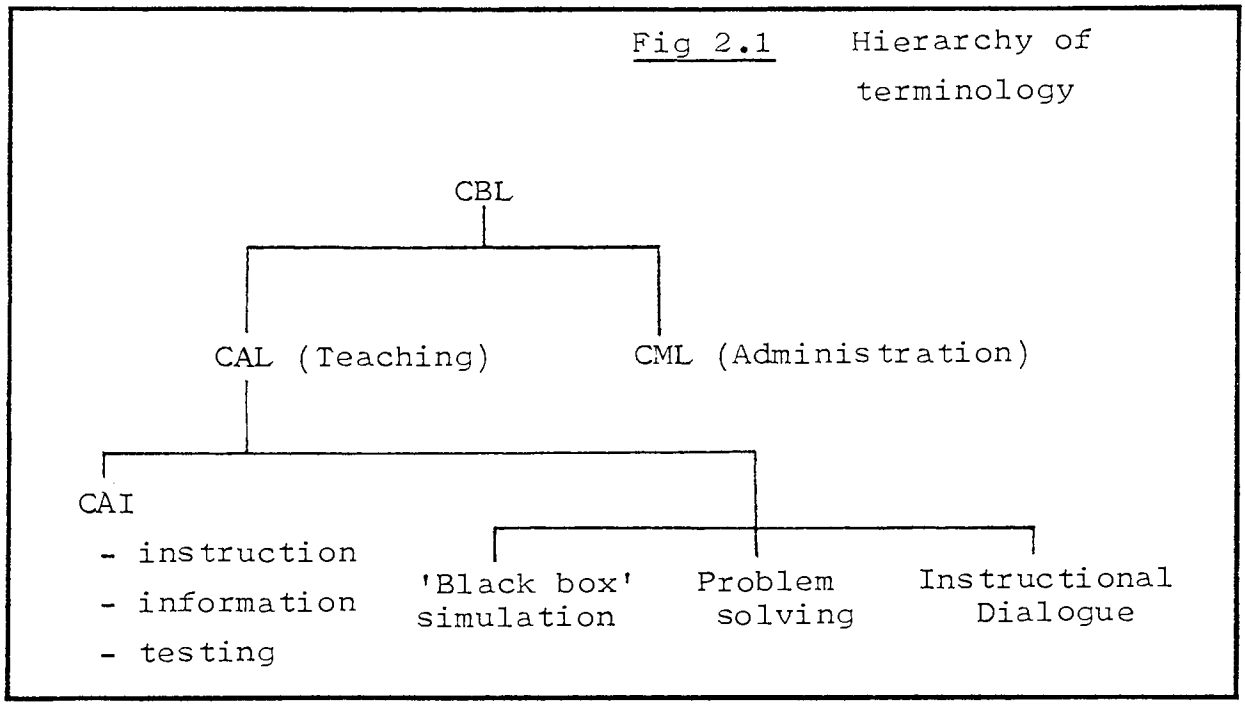


Fig 2.1 Hierarchy of terminology

The official title for this thesis includes the term 'Computer-Assisted Instruction'. As the project has evolved and in particular, since the above hierarchy was produced, it now appears to the author to be more correct to consider the title as being:

'The development of a multi-function Computer-based Learning system using a hierarchial database structure as the teaching and administrative control mechanism'.

2.1.2 HISTORY OF USING COMPUTERS IN EDUCATION

(a) Prior to 1965

Computer-Based Learning began as a spin-off from the development of remote computer access in the late 1950's. These first tentative steps towards Interactive Processing utilised converted electric typewriters and teletypes linked to systems which provided and controlled user programming.

The educational potential of these early conversational systems was soon recognised, and simple tutorials were implemented. These were effected using existing programming languages, but by 1960 sufficient interest had accumulated to produce the first purpose-built CBL Author Language - IBM Coursewriter (Suppes & Macken, 15), representing the first serious attempt to make the generation of CBL tutorials less dependent on computer professional expertise.

Stanford University, California in 1963 began the development of the first viable school CAI system and by 1965 teletype terminals were installed on school premises giving daily elementary arithmetic and reading lessons.

Also at this time development of system PLATØ (Programmed Logic for Automatic Teaching Operations) was started at the University of Illinois (Hooper, 16), and this has subsequently evolved into the world's largest, most comprehensive and commercially successful CBL system.

(b) 1965 to 1970

This was a period of contradictions - first a widespread expansion of CBL to many new fields and then a contraction of development as implementation costs started to become prohibitive.

The Stanford University system continued to grow: by 1966 three schools had installed terminal equipment, and by 1967 visual display devices and multi-media University level teaching (e.g. Russian, Mathematics) had been implemented - in the instance of Russian, special Cyrillic keyboards were being used and all normal classroom activity had been replaced.

PLATØ continued to evolve and computer-controlled graphics, animation etc. were incorporated to excellent effect.

Throughout the computing fraternity, CBL blossomed and tutorials over a wide spectrum of topics were produced. It was however a lengthy and complex task to generate this software, and with the exception of IBM Coursewriter and PLATØ, specific systems were written for specific topics - and this was expensive.

Consequently the envisaged low CBL session costs were not realised and this, plus lack of authoring flexibility slowed investment, research and development to walking pace.

(c) 1970 to 1975

The Seventies began, from a CBL standpoint, with very little work being done, and for very sound reasons:

- course and software development costs were prohibitive;
- processing and storage requirements were too large for most installations;
- teaching techniques available via CBL were very limited, especially using standard terminals;
- companies could fill their computers quite comfortably with more pressing (and profitable) tasks;
- computers were as yet not academically respectable;

Realism began to creep into what development was being done, best illustrated by the MITRE Corporation/Brigham Young University TICCIT (Time shared Interactive Computer-Controlled Information Television) system. Begun in 1972, TICCIT was the first CAI system to use the then recently developed mini-computer (Suppes & Macken (15), Flockhart (17), Alderman et al (18)). Standard colour TV receivers were linked to twin Nova 800 mini-computers to provide a small, local CAI capability - the major aim being to maintain low cost. Each TICCIT system was self-contained and capable of supporting up to 128 TV terminals, and several colleges still use TICCIT 'courseware' as an integral part of their curricula.

As hardware became cheaper and more powerful, computers became more widespread. Computer Science began to be taught academically, and the machines

were used in many other subject areas.

This new found respectability, and spare machine capacity brought about a gradual resurgence of interest in CBL, spreading to industry, where the first viable on-line systems were in use (typified by Compower's CPS/CAI System).

(d) 1975 to the Present

The key to CBL development over the last few years has been machine power - i.e. the specific abilities to support large numbers of students/ authors simultaneously (possibly alongside other systems), provide wide ranging facilities and support more than 1 concurrent subject. CBL software has consequently tended to become more generalised and several computer corporations have taken to it commercially during this period, as the following list (Baker, 19) shows:

CDC	PLATØ
Sperry Univac	ASET
Hewlett-Packard	IDF
IBM	IIS and ITS
DEC	DECAL
ICL	LØRA

Some of these appear to be enhanced BASIC language programming, but others are significant enough to warrant separate case studies - see section 2.4.

2.2 CAL - PRINCIPLES AND STRUCTURES

Any CAL system consists of two fundamental components:

(i) CONTENT - the substance of the material to be presented;

(ii) PROCEDURES - the vehicle which controls and delivers the Content;

Traditionally, this dichotomy has resulted in two separate skills requirements (Zinn, 20, 21). Firstly organising and constructing the CONTENT; obviously this requires subject matter expertise and an understanding of the teaching commitment (i.e. objectives and means of achieving these objectives), but will also involve knowledge of the specific characteristics of a host computer system.

Secondly, transposing the content into an appropriate form for computer hardware - i.e. PROCEDURE preparation. The skill requirement here is some form of programming. Fig. 2.2 gives an overview of these components and skill requirements.

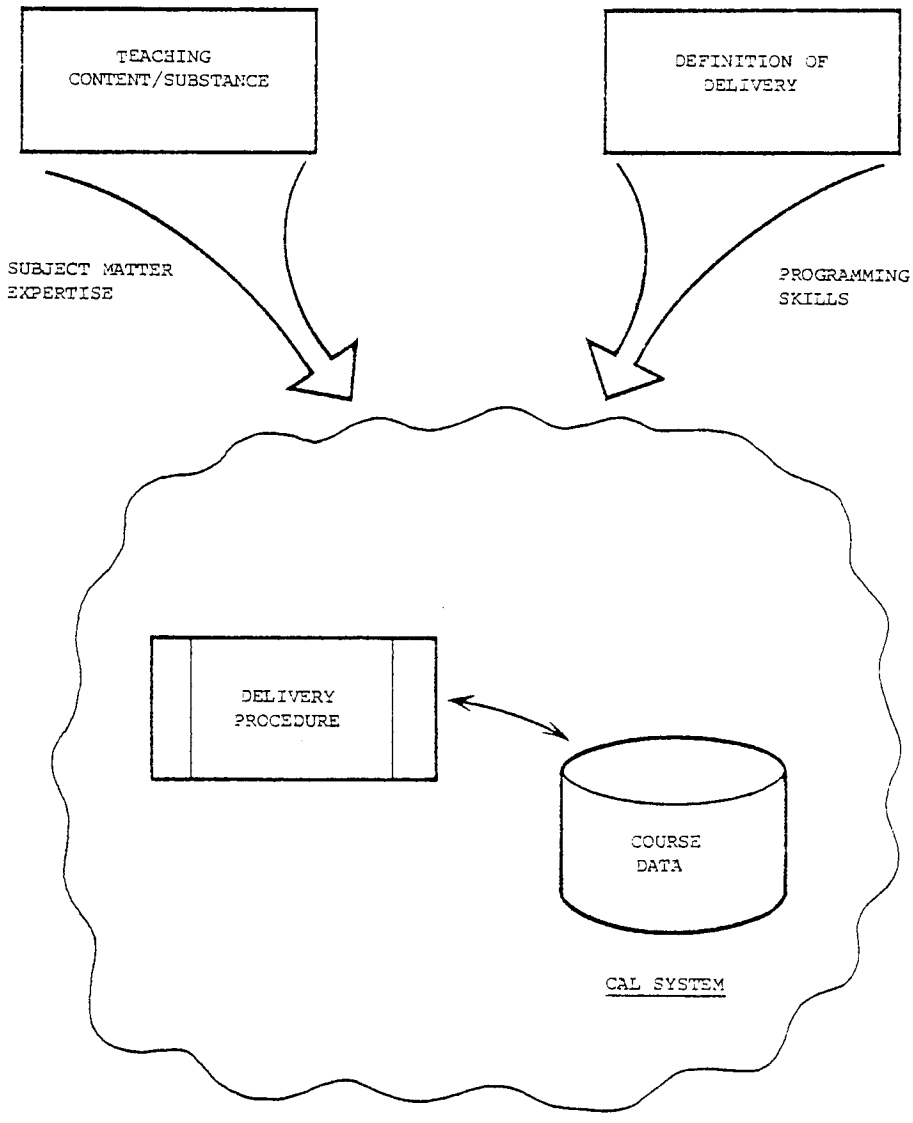


Fig. 2.2 PRINCIPLES AND STRUCTURE OF A TRADITIONAL CAL SYSTEM.

Early CAL systems depended therefore on at least two types of skills availability:

- the discipline or subject matter expert
- the procedure definition expert, or programmer

and normally this implied at least two individuals.

To simplify this procedure, three approaches became apparent:

(i) the subject matter expert becomes proficient in computer programming;

(ii) the programming expert becomes conversant with the subject matter;

(iii) the procedure definition process is simplified to allow CAL construction without recourse to conventional programming skills;

Of these 3, the last one is obviously the most preferable, and several attempts have been made to achieve this end - typically via the production of CAL Author Languages.

2.2.1 CAL - Author Languages

A large proportion of instructional programming is done using conventional languages - typically, high-level examples such as ALGØL, APL, BASIC, FORTRAN, PL/1.

Whilst these are not necessarily inadequate (APL and PL/1 for instance being extremely powerful), the production of a non-trivial CAL system requires considerable programming expertise and other limitations may be encountered (e.g. text handling, upper and lower case characters, answer definition/recognition, etc). On the other hand, standard data processing languages

usually have some favourable characteristics which will continue to make them attractive:

- good documentation;
- well tried compiler/interpreter;
- good debugging facilities;
- portability;

Nonetheless, these are generalised languages designed and produced for other purposes and consequently a number of custom-built CAL Author Languages have evolved over the last 20 years.

Zinn (21), categories these in 3 groups:

(a) successive frame languages:

- in essence computerised programmed instruction;
- languages of this type generally have good text display facilities, accept/ analyse simple student responses, record performance and provide explicit or implicit branching capabilities;
- best known languages within this category are COURSEWRITER (IBM) and TUTOR (Computer-based Education Research Lab, University of Illinois);

(b) languages which provide conversation (albeit strictly limited):

- typically this means extensions to category (a) to facilitate condition testing, text processing, storing replies, etc.;

- languages within this category normally have the ability to record a sequence of replies (or 'conversation') and to make decisions based on both the most immediate reply and the history of the conversation so far;
- MENTØR (Department of Educational Technology, Cambridge, Mass) is an example of such a language;

(c) framework or strategy languages:

- effectively a standard delivery procedure is defined without embedded course material;
- once a strategy (or 'framework') has been defined, the author identifies the information to be used by it and the appropriate teaching sequence can take place;
- the most notable author language in this category is CATØ (Compiler for Automatic Teaching Operation - part of the PLATØ system). Other examples are CØMBAT (Mills & Allen, 22), ITS (Interactive Training System - IBM), ASET (Sperry Univac);

All of the above categories represent languages, and whilst they are significantly more tailored than the general purpose languages, they still incorporate the usual language-related characteristics:

- learning a syntax;
- coding;
- compilation/interpretation;
- testing and debugging;
- alterations/editing;

From the above it is apparent that CAL Author Languages still require significant skill levels - in fact the more complex examples approach normal high-level languages.

Recent work has evolved the generative CAL Author System - an attempt to address the programming skill problem.

2.2.2 CAL - Author Systems

The basic principle of an Author System is that no language is required at all. The Author still has control over the procedure by which his instructional material is delivered, but this is defined via simple parameters rather than language statements.

Different mechanisms exist for parameter definition, including:

- prompt and response;
- macro statements, e.g. ITS (IBM, 23);
- author documents (i.e. specially designed data entry forms defining both structure parameters and text); ACATS (Flockhart, 17) uses this approach, as does the Extended Course Structuring Facility of ITS;

A computer-Assisted Learning System employing this approach can now be represented as 3 related logical subsystems, as Fig. 2.3 illustrates:

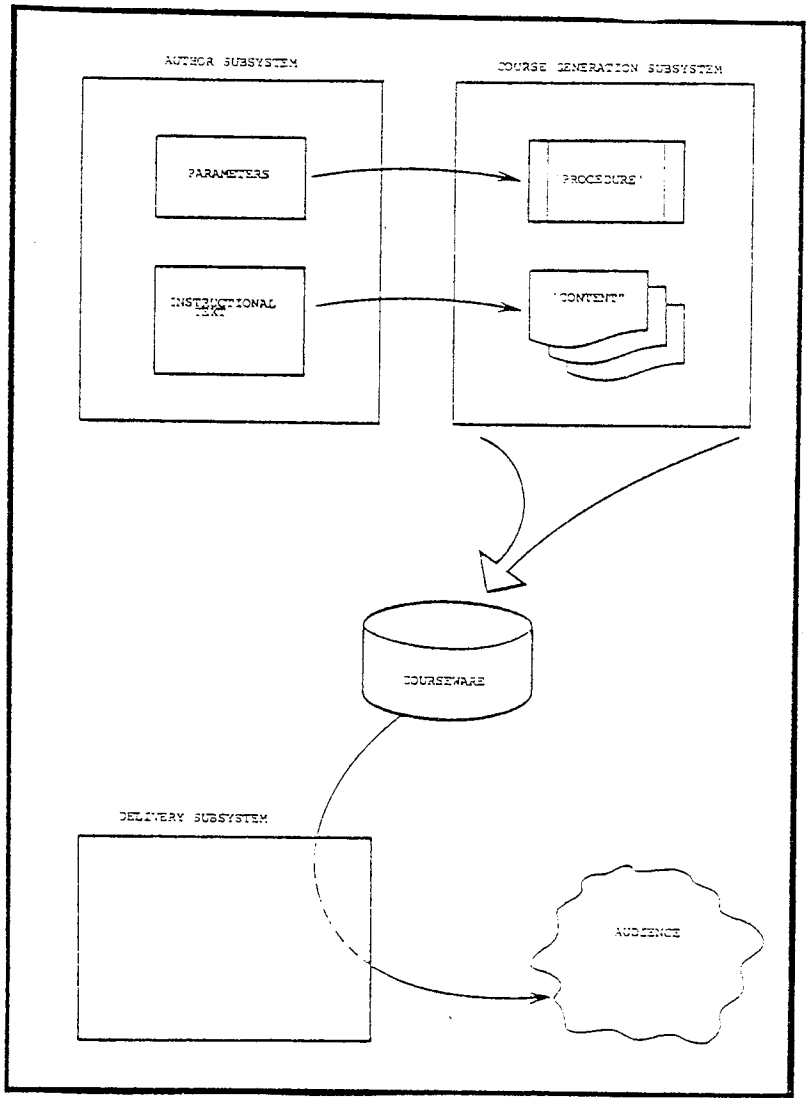


Fig. 2.3

CAL Logical Subsystems

A spin-off advantage of this structure is that the various components need not necessarily be cohabitant - COMBAT (Mills & Allen, 22) for instance, is designed to run with Authoring and Generation subsystems on one machine, and the Courseware Delivery subsystem on several others.

2.3 CML - PRINCIPLES AND STRUCTURE

Computer Managed Learning (CML) is sub-divided into major 3 components:

(i) TESTING

(ii) ROUTING

(iii) RECORD KEEPING

These can be applied in various forms to both conventional and CAL processes. Few successful examples of the former exist, although the Hertfordshire Computer Managed Mathematics Project (Tagg, 24) and CML systems implemented within military training (Miles, 25), are worthy of note.

Application of CML to computer-assisted learning systems can be done in a variety of ways, typically via interface datasets, or calls between parallel CAL and CML software. Very complex systems may in fact be subdivided into functional sub-components, one controlling the learning processes, the other the management functions, as Fig. 2.4 illustrates:

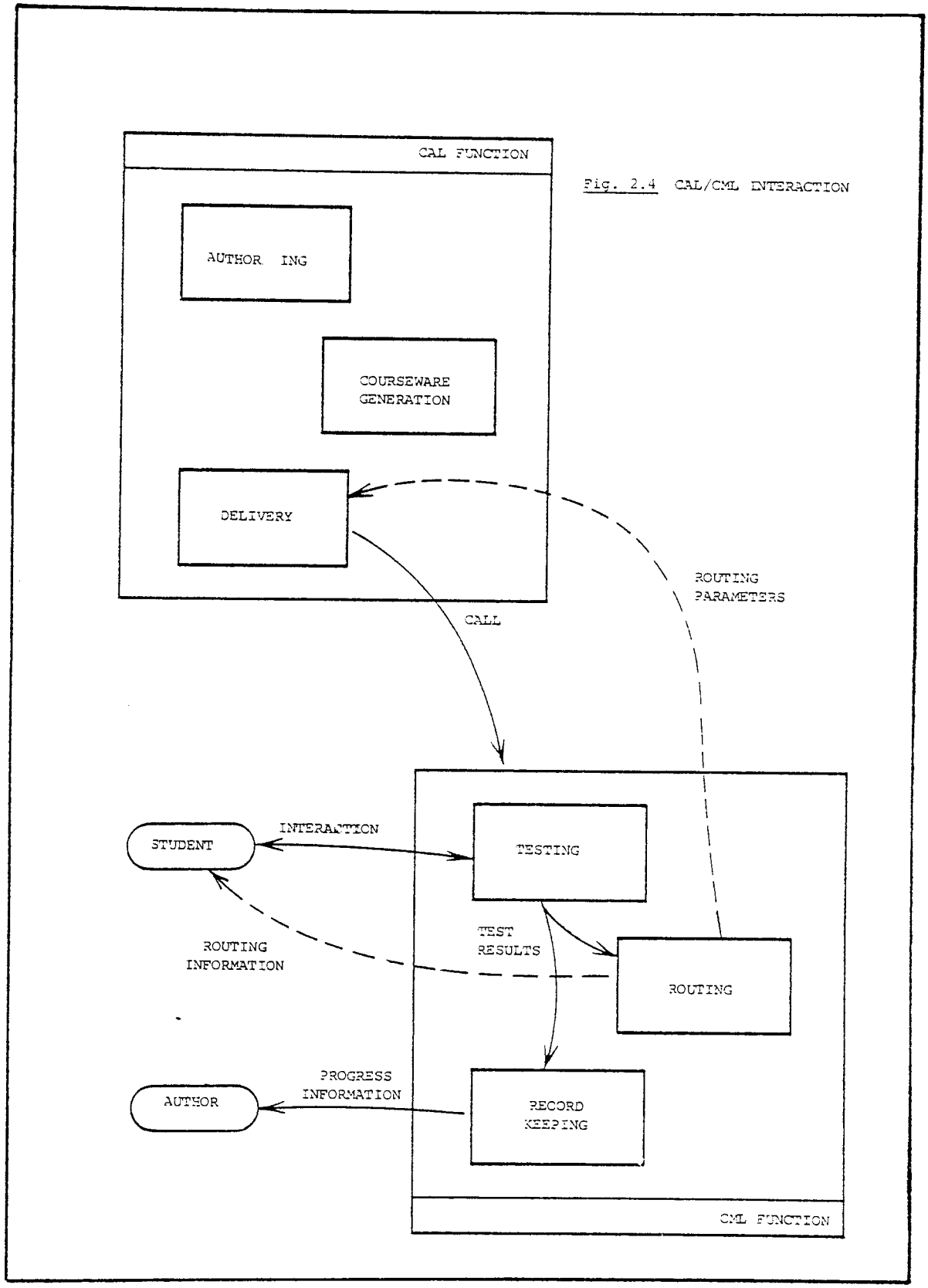


Fig. 2.4 CAL/CML INTERACTION

N.B. (Relative to Fig. 2.4)

(i) the CML Testing component may (in part) be resident in the Delivery component of the CAL function;

(ii) the algorithms used to control student Routing may get fairly complex, and may incorporate some degree of manual control (student, or teacher/author) and/or historical information;

2.3.1 CML - the Testing Function

An enormous range of techniques have been applied to proficiency assessment.

Examples are:

(a) simple question/answer:

- producing a score which permits further progress or otherwise;

(b) diagnostic assessment:

- the student is tested and a 'profile' of his performance is built up; this is then used to select subsequent learning materials; examples may be found in Compower's CPS/CAI System and IBM's ITS (IBM, 23);

(c) mastery learning:

- often used in operational simulation (e.g. process simulators, technical fault finding, etc).

Some attempts have also been made to produce CML systems which generate test material automatically, either selecting from a (lengthy) list of predefined questions, or building material up according to various student criteria. Information on this approach is thin on the ground.

2.3.2 CML - the Routing Function

Routing refers to the direction and control of students through a particular course of study. At its simplest, this will be more or less a straightforward monitored sequence, but at its most complex, routing can involve intricate branching algorithms that attempt to emulate the intuition and adaptability of a human teacher.

Almost invariably, CML routing and CML testing are related in some way, with decisions being made at a number of stages - ranging from immediate to deferred (i.e. performance within one part of the subject influences another part).

2.3.3 CML - the Record Keeping Function

Performance recording can range from a simple sequence of marks obtained on examinations/tests to sophisticated monitoring procedures which will capture many details of student performance at the individual questions level, including the number of times assistance has been requested, the number of incorrect attempts and even details of how long the student took to answer.

The advantages of using a computer system for record keeping become more attractive as student numbers increase and as the data required becomes more difficult to obtain. It is also possible to append statistical analysis, result sorting and reporting procedures quite easily.

2.4 CBL SYSTEMS - CASE STUDIES

The generally disappointing impact of Computer-Based Learning has already been discussed, but it is nonetheless important to acknowledge that some systems have survived long enough, and been used widely enough, to warrant specific examination. Two systems have been chosen:

(i) PLATO (University of Illinois/Control Data Corporation)

(ii) IIS (IBM)

Both have marked similarities (in existence for many years; large user base; have evolved through many versions) and marked differences (dedicated v. shared mainframes; academic v. industrial background; specialised v. general purpose hardware) and a great deal of valuable information is to be gleaned from studying them in detail.

2.4.1 PLATO - An Overview

First started at the University of Illinois around 1960, PLATO (Programmed Logic for Automatic Teaching Operations) has evolved through many versions thanks mainly to considerable funding from the US Government and Control Data Corporation. Versions of PLATO have normally been significantly different from one another, (Bitzer & Skaperdas,

10), and two representative versions will be described:

(a) PLATO III:

- the most significant characteristic of this version (and subsequent University of Illinois releases) was that it used special display stations, comprising TV monitor (used to display a conglomerate of slides, diagrams and computer-generated alphanumeric text, all under PLATO control) and a keyboard used for student replies. Fig. 2.5 illustrates the PLATO III hardware organisation:

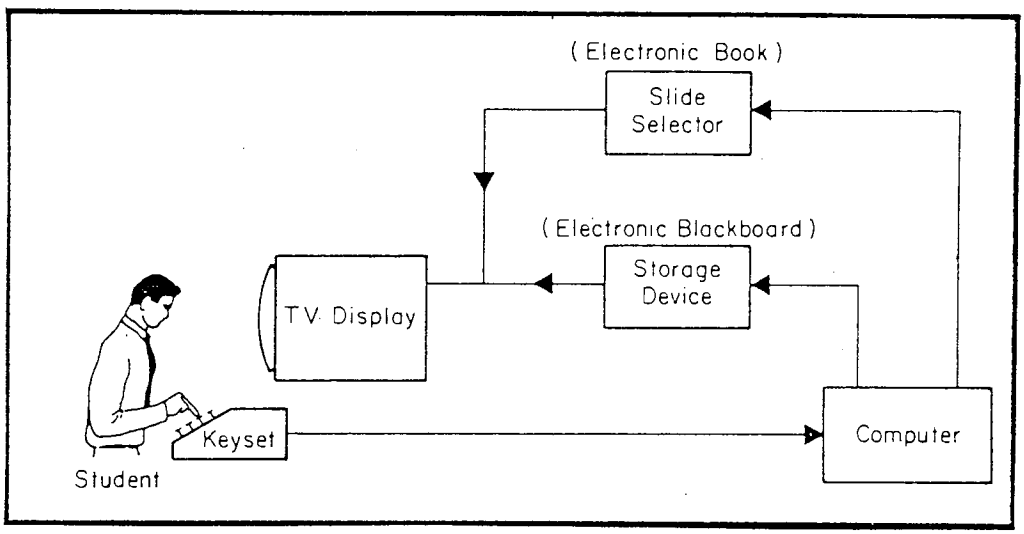


Fig. 2.5 PLATO III Hardware

Course authoring was via the CATO author language, instructions being incorporated into each 'program' to select and display elements from a large sheet of transparencies held in the Slide Selector. These had to be photographically produced and pre-loaded into the device - the result being high quality displays (better than any contemporary computer produced equivalent) but expensive course preparation, expensive student equipment and limited author flexibility.

(b) PLATO IV:

- similar in concept to version III, special student station hardware was again used - in this instance plasma-display screens with the Slide Selector mechanism now using microfiche and built into the terminal itself (see Fig. 2.6):

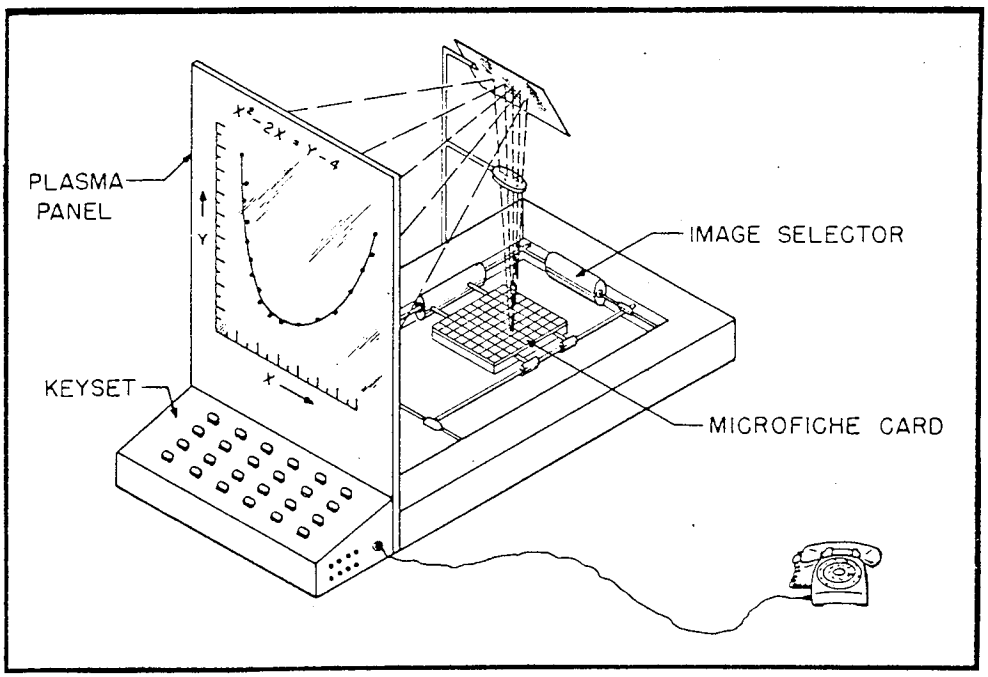


Fig. 2.6 PLATO IV Terminal

The major advantages were cost and a much greater level of software control over the display layout. The plasma display technique has other advantages - it does not require regeneration, is bright and has very good definition.

- A higher level author language (TUTOR) was also introduced, and a range of other facilities such as comment files, inter-student communication and games were incorporated (Denenberg, 26). As with all PLATO systems, versions IV was run on a very large, dedicated CDC mainframe computer, and towards the late 70's, several such Computer Centres had been set up in the U.S.A.

During 1977, an important split in the development of PLATO took place, producing:

(i) Academic PLATO:

- University of Illinois;
- specific emphasis on research, CAI hardware and providing a service for educational establishments;

(ii) Industrial PLATO:

- Control Data Corporation;
- specific emphasis on using and developing those aspects of the original system which were commercially viable;

The latter version is now probably the best known CBL system in the world. The commercial philosophy of CDC has been to set up Learning Centres, where clients can go to study, as well as leasing terminal equipment. Over sixty such centres have been set up throughout North America and U.K. with up to 12,000 hours of study available (Box, 27). Briefly, the current CDC PLATO system has the following characteristics:

- simplified VDU hardware (i.e. no slide or microfiche capability) although full monochrome graphics (both static and dynamic) can be used;

- terminals dedicated to the PLATO system and on-line to a very large, dedicated CDC Cyber 7600 mainframe. One particular attractive aspect of the PLATO screens (especially to children) is that they have touch response - i.e. areas of the screen can be activated so that if touched by the student an appropriate response is generated;

- text generation and display can be done in a variety of character fonts and be paced (i.e. rather than instantaneous display, the student can watch the text being written);

- authoring is via the TUTOR Author Language and a comprehensive, but not necessarily easy to use range of facilities is provided.

2.4.2 PLATO - An Appraisal

There is no doubting the power and impact of a professionally produced PLATO course. They have, however, been produced by a team structure, comprising both subject matter and PLATO programming experts - the courses are simply too complex and the TUTOR language too difficult for it to be otherwise. Denenburg (26) confirms this, and emphasises that whilst simple CAI programs can be produced quite easily, '.... the more useful lessons require a degree of expertise not commonly found even in many professional programmers'.

The use of dedicated hardware - both terminal and mainframe - is also a major drawback, resulting in the PLATO system being inordinately expensive if using equipment installed on user premises. This is only justifiable given a very high level of CBL usage. It is felt therefore that the main success area for CDC PLATO will be the Learning Centre concept (i.e. off-the-shelf non-specific courses, professionally authored and marketed to industry) rather than locally installed, user-driven Computer-Based Learning.

2.4.3 IIS - An Overview

The IBM Corporation software product IIS (Interactive Instructional System) represents 20 years of CBL system development, during which time three distinct phases have evolved:-

(i) 1960 - present COURSEWRITER author language

(ii) 1971 - 1977 ITS (Interactive Training System)

(iii) 1977 - present IIS.

The largest user of IIS (or its predecessors) is the IBM Corporation itself - it is claimed for instance that the world's largest CBT system is IBM's internal Field Instructional System (FIS). This is based on a combination of the above products and is used to train technical, engineering, marketing and management staff via several thousand terminals throughout the Western hemisphere (Baker, 19).

The COURSEWRITER language, developed at IBM software laboratories in New York during 1959-60 represented the first purpose-built CBL Author Language. Considerably enhanced over the years, COURSEWRITER is still available to authors as part of IIS, and also forms the 'low level' base of the complete system (IBM, 23). Language statements are separated into three groups:-

(i) PROBLEM - deliver a question or frame to the student, await and read in the response;

(ii) MAJOR - classify the response (e.g. correct, anticipated but incorrect, unanticipated);

(iii) MINOR - processing to be carried out as a result of response classification;

Statements are coded sequentially and a CAI 'program' results, as the following example illustrates:

```
1   qu  When added together, what numbers = 4?
2   ca  2 and 2
3   ty  Very good.
4   cb  1 and 3
5   cb  3 and 1
6   ty  That is correct.
7   wa  0 and 4
8   wb  4 and 0
9   ty  Do not use the digit 0.  Try again.
10  wb  4 and 4
11  ty  This results in 8.  We want 4.
12  un  I do not understand.  Try a different answer.
13  un  I do not understand.  Try again.
14  qu  What is the result of 4 times 4?
```

The operation codes used in the above example only represent a small part of the available repertoire, see below:

Problem	Meaning
qu	Question
rd	Read
pr	Problem
st	Set format

Major	Meaning
ca	Correct answer
cb	Alternate correct answer
wa	Wrong answer
wb	Alternate wrong answer
un	Unanticipated answer
aa	Anticipated answer
ab	Alternate anticipated answer
nx	No match
ea	End of answer
gt	Get field

Minor	Meaning
ep	Enter and proceed
	Continue (2 spaces
ty	Type
ad	Add
sb	Subtract
mp	Multiply
dv	Divide
ld	Load
ed	Edit
br	Branch
fn	Function
pa	Pause
de	Display erase
pt	Put field
pc	Put continue
pe	Put end

The basic philosophy of COURSEWRITER revolves around implicit conditional branch operations (e.g. ca, wa, ab, etc.), making it easier for the author to define courseware logic, but it still retains the major disadvantages of all Author Languages - i.e. programming skill requirements, language syntax, debugging problems.

As an attempt to produce a 'higher-level' authoring facility, IBM evolved the ITS package.

ITS (Interactive Training System) was an attempt to package COURSEWRITER into a more complete CAI system. These enhancements produced the following range of facilities:

(i) the COURSEWRITER Language:

(ii) FUNCTIONS:

- the ability to extend basic ITS features with user-written Assembler coding;

(iii) VDU FIELD CONTROL:

- full control over the layout and characteristics of the Visual Display Unit Screen contents;

(iv) AUTHOR FACILITIES:

- a wide range of services for course creation and control, text editing and performance monitoring;

(v) EXTENDED COURSE STRUCTURING FEATURE (XCSF):

- a mechanism for defining the organisation, structure and contents of a course without using an Author Language.

XCSF is possibly the most unusual facet of ITS, whereby a course is subdivided into sessions, each of which may comprise:

(i) Pretest - to determine initial level of knowledge;

(ii) Tutorial - to present required material;

(iii) Posttest - to establish level of knowledge after instruction;

(iv) Review - further testing/instruction if required;

Short segments of text may also be delivered by way of session introduction/summary, and it is also possible to break-down a session level into topics, each with an individual pretest/tutorial/posttest/review structure.

The delivery of the various components is defined to the Expanded Course Structuring Feature via a series of author Macros (possibly coded on ITS Worksheets) each of which is interpreted into COURSEWRITER statements. One of the options available to the Author is whether tutorial presentation is to be delayed or immediate, i.e.

Delayed - all pretest question are presented in a sequence followed by tutorial components for all questions answered incorrectly.

Immediate - the related tutorial is presented immediately after an incorrect pretest question;

The final evolution from ITS into IIS (Interactive Instructional System) was, of minimal CBL significance - most of the alterations being to the software internals (e.g. access methods, compatibility with host systems etc).

2.4.4 IIS - An Appraisal

Whilst not in the same league as PLATO as regards presentation techniques, IIS has certain significant attractions - it can run alongside other applications in a mainframe, use conventional terminal equipment and is widely available from, and supported by, IBM.

It is however undeniably complex, and a 2½ day Course and 200-page manual are, in the Author's opinion, minimum requirements for use. Certainly the likelihood of a non-programmer getting to grips with ITS Course generation seems remote.

Obviously the range of Author facilities has been restricted to facilitate the use of conventional terminal equipment, but with the ability to use these devices and run on standard IBM computers, plus the product's low cost, the potential market-place for IIS is enormous.

The relatively low intrusion into this market-place is probably a testimony to its complexity.

2.5 CONCLUSIONS

Investigation into available Computer-Based Learning systems was triggered off within Compower by two related considerations:

(i) successful experience of using similar, albeit limited, software in-house;

(ii) a need to train a widely-dispersed terminal user population;

It was surprising to discover the lack of impact that CBL had achieved, particularly within commerce/industry, and a variety of reasons for this became apparent:

- Expense. The most common complaint and it appears to be largely vindicated. Certainly the favourable forecasts of Bitzer & Skapperdas (10) have not proved realistic;

- Special purpose hardware. This ties in with the previous category in that only massive and complex requirements can justify the kind of expenditure typified by dedicated systems such as PLATO. Most potential CBL users already have normal computer hardware that they wish to exploit fully;

- Inefficiency. Those systems which can cohabit with other applications (e.g. IIS) are generally relatively inefficient;

- Authoring complexity. Most author languages appear to have been produced by DP professionals rather than educationalists. As a consequence a considerable leaning towards programming becomes evident and attempts to alleviate this via authoring systems have also largely failed;

- Distrust. Effectively the combined product of the previous reasons, a significant level of distrust exists towards CBL software (particularly on centralised mainframe computers). Contributing towards this is the common trade-off between presentation and cost - i.e. those systems with the greatest presentation impact. (e.g. PLATO) tend to be expensive, and those systems which are cheap tend to be mundane.

Based on the above, it was concluded that no available CBL product suited Compower's overall requirements. It was decided therefore to design and develop an in-house system with the following basic philosophy:

(i) the package should be 'lightweight'. (i.e. efficient software, optimum utilisation of data storage etc);

(ii) existing terminal and telecommunications equipment should be used;

(iii) courseware authoring should be straight-forward, and no Author language should be involved;

(iv) the entire system should be easy, attractive and interesting to use;

The remainder of this thesis describes the evolution of the Compower system SCHOOL: (System Controlling Hierarchical Organisation of Online Lessons).

chapter three

- SCHOOL DATABASE STRUCTURE
  AND I/O HANDLING

3.1 INTRODUCTION

Conventional education syllabi follow very similar outlines - they have a start point, a connecting structure (topics, sub-topics and items of information put together in a progressive sequence) and a desired end point. Occasionally, other approaches occur - less formalised, Socratic' forms, whereby only the limits of the subject are defined, the means of achieving these limits being completely fluid.

Research CAI systems now exist which belong to either camp - those which provide structured education and those which demonstrate intelligence and a 'Socratic' capability. Of these two options, the very nature of the subjects that SCHOOL was designed for dictated that it should provide education of the structured and controlled form , and it became one of the most fundamental considerations in the development of SCHOOL to tailor Course data storage to assist towards this end.

3.2 CONCEPTS OF DATABASE STRUCTURE

As with all Data Base Management Systems (DBMS), the SCHOOL database appears to be different things from different viewpoints:

- to the user (either Author or student) it appears to be a hierarchy of subjects, lessons, frames, etc.
- to the database handling software it appears to be a fluid collection of data blocks linked together via complex system of pointers.

The terminology that has been adopted to describe these apparent structures is similar to that employed by IBM in describing Information Management System (IMS) databases, i.e.

LOGICAL Database - the data structure as it appears to the user or application

PHYSICAL Database - the internal data structure (blocks, pointers etc.) as it appears to the I/O software.

Fig. 3.1 illustrates this dual structure:

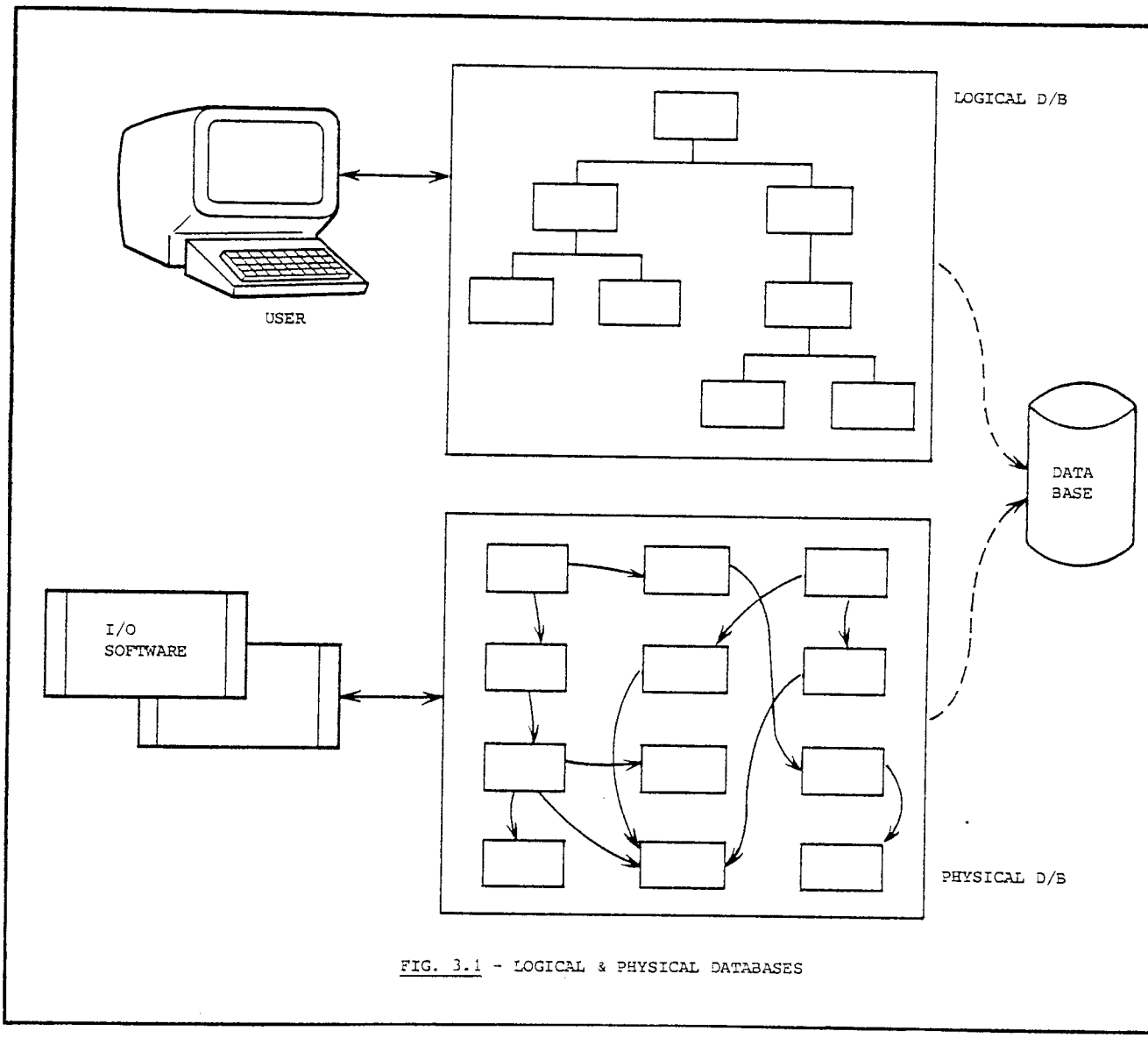
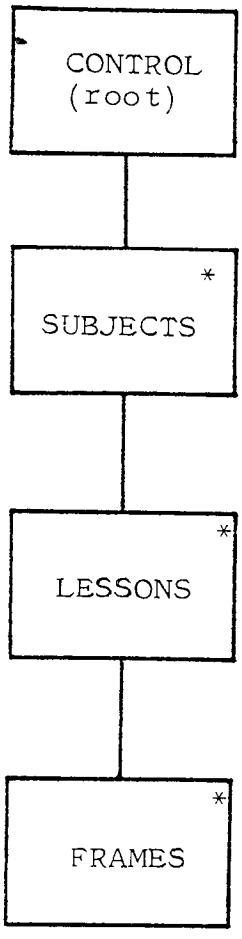


FIG. 3.1 - LOGICAL & PHYSICAL DATABASES

3.3 LOGICAL DATABASE STRUCTURE - General Notes

The SCHOOL Database appears to the user as a hierarchy of Subjects, Lessons, Frames, etc. There are 4 levels to this hierarchy, termed Logical Segments:

Fig. 3.2



N.B. '*' signifies 'one or more'

Each logical segment expands laterally into a series of Components, or Logical Records.

For instance, the Control Segment expands into:-

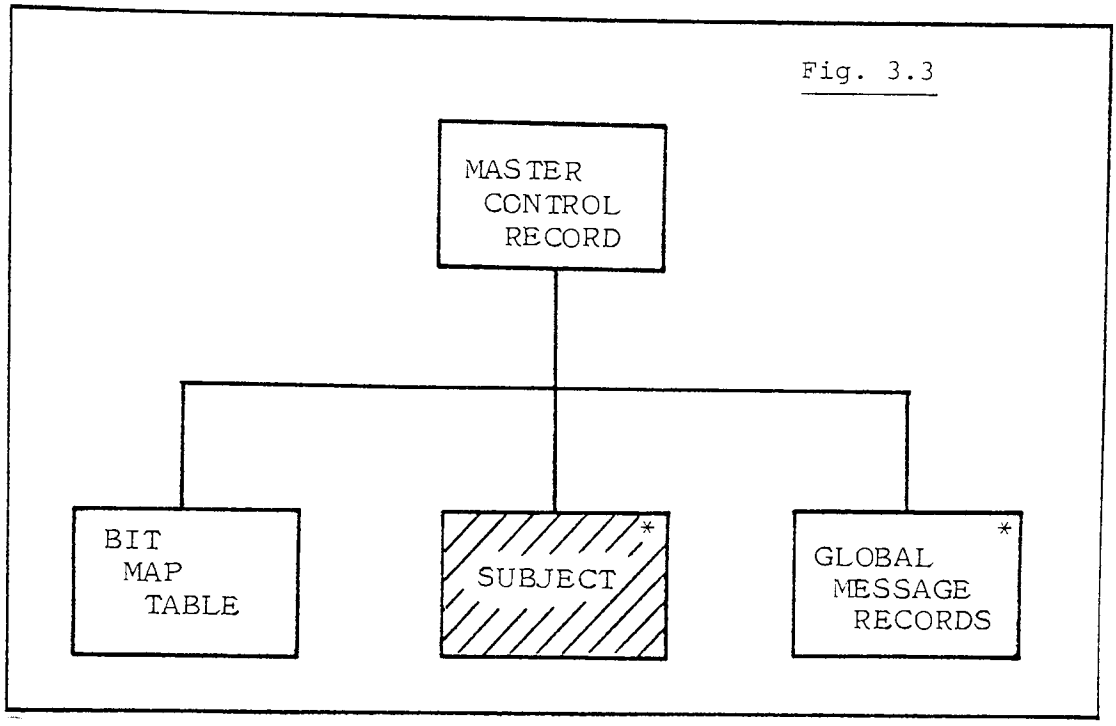


Fig. 3.3

Within this structure all the boxes represent Logical Records, with the exception of the shaded 'Subject' box, which signifies an entire Logical Segment, the next level down the hierarchy.

Relationships between logical records within segments are defined using the terms PARENT, CHILD, SIBLING and TWIN:

CHILD - a dependent of a logical segment or record higher up the hierarchy,
e.g. in the above, the Bit Map Table is a logical child of the Master Control Record.

PARENT - the logical segment or record to which a logical child is dependent,
e.g. in the above, the Master Control record is the logical parent of all Global Message Records.

SIBLING - logical records which have the same logical parent, although they themselves are not of the same type,

e.g. in the above, the Bit Map Table and any Global Message Record are logical siblings of one another.

TWIN - logical records of the same type which have a common logical parent,

e.g. in the above, all Global Message Records are logical twins of one another.

3.4 SCHOOL DATABASE LOGICAL SEGMENTS

3.4.1 Control Segment

The basic function of this is to define current database status, including:

- system version number
- passwords
- system messages
- subjects on system
- error handling information
- database free space handling

Components within the Control Segment are organised as in Fig. 3.4:

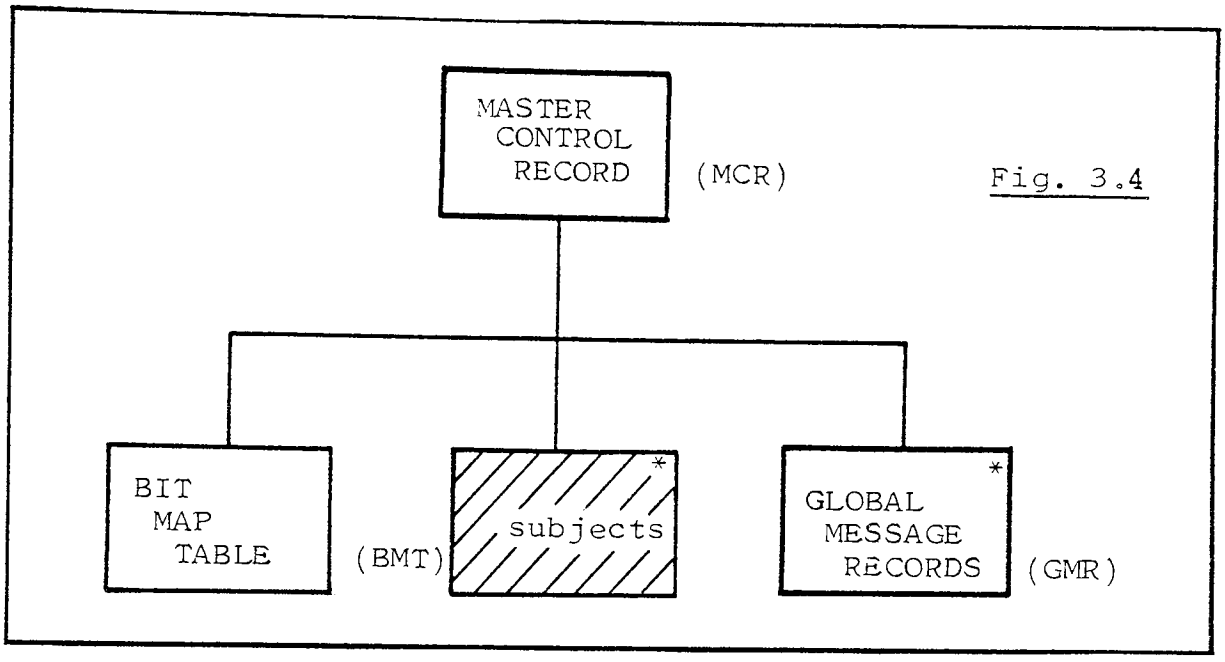


Fig. 3.4

Note: The exact logical record formats may be found by reference to Appendix 3.1 - this applies to all components discussed within this chapter.

Each of the logical records within the Control Segment contributes some part of the overall control function:

(i) Master Control Record (MCR):

The initial database 'root', this defines:

- current system software version number
- system passwords (held on the Database to facilitate frequent alterations without the need for any program recompilation);

- number of subjects on system, and pointers to the Subject Control Records that head each Subject segment;
- error handling information. Whenever a non-recoverable error is encountered, SCHOOL intercepts the normal abend procedure and builds an Error Data Block, the contents of which are written to the Database MCR. Details include abend code, location of abend (instruction and module). This enables the System Supervisor to re-trace the events which caused the problem and subsequently debug the system;
- Global Time Switch. This is used to suspend system operations (for priority maintenance or recovery work) until a particular date and time, to either students or authors or both. After the defined point in time the system is automatically reactivated;

(ii) Bit Map Table (BMT):

- defines space utilisation within the SCHOOL database. Full details of how this is used will be discussed in Section 3.5;

(iii) Global Message Records (GMR):

- a series of up to 10 messages (each up to 16 screen lines), which are displayed in sequence to all users as they sign-on to the system. Each message has an expiry date attached to it, after which it is no longer active.

3.4.2 Subject segment

The basic function of this is to define the status of an individual subject, including:

- subject identification
- no. lessons within subject
- pointers to individual lesson segments
- subject messages
- student identification and performance details

Components within a Subject Segment are organised as in Fig. 3.5:

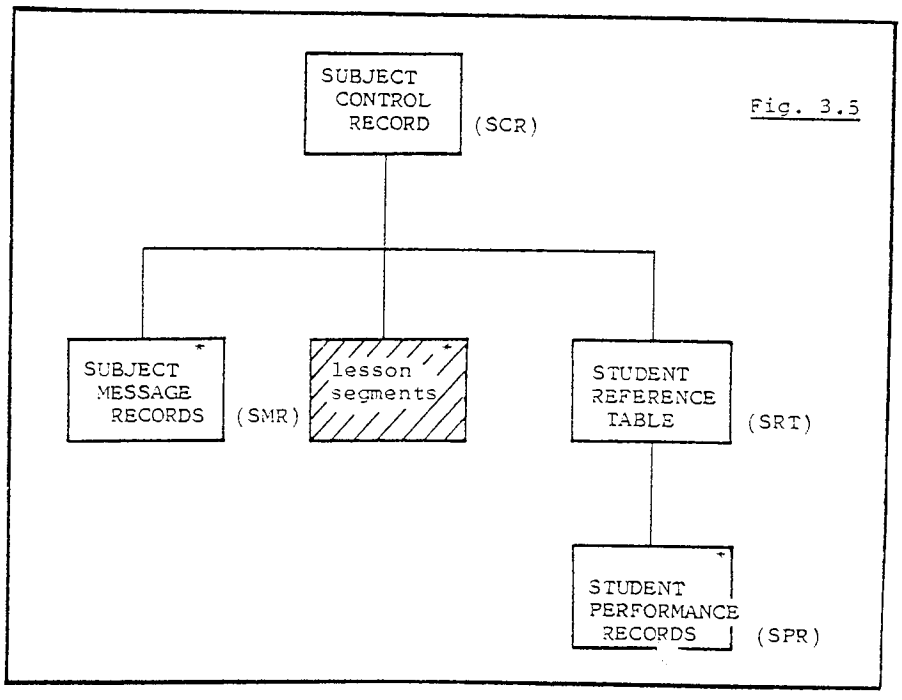


Fig. 3.5

The functional responsibility of each of the logical records within the Subject segment is as detailed below:

(i) Subject Control Record (SCR):

Defines
- subject name and reference no.;
- creation date;
- subject manager name and location;
- number and location of the various lesson segments;
- location and expiry date of the various Subject Message records;
- location of the Student Reference Table;
- a block of introductory text to be delivered whenever a subject session is started.

(ii) Subject Message Records (SMR):

- a series of up to 10 messages (each up to 16 screen lines) delivered as appropriate to students as they start each subject session. Each student has recorded, as part of his performance statistics, a message status identifying which subject message he has already received and which is next.
- each message has an expiry date attached after which the message becomes defunct.

(iii) Student Reference Table (SRT):

- a list of all students signed-on to this particular subject. Information recorded for each comprises:
  - name
  - location
  - identification no. (unique within entire SCHOOL system)
  - a link to the corresponding student performance record;
- a maximum of 100 students may be registered on any one subject at any one time;
- note that no performance data is held within the SRT, this being the function of each student's SPR. The main advantage of using this 2-level technique arises whenever a student signs-on. The identification details that he submits must be compared to those entries already stored. If this information was only held within individual performance records, a lengthy sequence of read operations would be necessary to find either:

  (a) the required record

  or (b) the fact that it does not exist.

Retaining all student identifications within one table, means only 2 read operations are required - one to access the Student Reference

Table and one to read the corresponding Student Performance Record.

(iv) Student Performance Record (SPR):

- a definition of the performance of a particular student within this subject. Each student has a unique SPR allocated to him, and the information stored includes:
  - subject message status;
  - dates (original sign-on to subject and latest session);
  - total session time;
  - last session status (i.e. stage reached and type of termination - system abend, normal lesson end etc.);
  - individual lesson statistics, including mark achieved, scoring mode (hard/easy), lesson duration.

3.4.3 Lesson segment

Subordinate to a particular subject segment, one Lesson Segment is used to define the characteristics and structure of each lesson within that subject, including:

- lesson title, author, etc.
- lesson analysis data;
- a glossary of terms peculiar to the lesson;
- a definition of required syntax processing to be applied to incoming replies;
- pointers to, and a definition of the structure of all incorporated frame segments.

Components within a Lesson Segment are organised as in Fig. 3.6

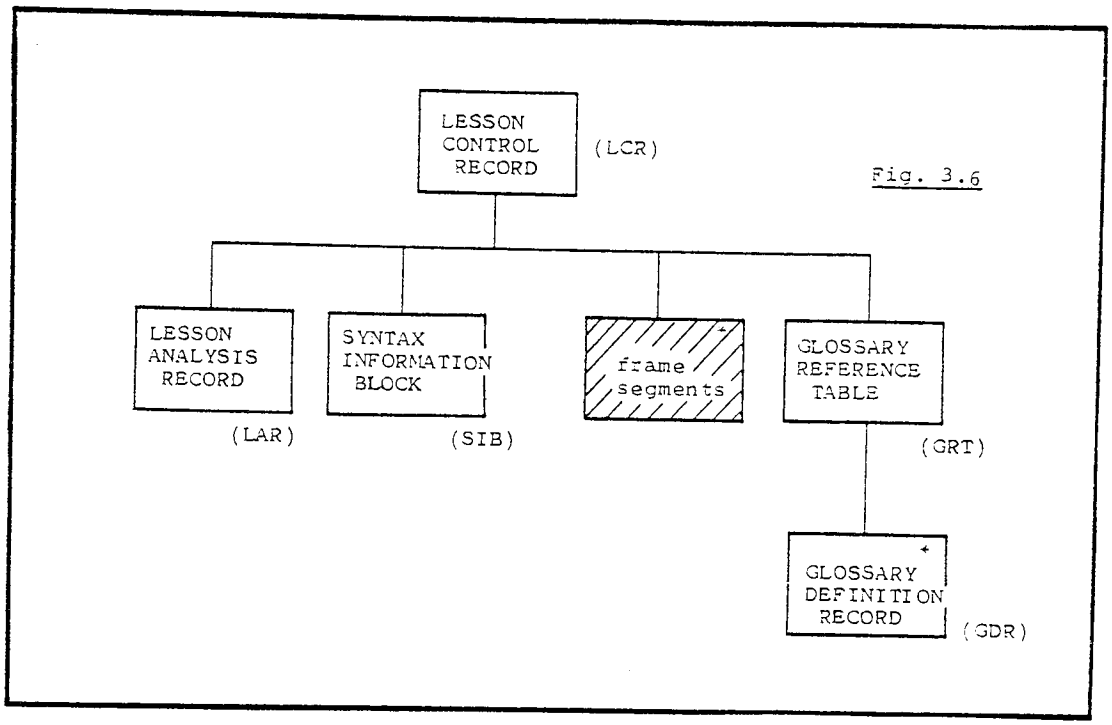


Fig. 3.6

The purpose of each logical record is as follows:

(i) Lesson Control Record (LCR):

- basically the lesson 'root', the LCR controls and defines the lesson structure.

Information held includes:

- lesson title and author;
- a security password restricting author access to the lesson contents;
- the location of Lesson Analysis, Syntax and Glossary reference information records;

- current scoring mode and passmark options;
- hardcopy status;
- the number of frames within this lesson and pointers to all frame segment logical records;
- introductory text (max. 5 lines) to be displayed to students whenever this lesson is activated.

(ii) Lesson Analysis Record (LAR):

- a facility has been built into SCHOOL whereby Lessons can be made to monitor their own performance, in terms of:
  - how often used and under what conditions it was terminated;
  - the average session duration;
  - the difficulty of each question, i.e. how often attempted, how often correct, how often hints or the answer has been requested.

(iii) Syntax Information Block (SIB):

- if present, a Syntax Information Block defines the lexical and syntactic preprocessing to be carried out on all incoming student's answer strings;
- the definitions included within the SIB encompass:-

- significance of space characters
- significance of punctuation
- redundant characters
- character substitutions
- words to be ignored
  (stop words)
- answer keywords
- max. no. words allowable in answer and, if required, the max. length of each.

(iv) Glossary Reference Table (GRT):

- each lesson within the SCHOOL database can have appended to it a Glossary facility, used to explain or monitor words/phrases that the students do not understand. The function of the GRT is to define:
  - words/phrases requested or explained;
  - whether an explanation exists (if so, a pointer to the explanatory Glossary Definition Record is included);
  - how often this word or phrase has been requested.

(v) Glossary Definition Record (GDR):

- one GDR is created for every Glossary explanation associated with a particular Lesson, and this is linked back to the corresponding GRT entry.

3.4.4 Frame segment

The lowest level within the SCHOOL database are Frame Segments, these being definitions of:

- the question or frame text to be displayed to the student;
- hints that may be of assistance in arriving at an answer;
- correct and anticipated incorrect replies;
- comments to be delivered on receiving a particular reply.

Components within a Frame Segment are organised as in Fig. 3.7.

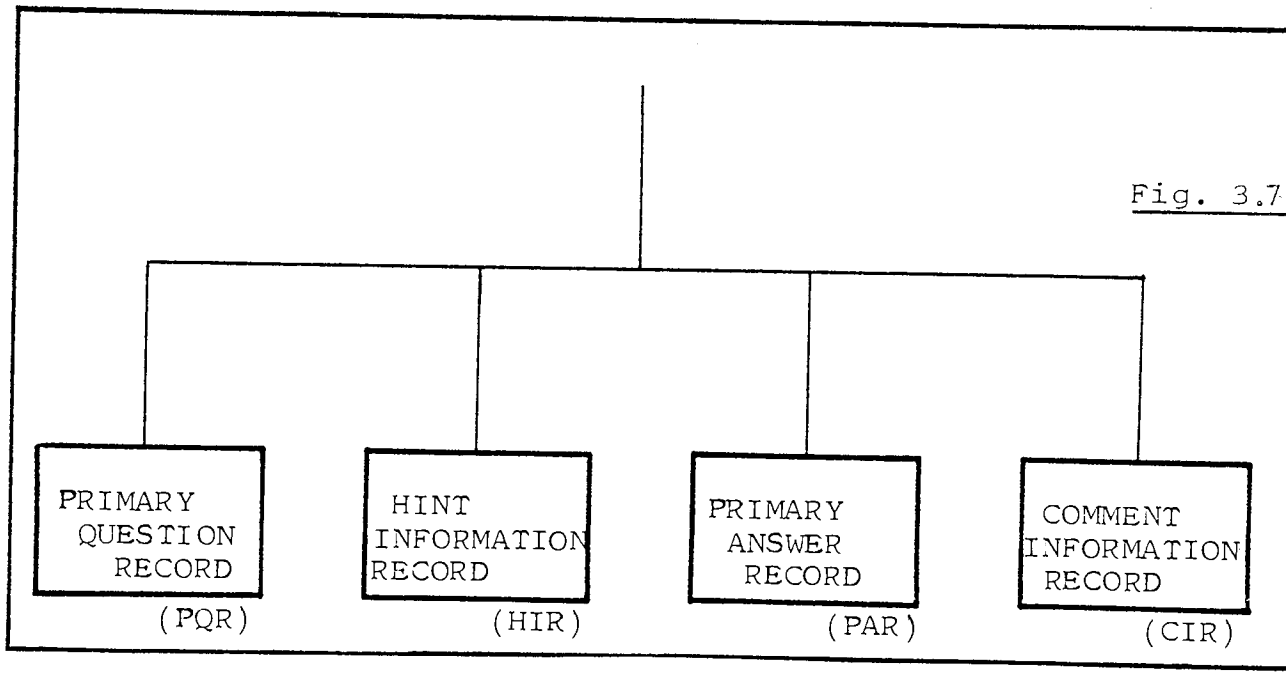


Fig. 3.7

The purpose of each logical record is as follows:

(i) Primary Question Record (PQR):

- defines the student-display including:
  - frame display (maximum of 16 screen lines). The format of this is exactly as originally defined by the Author;
  - associated marks and penalties for requesting hints;
  - next step(s) in the tutorial sequence (i.e. which frame to advance to if this one is answered correctly/ incorrectly).

(ii) Hint Information Record (HIR):

- a Hint facility has been incorporated whereby it is possible for a student to request addition supportive information. The HIR defines these hints - up to a maximum of 3 Hints, each up to 4 lines in length.

(iii) Primary Answer Record (PAR):

- the Author may define 2 types of answer:
  (a) those considered correct;
  (b) those which the student is likely to enter but which are incorrect, i.e. anticipated errors.

The PAR stores these definitions (up to 8 correct answers, up to 4 incorrect) plus associated control information, i.e.:

- maximum number of attempts allowed at this question;
- trivial answer string pre-processing (before final analysis);
- the type of replies that are to be returned to the student after answer analysis.

N.B. (a) It is perfectly feasible for the Author to specify that frame display is informative only - i.e. no student replies are expected or necessary. In this instance no PAR would be created.

(b) Processing of incoming replies against defined answers can be extremely complex - full details of answer processing may be found in Chapter 9.

(iv) Comment Information Record (CIR):

- defines to the system the exact replies the Author requires to be generated on matching an answer string with a PAR entry. A maximum of 7 2-line comments may be created, each being cross-referenced to one PAR entry.

N.B. This logical record is optional, and if not defined simple 'Correct/ Incorrect' messages will be generated.

3.4.5 Overall Structure

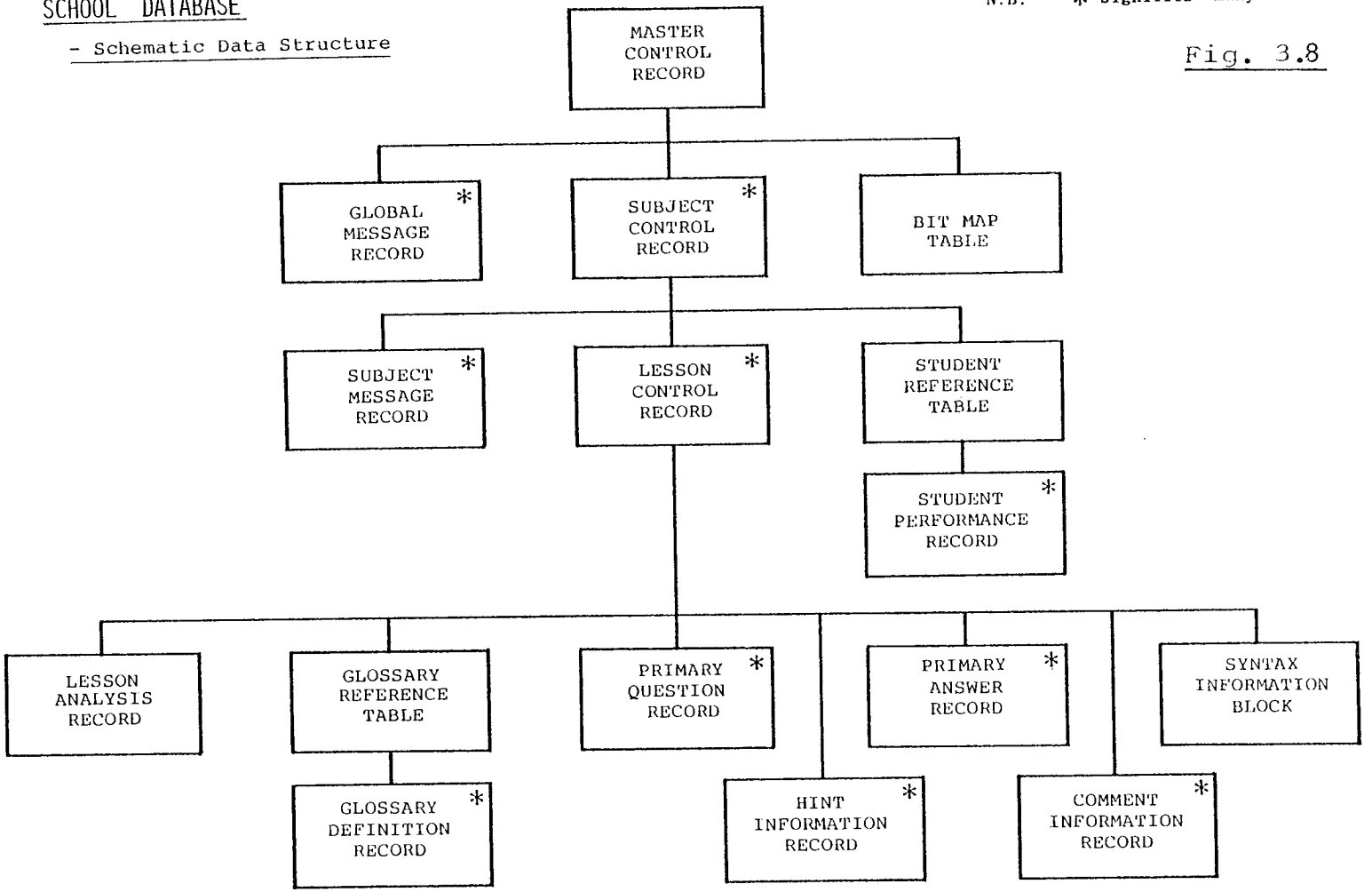
To summarise the overall logical structure of the SCHOOL database, refer to Fig. 3.8 overleaf:

SCHOOL DATABASE

- Schematic Data Structure

N.B. * signifies 'many of'

Fig. 3.8



3.5 PHYSICAL DATABASE STRUCTURE - General Notes

It is now necessary to define the internal organisation of the database as it appears to the associated Input/Output software - i.e. the Physical Database.

The first, and most important aspect governing the physical organisation of any database is what techniques are available for the Input and Output of the information in that database - i.e. the Access Method. The optimum solution is:

(i) devise a completely new Access Method, tailor-made to the database protocol;

(ii) develop all required software, and incorporate this as necessary into the host operating system;

(iii) introduce a new series of application/supervisor interfaces to activate this Access Method on demand.

This is the approach normally adopted by machine manufacturers, or specialist software houses, and it works very well - a typical example is IBM's Information Management System (IMS) which has a range of available access methods, from which can be chosen one to best suit the required application.

A variety of factors preclude this approach for SCHOOL:

- it would not be possible to alter the host operating system(s) to suit (this would be immensely complex and would have to be repeated for each new release of the operating system);

- SCHOOL is intended to run under a wide range of IBM host systems;

- often there are hardware limitations to purpose-built access methods (e.g. certain machine instructions may not be available, similar machines may not support the particular disk-device required, etc.)

It was necessary therefore to use standard access methods, and refine their use to suit the SCHOOL Database.

3.5.1 Choice of Access Method

The main considerations influencing choice of physical database access method are as follows:-

(a) record sizes will vary considerably;

(b) the database will be volatile, any may become fragmented - this must not affect performance;

(c) available database space will be limited, and efficient storage control will be imperative;

(d) data access must be very fast;

(e) whilst initial space will be limited, the technique chosen must be flexible enough to cater for a possible database expansion.

Several approaches were evaluated, taking into consideration read/write techniques (sequential, indexed, direct, relative), and record organisation, i.e.

- Fixed length physical records;
- Variable length physical records;
- Spanned physical records;

3.5.1.1 Fixed Length Records

- actual (physical) record size being dictated by the largest logical record. This has several advantages:-

  (i) Flexibility - processing can be via sequential, relative, indexed or direct access;

  (ii) Speed - 'direct' access can be achieved (without the use of algorithms) using relative record numbers - this being faster than indexed or direct access;

  (iii) Simplicity - relative record I/O is extremely easy and Database fragmentation could be easily controlled via a 'pool' of available record numbers.

The major disadvantage of this approach is of course the large waste of storage, especially if there is a considerable discrepancy between largest and smallest physical records, and if the largest occurs relatively infrequently.

3.5.1.2 Variable Length Records:

- the actual size of the physical record is set to that of the logical record (plus the mandatory record size information). This method has one enormous advantage - space wastage is minimised. There are however some important disadvantages:

  (i) 'random' access can only be achieved by direct or indexed access methods (these being significantly slower than the

relative approach). Under some host systems simulation overheads may be involved;

(ii) free space management can be very complex. A typical approach is to use several levels of free space table, each one responsible for records within a particular size range. See Fig. 3.9 below:

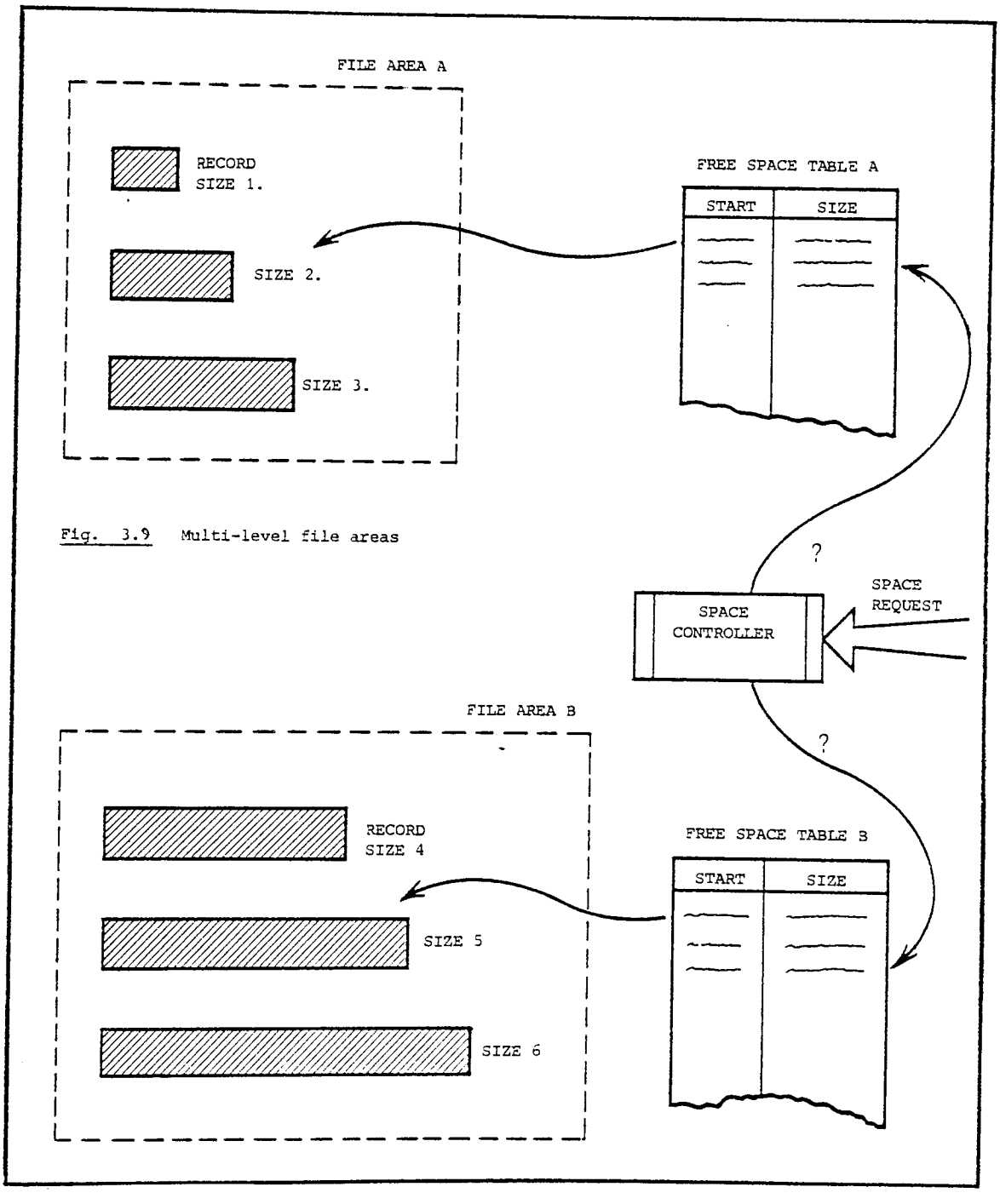
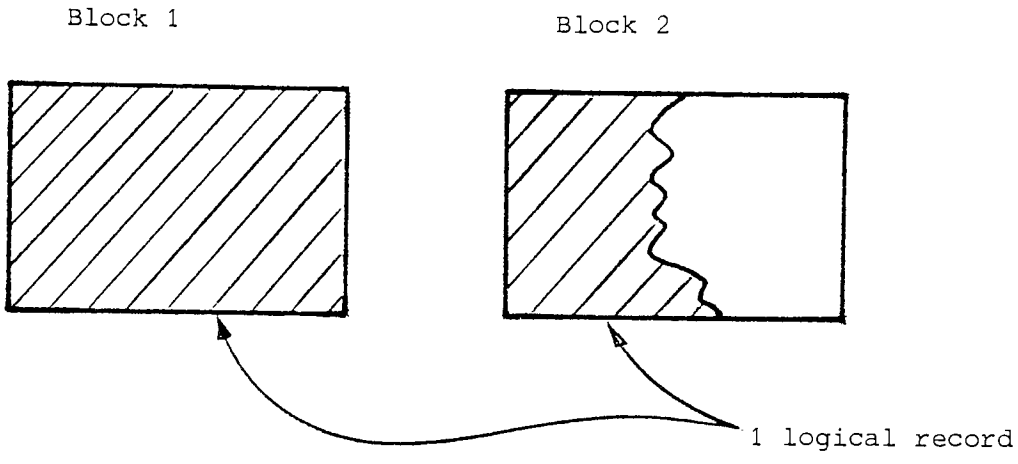


Fig. 3.9 Multi-level file areas

3.5.1.3 Spanned Records:

- the logical record occupies more than 1 fixed size physical record, or 'transfer block'. i.e.

Fig. 3.10



There are two important disadvantages with this approach:

(i) Space wastage - there will normally be some unused space in the last block of the spanned record (i.e. the unshaded portion in Fig 3.10).

(ii) Multiple I/O - it may be necessary to read several blocks before an entire logical record has been accessed.

Alternatively, the spanned record technique has several attributes to recommend it:-

(i) Flexibility - records may be read sequentially, or randomly (via relative record no. as individual blocks are of fixed size). It will be necessary to know which block represents the start of a record.

(ii) Speed - relative record access is generally fast. However this must be offset against the possible need for multiple I/O.

(iii) Simplicity - control of I/O and data-base free space is straightforward via the use of relative record (or block) numbers and tables of available units.

3.5.1.4 SCHOOL Database Access Method

None of the above, 'ready-made' access methods is immediately indentifiable as being the most suitable solution, even though they all have something to recommend them. However if it were possible to combine the various good points of each a very acceptable solution could be achieved. This is in fact the approach adopted. The start point is the Spanned Record approach - this has advantages of flexibility, speed and ease of free space control. Furthermore, if blocks are chained via a pointer system, logical records need not span physically adjacent blocks. By using relatively small I/O block sizes it is also possible to reduce space wastage - although this can result in an increase in I/O operations necessary to read any particular logical record. The outstanding problem therefore is that of multiple I/O, and as SCHOOL is to run (initially at least) on IBM machines there is a method of getting round this.

Within IBM Operating Systems (e.g. MVS, CMS) all Input/Output is controlled by a linked sequence of Control Blocks. One of these (the Data Control Block (DCB) in MVS or the File System Control Block (FSCB) in CMS) has the specific purpose of defining the characteristics of the file being processed - e.g. logical record length, block size, filename etc. Furthermore, some of these blocks are normally defined within the program area and as such are capable of dynamic alteration. Fig 3.11 overleaf illustrates the approach).

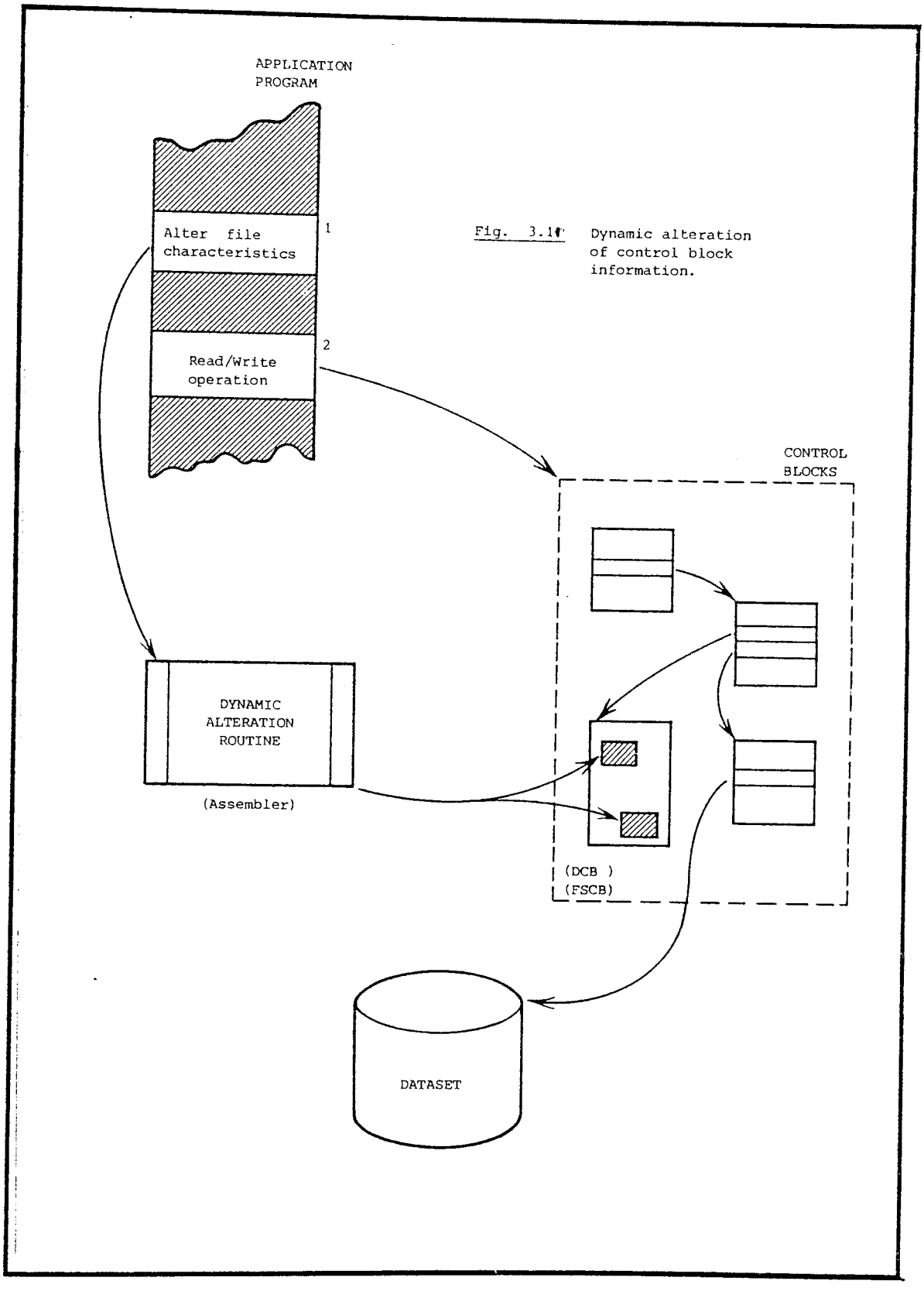


Fig. 3.1f Dynamic alteration of control block information.

It is possible therefore (via Assembler Coding) to alter the number of bytes actually transferred in any one Read or Write process - although obviously the organisation of the file must be such that contiguous storage can represent logically contiguous data.

The physical database access method chosen can be summarised as follows:-

- database storage is divided into relatively small fixed-length blocks;
- logical records span these blocks (as many as necessary) and whenever possible, a multi-block logical record is written to the database with the maximum possible number of blocks physically adjacent; data base I/O Control Blocks are then dynamically altered to process this amount of storage in one operation;
- relative block pointers are used so that although desirable, it is not imperative for one logical record to span physically contiguous blocks;
- free space is controlled at the block level, i.e. each block has a flag associated with it to identify whether it is in use or not.

3.6 IMPLEMENTATION

3.6.1. Database Transfer Units (DBTU's):

- DBTU's are the blocks from which spanned logical records within the SCHOOL D/B are built up.

- in the initial version of SCHOOL, the size chosen for DBTU's is 200 bytes. There are a variety of reasons for this:

(a) CMS (under which the development version of SCHOOL runs) imposes a protocol of 800-byte blocks on all files used under its control - hence any dataset with a record/block size which is a factor or multiple of 800 is very efficient.

(b) Whilst SCHOOL D/B logical records will vary in size, 200 bytes represents a reasonable compromise between wasted space and multiple I/O operations - especially important in the light of sub-section 3.6.3 below.

(c) I/O software associated with DBTU processing has been designed in such a way that DBTU size can be very easily altered - typically if 200 bytes under CMS should prove inefficient, or when running under other operating systems.

3.6.2 DBTU allocation and control:

- this is controlled entirely via the Bit Map Table (BMT). Each DBTU within the database has an associated bit in the BMT, the settings being:-

  0 - corresponding DBTU free

  1 - corresponding DBTU in use

- the Bit Map Table resides within the SCHOOL Database, in a fixed position immediately after the Master Control Record. As DBTU's are allocated to logical records, the corresponding bit(s) are set within the BMT, and as DBTU's become free, the BMT bits are zeroised.

- the initial size of the BMT is 4000 bytes (again easily altered). This is structured as a sequence of DBTU's in the normal way (although in this instance occupying contiguous storage in a fixed physical position.)

N.B. The size of the Bit Map Table defines the maximum size of the SCHOOL Database.

The initial allocation of 4000 bytes gives:

4000 x 8 = 32000 allocation bits

(32000 DBTU's
(
i.e. (6400000 bytes
(
(6 Mb approximately

3.6.3 Adjacency

- a technique was discussed in sub-section 3.5.1.4 whereby dynamic alteration of dataset control blocks can be used to alter the 'apparent' characteristics of a file. Using this approach it is possible to transfer any number of physically adjacent DBTU's to or from disk in one operation, and so when creating logical records, the SCHOOL D/B Management software attempts to put as many related DBTU's as possible in contiguous positions.

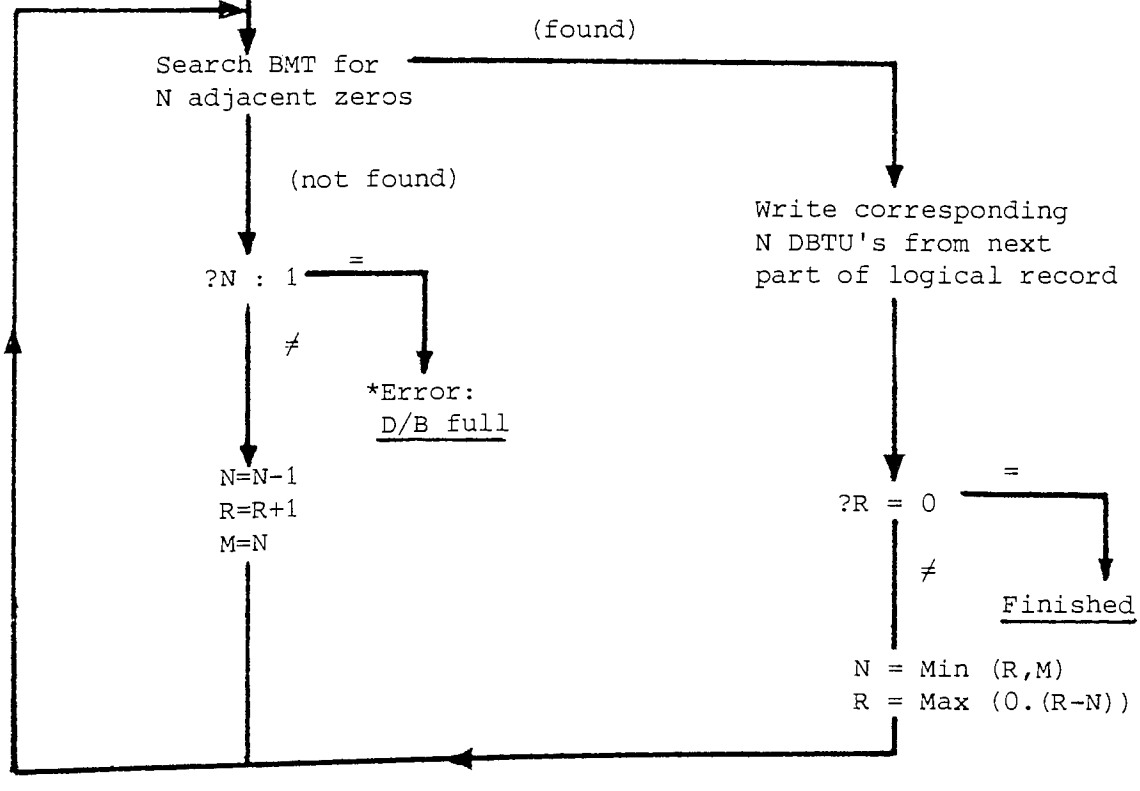
For example:

A Lesson Control Record (LCR) is to be written to the database in a form which will require 5 DBTU's. The logic involved is as follows:-

N = required no. DBTU's (i.e. 5 for this LCR)

M = N

R = O



The sequence of contiguous DBTU allocations that will be attempted is:

5
4 1
3 2
3 1 1
2 2 1
2 1 1 1
1 1 1 1 1

Non-adjacent DBTU's are chained, as the next sub-section describes.

3.6.4 Pointers and Identification codes

- each DBTU within the database (except the Master Control Record and the Bit Map Table) has as its first 10 bytes a fixed control area, the purpose of which is threefold:

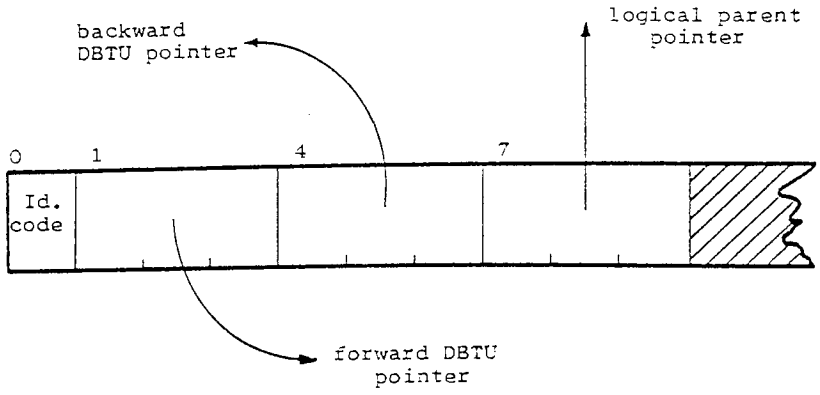
  (i) to identify the logical record type;

  (ii) to chain this DBTU forwards and backwards to other DBTU's (if any) within the logical record;

  (iii) to chain the logical record 'upwards' to its logical parent.

The structure of this control area is as per Fig 3.12.

Fig 3.12 DBTU Control Area



(a) Id. code - a 1 byte hexadecimal code identifying the logical record type. Refer to Appendix 3.3 for all allocated values;

(b) Forward DBTU pointer - points to next DBTU in logical record chain. This value may be one of:

(i) a valid non-zero DBTU Number

(ii) binary zeros - i.e. next DBTU in chain is that physically following this one;

(iii) binary ones - i.e. end of logical record chain.

(c) Backward DBTU pointer - points to previous DBTU in logical record chain. This value may be one of:-

(i) a valid non-zero DBTU No.

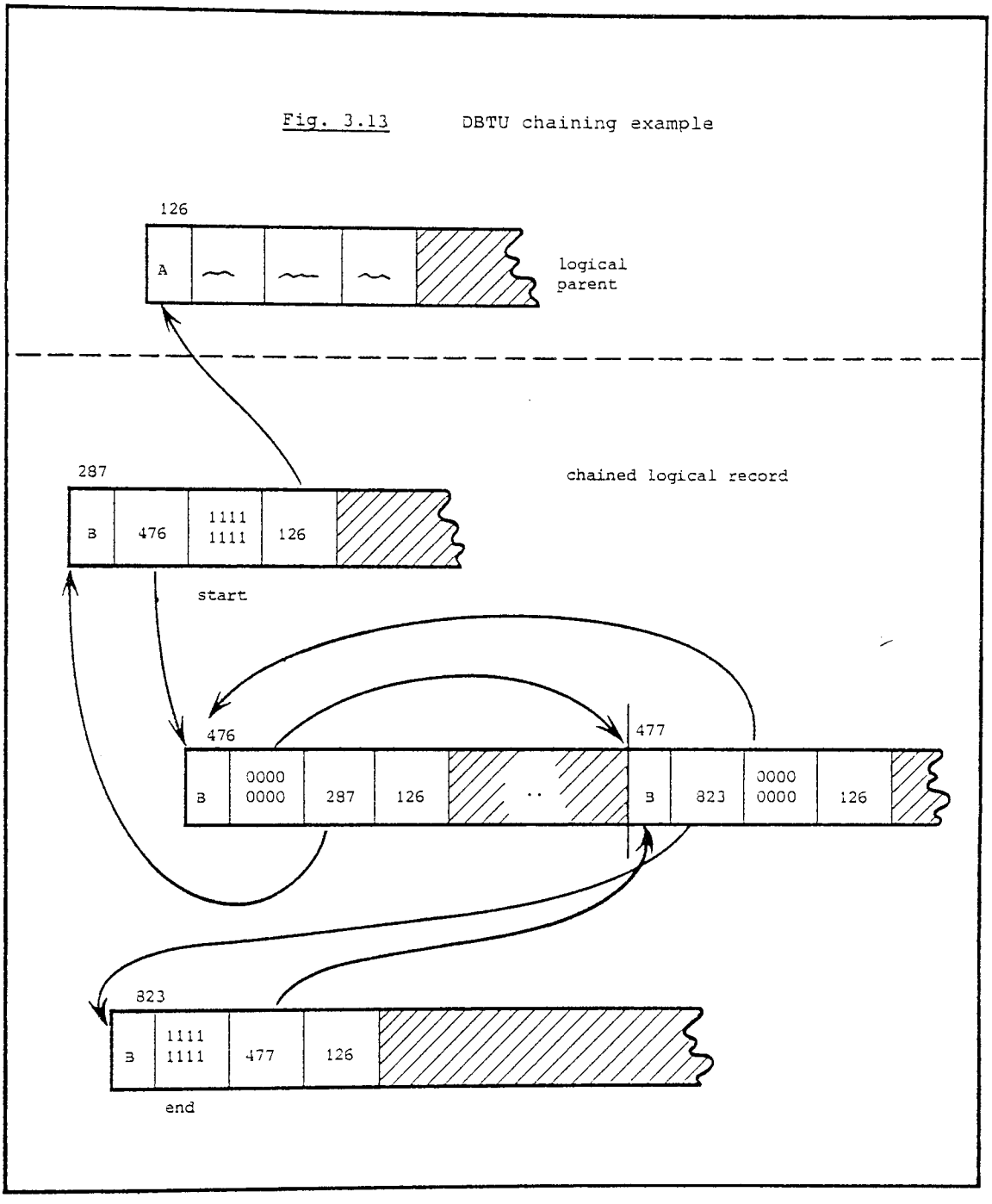
(ii) binary zeros - i.e. previous DBTU in chain is that physically immediately prior to this one;

(iii) binary ones - i.e. start of logical record chain.

(d) Logical Parent pointer - points to the first DBTU in the chain of the logical record that constitutes the parent of the current one.

N.B. To illustrate permutations of pointer values, refer to Fig 3.13 overleaf.

Fig. 3.13 DBTU chaining example



Each pointer has also a definite structure within its allocated 3 bytes. This is:

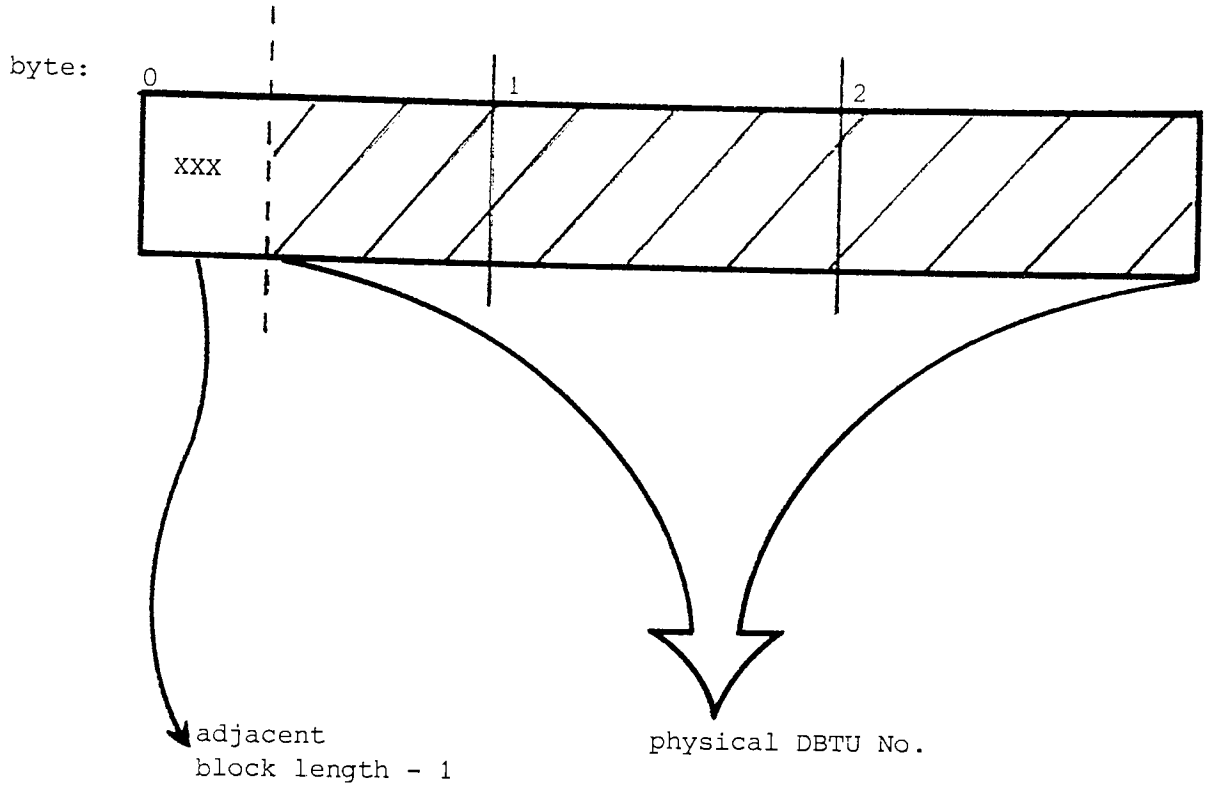


Fig. 3.14 Pointer structure

N.B. Adjacent Block Length (bits 0-2, byte 0).

- identified how many related DBTU's are physically adjacent. This value is used to alter the appropriate File control blocks in I/O operations (the value is held as 1 less than actual, i.e.

  000 - 1 DBTU

  101 - 6 DBTU's

  giving a maximum single transfer of 8 DBTU's)

The remaining pointer structure of 21 bits determines the physical size limit of the SCHOOL Database. This is:

$2^{21} - 1$ DBTU's

- i.e. 2,097,151 DBTU's

= 419,430,200 bytes

(400 megabytes).

(Should there ever be a requirement to exceed this parallel SCHOOL databases can be easily configured).

3.6.5 Physical/Logical Transformations:

- under normal circumstances, whenever information is transferred to or from the SCHOOL database, a transformation between logical and physical formats is involved. The exact nature of this transformation depends on the characteristics of the data and the direction of transfer:

(a) Writing a new logical record:

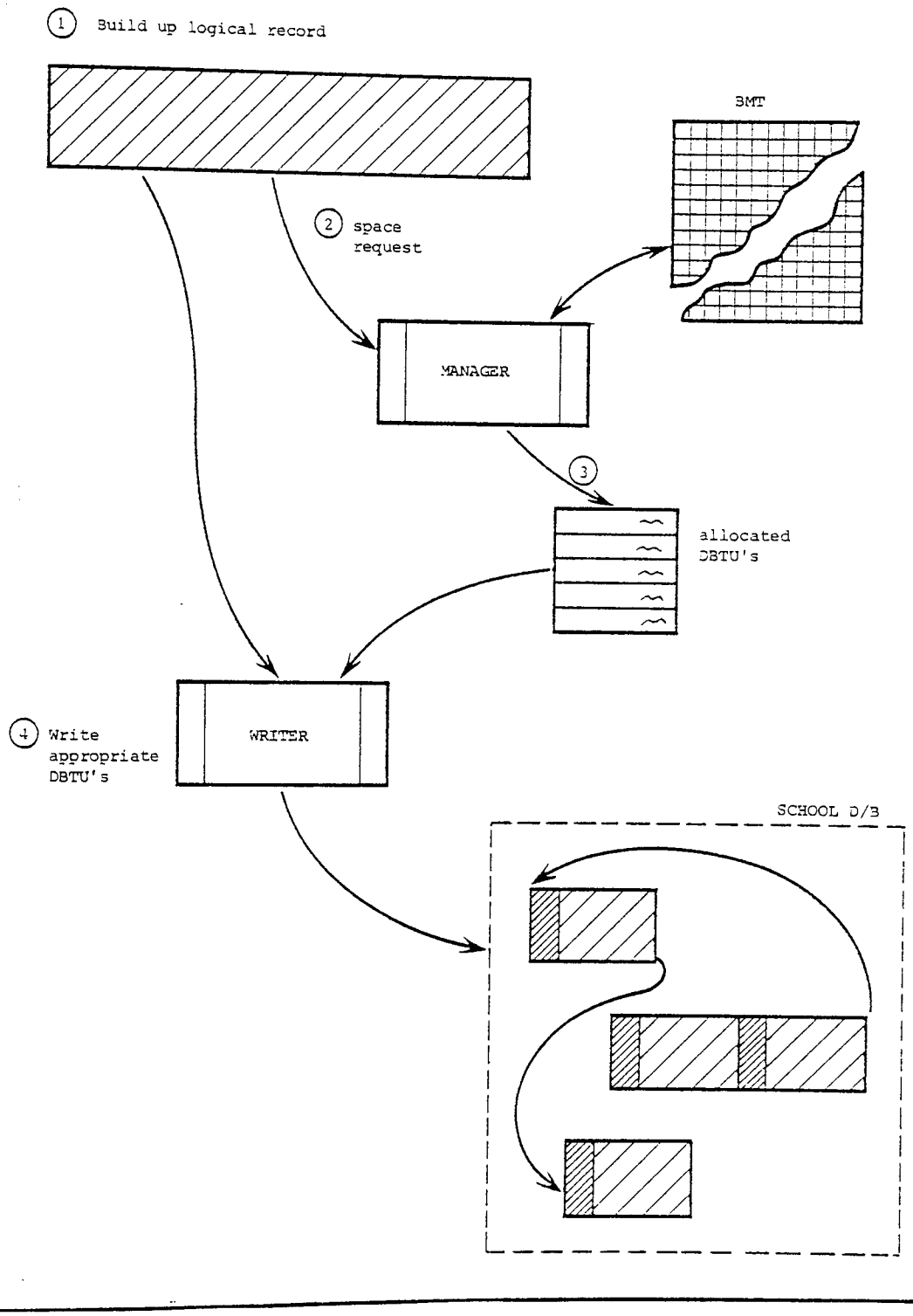
1. logical record is built up in a buffer;
2. DBTU requirement is assessed via

$$\text{No. DBTU's} = \text{CEILING}\left(\frac{\text{(logical record size)}}{190}\right)$$

3. A request is issued to the database free space manager for the identified no. DBTU's;
4. using the Bit Map Table, and the optimisation algorithm described earlier, a table of allocated DBTU no(s), is returned;
5. logical record is then written to the database using the allocated DBTU(s), plus forward and backward chaining as required.

Fig. 3.15 illustrates the processing sequence described above.

Fig. 3.15 Writing logical records to the SCHOOL Database

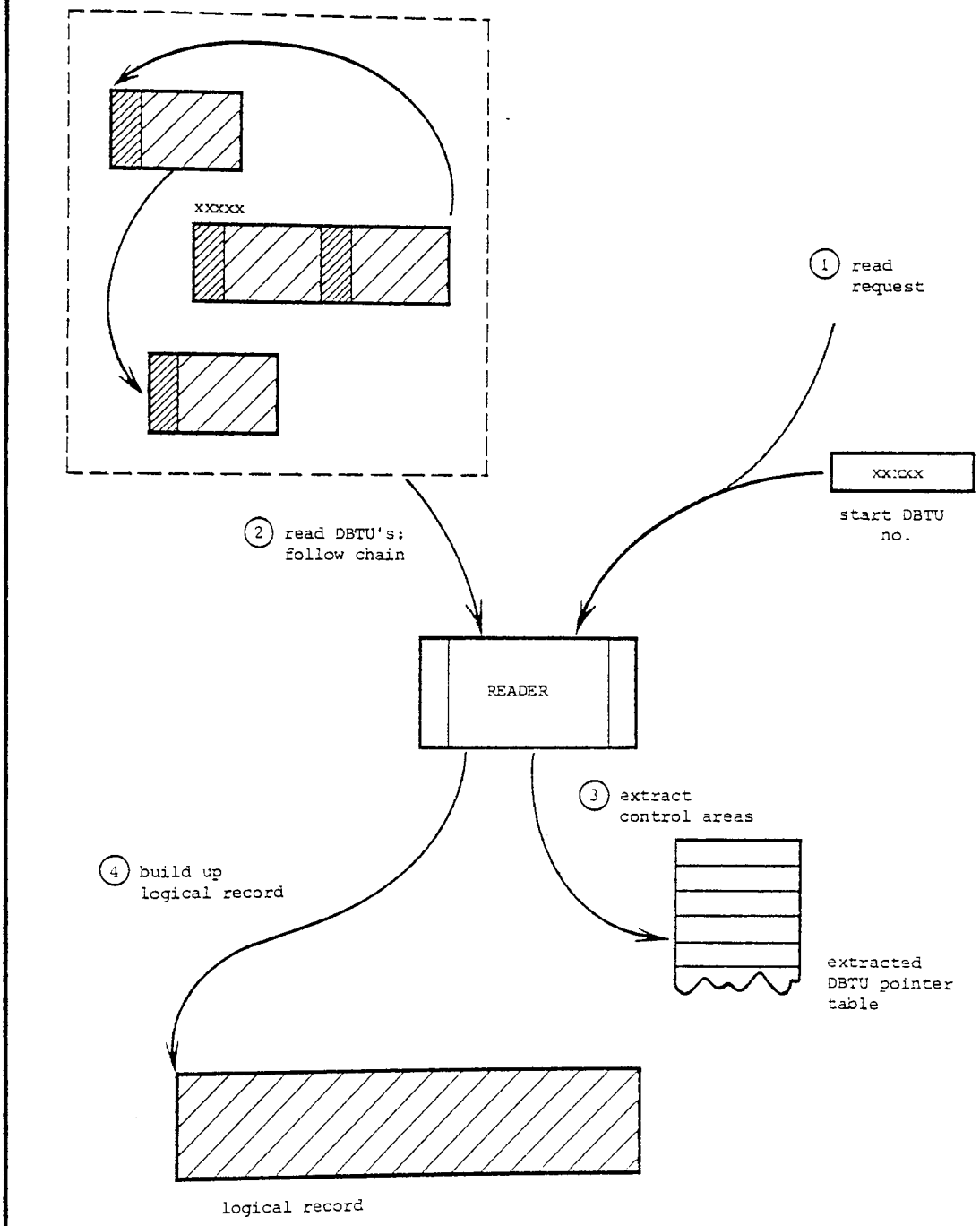


(b) Reading existing logical records:

1. start DBTU pointer is extracted (usually from the logical parent);
2. specified DBTU(s) are read into a buffer - the actual number of transfer operations may vary depending on DBTU adjacency;
3. as the DBTU chain is read, the control areas within each DBTU are extracted and stored in a pointer table for possible further use. The data part of the DBTU is then added to the logical record buffer.

Fig. 3.16 illustrates the above sequence:

Fig. 3.16 Reading logical records from the SCHOOL Database



(c) Rewriting existing logical records:

- if the logical record has not changed its DBTU requirements, then:

  1. the required DBTU locations are identified from the extracted pointer table built up at the original read phase;

  2. the logical record is then written to the identified location(s);

- if the logical record has altered sufficiently to require a different number of DBTU's. then:

  1. the procedure for creating a new logical record (steps a(2) to a(5) inclusive) is executed;

  2. the originally-occupied DBTU's (defined by the extracted DBTU pointer table created at the read phase) are flagged as now being available;

3.6.6 Transparency

- this ability to present requesting SCHOOL software with logical records, devoid of control information, has been termed TRANSPARENCY, and it can be actioned irrespective of the complexity of the physical DBTU chain. It is not mandatory however that the physical/logical transformation be actioned - in some instances (e.g. MCR and BMT handling) physical DBTU's need to be processed intact.

3.6.7 SCHOOL Database I/O Macros

It will by now be appreciated that SCHOOL database I/O processing can be an extremely complex procedure. Associated program modules (READER and WRITER) have therefore a variety of modus operandi, and consequently a range of parameters which may be passed into them. In order to facilitate the use of these modules, a simple macro language has been developed, the basic functions of which are:

SDBREAD - read from the SCHOOL Database

SDBRITE - write a new record to the SCHOOL Database

SDBBACK - rewrite a record in its original position in the SCHOOL Database

It would be inappropriate at this point to document these macros in detail, but the following examples are representative of how they are used:

(a) SDBREAD LARPOINT, CHAIN=NO, FORMAT=PHYSICAL

(b) SDBRITE FLRADDR, BUFSIZE=FLRLEN, PARENT=SUBDBTU, RECID=05

(c) SDBACK LARPOINT, BUFFER=(R1), BUFSIZE=RTOTAL

A full definition of SCHOOL Database I/O Macros may be found in the SCHOOL program reference manual.

3.6.8 Head of Database Organisation

The philosophy behind the SCHOOL database access method is based around a multiplicity of small fixed-length blocks, chained together to form logical records, in effectively random locations. This is however not true of the 2 logical record types which

form the Head of the Database. These are:-

(i) Master Control Record (MCR):

- always occupies the first 6 physical DBTU's on the database;

(ii) Bit Map Table (BMT):

- immediately follows the MCR, and currently occupies 20 DBTU's;

Having these 2 logical records fixed, both in size and location has a range of important advantages:

- the database 'root' can always be located;
- the description of database status is always available;
- the BMT can be processed very efficiently (no DBTU - chain following is necessary);
- the BMT (and by implication the upper limit for the database size) can be easily extended;
- a security mechanism can be easily built into Write/Rewrite modules to prevent accidental corruption of the MCR or BMT;
- recovery and reset utility programs become structurally simple.

chapter four

- TEXT COMPRESSION/EXPANSION TECHNIQUES

4.1 INTRODUCTION

It may initially seem incongruous to develop sohpisticated techniques for text data compression in the present climate of declining storage costs.

SCHOOL however cohabits a large, complex Multi-Access system with many other tasks and as such is designed to have the minimum impact on the total System environment both in terms of dataset storage and CPU time. To illustrate the advantages of saving disk space, consider an operational lesson database of 2 megabytes in size - a saving of only 10% represents 200,000 bytes (sufficient space to hold five 500-line COBOL source programs or allow one extra small user onto the service.) Another significant advantage is the associated saving in Database/Machine transfer time. A possible reduction of 10% here in what is essentially an I/O-bound environment is too important to be dismissed without further examination.

4.2 INTENDED IMPLEMENTATION

The logical relationship between the author, lesson text, database and student can be seen at Fig. 4.1 overleaf:

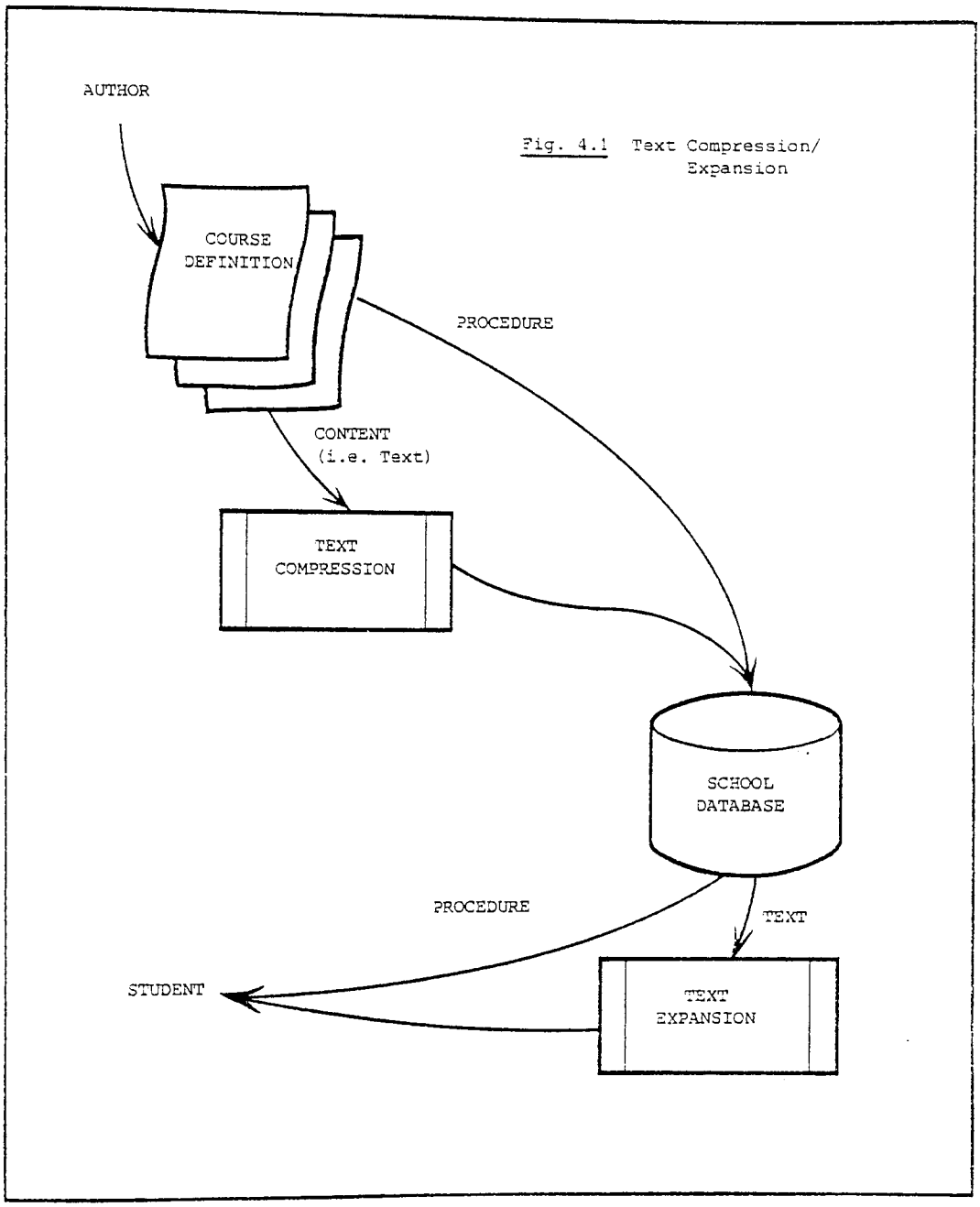


Fig. 4.1 Text Compression/ Expansion

To summarise, the Text Compression/Expansion mechanism should satisfy certain minimum criteria:

(a) Storage requirement reduction of approximately 10%;

(b) Extremely fast expansion of text from compressed state, (this will always be part of an interactive student session);

(c) Efficient compression of input text to reduced database format. Speed of performance here is not as critical - text creation being done relatively infrequently (it could in fact be a Batch activity);

4.3 INVESTIGATIONS

Several techniques for text compression already exist and these can be split roughly into two groups:

(1) methods whereby characters are coded in less bits than the manufacturers character code (e.g. ASCII,EBCDIC);

(2) methods whereby common character sequences are replaced by a unique abbreviated code;

Various compression algorithms were therefore investigated, with special consideration of the specific performance criteria laid down at the end of Section 4.2:

4.3.1 Variable Length Character Code

With this technique the number of bits used to encode text characters varies, i.e.

Most common symbols - few bits

Less frequent symbols - more bits

Using this mechanism it is possible to develop minimum redundancy codes which have the property of requiring (generally) the minimum number of bits for encoding any particular message.

Binary Huffman codes (Huffman, 32) are examples of MRC's and they approach the lower limit of bits per symbol at least as closely as any other code.

Results obtained by Wells (33) illustrate that such a code operating on approximately 130 symbols requires a minimum average ('entropy') of 4.45 to 5.58 bits per symbol, depending on the data used (compared to the original 8 bits per symbol). Informally this could produce a file compression in the region of 30 - 45%.

However, each set of data used had an optimum code created for it, and a generalised code would give less favourable results, estimated around 20%.

The decisive disadvantage of this method is that of speed. Wells in fact discussed two hardware-driven methods and states:
"The transformation from a variable-length to a fixed-length encoding is not straightforward and if implemented entirely by software may be disastrously slow".

Within SCHOOL all processes are software-driven, and as such a variable/fixed transformation is likely to be prohibitively slow.

Furthermore, a 20% compression factor is, as we shall see, not particularly unusual - of course the definition of a 'symbol' could be extended beyond single characters to include say, common English words such as THE, OF, AND, TO etc. This would then improve compression but not, unfortunately speed.

4.3.2 Fixed-length Shortened Character Code

The common EBCDIC symbol code uses 8 bits per character, giving a possible range of 256 entries.

In practice only 99 of these codes represent printable characters and it is possible to encode these characters into only 7 bits (max. no. combinations = 128). This method however has little to recommend it, as it renders only 12.5% compression and would be clumsy to implement. If however the code is reduced to 6 bits (64 combinations) a better technique can be devised, omitting certain obscure special characters and lower-case alphabetic.

Also, the use of an "8-bit/6-bit" code shortening brings about a guaranteed 25% compression, and the mapping process involved is conceptually quite simple.

Expansion is a direct reversal of compression, and this is one of the drawbacks of this technique, as it proved to be significantly slower than the Table-driven methods that will be examined in following sub-sections.

4.3.3 Dictionary Tree

With this method, a "dictionary" is created from the text as it is inserted into the Database.
This dictionary takes the form of a tree structure, words being overlapped according to the letters they contain.
To illustrate the basic principles of this technique, consider the words:

TAP TEA TERM THAT THE THESE THIN THIS

The associated segment of the Dictionary tree would appear:

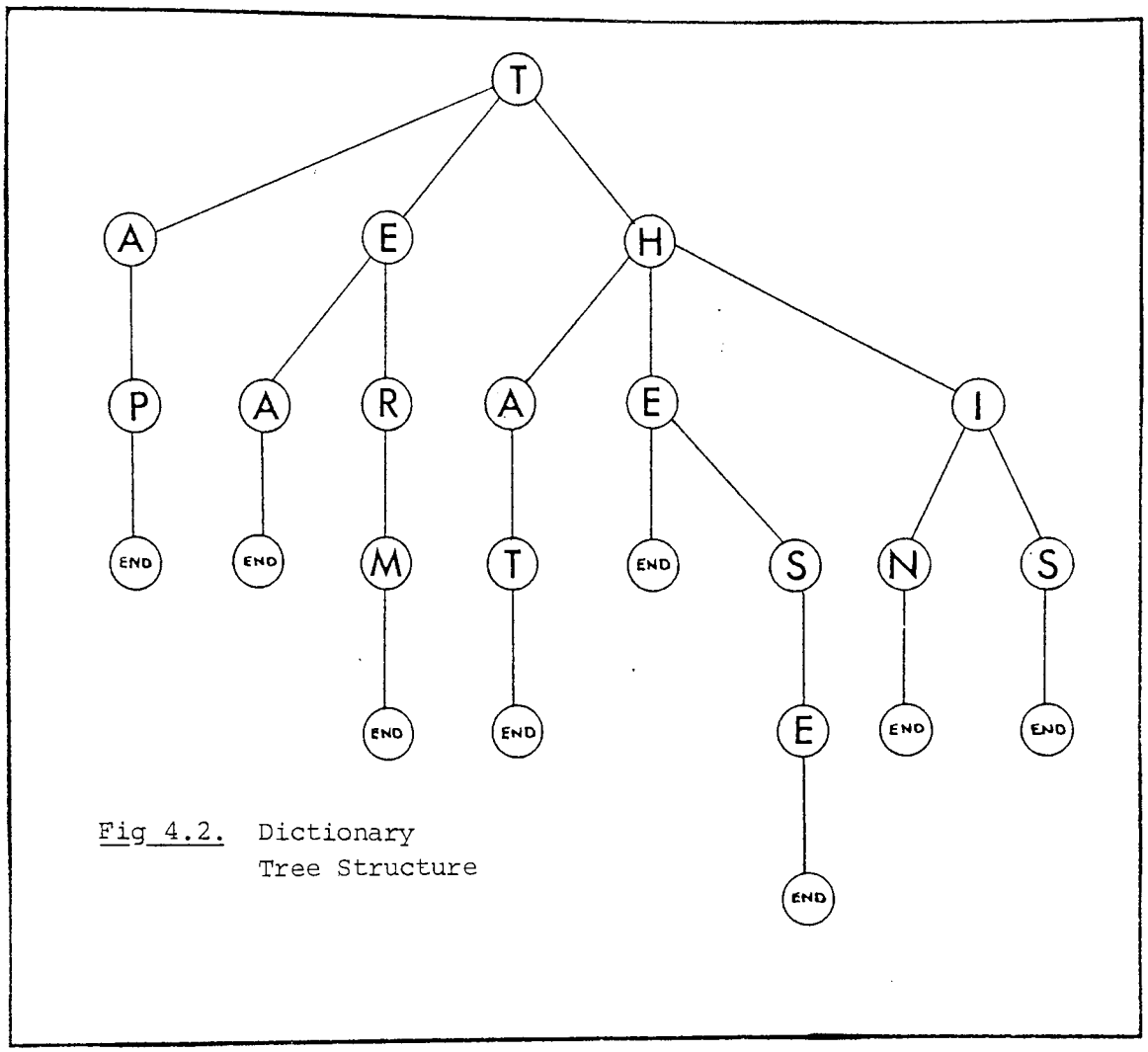


Fig 4.2. Dictionary Tree Structure

To demonstrate the effectiveness of this method, consider the 8 words in the earlier example:

No. characters in text = 30

No. characters in tree:

(1) representing a 'branch' end with a special character: 24

(2) representing a 'branch' end with a special bit setting in the last character: 16

Percentage compression (using bit-setting as (terminator) = 46.67%

These figures do not however allow for the coding necessary to represent the 'route' through the tree for a particular word - this will be discussed later.

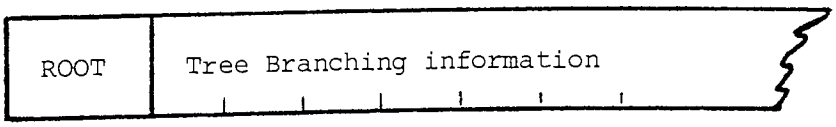
The main advantage of this technique (which has been employed in machine translation of natural languages) is that as the text vocabulary widens, then so does the scope for text compression, e.g.

COMPLETE COMPLETED COMPLETENESS COMPLETELY

There are two major disadvantages of the Dictionary Tree technique. Firstly, to be truly effective it needs very large volumes of text so that every Dictionary entry is statistically likely to occur several times. If a word is only encountered once it will require more storage than the text original (i.e. Dictionary entry + coded representation).

Secondly, the code used to substitute text words and cross-reference them to the Dictionary will be complex. A typical format could be:

Fig. 4.3



The Root would normally be one character (although IBM STAIRS (40)uses 2 for its text Dictionary). In addition, each subsequent text character would need 5 bits to represent associated tree branching (5 bits gives 32 branch options).

An estimate of the approximate compression available with this method can be calculated using data derived from West (34), Dewey, (35) and Thorndike & Lorge (36). Assuming the following:-

Sample size: 100,000 words

Average length: 5 characters

Average occurence: 10 times

Compression could be calculated as follows:-

Total free text size = No. words x average length
= 500,000 (ignoring spaces, punctuation).

No. different words = No. words ÷ average frequency
= 10,000

Dictionary size (estimated)
= 10,000 characters

Encoded word length = (Av. length-1) x 5 bits per character + 1 character for root.
= 20 bits + 1 character
= 4 bytes

Encoded text size = No. words x encoded word length
= 400,000 (ignoring spaces, punctuation)

Possible compression therefore is:

$$\frac{\text{original text size} - (\text{encoded size} + \text{dictionary size}))}{\text{original size}}$$

$$= \frac{(500{,}000 - (400{,}000 + 10{,}000))}{500{,}000}$$

= 18%

Other dictionary overheads would reduce this further.

This reduction is no better than other methods investigated. Compression also could become prohibitively slow as the Dictionary becomes large, and expansion back into normal text, although relatively simple, could not achieve the high speeds obtained by more straightforward Table-driven methods. It was not therefore adopted.

4.3.4 Common Word Removal

Incoming text is scanned for "common" words, as defined by a pre-created Common Word Dictionary. Any located word is then replaced in the text by a unique code, significantly shorter than itself.

Fig 4.4 illustrates the technique:

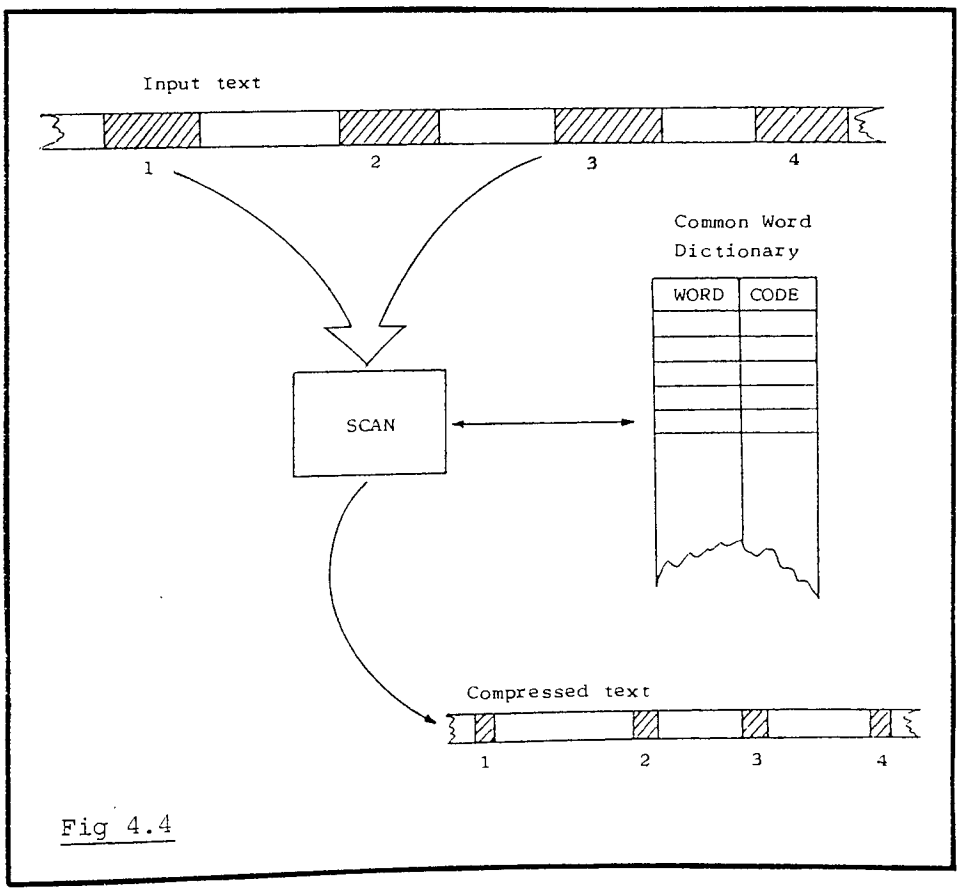


Fig 4.4

Expansion is achieved by scanning compressed text for embedded word codes; once found, the code is replaced by the corresponding word, as extracted from the Common Word Dictionary.

One significant disadvantage of this technique is that of the Dictionary Table overhead. No matter what storage reduction is achieved, this will be eroded by the size of the resident Dictionary. Any implementation of a Common Word compression method must therefore compromise between the number of Dictionary entries, their maximum length and the achievable Text compression.

This method has, however, one important advantage - it can be made to work extremely quickly (one of the original design criteria). If the Dictionary Search algorithm is carefully organised and if the Table entry number is used to replace the common word, then both compression and expansion functions will be extremely fast.

The achievable degree of compression is not easy to assess theoretically, but would seem (on the basis of informal examination) to be potentially of the same order as other techniques discussed

It was decided therefore to test Common Word Replacement experimentally, carried out in a series of stages:

(i) Dictionary creation:

compressible text within SCHOOL will mostly be normal English prose, and as such the Dictionary was created from the commonest words in written English;

(ii) Evaluation of Table look-up algoriths:

to ensure the fastest possible Table searching during text Compression. (The same algorithm is also used to insert common words initially into the Dictionary Table).

(iii) Performance testing the finalised method:

to identify the compression performance of this method against typical text streams.

4.3.4.1 Creation of a "Common Word" Dictionary

Two steps were involved:

(a) Establish a list of the commonest words in the English Language;

(b) Devise an efficient hashing function and use this to create a Dictionary Table;

Step (a) - evaluating the commonest words in the English Language.

The required information was gleaned from three sources: West (34), Dewey (35), Thorndike & Lorge (36).

Theoretically one source should have been adequate, but the discrepancies between the three were so large (between 10 and 24%) that it was felt that a compound list was necessary to give a true representation.

The compound word list was fixed at approximately 500 entries - it was felt that this would give adequate coverage of common words for good compression without prohibitive dictionary overheads.

The compound word list was produced as follows:-

(i) the most frequent 500 words and their frequency (in a 100,000 sample) were taken from Dewey as the "Master" list. This source was chosen in preference to the others (although oldest) as it accurately represents all word derivatives instead of only including the root form.

(ii) Similar lists were extracted from:

West - words occuring more than 1000 times in a sample of 5 million (441 entries)

Thorndike & Lorge - most frequent 500 words (no frequency information defined) including some Proper nouns later discarded.

(iii) The above three lists were then merged with certain special considerations being made:

- frequencies were taken from Dewey where the same word occurred in more than 1 list;
- words unique to Thorndike & Lorge were added to the compound list with minimum frequency (i.e. 20/100000):
- words unique to West were added to the compound list with their frequency adjusted to the same scale;

(iv) Root/derivative conflicts were identified and appropriate action taken. Both Thorndike & Lorge and West list words in "root" form only, e.g. SAY

(SAID/SAYS/SAYING etc. would be omitted), and a quoted frequency refers to this root and all derivatives.

The merged list was examined therefore and any words taken from either Thorndike & Lorge or West were studied in relation to any Dewey - included derivatives. If the condition

$$fr - \sum_{i} fd_i \geq 20$$

where fr - frequency of root

fd - frequency of a derivative

was found to be true, then the root was retained, otherwise it was discarded.

Note: On completion of the above a compound word list of some 679 entries was produced. This list may be found at Appendix 4.1.

(v) All words outside the required experimental limits were then deleted;

- maximum lengths 5 and 6 characters;
- minimum length 2;

Two lists resulted (see Appendix 4.2):

2 - 5 characters - 472 entries

2 - 6 characters - 572 entries

Step (b) - creating the Dictionary Table.

The Dictionary Table was created by running both ≤ 5 and ≤ 6 character compound word lists against a special Hashing program which created a Dictionary Table,

monitored the efficiency of the access algorithm, and printed out details of the generated Table.

Under normal circumstances the most important consideration of Table access algorithms is speed of identification of the relevant table entry, normally represented by the average search length (Hopgood, 37). This was used as the major performance benchmark.

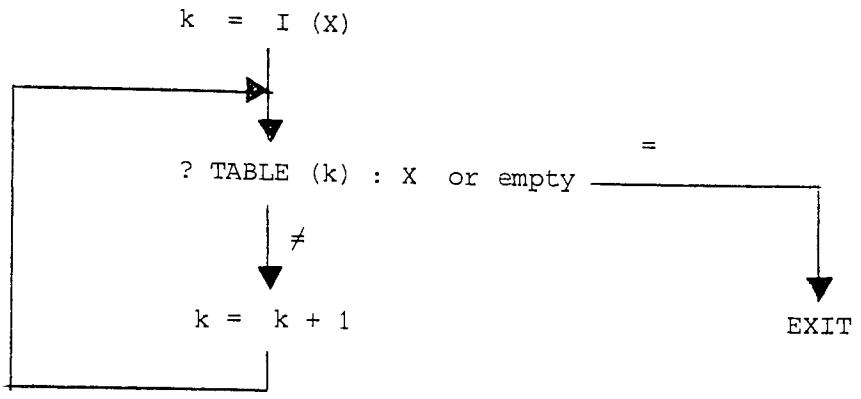
4.3.4.2 Examination of Hashing Algorithms

The usual Table access methods as summarised in Hopgood (37) were examined, and those which involve chain pointers to resolve synonyms were discarded because of associated pointer storage overheads:

The most favourable method to eliminate chaining is that known as the open hash or linear search method. Synonyms are resolved by examining successive entries after the calculated one until an empty slot (for insertion) or a match (for retrieval) is found. Chaining is therefore implicit and requires no storage overhead.

This can however produce a "clustering effect" whereby two sequences starting at positions k_1 and k_2 can come together(at k_1 or k_2, whichever is the later) and subsequently remain together resulting in excessive search lengths.

This method may be represented algorithmically as:



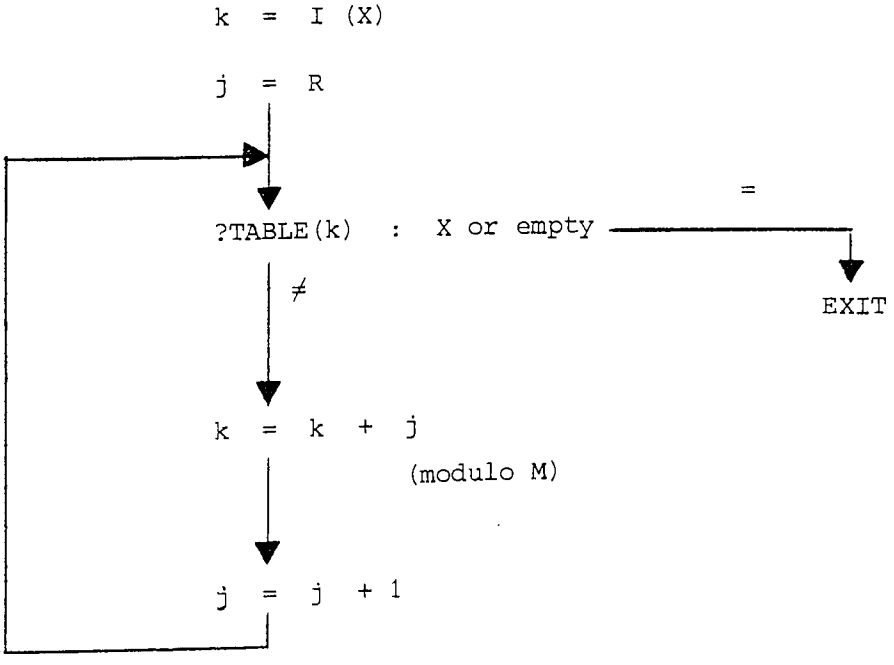
where I(X) is a mapping function applied to word X.

Hopgood and Davenport (38) have shown that this clustering effect may be eliminated by replacing k = k + 1 with

$$k = k + ai + bi^2$$

for the ith position in the sequence, with a and b arbitrary constants.

This technique is termed the Quadratic Hash method and it can be implemented very efficiently using a table size M (where M is a power of 2), R (an arbitrary constant), and altering the algorithms to:-



This was the Hashing algorithm adopted for the creation of the Common Word Dictionary, and subsequently employed in Text compression testing.

Evaluation of a suitable Mapping Function

Mapping function I(X) was also carefully chosen to give a wide spectrum of values within the range 0 to M-1, based on the following steps:-

(i) Place text word in an 8-byte field, right aligned and padded to the left with leading zeros, e.g.

WOULD	00	00	00	W	O	U	L	D

(ii) Treat the resultant EBCDIC code as a binary value, e.g. (in hexadecimal):

WOULD	00	00	00	E6	D6	E4	D3	C4

and then divide by M-1 (i.e. 511 or 1023 depending on current table size). The remainder then gives k, the initial table position. e.g.

WOULD - 208(511) / 515 (1023)

N.B.

The divisor is chosen as M-1 (where M is the table size and a power of 2) to eliminate dependence on the trailing characters of individual words - for example, if M = 1024 and was taken as the divisor, this is equivalent to masking off the last 10 bits of the EBCDIC pattern and would generate an unacceptably high synonym level.

This mapping function was therefore incorporated into the Hashing algorithm described earlier and the complete entity was coded into two Common Word Dictionary Creation/ evaluation programs, (CATCOMP1/2).

Full details of these may be found at Appendix 4.3.

Experimentation and Results:

Programs CATCOMP1/2 were run several times varying:

(i) input data - i.e. ≤5 or ≤6 - character word lists

(ii) sorted order of the input data - i.e. ascending/ descending order of frequency.

(iii) R, the quadratic constant

(iv) M, the table size.

The best results obtained are tabulated in Fig. 4.5; comparative results obtained by other sources may be found in Fig. 4.6. A complete breakdown of results may be found at Appendix 4.3.

Fig. 4.5: CATCOMP1/2 best results.

MAX. WORD LENGTH	FREQUENCY ORDER	TABLE SIZE/ FULLNESS	R	N	A
5	Ascending	512/0.9219	15	1349	2.8580
		1024/0.4609	9 or 11	674	1.4279
	Descending	512/0.9219	16	1330	2.8177
		1024/0.4609	7	677	1.4343
6	Ascending	1024/0.5586	9	887	1.5506
	Descending	1024/0.5586	3	893	1.5611

where R - search constant

N - total no. insertion attempts

A - average search length

and 472 words ≤5 characters in length

572 words ≤6 characters in length

Fig. 4.6 Comparative results for average search length, as obtained from other sources.

Source \ Fullness	0.5	0.6	0.9
1.Schay & Spruth(1962) - linear search; predicted values	1.500	1.750	5.500
2.Peterson (1957) - linear search; random data	1.541	1.823	5.526
3.Hopgood & Davenport (1972)			
- linear search; random data	1.492	1.733	5.579
- linear search; non-random data	2.748	3.762	16.674
- modified linear search; non-random data	2.689	3.199	8.927
- quadratic search; random data	1.441	1.605	2.818
- quadratic search; non-random data	2.234	2.648	4.542

Conclusions:

(i) The results by other researchers confirm the superiority of the quadratic search method over normal and modified linear techniques;

(ii) The best figures obtained by the Common Word Dictionary creation programs compare very favourably with the best obtained by others, for example:

CWD creation:

Table 0.9219 full/descending f - 2.8177

Hopgood & Davenport:

Table 0.9 full/random data - 2.818

/non-random data - 4.542

The fact that the common word data is not completely random indicates the relative efficiency of this implementation of the Quadratic Search technique, especially the adopted mapping function.

(iii) Table storage overheads using a table size (M) of 1024 may not be acceptable (although searching is very fast):

For words $\leq$5 characters:

- 472 entries used out of 1024
- wastage: 552 x 5 = 2750 bytes

For words $\leq$6 characters:

- 572 entries used out of 1024
- wastage: 452 x 6 = 2712 bytes

For a Common Word Dictionary with words $\leq$6 characters long, a modification was developed as detailed in the following Extension sub-section.

(iv) It is important to take into consideration the relative frequencies of common words when deciding which method of Dictionary creation to adopt.

To maintain a satisfactory practical performance the more common words should be inserted into the Dictionary Table first, thus ensuring a minimal search length.

Only Dictionaries created from descending frequency word lists will be used in performance tests therefore.

Extension:

As previously detailed, using a table size of 1024 for 572 words gives an unacceptably large amount of wasted storage. Two alternatives were considered:

(a) reduce the words to approximately 470-480 and reset M, the table size, to 512.

(b) adopt an earlier type of quadratic search using a prime value of M, (instead of a power of 2) in the region of 638 - to retain a 0.9 table utilisation figure.

The former was rejected as it negates one of the advantages of having an 'up to 6 - character' word list - i.e. a wider word spectrum to justify the extra character per table entry.

The latter was implemented with M = 631 (table utilisation 0.9065) and another derivative of the test program was produced and run - CATCOMP3 (see Appendix 4.3). This produced a best result of:

N = 1475

A = 2.5786

R = 8 (with descending frequency order)

4.3.4.3 Performance Testing the Common Word Dictionary Method

Step 1: The two most efficient algorithms as previously established were taken and using descending frequency word lists two CMS disk files containing the generated Dictionary Tables were created, i.e.

NUMBER CHARACTERS PER WORD	TABLE UTILISATION	TABLE SIZE	R	A
≤5	0.9219	512	16	2.8177
≤6	0.9065	631	8	2.5786

Fig. 4.7 Compression test parameters

N.B.
The Tables generated may be seen at Appendix 4.4

Step 2: - These files were then converted into program arrays, in standard Assembler language format.

To resolve the problem of determining when the end of a search sequence is reached, Hopgood & Davenport (38) state that with the quadratic hash method a search end is readily identified when the ith and i+1th entries in the sequence both map onto the same Table position. This may be suitable for the general case, but in a specific application, a tailor-made sequence end identification can be devised which is immensely more efficient. The following technique was therefore produced:

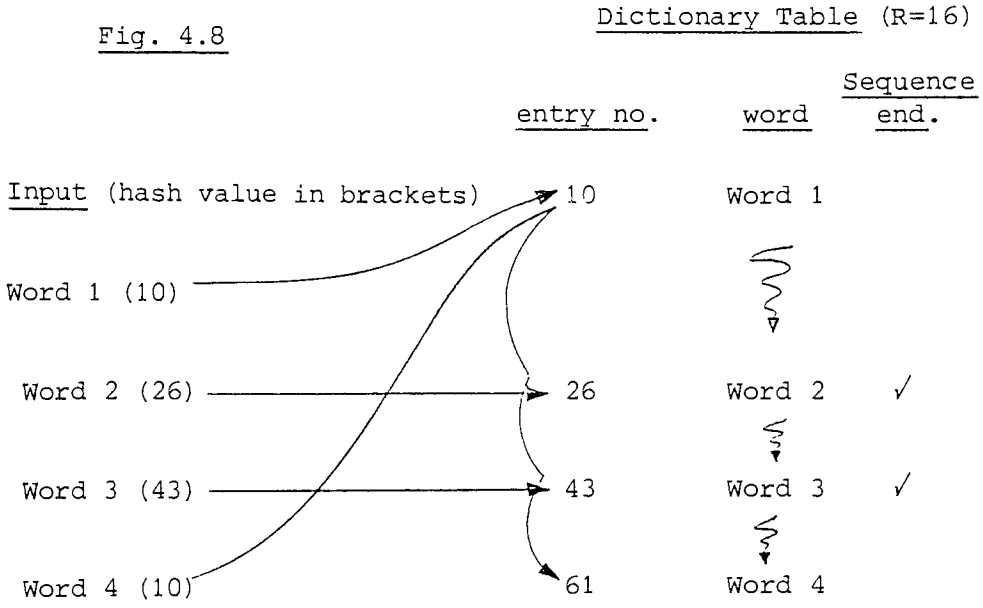
(i) From the generated Dictionary Tables the hash value sequences were evaluated;

(ii) The last entry in the sequence was established, allowing for 'wrap-round' (sometimes more than once);

(iii) The first character of the sequence-end word was complemented, and program arrays altered to reflect this;

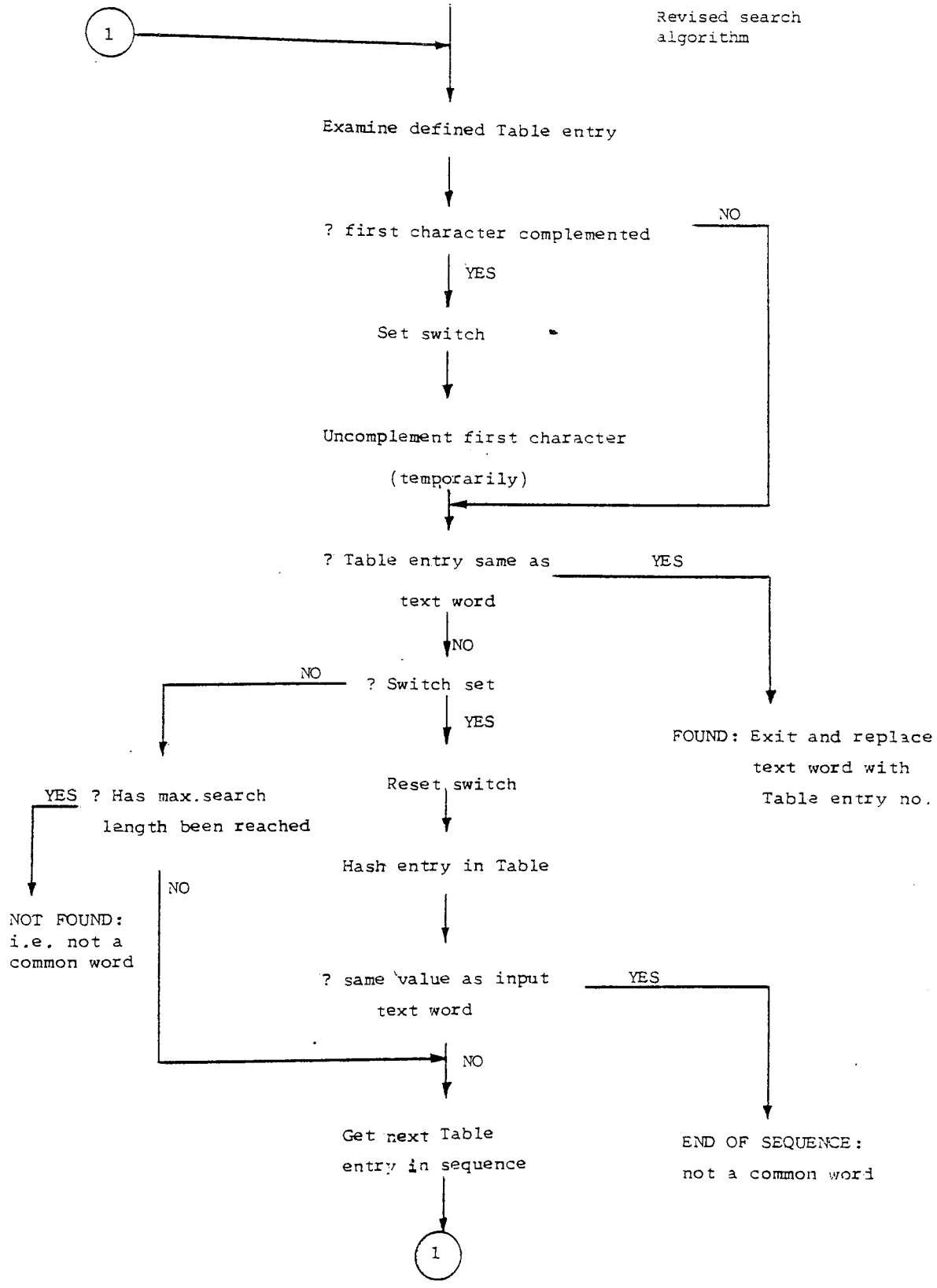
This alone does not uniquely define search sequence terminators, as the following example illustrates:

Fig. 4.8



To accommodate this 'false end' problem, the Search algorithm was extended as follows:-

Fig. 4.9


1
Revised search algorithm
Examine defined Table entry
? first character complemented
NO
YES
Set switch
Uncomplement first character (temporarily)
? Table entry same as text word
YES
NO
? Switch set
NO
YES
FOUND: Exit and replace text word with Table entry no.
Reset switch
YES ? Has max.search length been reached
Hash entry in Table
NO
NOT FOUND: i.e. not a common word
? same value as input text word
YES
NO
Get next Table entry in sequence
END OF SEQUENCE: not a common word
1

Some points of special significance:

(i) The maximum search sequence length with the established values for R was 37;

(ii) The overhead incurred by the extra Hash operations can be shown to be more than offset by the shortened search length;

(iii) Some calculations were carried out (using the Dictionary Table for words ⩽5 characters, Table size = 512, R = 16) to examine the effectiveness of this method and <u>on average</u> the no. extra hash operations for a word that <u>is present</u> in the CWD is:

average no. matches (2.8177) x $\frac{305}{512}$ = 1.679

<u>N.B.</u> 305 is the number of different Hash values in the Table, i.e. the no. different search paths.

<u>Step 3</u>: Two analysis programs were then written incorporating the above Common Word arrays:

CATCOMP4 - words ⩽5 characters)
) See Appendix 4.5 for full details
CATCOMP5 - WORDS ⩽6 characters)

Briefly, CATCOMP4/5 follow the following pattern:

(i) Read in text:

the text chosen for testing was the entire instructional text from Compower's existing CPS/CAI system, comprising Introductions, questions, responses, sample CPS routines etc.

(ii) Search for common words:

using the evolved CWD Table, Hashing algorithm and search sequence termination techniques;

(iii) Replace any matched word with a 2-byte code indicating its Table position;

(iv) Examine the character following a common word. If non-blank, set bit 3/byte 1 of the compression code. If the character is a space, delete it from the compressed text altogether.

Examples:

THIS (Table entry 499, i.e. x 'IF3')

THIS␣ - | 01 | F3 |

- | 0000 0001 | 1111 0011 |

THIS. - | 1 1 | F 3 | 4 B |

- | 0001 0001 | 1111 0011 | 01... |

bit 3 set to indicate that no space is to be inserted after the word during expansion.

(v) Print out the original text line and equivalent compressed line (representing the 2-byte compression code by '||');

(vi) Finally, print an analysis of the compression achieved;

Step 4: Test runs of CATCOMP4/5 were then carried out on:

(i) The complete CPS/CAI System text

(ii) The Introduction to the above (nearer free English text with no PL/I statements incorporated.)

Output from some of the above tests may be found at Appendix 4.6, and tabular analyses of the results follow:-

Fig. 4.10 Results from CATCOMP4 (common words ≤5 characters long).

	Full Text	Introduction Text.
No. characters input	22331	2605
Total no. words	3958	463
Average word length	4.131 chars	4.434 chars
No. common words	2122	285
Hit rate	53.613%	61.555%
Overall storage reduction	4434 bytes	583 bytes
Compression factor	19.856%	22.380%
Process time *	5.00	0.78

* Virtual machine CPU time (in seconds) as evaluated and returned by the VM/370 System, run on IBM 370/148;

Fig. 4.11 Results from CATCOMP5 (common words ≤6 characters long)

	Full Text	Introduction Text
No. characters input	22331	2605
Total no. words	3958	463
Average word length	4.131 chars	4.434 chars
No. common words	2268	312
Hit rate	57.302%	67.387%
Overall storage reduction	5126 bytes	714 bytes
Compression factor	22.954%	27.408%
Process time *	5.29	0.87

* Virtual CPU time.

4.3.5 Common String Removal

Similar in concept to Common Word Removal, this involves the identification and replacement of common character strings within the source text.

Two approaches for identification of common strings are possible:

(i) preprocess the original text to assess its contents and generate a Specific Common String Dictionary (CSD);

(ii) use a Standard CSD as evaluated independently from general text samples.

Mayne and James (39) have produced a technique within the first category, broken down into four phases:

(i) Preprocess - establish string frequencies (these may vary in length);

(ii) Generate CSD;

(iii) Compress input text and store;

(iv) Decode compressed text for display as required.

Compression factors between 40.6% (special format data) and 37% (normal English) were obtained. This however is at the expense of extremely slow text compression (phases (i), (ii) and (iii)) and is not suitable for volatile text - each alteration should theoretically be accompanied by a regeneration of the CSD. This is acknowledged by Mayne & James.

The use of a standard Common String Dictionary is much better suited to text which is subject to change. Two formats for this standard CSD are available:

(i) fixed length strings;

(ii) variable length strings.

In either instance, 2 bytes will be required for a compression code (as evolved for Common Word Removal).

4.3.5.1 Fixed length:

(a) 3-character strings (e.g. THE, ING, ERE, BLE):

- these would reduce to 2 bytes, giving a maximum compression of 33%. Actual results would probably be much less than this, as not all 3-character sequences will occur in a CSD of convenient size;

(b) 4-character strings (e.g. MENT, ØULD, RING, TIØN.

- these would also reduce to 2 bytes, giving a theoretical maximum compression of 50%, but practical results using an acceptable CSD size would be significantly less (particularly as many 3-character words would not be compressed).

4.3.5.2 Variable length:

- common strings within a range of 3 to 6 characters in length would be compressed. Whilst this would initially appear to be more flexible and capable of greater compressions than Fixed-length approaches, actual results would not exceed 37%, as achieved by Mayne & James using a specific CSD. Brief experiments using this technique have failed to achieve 20% compression (see Appendix 4.7) and proved slower than the Common Word Removal approach.

4.4 ACTUAL IMPLEMENTATION

The objectives of database text compression were laid down at the start of this Chapter (p.86), and based on these and the nature of CBL text and its volatility, the Text Compression method adopted was that of Common Word Removal (words up to 6 characters long, table size: 631). Whilst actual text reductions will vary, it is expected that these will normally exceed 25% with very rapid expansion and compression processing. Further 'format compression' improvements have been introduced - see Chapter 8, Section 8.6 - which enhance compressions to over 50%.

N.B. For a discussion of Text Compression modules, refer to Chapter 7, Section 7.4.3.

Corollary

Words longer than 6 characters (which are currently not compressed) can have their leading 6, 5 or 4 characters compared to the Common Word Dictionary entries and be compressed appropriately, (e.g.BECOMES, HAPPENING, TRUTHFUL, MOMENTOUS, THATCHER). Whilst this has not been incorporated as yet, useful further reductions would be achieved with only minor processing overheads.

chapter five

- SCHOOL SYSTEM STRUCTURE
  AND CONTROL LOGIC

5.1 INTRODUCTION

As with most complex software, SCHOOL has been organised as a series of individual functional components, each with a particular responsibility towards the whole. However, what is not necessarily true of other systems, is the degree of formalisation that has been applied to this structure and the techniques used during software development. This chapter describes these approaches.

5.2 SCHOOL SYSTEM ORGANISATION

The software of SCHOOL is designed and organised on a number of levels:

(i) System Control Manager:

- overall control of processing

(ii) Subsystems:

- the various procedural function of the system

(iii) Modules:

- individual programs which together constitute any subsystem .

Hence, in keeping with the SCHOOL database, the SCHOOL software can be represented as a hierarchy, as Fig. 5.1 illustrates:

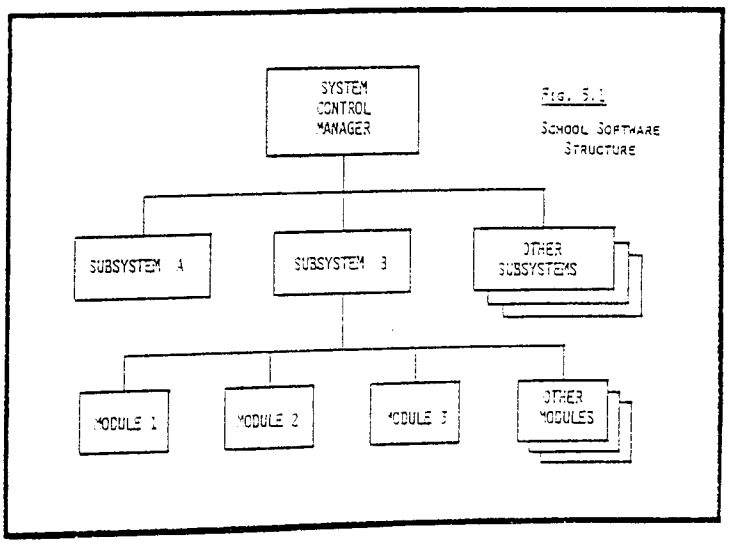


Fig. 5.1
School Software Structure

Not all flow is strictly 'top-down' however, and inter-subsystem communication is possible - facilitated by the definition of two types of subsystem, Major Function and Service Function.

This subdivision into multiple separate components does however introduce a problem when considering subsystem and module inter-communication - normal parameter interchange techniques would involve a considerable processing overhead as data (or addresses) are moved from module to module. To alleviate this the concept of Data Nuclei has been introduced, the basis of which is that each subsystem has as part of its structure an area of storage used as a communications 'mailbox' between itself and the remainder of the SCHOOL environment.

5.3 SYSTEM CONTROL MANAGER

The function of the System Control Manager is to activate SCHOOL, decide on the type of use (including appropriate security checks) and then pass control to the requisite Major Function Subsystem. Once the user has finished his work, the System Control Manager resumes control and closes the system down. This is all controlled by a single module (CYBER) and the exact sequence of events is as follows:-

(i) establish all internal run time addresses;

(ii) open the SCHOOL Database and read in the Master Control Record (MCR);

(iii) read in the current Bit Map Table (BMT);

(iv) request details of the current terminal type from the host Operating System (these are used for Output optimisation purposes by the Display Format Subsystem);

(v) generate SCHOOL logo on user terminal (exact format is device-dependent).

(vi) request and identify Operator Mode - i.e. type of user (Student, Author or System Supervisor). Note that in the instance of the latter two, passwords are requested - should the user not enter this correctly within 4 attempts CYBER will automatically terminate the system;

(vii) delegate control to the requested Major Function Subsystem;

On completion of processing within the Major Function Subsystem, CYBER once more takes charge and the following sequence ensues:

(viii) request and identify whether or not the user wishes to continue the session. If the answer is affirmative proceed as per (vi) above, otherwise as below;

(ix) close down the SCHOOL database, deliver a closedown message and exit;

An organisation diagram of the System Control Manager is as in Fig 5.2:

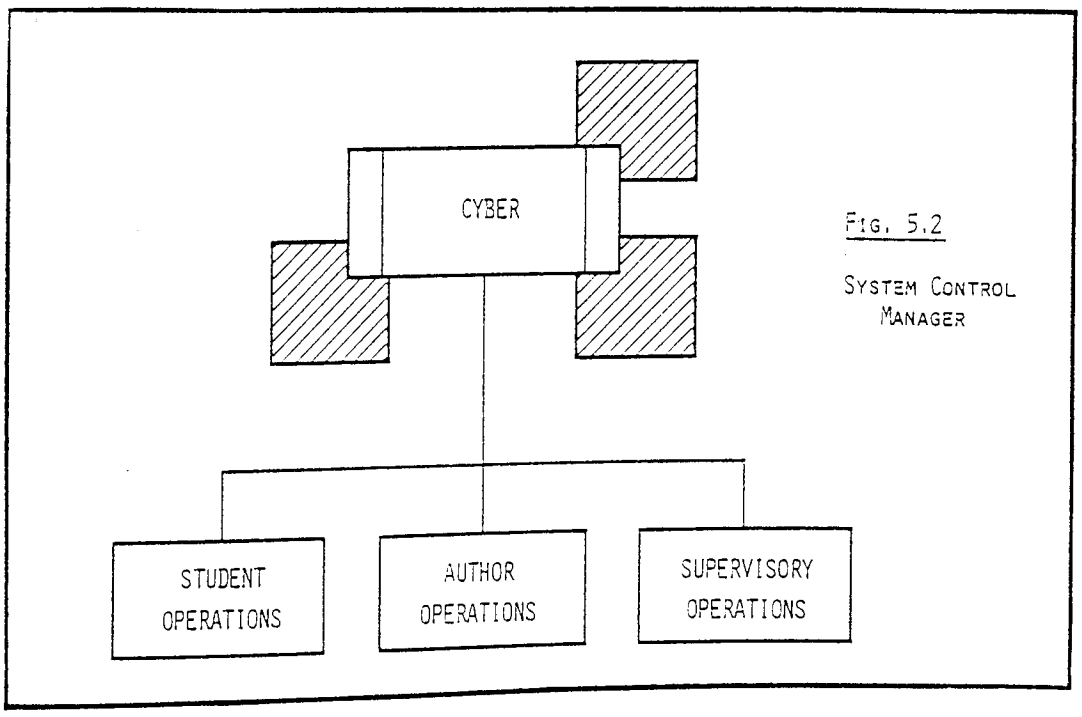


FIG. 5.2

SYSTEM CONTROL MANAGER

In Fig. 5.2 the shaded boxes represent the 3 Data Nuclei that CYBER uses. These are concerned with:

- overall system operation (i.e. the master Data Nucleus)
- database free space management (i.e. the Bit Map Table).
- error handling.

5.4 SUBSYSTEMS

All of the remaining software of SCHOOL is organised into discreet subsystems, each comprising one or more program modules.

These subsystems are further subdivided into 2 categories:

(i) Major Function Subsystems:

- i.e. one subsystem for each of the operational modes of SCHOOL (student, author and supervisor).

(ii) Service Function Subsystems:

- i.e. a subsystem for each operational function within SCHOOL (e.g. monitoring keyboard input, handling database I/O etc.).

Major Function Subsystems can only be called from CYBER, whereas Service Function Subsystems can be called from any source (recursion excluded). Communication between subsystems is effected via mutually visible Data Nuclei - which may vary considerably in size and complexity.

5.4.1 MAJOR FUNCTION SUBSYSTEMS

The three primary operating modes of SCHOOL are each controlled and organised by a Major Function Subsystem. These are:

(i) Tutorial Logic Control Subsystem (TLCS):

- controls all student-related activities (fully described in Chapter 9).

(ii) Author Control Subsystem (ACS):

- responsible for all author-related activities (fully described in Chapter 6).

(iii) Supervisor Control Subsystem (SCS):

- used by the SCHOOL System Supervisor to control, monitor and tune the performance of the system (fully described in Chapter 12).

5.4.2 SERVICE FUNCTION SUBSYSTEMS

The more mundane functions of SCHOOL are handled by Service Function Subsystems. These are obviously more numerous than Major Function Subsystems and tend to be smaller. Subsystems within this category are:-

(i) Data Input Subsystem (DIS):

- creation of courseware by SCHOOL authors. This has a major relationship to the Author Control Subsystem (ACS) and is fully described in Chapter 7.

(ii) Keyboard Response Evaluation Subsystem (KRES):

- responsible for monitoring all keyboard activity, accepting input and performing any required lexical analysis prior to passing the input on to the requesting subsystem. See Chapter 10.

(iii) Immediate Command Execution Subsystem (ICES):

- subordinate to KRES, this subsystem intercepts special user commands returned instead of the expected response. These result in some immediate

function being carried out prior to returning to await the response originally requested. See Chapter 11 for full details.

(iv) Display Format Subsystem (DFS):

- control of all terminal output. This involves the expansion of data extracted from the SCHOOL Database, conversion into optimum device format and display. Chapter 8 gives full details.

(v) Database Input/Output Subsystem (DIØS):

- as has been described in Chapter 3, SCHOOL courseware is stored in a complex hierarchial structure and as such, input/output operations are far from simple. DIØS is provided therefore to manage the SCHOOL Database and all associated I/O processing.

(vi) Error Management Subsystem (EMS):

- responsible for handling non-recoverable system errors. Details are passed into EMS from the appropriate point of system execution and all relevant debugging information is then generated for the benefits of the SCHOOL System Supervisor.

5.4.3 SUBSYSTEM STRUCTURE

Subsystems are organised as a hierarchy of program modules, the number and complexity of which obviously vary. This produces a typical structure as in Fig 5.3 overleaf.

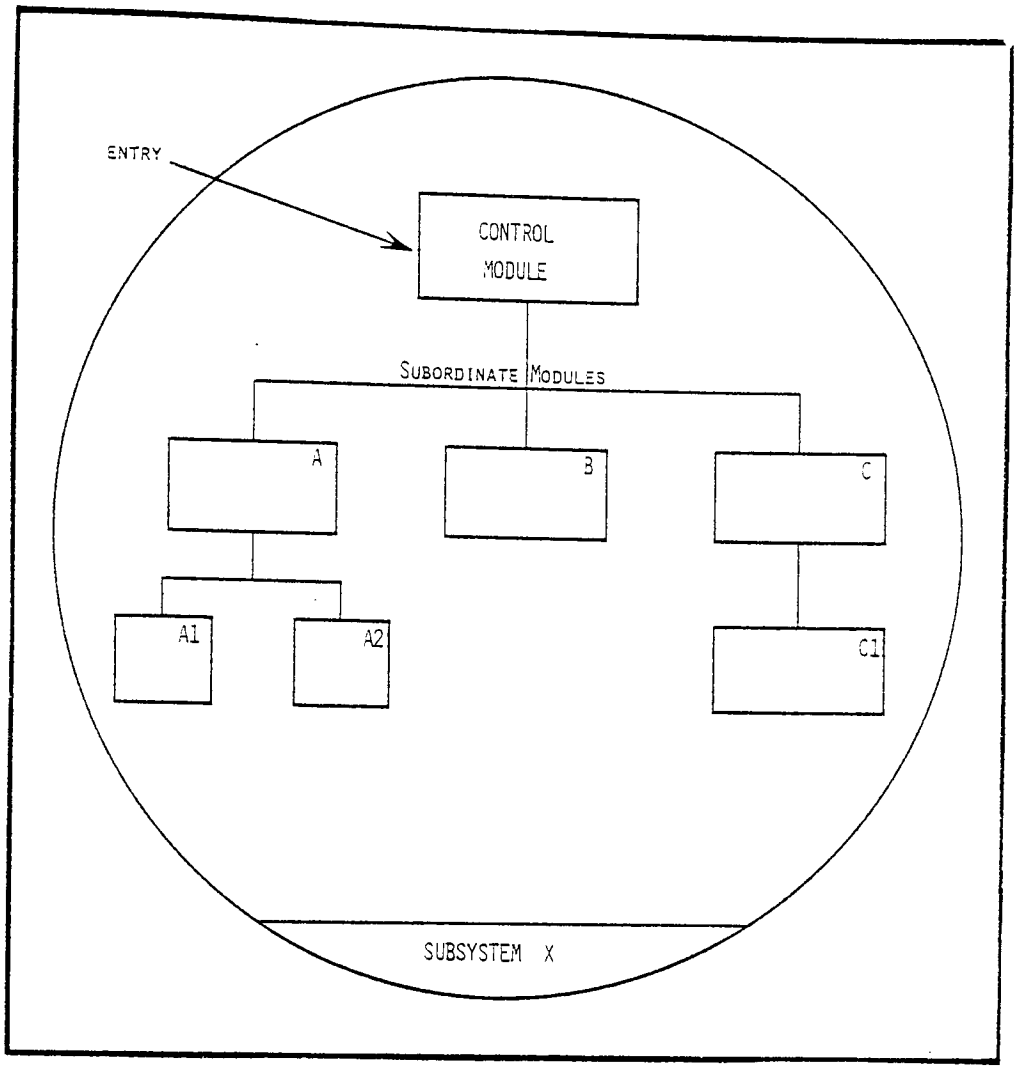


Fig 5.3 Subsystem structure

N.B. Whenever control is passed into a particular subsystem, it is always done via the subsystem control module - this is in keeping with the top-down structured design/programming philosophy described in subsection 5.7

5.4.4 SUBSYSTEM RELATIONSHIPS

Within any Major Function Subsystem, a complicated interaction can build up between its own modules and various Service Function Subsystems. It is not possible to generalise this, but as a fairly representative

example, consider the Course creation function provided as part of the Author Control Subsystem, illustrated by Fig. 5.4:

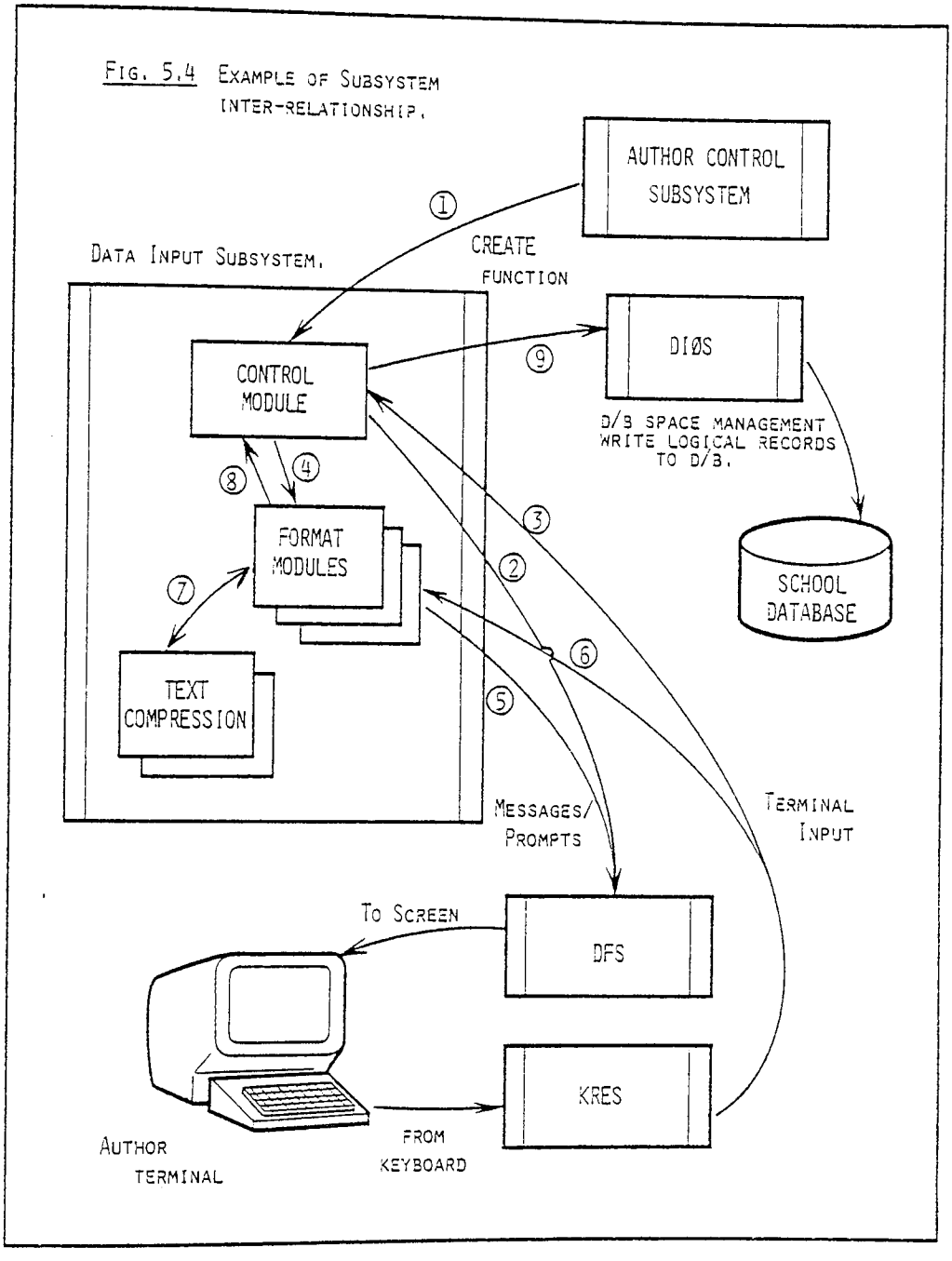


FIG. 5.4 EXAMPLE OF SUBSYSTEM INTER-RELATIONSHIP.

Notes on Fig. 5.4

The typical sequence of events is:

1 - a CREATE function request is intercepted by the Author Control Subsystem (ACS) and the Data Input Subsystem (DIS) is immediately invoked;

2,3 - DIS requests (via DFS, the Display Format Subsystem) and reads in (via KRES, the Keyboard Response Evaluation Subsystem) details of the course data to be created;

4,5,6 - once identified, the corresponding DIS Format Module is called. This then uses DFS/KRES to request/read tutorial text and control parameters;

7 - text input may be compressed prior to being written to the SCHOOL database. This is optional;

8 - once all required data has been received, the formatted logical record is passed back from the DIS Format Module to the Control Module;

9 - the Database Input/Output Subsystem (DIØS) is then invoked and the logical record is written to the SCHOOL database as a physical record.

5.5 MODULES

SCHOOL modules are written in one of two programming languages:-

(i) Assembler - standard IBM System 370 Assembly language (IBM, 28).

(ii) CØBØL - American National Standard Version 4 Commercial Business Oriented Language (IBM, 29).

Anything that is used infrequently and involves a lot of data manipulation or decision making is written in CØBØL, everything else (normally heavily used, needs to be efficient and control hardware functions not accessible from high-level) is written in Assembler.

Other languages were considered:

PL/1 - rejected because of limited experience on the Author's behalf (and within Compower); however, the resultant software, if compiled using the PL/1 Optimising Compiler (IBM, 30), would no doubt have been fairly efficient;

FORTRAN - rejected because of limitations in text handling and slightly worse machine performance than CØBØL;

APL - this would have been probably the best language for speed of development, but operational performance would have been poor (it is interpretive) software security would be restricted and there are problems in interfacing with Assembler modules;

The marriage of Assembler and CØBØL has proved very satisfactory, from both development and operational standpoints. Module interfacing is very straightforward via standard CALL statements and both use the normal IBM Register 1 linkage convention.

5.6 DATA NUCLEI

SCHOOL has as part of its structure special areas of storage (maximum 4096 bytes) termed Data Nuclei , which are used by sybsystems when communicating with one another.

Consider Fig. 5.5:

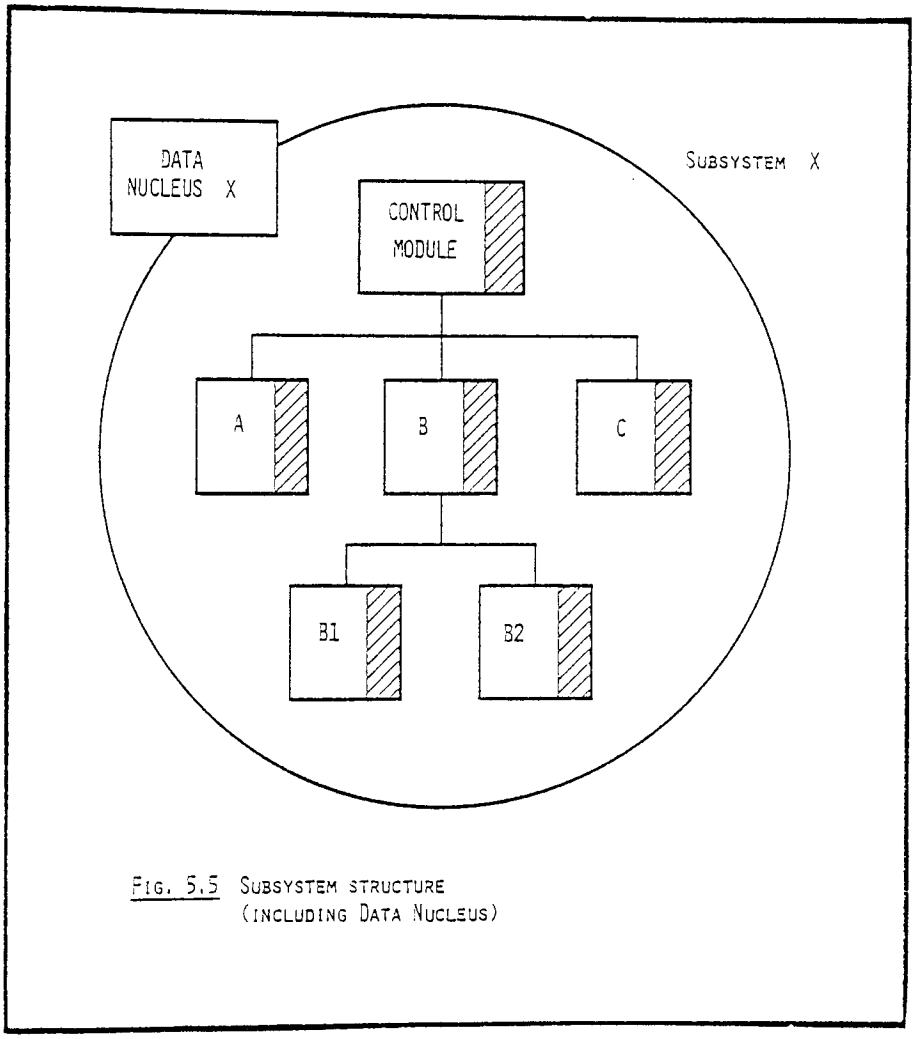


FIG. 5.5 SUBSYSTEM STRUCTURE
(INCLUDING DATA NUCLEUS)

The shaded areas represent 'local' data - i.e. used by, and pertinent to, only a single subsystem module. The Data Nucleus for Subsystem X is however 'visible' from all subsystem modules (which can therefore use and modify data within it as required). Furthermore, external subsystems which are used by,

or communicate with Subsystem X also have access to its Data Nucleus. This technique provides rapid and straightforward inter-subsystem communications and saves a great deal of parameter and address manipulation. The Data Nucleus associated with a subsystem need not however be physically contiguous with the subsystem software itself - this can be particularly beneficial when considering host operating system paging (i.e. only the Data Nucleus, not the subsystem coding need be resident in real storage whilst being used by other subsystems). This has also proved invaluable during software debugging (i.e. all relevant data is available in one large adjacent block).

N.B. For specific details of SCHOOL software structure, refer to Appendices:-

5.1 - list of subsystems/modules

5.2 - structure chart

5.3 - data nucleus/subsystem cross-reference

5.7 SCHOOL PROGRAMMING AND TESTING

Wherever possible, Structured design, programming and testing techniques have been applied during the development of SCHOOL. The range of techniques used includes:

(i) Top-down system and program design:

- hierarchical structure and step-wise refinement of system functions.

(ii) Top-down coding:

- coding modularised, extensive use of 'perform' and 'call' concepts to execute subordinate levels. This technique is most effective within the CØBØL segments of the software.

(iii) Top-down testing:

- major interfaces between programs/modules tested first, then sequence of major functions and so on down to the lowest level of coding.

(iv) Structured programming:

- only relevant to COBOL segments of the software and signifies that 'GØ TØ' statements are eliminated as far as possible, and coding is built up from sequential statements, 'do-while' constructs and binary 'if-than-else' decisions.

(v) Program annotation:

- comprehensive documentation built into coding, meaningful data names, sequences of coding kept short.

These techniques can be seen by reference to any part of the SCHOOL coding, and earlier parts of this Chapter.

5.8 SYSTEM HABITATS

As far as practical, SCHOOL has been constructed to be independent of the Operating System which hosts it. However, as is true of almost all software products, it is not completely portable and more than one version will need to be produced should it be intended to run under different host systems. The development version of SCHOOL has been evolved within IBM Conversational Monitoring System, CMS, which itself is part of VM/370 - Virtual Machine Facility/System 370 (IBM, 4). It is important also to address the other equivalent and planned environment, VSPC (Virtual Storage Personal Computing - IBM, 3).

5.8.1 Virtual Machine Facility System 370 (VM/370).

Within VM/370, the resources of a single real machine are subdivided to emulate the resources of multiple imaginary, or virtual machines, each of which supports a single conventional Operating system. Each virtual machine must therefore emulate all aspects of a normal computing environment, as Fig. 5.6 illustrates:

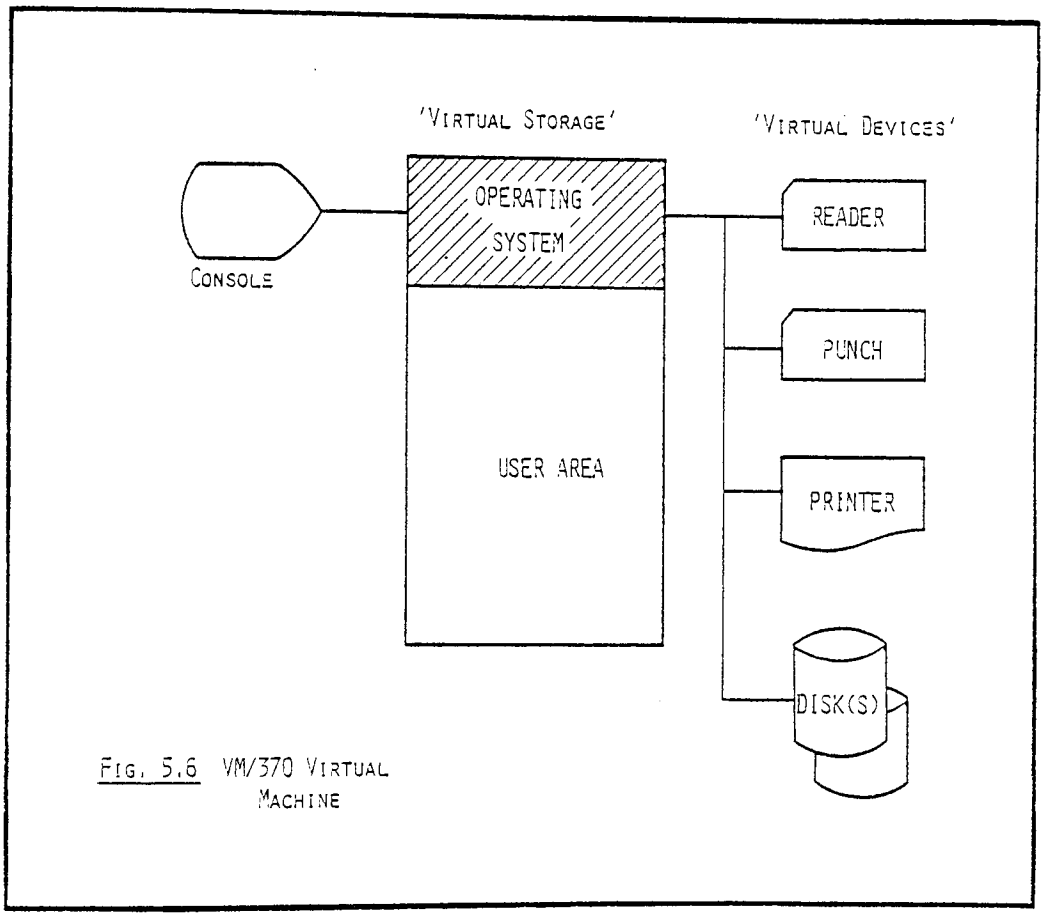


FIG. 5.6 VM/370 VIRTUAL MACHINE

The various components are emulated by VM/370 as follows:

(i) Operating Console - functions provided at the user's terminal;

(ii) Main Storage - paged virtual storage;

(iii) I/Ø Devices - reader/punch/printer (generically termed 'unit record' devices) all simulated by areas of disk storage, which can interface to real devices if required;

- user disk storage provided by small allocated sections of real disks.

Most normal IBM operating systems can run within a VM/370 virtual machine, but of particular relevance here is the provision of CMS. This is a single-user single-task Supervisor which will manage the resources of one virtual machine. The development version of SCHOOL runs under CMS, as shown in Fig. 5.7:

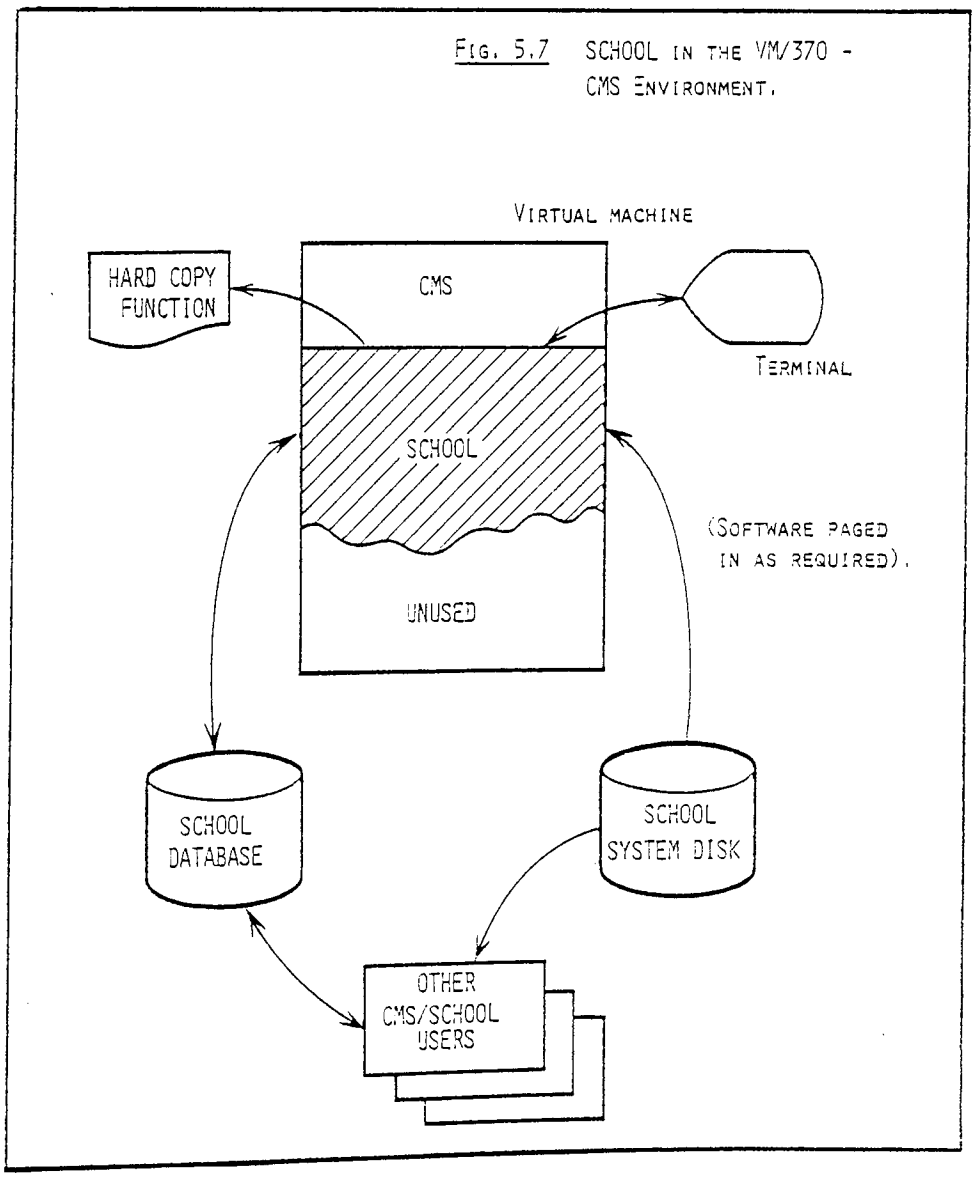


FIG. 5.7 SCHOOL IN THE VM/370 - CMS ENVIRONMENT.

Both the SCHOOL Database and software disks are shared by all users, but any active SCHOOL software is unique within each user's virtual machine - i.e. it is not necessary under CMS to have SCHOOL capable of multi-tasking in its own right, or to be formally re-entrant.

5.8.2 Virtual Storage Personal Computing (VSPC)

A recent addition to Compower's Teleprocessing facilities is that of VSPC (Virtual Storage Personal Computing). This is similar to CMS insofar as it offers remote program development access to large IBM mainframe equipment, but it is angled towards a generally less expert type of user - who typically will use BASIC, APL or FORTRAN. Each user is allocated an ACCOUNT under VSPC, which consist of 2 fundamental parts:

(i) User Library - an area of disk storage where user programs and data are retained;

(ii) Active Workspace - an allocation of machine (virtual) storage used to create, edit and execute programs.

Note that unlike the much more powerful CMS, no 'virtual devices' are provided under VSPC.

It is hoped to produce a VSPC version of SCHOOL, and to do this the approach will be as per Fig. 5.8 (IBM, 31). See overleaf.

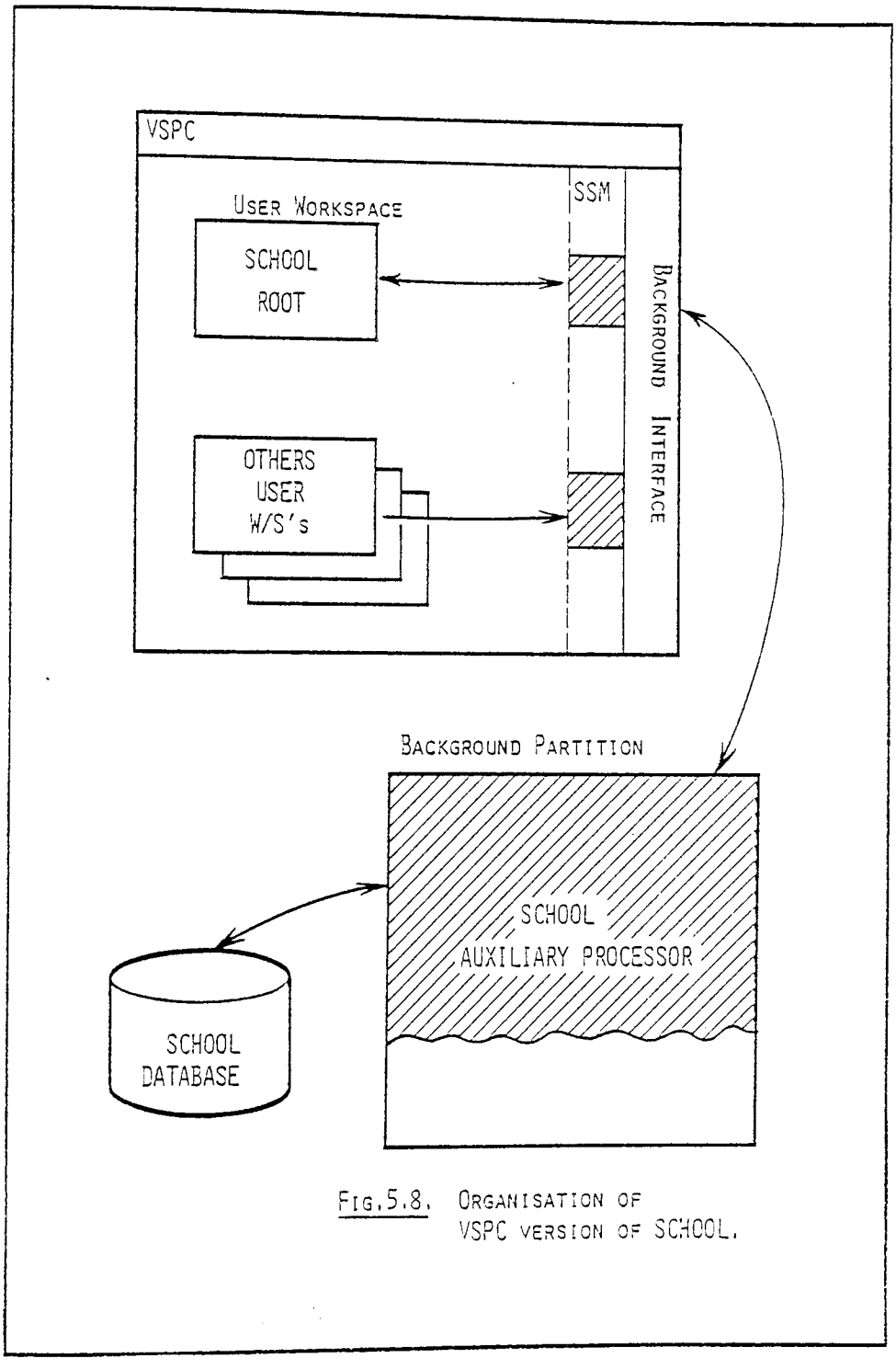


FIG.5.8. ORGANISATION OF VSPC VERSION OF SCHOOL.

Notes on Fig. 5.8

(i) Each SCHOOL user will have resident in his workspace a SCHOOL Root. This would probably be written in APL;

(ii) Each user's SCHOOL Root will contain 'local' processing logic (e.g. entire subsystems such as KRES and DFS, parts of others such as DIS and TLCS) and temporary data (e.g. status and performance within a tutorial session, working version of a frame as it is being authored);

(iii) Communication with the main body of SCHOOL (the SCHOOL AP) is via Shared Storage Management (SSM). This, and the VSPC Background Interface, enable Background Auxiliary Processors to communicate with VSPC and user programs running within it (BASIC, APL or FORTRAN). These programs are termed Foreground Processors, FP's);

(iv) Each SCHOOL Root calls the SCHOOL AP via SSM, passing blocks of data (termed Shared Variables - maximum size currently 4K) into the AP as required. A similar process applies in reverse;

(v) SCHOOL AP coding will need to be serially reusable (most current coding already is);

(vi) Macros used to handle SCHOOL Database I/Ø will require alteration to those compatible with the AP host environment;

chapter six

- AUTHOR CONTROL SUBSYSTEM

6.1 INTRODUCTION

A major emphasis in the design of SCHOOL is towards the courseware Author, as has been discussed in earlier Chapters. The Author is considered to have 3 major functional responsibilities:

(i) COURSEWARE CREATION

- new material, or extending existing data;

(ii) MONITORING

- who is using the course, and how well both students and course are performing;

(iii) MAINTENANCE

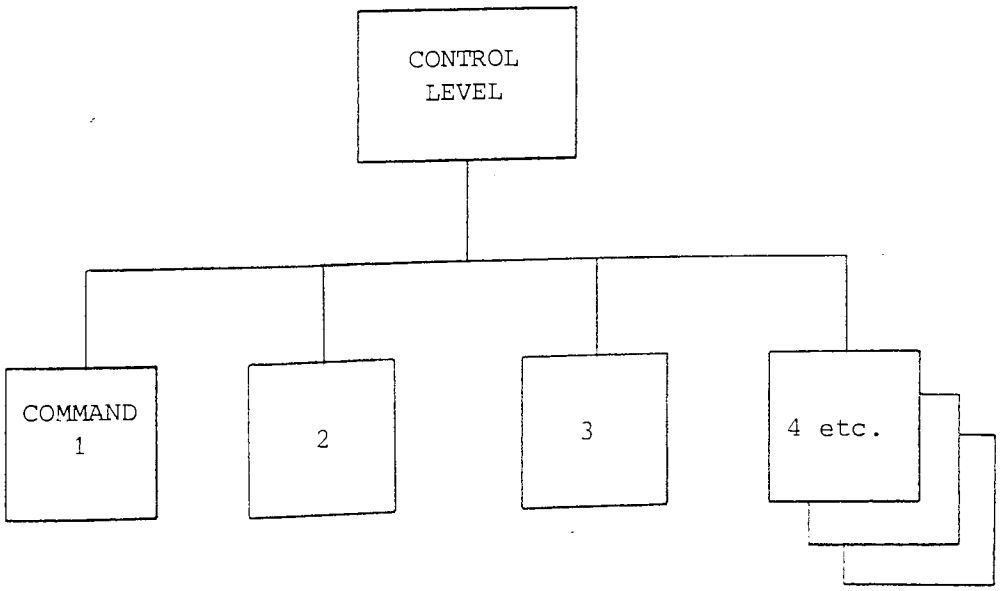
- altering course procedure and content as required, deleting redundant information and using other special facilities (e.g. user messages);

The above functions are provided within SCHOOL by the Author Control Subsystem (ACS), as the remainder of this Chapter will describe.

6.2 AUTHOR CONTROL SUBSYSTEM - ORGANISATION

ACS is organised in two levels, as Fig. 6.1 illustrates:

Fig. 6.1 ACS Organisation



Entry is always at the control level, being achieved by invoking SCHOOL, specifying AUTHOR as the Operation mode and supplying the correct (and current) 'Author mode' password (4 attempts allowed).

Once ACS Control level is entered successfully, the Author is prompted to enter an 'Author Command', plus the name and Subject Reference Number of the SCHOOL subject being manipulated. This latter 2-digit value is assigned by the SCHOOL System manager and is unique throughout the System. Furthermore it is not capable of being interrogated by Authors.

6.3 ACS COMMAND GROUPS

Author commands, which are usually in near normal English form, are subdivided into 6 groups:-

6.3.1 Courseware Creation Commands

- only one ACS command is in this group:

  CREATE

- the effect is to switch the Author into the Data Input System, DIS. This has its own organisation/subcommands etc., and is fully described in Chapter 7.

6.3.2 Course Monitoring Commands

- designed to interrogate current course structure and status, along with the built-in Course Analysis aids (Lesson Analysis, Glossary requests). Commands within this group are:

  ANALYSIS

  CONTENTS

  DESCRIBE

  GLOSSARY

  RESETLAR

  ROUTE

6.3.3 Student Control Commands

- these provide full interrogation and control facilities over the students registered for a particular subject. Commands are:

HISTORY

REMOVE

STUDENTS

6.3.4 Courseware Maintenance Commands

- provide the ability to alter existing course data (either Procedure or Content), or delete unwanted material. Two commands exist:

ALTER

DELETE

6.3.5 Subject Message Maintenance Commands

- a unique facility within ACS is the creation and maintenance of Subject Messages. These are displayed to each subject user on invocation, and the following commands are used by the Author to control this feature:

MSGALTER

MSGBUILD

MSGSCRUB

MSGSHOW

6.3.6 Miscellaneous commands:

- commands not otherwise classified. Only one command currently exists:-

END

6.4 ACS COMMANDS

This Section defines the functions of all Author Commands, listed alphabetically.

Fig. 6.2 A.C.S. Commands.

COMMAND	FUNCTION	OTHER DATA?
ALTER	Provides a courseware edit facility, the exact approach to which varies depending on Author requirements.	See Note (ii)
ANALYSIS	Returns lesson analysis data detailing the performance of the defined lesson (e.g. times used, completion characteristics, length of time needed)	S, L See Note (i)
CONTENTS	Produces a full definition of the current status and structure of the chosen part(s) of the named subject. This data may be displayed at the terminal or printer.	S,(L)
CREATE	Enters the Data Input Subsystem.	See Chapter 7.
DELETE	Used to remove all or part of the specified lesson. The exact scope of deletion is defined via a menu display.	S,L
DESCRIBE	Returns an overview of the subject/ lesson structure. This is effectively a precis of the CONTENTS command.	S,(L)
END	Leaves AUTHOR mode.	-
GLOSSARY	Gives the current glossary status for the defined lesson (both supported and unsupported/requested entries). This data may be displayed or printed or both.	S,L
HISTORY	Provides an analysis of the current status of the named student(s) (e.g. progress so far, performance details.) This data may be displayed or printed or both - normally printed if a complete course's students are being analysed.	S

COMMAND	FUNCTION	OTHER DATA?
MSGALTER	Enables the examination and alteration of an existing Subject Message (both text and expiry date information).	S
MSGBUILD	Provides an interactive facility for the author to create a new Subject Message (data and text information).	S
MSGSCRUB	Used to delete existing Subject Messages. This can be done either explicitly by specifying the Message number or by expiry date.	S
MSGSHOW	Returns a display giving the current contents of the identified Subject Message. Both text and expiry date information is displayed.	S
REMOVE	Deletes the record(s) pertinent to one or more registered students. This may be by student name/number or by 'date-last-active'.	S
RESETLAR	Resets all fields in the defined Lesson Analysis record to their initial value (mostly zeros).	S,L
STUDENTS	Returns a list of currently registered students within the defined subject. This may be either displayed at the terminal or printed or both.	S

Notes on Fig. 6.2:

(i) S - Subject Reference Number, and/or Subject Name;

L - Lesson number (brackets denotes optional, typically if a complete subject is being manipulated);

(ii) Other data requirements for ALTER command can vary considerably according to the intended scope of alteration. These are selected by menu. See Section 6.8.1.

All of the above commands are described in detail in the SCHOOL Author Manual, but the following sections outline some of the more important facilities:

6.5 COURSE CREATION PROCEDURES

This is handled entirely by DIS, the Data Input Subsystem, and is invoked by the CREATE command. A comprehensive range of courseware creation subcommands is then available to the Author - See Chapter 7.

6.6 COURSE MONITORING FACILITIES

There are two aspects available to the Author:

(a) peruse current courseware content and procedure definitions;

(b) check status of monitoring aids (lesson analysis and glossary).

6.6.1 Current Course Definition

- courses can get very complex and it can be most useful to scan the current status of a course structure. Some of the available commands and the information returned follow:

(i) DESCRIBE - produces an overview of subject/lesson structure (as requested), e.g.

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                SCREEN  23

 0
 1
 2        SUBJECT:  LOTUS CARS
 3
 4                  MANAGER:        MIKE LITTLE
 5                  REF. NO.:       17
 6                  NO. LESSONS:    3
 7
 8                  MESSAGES:       00 ACTIVE      02 EXPIRED
 9
10                                                 /PRESS ENTER TO CONTINUE
11
12
13
14
15
16
17
18
19
20
21
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  24

 0  *
 1  *          SUBJECT:  LOTUS CARS          LESSON DETAILS:
 2  *     NO.       TITLE                    AUTHOR
 3  *
 4  *      1        HISTORY                  COLIN CHAPMAN
 5  *
 6  *               NO. FRAMES:    12        ANALYSIS:   Y
 7  *                                        SYNTAX:     N
 8  *                                        GLOSSARY:   Y
 9  *
10  *      2        ELANS & EUROPAS          MIKE LITTLE
11  *
12  *               NO. FRAMES:    18        ANALYSIS:   Y
13  *                                        SYNTAX:     N
14  *                                        GLOSSARY:   Y
15  *
16  *      3        CURRENT RANGE            MIKE KIMBERLEY
17  *
18  *               NO. FRAMES:     9        ANALYSIS:   Y
19  *                                        SYNTAX:     N
20  *                                        GLOSSARY:   Y
21  *                                       /PRESS ENTER TO CONTINUE...
```

(ii) CONTENTS - produces a display giving details of the chosen part of a course. For example:

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  29

 0  *
 1  *          SUBJECT:        LOTUS CARS
 2  *
 3  *          LESSON NO.2          TITLE:    ELANS & EUROPAS
 4  *                               AUTHOR:   MIKE LITTLE
 5  *
 6  *          ANALYSIS:  Y         SYNTAX:  N        GLOSSARY:  Y
 7  *
 8  *          PASSMARKS:  50(EASY)  80(HARD)        MODE:    EASY
 9  *
10  *          HARDCOPY:   OFF
11  *
12  *          INTRODUCTORY TEXT:
13  *
14  *          1962 SAW THE INRODUCTION OF WHAT WAS TO BECOME ONE OF THE
15  *          MOST RESPECTED SPORTS CARS OF ALL TIME  -  THE LOTUS ELAN.
16  *
17  *              THIS SEGMENT OF THE COURSE DISCUSSES THE EVOLUTION OF
18  *          THIS MODEL, ITS '+2' DERIVATIVES, AND THE MID-ENGINED EUROPA.
19  *
20  *                                /PRESS ENTER FOR FRAME BREAKDOWN
21  *
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  30

 0  *
 1  *          SUBJECT:      LOTUS CARS
 2  *          LESSON NO.    2                    NO. FRAMES:   18
 3  *
 4  *          FRAME DETAILS:-
 5  *
 6  *           NO.     .    Q     A     H     C
 7  *
 8  *             1          X     -     -     -
 9  *             2          X     -     -     -
10  *             3          X     -     -     -
11  *             4          X     -     -     -
12  *             5          X     X     -     X
13  *             6          X     X     X     X
14  *             7          X     -     -     -
15  *             8          X     X     -     X
16  *             9          X     -     -     -
17  *            10          X     -     -     -
18  *            11          X     X     X     X
19  *            12          X     -     -     -
20  *            13          X     -     -     -
21  *                               /PRESS ENTER FOR FURTHER FRAMES ...
```

(ii) ROUTE - generates a table of possible frame sequences within the named lesson, e.g.

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  33

 0  *       SUBJECT:    SCHOOL TEST                                          *
 1  *                                                  LESSON NO.   01      *
 2  *        FRAME      SEQUENCE:      NORMAL     REMEDIAL                   *
 3  *                                                                        *
 4  *           1                                                            *
 5  *           2                          2          2                      *
 6  *           3                          3          3                      *
 7  *           4                          7          4                      *
 8  *           5                          5          5                      *
 9  *           6                          7          6                      *
10  *           7                          7          7                      *
11  *           8                          8          8                      *
12  *           9                          9          9                      *
13  *          10                         15         10                      *
14  *          11                         11         11                      *
15  *          12                         14         12                      *
16  *          13                         14         13                      *
17  *          14                         14         14                      *
18  *          15                         15         15                      *
19  *                                    END                                 *
20  *                                          /ENTER TO CONTINUE            *
21  *                                                                        *
    *                                                                        *
    **************************************************************************
```

6.6.2 Monitoring aid status checks

- designated to provide the Author with an insight into how well his work is performing, typically in terms of Lesson Analysis and the use of Lesson Glossaries. Two commands are available:

(i) ANALYSIS - returns lesson analysis information at 2 distinct levels. Firstly, overall status:

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  35

 0  *       SUBJECT:    SCHOOL TEST                                          *
 1  *       LESSON NO.  01                  NO. FRAMES:    15               *
 2  *                                                                        *
 3  *       OVERALL:    NO. TIMES USED:        23                            *
 4  *                   NO. TIMES COMPLETED:   15                            *
 5  *                   NO. TIMES SUSPENDED:    4                            *
 6  *                   NO. TIMES ABANDONED:    4                            *
 7  *                                                                        *
 8  *                   TOTAL DURATION:      2:34   (H:M)                    *
 9  *                                                                        *
10  *                                                                        *
11  *                                                                        *
12  *                          /CONTINUED......PRESS ENTER....               *
13  *                                                                        *
14  *                                                                        *
15  *                                                                        *
16  *                                                                        *
17  *                                                                        *
18  *                                                                        *
19  *                                                                        *
20  *                                                                        *
21  *                                                                        *
    *                                                                        *
    **************************************************************************
```

- and secondly frame analysis information:

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN  36

        SUBJECT:     SCHOOL TEST
        LESSON NO.   01

                                    ANALYSIS REPORT

  FRAME   USED   ATTEMPTS   INCORRECT   HINT REQ   ANS REQ    MARKS    PERCENT
    1      23       31          8          12          4     126/230     54.8
    2      23       28          1           1          0     218/230     94.8
    3      23       47         10          25          7      80/230     34.8
    4      10       10          3           6          1      58/100     58.0
    5      10       16          5          11          4      28/100     28.0
    6       5        6          1           2          0      36/ 50     72.0
    7      23       25          1           2          0     216/230     93.9
    8      23       27          5          12          2     156/230     67.8
    9      23       31         11           8          2     104/230     45.2
   10      11       12          0           1          0     108/110     98.2
   11      11       20          6           7          3      36/110     32.7
   12       6       10          3           4          1      22/ 60     36.7
   13       3        3          0           0          0      30/ 30    100.0
   14      11       15          2           4          1      82/110     74.5
   15      23       23          0           0          0     230/230    100.0

                                    /CONTINUED.....
```

(ii) GLOSSARY - produces a report on the current Glossary contents for the defined lesson, e.g.

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN  42

        SUBJECT:    LOTUS CARS              LESSON NO.   2

        GLOSSARY ANALYSIS:

            ENTRY    WORD/PHRASE                SUPPORTED   USED/REQ

              1      WISHBONE                       Y           3
              2      BACKBONE CHASSIS               Y           0
              3      T.O.H.C.                       Y          11
              4      TOHC                           N           4
              5      GRP                            N           6
              6      K.O.                           Y           6
              7      TWINCAM                        N           1
              8      CHAPMAN STRUT                  Y           2

        TOTALS:       8 ENTRIES      5 SUPPORTED

                                    /PRESS ENTER TO CONTINUE...
```

(Note that this identifies both words/phrases that have an explanation on the Database (i.e. 'supported') and those that have been requested but have no entry as yet.)

6.7 STUDENT CONTROL FACILITIES

These facilities divide conveniently into two types:-

(a) interrogating the SCHOOL Database for student status information;

(b) deleting student registrations from the Database. This is achieved very simply via the REMOVE command - see Appendix 6.1.

Two commands make up the first category:

(i) STUDENTS - lists all students currently registered to a particular subject. For example:

```
     SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  11

     ********************************************************************************
     *                                                                              *
   0 *          SUBJECT:   SAMPLE 82                                                *
   1 *                                                                              *
   2 *          REGISTERED STUDENTS:                (TOTAL:   23)                   *
   3 *                                                                              *
   4 *       ID.      NAME                  LOCATION                                *
   5 *                                                                              *
   6 *      3101      MIKE LITTLE           CANNOCK                                 *
   7 *      3102      J.SMITH               O.R.E.                                  *
   8 *      3103      F.BLOGGS              O.R.E.                                  *
   9 *      3104      L.WILKINS             EDINBURGH                               *
  10 *      3105      F J SMITH             GATESHEAD                               *
  11 *      3106      BILL HARRIS           DONCASTER                               *
  12 *      3107      L.J.K.SETRIGHT        PUBLICITY                               *
  13 *      3108      A.HOBBS               HQ IND.RELS.                            *
  14 *      3109      P WILSON              MINING                                  *
  15 *      3110      A.ROBINSON            T.SCHOOL                                *
  16 *      3111      A.GREENWELL           ACCOUNTS                                *
  17 *      3112      ROBERT REID           PROD.DEV.                               *
  18 *      3113      E.WILLE               STAFF DEPT.                             *
  19 *      3114      M.DOUBLEDAY           ASTON UNIV.                             *
  20 *      3115      J.B.DUNNING           HQ TRAINING                             *
  21 *                                       /PRESS ENTER FOR MORE DATA             *
     *                                                                              *
     ********************************************************************************
```

(ii) HISTORY - returns a detailed Student Performance Report for the defined student(s). These may be defined individually by name or via an 'ALL' option which will give details of all students registered to the defined subject. A sample follows:

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  14

  0
  1        SUBJECT:    SAMPLE 82              STUDENT PERFORMANCE(1)
  2     ID.  3105      NAME:  F J SMITH       LOCATION:  GATESHEAD
  4
  5                    MESSAGE STATUS:  02
  6     DATES:
  7                    START:  82/033         LATEST:  82/069
  8
  9                    TOTAL DURATION:  2:14    (HH:MM)
 10     CURRENT STATUS:    LESSON:  5    FRAME:  12    COMP. CODE:  N
 11
 12                                    /ENTER TO CONTINUE ....
 13
 14
 15
 16
 17
 18
 19
 20
 21
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  15

  0
  1        SUBJECT:    SAMPLE 82              STUDENT PERFORMANCE(2)
  2                                           ID.  3105
  3
  4           LESSON       MARK       MODE      DURATION(HH:MM)
  5
  6              1           23          H              0:24
  7              2           94          E              0:31
  8              3           87          E              0:21
  9              4           71          E              0:25
 10              5           57          E              0:33
 11
 12     ****  STUDENT DETAILS COMPLETE  -  PRESS ENTER TO PROCEED  ****
 13
 14
 15
 16
 17
 18
 19
 20
 21
```

6.8 COURSEWARE MAINTENANCE

Three specific considerations are involved here:

(a) physical alteration of the appropriate course information. This may be either course content (i.e. tutorial text) or procedure (i.e. parameters controlling the processing of the course content);

(b) rewriting the altered data back to the SCHOOL database;

(c) deleting information entirely from a Database Subject structure.

6.8.1 Courseware Alteration - the ALTER command

Only certain parts of a SCHOOL subject are capable of direct alteration, as defined in Fig. 6.3:

Fig. 6.3

LEVEL	FIELDS ALTERABLE	MNEMONIC
SUBJECT DEFINITION	Subject Manager details - name - location Introductory text	SDEF
LESSON DEFINITION	Lesson title Author name Introductory text Passmarks - easy - hard Current scoring mode Hardcopy setting	LDEF
GLOSSARY DEFINITION	Existing word/phrase entry Explanatory text	GDEF
SYNTAX INFORM-ATION	Space removal - leading - trailing - non-delimiting - all Removal of - punctuation - leading zeros - redundant brackets List of redundant characters Substitutions - to upper case - to lower case - characters	SINF

LEVEL	FIELDS ALTERABLE	MNEMONIC
QUESTION DEFINITION	Frame text Hints - thresholds - penalities Screen erase settings Tutorial sequence - normal - remedial Maximum scores - easy - hard	QDEF
HINT INFORMATION	Hint text - A - B - C	HINF
ANSWER INFORMATION	Correct answers - 01 - 02 (text and comment - 03 cross-reference) - 04 - 05 - 06 - 07 - 08 Incorrect answers - 01 - 02 (text and cross - 03 reference) - 04 Max. no. attempts Standard reply requirement Answer display requirement Punctuation significance Space significance	ADEF
COMMENT DEFINITION	Comment text - A - B - C - D - E - F - G	CDEF

Should an Author wish to alter part of his courseware, the sequence of events that he will use is as follows:-

(i) enter ALTER command;

(ii) enter Subject name and reference number as requested (if these do not match current D/B entries, the ALTER request is refused);

(iii) specify the alteration level Mnemonic (as defined in Fig. 6.3). This in turn may result in prerequisite information being prompted, as shown in Fig. 6.4:

Fig. 6.4

MNEMONIC	PREREQUISITE INFORMATION
SDEF	—
LDEF	Lesson number (+ password if the Author has password - protected this particular lesson).
GDEF	as for LDEF
SINF	as for LDEF
QDEF	1. as for LDEF 2. question number
HINF	as for QDEF
ADEF	as for QDEF
CDEF	as for QDEF

(iv) once all prerequisites are specified and accepted, the Author is presented with a 'menu' of items which can be altered at the defined level, for example:

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                SCREEN  23

 0  *      ALTERATION LEVEL:   QDEF          LESSON:  6    QUESTION:  8
 1  *
 2  *                ALTERABLE DATA:
 3  *
 4  *                  1.  FRAME TEXT
 5  *                  2.  HINT THRESHOLDS
 6  *                  3.  HINT PENALTIES
 7  *                  4.  SCREEN ERASE
 8  *                  5.  TUTORIAL SEQUENCE  -  NORMAL
 9  *                  6.  TUTORIAL SEQUENCE  -  REMEDIAL
10  *                  7.  MAX. SCORE  -  EASY
11  *                  8.  MAX. SCORE  -  HARD
12  *
13  *
14  *   ==>  SELECT OPTION(S):
15  *   2
16  *
17  *
18  *
19  *
20  *
21  *
```

(v) the items to be changed are then specified in the format

$$x_1[,x_2[,x_3.....]]$$

e.g. 2,5,6 would flag:

- Hint Thresholds
- Tutorial sequences (normal and remedial)

in the above menu.

(vi) the next stage depends on the type of data being altered:

(a) Non-text fields:

- a revised version of the previous display is presented, for example:

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                SCREEN  24

 0  *      ALTERATION LEVEL:   QDEF          LESSON:  6    QUESTION:  8
 1  *
 2  *                SELECTED DATA:
 3  *
 4  *                    1.  FRAME TEXT
 5  *                ==> 2.  HINT THRESHOLDS
 6  *                    3.  HINT PENALTIES
 7  *                    4.  SCREEN ERASE
 8  *                    5.  TUTORIAL SEQUENCE  -  NORMAL
 9  *                    6.  TUTORIAL SEQUENCE  -  REMEDIAL
10  *                    7.  MAX. SCORE  -  EASY
11  *                    8.  MAX. SCORE  -  HARD
12  *
13  *
14  *   ***  CURRENT DATA VALUE(S)    1  2  4
15  *
16  *   ==>  ENTER NEW VALUE(S):
17  *   1,2,3
18  *
19  *
20  *
21  *
```

- each revision is entered in turn, with the above display being updated each time.

N.B. If (as on the above display) a text field can be selected, this is requested last, in the format described below:

(b) Text fields

- these are displayed on a revised screen with line numbers added, e.g.

```
    SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  25

    ********************************************************************************
    *                                                                              *
  0 *     ALTERATION LEVEL:   QDEF          LESSON:  6    QUESTION:  8             *
  1 *                                                                              *
  2 *    LINE    TEXT                                                              *
  3 *                                                                              *
  4 *       1          SCHOOL USERS CAN BE GROUPED INTO 3 CLASSES:                 *
  5 *       2                                                                      *
  6 *       3               1)  PUPILS                                             *
  7 *       4               2)  COURSE AUTHORS                                     *
  8 *       5             AND 3)  THE SYSTEM SUPERVISOR                            *
  9 *       6                                                                      *
 10 *       7   THE FUNCTION OF THE SYSTEM ADMINSTRATOR IS TO MONITOR              *
 11 *       8   AND CONTROL THE ENTIRE SYSTEM TO ENSURE THAT EVERYTHING            *
 12 *       9   WORKS SATISFACTORILY. AS THIS IS A VERY TECHNICAL                  *
 13 *      10   PROCEDURE, IT WILL NOT BE DISCUSSED FURTHER WITHIN THIS           *
 14 *      11   COURSE.                                                           *
 15 *      12                                                                     *
 16 *      13         TO PROCEED TO A DISCUSSION ON THE ROLE OF                   *
 17 *      14   SCHOOL STUDENTS ..... PRESS ENTER .....                          *
 18 *      15                                                                     *
 19 *      16                                                                     *
 20 *                                                                             *
 21 *  ==>  ENTER LINE NOS. TO BE ALTERED:                                        *
    *                                                                             *
    ********************************************************************************
```

- a sequence of one or more line numbers, or the keyword 'ALL', is then input (e.g. 1,3,7);

- if using an IBM 3270-type VDU, each selected line is placed in the terminal input area, and the built-in 3270 features (cursor movement, insert, character delete, delete to end-of-line) may be used to edit the text. Pressing ENTER completes alteration to this line, and the screen is redisplayed with the new data;

- if using a Teletype terminal each selected line is prompted and must be retyped in full;

- the operation above is repeated (either 3270 or Teletype) for all selected line(s);

- once complete, the revised text is displayed at the terminal in full, e.g.

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN  30

0  *     ALTERATION LEVEL:   QDEF          LESSON:  6    QUESTION:  8
1  *
2  *    LINE    TEXT
3  *
4  *  ==> 1          SCHOOL USERS CAN BE GROUPED INTO 3 CATEGORIES:
5  *      2
6  *  ==> 3               1)  STUDENTS
7  *      4               2)  COURSE AUTHORS
8  *      5             AND 3)  THE SYSTEM SUPERVISOR
9  *      6
10 *  ==> 7    THE FUNCTION OF THE SYSTEM SUPERVISOR IS TO MONITOR
11 *      8    AND CONTROL THE ENTIRE SYSTEM TO ENSURE THAT EVERYTHING
12 *      9    WORKS SATISFACTORILY. AS THIS IS A VERY TECHNICAL
13 *     10    PROCEDURE, IT WILL NOT BE DISCUSSED FURTHER WITHIN THIS
14 *     11    COURSE.
15 *     12
16 *     13         TO PROCEED TO A DISCUSSION ON THE ROLE OF
17 *     14    SCHOOL STUDENTS ..... PRESS ENTER .....
18 *     15
19 *     16
20 *                                          (entering NONE terminates
21 *  ==>  ENTER LINE NOS. TO BE ALTERED:      text alteration)
```

(note that altered lines are highlighted)

- the author is then given the option of applying further alterations to this block of text or proceeding. If he opts for further alterations, control return to the 'Enter Line Numbers' stage, otherwise the text alterations are retained.

Notes: (i) for major alterations to courseware, it is recommended that Authors use DELETE and CREATE;

(ii) a more powerful and easier-to-use full-screen text editor is under development - see 'Conclusions' chapter.

6.8.2 Updating Database logical records

Once the ALTER command sequence has been completed, ACS will update the appropriate part(s) of the SCHOOL database. This may be done in one of two ways, depending on which alterations the Author has effected.

(i) Non-text fields:

- D/B logical records will be the same size and so existing physical records (DBTU's) are rewritten in-situ;

(ii) Text fields:

- text data is now almost guaranteed to be a different length from the original, and so the procedure followed is:

  (a) delete current D/B phsyical records(s);

  (b) create a new logical record (including text compression etc.) and have this written to the database;

6.8.3 Courseware Deletion - the DELETE command

This is used to remove those parts of a subject which are either redundant or are to undergo major surgery. Once the DELETE command is entered, an options menu is displayed:

```
    SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                          SCREEN  37

    ********************************************************************************
    *                                                                              *
 0  *                                                                              *
 1  *       SUBJECT:   THE EYE                                                     *
 2  *                                                                              *
 3  *          DELETE OPTIONS:                                                     *
 4  *                                                                              *
 5  *               1.  LESSON                                                     *
 6  *               2.  LESSON ANALYSIS                                            *
 7  *               3.  GLOSSARY  -  ALL                                           *
 8  *               4.  GLOSSARY  -  SPECIFIC                                      *
 9  *               5.  SYNTAX INFORMATION                                         *
10  *               6.  FRAME  -  ALL                                              *
11  *               7.  HINT                                                       *
12  *               8.  ANSWER                                                     *
13  *               9.  COMMENT                                                    *
14  *                                                                              *
15  *                                                                              *
16  *  ==>  ENTER SELECTION:                                                       *
17  *  7                                                                           *
18  *                                                                              *
19  *                                                                              *
20  *                                                                              *
21  *                                                                              *
    *                                                                              *
    ********************************************************************************
```

and the Author may specify his requirements (only one at a time). Note that there are some restrictions imposed by the above list:

(i) QUESTION (signifying the Question part of a frame) is not an option as deleting this will render void dependent parts of the frame;

(ii) ALL (signifying an entire Subject) is not available - only the System Supervisor can delete a complete subject (see Chapter 12);

Depending on the DELETE option selected, further information will be requested as illustrated by Fig. 6.5:

Fig. 6.5

DELETE OPTION	ASSOCIATED INFORMATION REQUIREMENTS
LESSON	Lesson number
LESSON ANALYSIS	Lesson number

DELETE OPTION	ASSOCIATED INFORMATION REQUIREMENTS
GLOSSARY - ALL	Lesson number
GLOSSARY - SPECIFIC	Lesson number Glossary entry (word/phrase)
SYNTAX INFORMATION	Lesson number
FRAME -ALL	Lesson number Question (frame) number
HINT	Lesson number Question number Hint letter
ANSWER	Lesson number Question number Answer type (correct/incorrect) Answer number
COMMENT	Lesson number Question number Comment letter

N.B. The 'associated information' is always requested in full - no defualts are assumed.

Consider the following example:

```
    SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                SCREEN  38

    ********************************************************************************
    *                                                                              *
 0  *                                                                              *
 1  *       SUBJECT:   THE EYE                                                     *
 2  *                                                                              *
 3  *       DELETE:    ****  HINT  ****                                            *
 4  *                                                                              *
 5  *  ENTER LESSON NO.                                                            *
 6  *  9                                                                           *
 7  *  ENTER FRAME NO.                                                             *
 8  *  X                                                                           *
 9  *  ILLEGAL FRAME NO.  -  RE-ENTER:                                             *
10  *  18                                                                          *
11  *  ENTER HINT LETTER:                                                          *
12  *  B                                                                           *
13  *                 ****  PROCESS COMPLETE:  9/18/HINT B REMOVED                 *
14  *                                                                              *
15  *                                    /PRESS ENTER TO CONTINUE                  *
16  *                                                                              *
17  *                                                                              *
18  *                                                                              *
19  *                                                                              *
20  *                                                                              *
21  *                                                                              *
    *                                                                              *
    ********************************************************************************
```

- the effect of this sequence is that Hint B within the defined Subject/Lesson/Question structure, and all references to it, are deleted.

6.9 SUBJECT MESSAGE MAINTENANCE

Each SCHOOL subject can have appended to it a maximum of 10 date-controlled Subject Messages, and the Author has a series of commands available to manipulate these:

MSGBUILD - interactive message creation (similar to some aspects of DIS - see Chapter 7);

MSGALTER - interactive message alteration (similar to ALTER command with text data);

MSGSCRUB - delete named or time - expired messages (similar to REMOVE commands);

MSGSHOW - display current contents of defined message, e.g.

```
    SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN  24

    ********************************************************************************
    *                                                                              *
  0 *  ENTER AUTHOR COMMAND:                                                       *
  1 *  MSGSHOW                                                                     *
  2 *                                                                              *
  3 *  ENTER MESSAGE NO.                                                           *
  4 *  1                                                                           *
  5 *                                                                              *
  6 *  **********  SUBJECT:  TESTRUN          MESSAGE:  01  **********            *
  7 *                                                                              *
  8 *          PLEASE NOTE THE FOLLOWING:                                          *
  9 *                                                                              *
 10 *                SUBJECT TESTRUN WILL BE SUSPENDED FOR MAINTENANCE             *
 11 *          WORK ON TUESDAY 16TH MARCH 1982 FROM 2.30 PM FOR                    *
 12 *          APPROXIMATELY 3 HOURS.                                              *
 13 *                                                                              *
 14 *                APOLOGIES FOR ANY INCONVENIENCE CAUSED.                       *
 15 *                                                  MJL  11/3/82                *
 16 *                                                       **********             *
 17 *  **********  MESSAGE EXPIRES:  82/076                                        *
 18 *                                                                              *
 19 *                                                                              *
 20 *                                                                              *
 21 *                                                                              *
    ********************************************************************************
```

6.10 AUTHOR CONTROL SUBSYSTEM - SOFTWARE

The control level of ACS (see Fig. 6.1) is coded as module AUTHOR. This handles the initial operations of ACS, i.e.

- command entry and recognition;
- entry and validation of subject name and reference number;
- branching to the appropriate ACS command coding.

AUTHOR also contains coding to provide some of the more trivial SCS functions (DESCRIBE, END, MSGSHOW, RESETLAR) as these involve minimal logical record processing.

All other ACS functions are supported by separate program modules, named ACS..... For a detailed description of these, see Appendix 6.2.

chapter seven

- DATA INPUT SUBSYSTEM

7.1 INTRODUCTION

The Data Input Subsystem (DIS) is the major method of creating course material ('courseware') and inserting it on the SCHOOL database.

DIS may be used to:

(i) create new courseware;

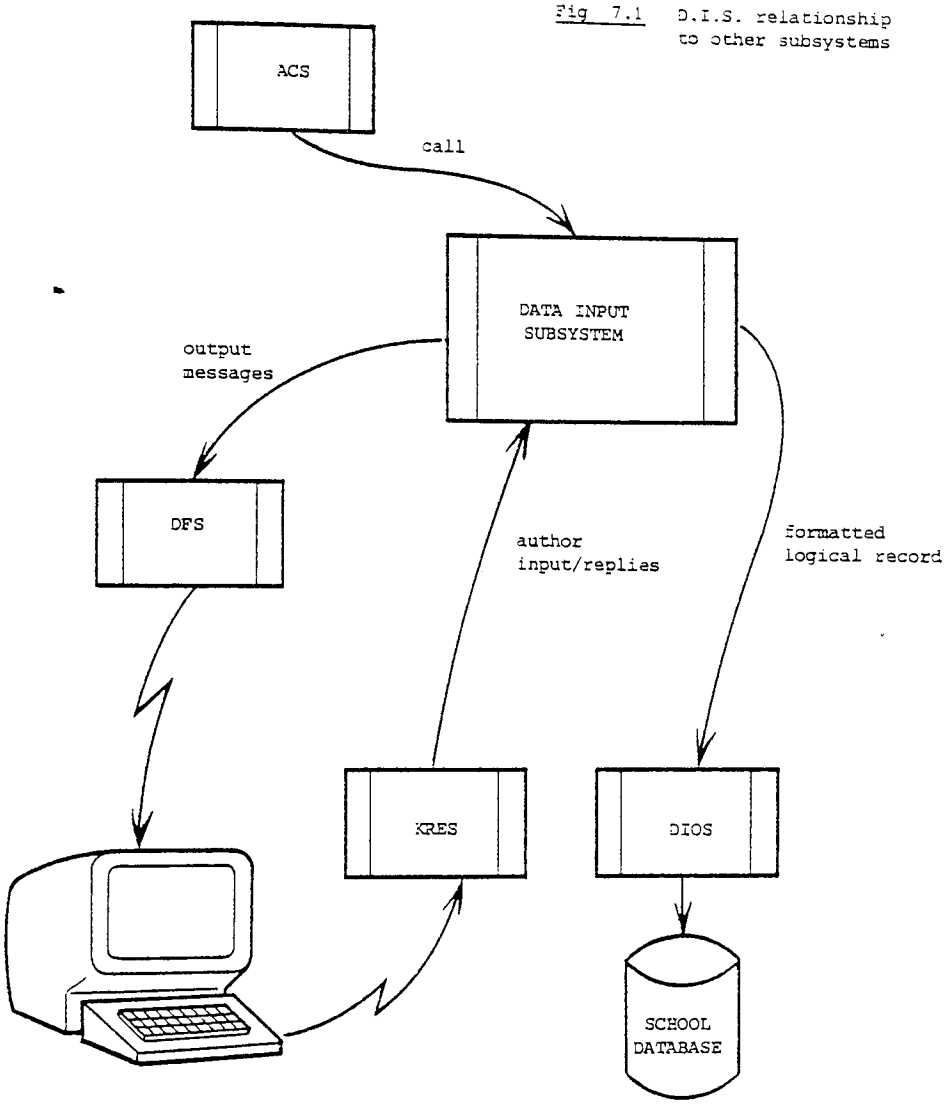
(ii) extend existing material;

and is invoked from the Author Control Subsystem (ACS) via the CREATE Author subcommand. It is by definition a Service Function Subsystem, but is a little unusual in that it can only be invoked from one source.

7.2 DIS - RELATIONSHIP TO OTHER SUBSYSTEMS

The Data Input subsystem interfaces with a number of other subsystems, as Fig 7.1 illustrates. (See overleaf)

Fig 7.1 D.I.S. relationship to other subsystems



N.B. DFS - Display Format Subsystem
KRES - Keyboard Reponse Evaluation Subsystem
DIOS - Database Input/Output Subsystem
ACS - Author Control Subsystem

The sequence of events described in Fig. 7.1 is as follows:-

(i) DIS is called from ACS;

(ii) a formatted logical record is built up from a dialogue with the Author (DFS displays prompts and messages, KRES accepts input). Note that Text Compression (part of DIS) may be involved here;

(iii) the created logical record is passed to DIØS for writing to the database;

7.3 COURSE AUTHORING

One of the original and fundamental aims of SCHOOL is that courseware creation should not involve Author Language programming. An alternative approach has been devised therefore, revolving around the use of DIS and Author Documents - specially designed worksheets used by the author to:-

- plan his course;
- design display layouts;
- specify control parameters;
- submit input to data preparation if large volumes of courseware are involved;
- use as reference material when creating courseware interactively.

Initially, authors will rely quite heavily on Author Documents, but this will tend to decrease as expertise is gained.

An overview of the course creation sequence is as per Fig. 7.2:

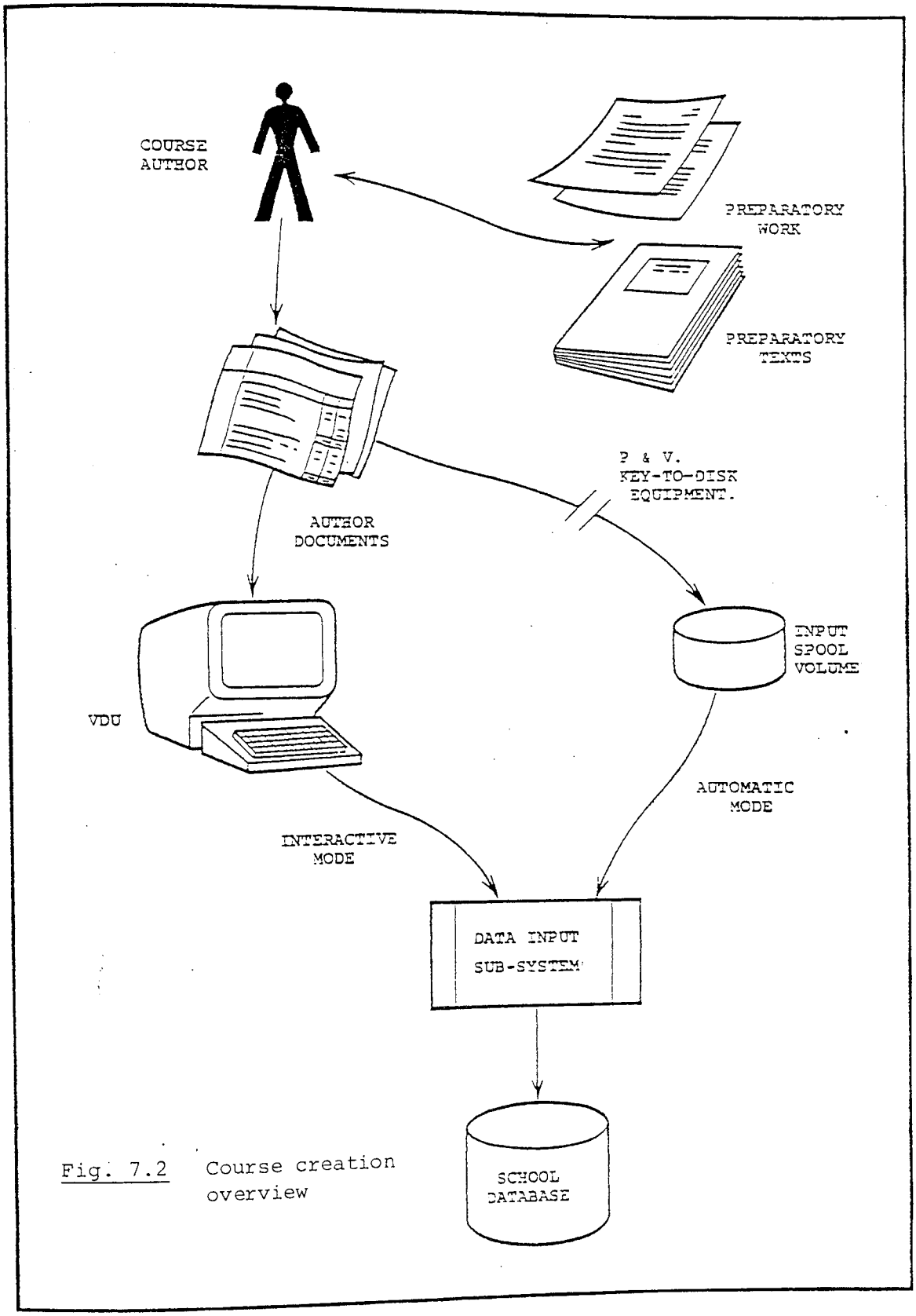


Fig. 7.2 Course creation overview

The sequence of events is basically as follows:

(a) the Author becomes fully conversant with the necessary subject matter, and adequately knowledgeable on the workings of SCHOOL. He then prepares a rough outline of the Course to be created, according to his teaching aims;

(b) The Author then defines his Course content and procedure on SCHOOL Author Documents;

(c) the Author Document data is then submitted to SCHOOL in one of two ways:

(i) Interactively on a prompt/reply basis via a terminal;

(ii) Automatically after suitable data preparation and transfer to an intermediate spool volume (via punched card, key-to-disk etc);

(d) once read in, the Author Document data is formatted and inserted into the SCHOOL Database by the Data Input Sub-system (DIS).

Each type of Author Document has a direct responsibility towards one or more database logical records. Consider Fig. 7.3:

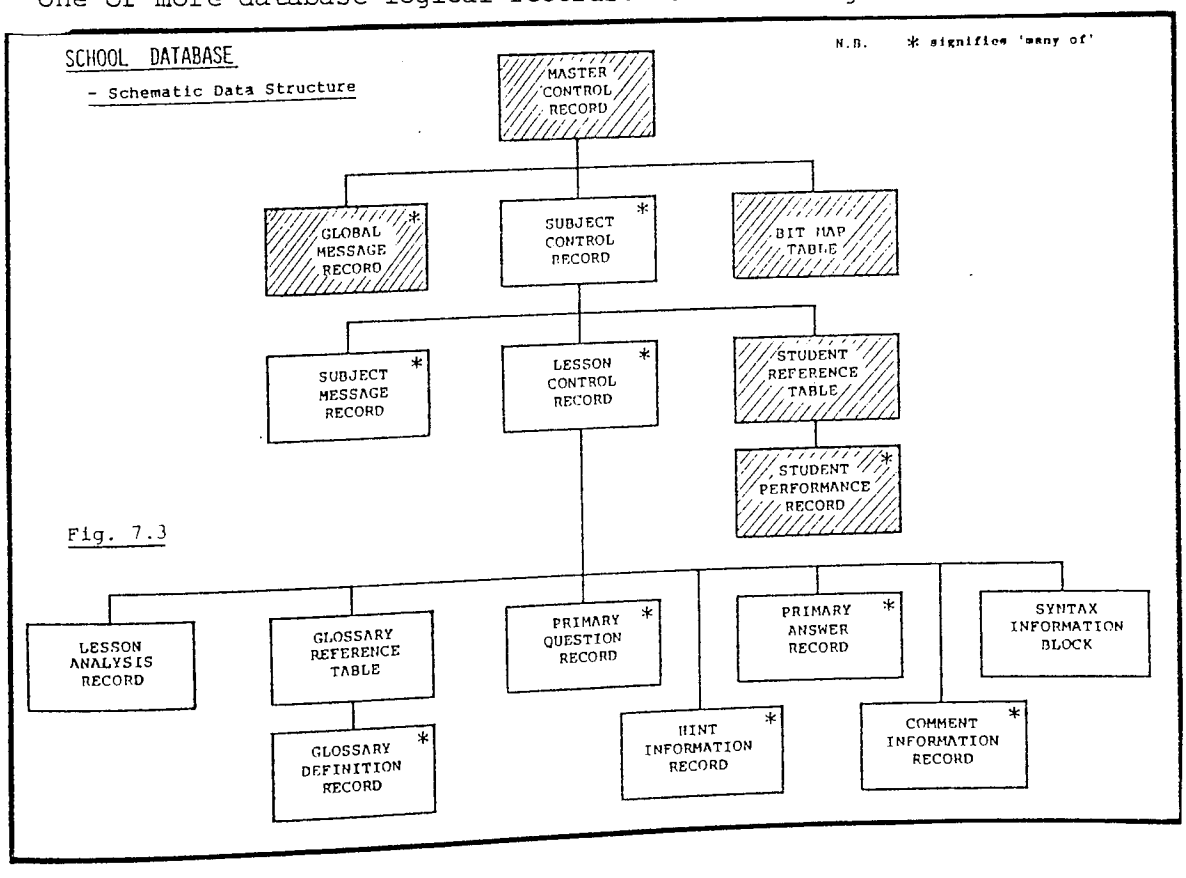


Fig. 7.3

Within Fig. 7.3, the unshaded boxes represent those components of the database that are within an Author's 'sphere of influence'. These may be created in different ways:

(i) Explicitly - a one-to-one correlation between Author Document and Logical record;

(ii) Implicitly - a particular parameter setting on one Author Document generates an extra logical record;

(iii) Interactively - no Author Document equivalent exists, and an Author Control Subsystem sequence is involved in the component's creation.

7.3.1 Author Documents

To correlate Author Documents, database logical records and methods of creation, consider Fig. 7.4.

Fig. 7.4

CREATION METHOD	LOGICAL RECORD	AUTHOR DOCUMENT
EXPLICIT	Subject Control Record (SCR)	Subject Definition
	Lesson Control Record (LCR)	Lesson Definition
	Glossary Definition Record (GDR)	Glossary Definition
	Syntax Informatiom Block (SIB)	Syntax Information
	Primary Question Record (PQR)	Question Definition
	Hint Information Record (HIR)	Hint Information
	Primary Answer Record (PAR)	Answer Definition
	Comment Information Record (CIR)	Comment Definition
IMPLICIT	Lesson Analysis Record (LAR)	Lesson Definition
	Syntax Information Block (SIB)	Lesson Definition (+ Syntax Info. as above)
	Glossary Reference Table (GRT)	Lesson Definition (+ Glossary Defn. as above).
INTERACTIVE	Subject Message Record (SMR)	-

For the creation of any particular course, a sequence of Author Documents builds up:

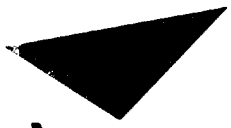
SUBJECT DEFINITION

repeat for each lesson:

- LESSON DEFINITION
- GLOSSARY DEFINITION) (optional)
- SYNTAX INFORMATION) (optional)
- repeat for each 'frame':
  - QUESTION DEFINITION
  - HINT INFORMATION (optional)
  - ANSWER DEFINITION
  - COMMENT DEFINITION (optional)

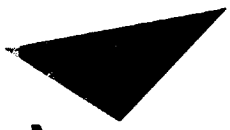
N.B. Within any subject there may be a maximum of 40 lessons, and within any lesson a maximum of 40 frames.

Samples of each type of Author Document follow:



Aston University

Content has been removed for copyright reasons

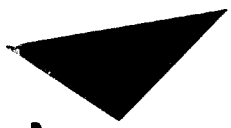


Aston University

Content has been removed for copyright reasons

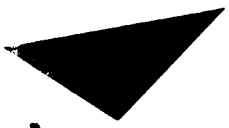
Aston University

Content has been removed for copyright reasons



Aston University

Content has been removed for copyright reasons

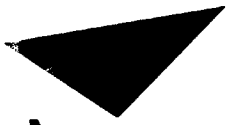


Aston University

Content has been removed for copyright reasons

Aston University

Content has been removed for copyright reasons



Aston University

Content has been removed for copyright reasons

Aston University

Content has been removed for copyright reasons

N.B. (i) preceding Author Document examples have been reduced to fit A4 paper. Normal dimensions are 13" x 8½";

(ii) what has not been described (for brevity) is the range of parameter options and defaults available on the various document types. This is described in considerable detail (along with all other aspects of courseware creation) in the SCHOOL Author Manual. Relevant extracts are reproduced at Appendix 7.1;

(iii) three of the options on the Lesson Definition document are of particular note:

SYNTAX CHECK REQUIRED?

AUTO ANALYSIS REQUIRED?

GLOSSARY REQUIRED?

- these imply the generation of Syntax, Lesson Analysis and Glossary logical records (SIB, LAR and GRT);

7.3.2 Courseware Creation Procedures

Two alternative approaches are available:

(i) Interactive

invoked via specific Data Input subcommands. A sequence of prompts is issued by DIS and the author reponds as appropriate, using the data prepared on his Author Documents. Once this sequence is completed, the corresponding logical record is built up and written to the database.

The following session hardcopies demonstrate typical approaches:

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN   5

 0  *
 1  *            SYSTEM SCHOOL:            AUTHOR CONTROL FACILITY.
 2  *            **************
 3  *  ENTER AUTHOR FUNCTION:
 4  *  COMMAND COMPLETE:  REPLY TO ORIGINAL REQUEST
 5  *  CREATE
 6  *  ENTER DATA CREATION TYPE:
 7  *  SUBJECT
 8  *
 9  *
10  *
11  *
12  *
13  *
14  *
15  *
16  *
17  *
18  *
19  *
20  *
21  *
```

response from *LIST Immediate Command (produces output on separate screen; not included here)

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN   6

 0  *
 1  *           AUTHOR FACILITY:    SUBJECT DEFINITION
 2  *           ****************
 3  *  ENTER SUBJECT NAME:
 4  *  SAMPLE 82
 5  *  ENTER ALLOCATED SUBJECT REFERENCE NO.
 6  *  31
 7  *  ENTER SUBJECT MANAGER DETAILS:
 8  *  NAME?
 9  *  MIKE LITTLE
10  *  LOCATION?
11  *  BLANK ANSWER UNACCEPTABLE:  RE-ENTER:-
12  *  COMPOWER CANNOCK
13  *  ENTER INTRODUCTORY TEXT - TYPE A COLON (:) WHEN A BLANK LINE IS REQUIRED:
14  *  :
15  *        SAMPLE INTERACTIVE COURSEWARE CREATION SESSION
16  *  :                          RUN DATE:  27/01/82    TIME:  13:46
17  *  :
18  *
19  *
20  *
21  *
```

A sequence of operations to create a lesson subordinate to the above subject, and carried out in another (later) session may be seen overleaf:

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN    5

            SYSTEM SCHOOL:              AUTHOR CONTROL FACILITY.
            *************
ENTER AUTHOR FUNCTION:
COMMAND COMPLETE:  REPLY TO ORIGINAL REQUEST
CREATE
ENTER DATA CREATION TYPE:
LESSON
ENTER SUBJECT NAME:
SAMPLE 82
ENTER LESSON NO:
1
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN    6

            AUTHOR FACILITY:    LESSON DEFINITION
            ***************
ENTER LESSON TITLE:
LESSON ONE
ENTER LESSON AUTHOR:
MIKE
ENTER PASSMARK - EASY:
50
ENTER PASSMARK - HARD:
80
ENTER INITIAL SCORING MODE:
DIFF
UNRECOGNISED SCORING MODE - REENTER:
EASY
IS AUTOMATIC LESSON ANALYSIS REQUIRED?
X
ANSWER YES OR NO:
NO
IS A HARD-COPY REQUIRED?
YE
ANSWER YES OR NO:
YES
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN    7

ENTER INTRODUCTORY TEXT - TYPE A COLON (:) WHEN A BLANK LINE IS REQUIRED
:
            CONTINUATION OF DEMO. CREATION SEQUENCE
:
:
:
```

(Several other interactive courseware creation sessions are documented at Appendix 7.2).

This current prompt/response approach is applicable to all types of terminal, but as most existing SCHOOL authors have access to IBM 3270-type devices, a screen-oriented equivalent is under development, as discussed in 'Conclusions and Proposed Extensions'.

(ii) Automatic

the Author Documents are treated as formal coding sheets and are submitted for conventional Data Preparation. Normal procedures are then followed, the end product being an input disk dataset which can be read into SCHOOL automatically.

By using a rigid naming convention for these datasets, (i.e. SUBxx CONTROL, where xx is an allocated subject reference no.), any set of courseware data may be prepared off-line and processed by DIS automatically. The actual generation of database records from this off-line data is done as an interactive session (using the AUTO subcommand), with appropriate messages being generated and (normally) a hardcopy print being taken. Typical examples follow:

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN   8

 0  ENTER DATA CREATION TYPE:
 1  AUTO
 2  ENTER ALLOCATED SUBJECT REFERENCE NO.
 3  01
 4            PROCESSING:  SUBJECT DEFINITION    - PLEASE WAIT
 5  SUBJECT NAME 'SYSTEM CHECKOUT' ALREADY ALLOCATED - ENTER A DIFFERENT NAME:
 6  SCHOOL TESTING
 7                 DEFAULT ASSUMED:  NO. LESSONS (0)
 8            PROCESS COMPLETED OK
 9  AUTOMATIC INPUT COMPLETE - PRESS ENTER TO CONTINUE
10
11
12
13
14
15
16
17
18
19
20
21
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN   9

 0  ENTER DATA CREATION TYPE:
 1  AUTO
 2  ENTER ALLOCATED SUBJECT REFERENCE NO.
 3  02
 4            PROCESSING:  SUBJECT DEFINITION    - PLEASE WAIT
 5  SUBJECT NAME 'CMS EDITING    ' ALREADY ALLOCATED - ENTER A DIFFERENT NAME:
 6  CMS EDITOR
 7            PROCESS COMPLETED OK
 8            PROCESSING:  LESSON DEFINITION     - PLEASE WAIT
 9                 DEFAULT ASSUMED:  SCORING MODE (EASY)
10                 DEFAULT ASSUMED:  SYNTAX REQUEST (NO)
11                 DEFAULT ASSUMED:  AUTO ANALYSIS (YES)
12                 DEFAULT ASSUMED:  HARD-COPY (NO)
13                 DEFAULT ASSUMED:  GLOSSARY REQD. (NO)
14            PROCESS COMPLETED OK
15            PROCESSING:  LESSON DEFINITION     - PLEASE WAIT
16            PROCESS COMPLETED OK
17            PROCESSING:  LESSON DEFINITION     - PLEASE WAIT
18                 DEFAULT ASSUMED:  NO. QUESTIONS (0)
19            PROCESS COMPLETED OK
20  AUTOMATIC INPUT COMPLETE - PRESS ENTER TO CONTINUE
21
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                   SCREEN  11

 0  ENTER DATA CREATION TYPE:
 1  END
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
```

7.4 DATA INPUT SUBSYSTEM STRUCTURE

The organisation of DIS is shown diagrammatically in Fig. 7.5:

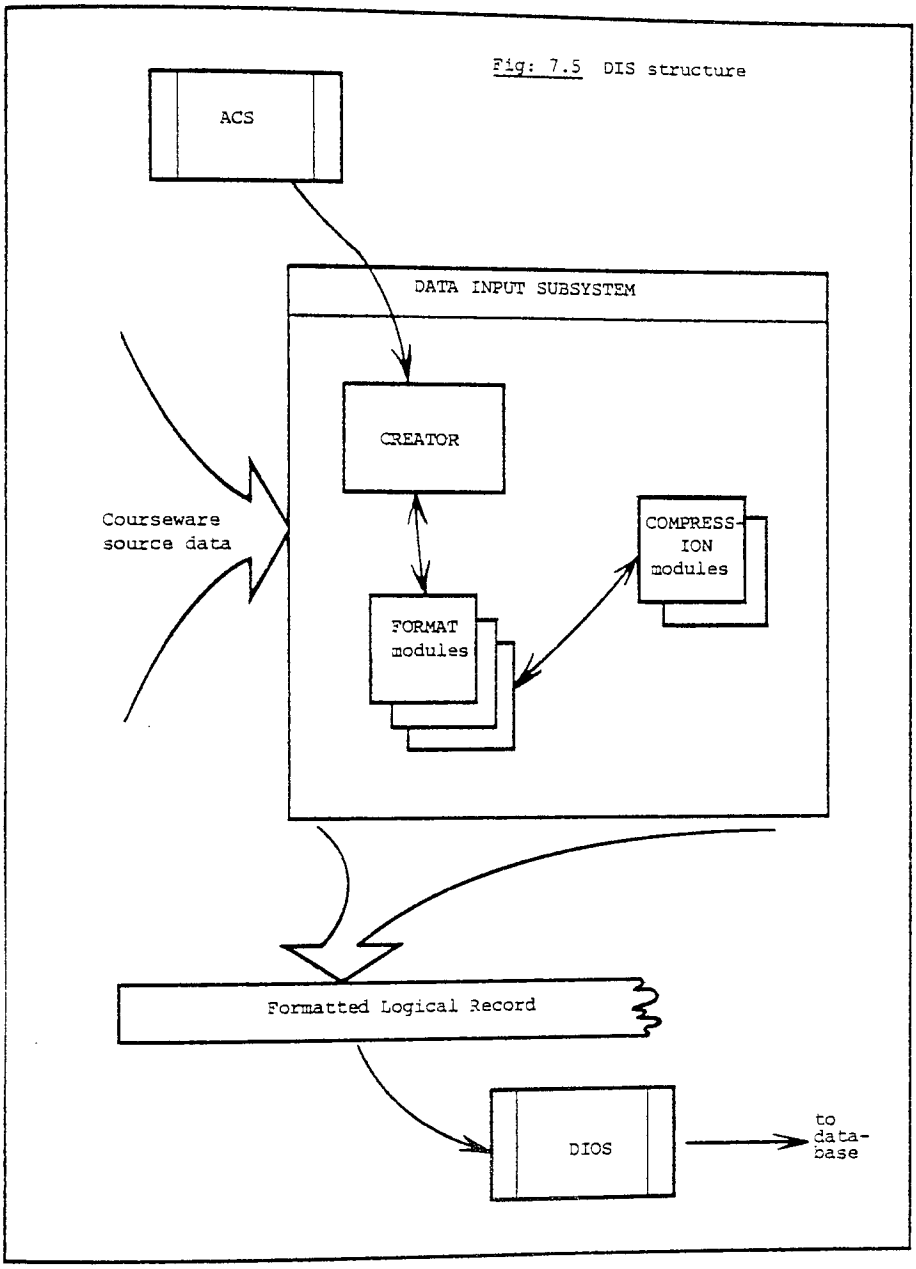


Fig: 7.5 DIS structure

As shown by Fig. 7.5, the modules which go together to provide DIS facilities are in 3 groups:

(i) module CREATOR

- overall control of DIS processing;

(ii) FORMAT modules:

- take Author Document data (either interactive or automatic) and produce a formatted SCHOOL database logical record;

(iii) COMPRESSION modules:

- called by the FORMAT modules, the effect is to take display text and produce Frame Data Block output (see later notes), with all embedded text compressed;

7.4.1 Module CREATOR

The logic associated with module CREATOR breaks down conveniently into three functions:

(i) requesting and handling Data Creation subcommands;

(ii) control of FORMAT modules;

(iii) cross-reference maintenance (i.e. creating any 'implied' logical records, inserting pointers and data values into any associated logical records);

The processing within these functions breaks down as per Fig. 7.6 overleaf.

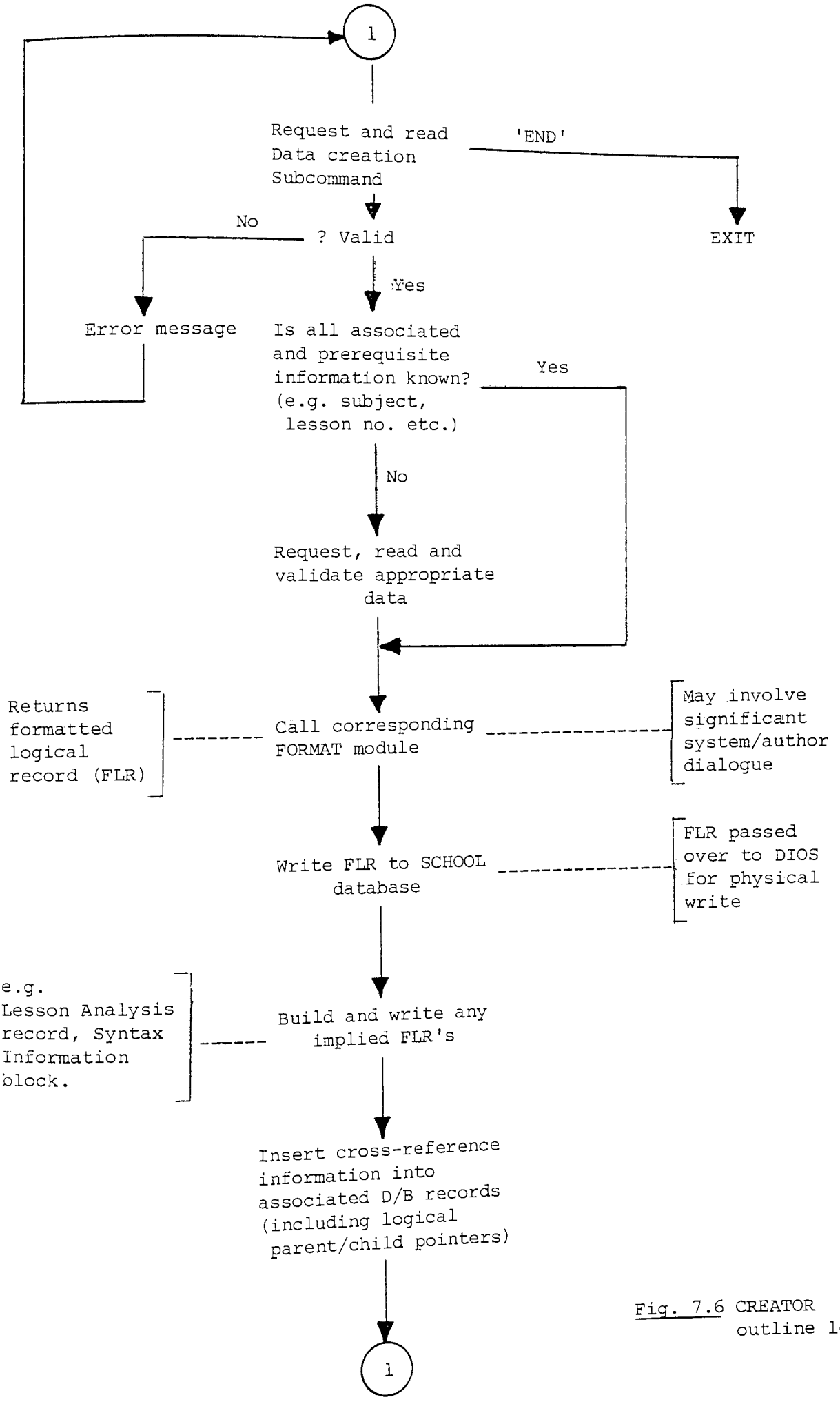


Fig. 7.6 CREATOR outline logic

Notes on Fig. 7.6:

(i) the available Data Creation subcommands and the corresponding database logical records are as follows:-

Fig. 7.7

SUBCOMMAND	RELATED LOGICAL RECORD(S)	COMMENTS
AUTO	Multiple	Actual database logical records created operands on contents of the defined Author Document file.
END	-	Terminates data creation sequence.
GLOBMSG	Global Message Record (GMR).	Always created interactively (no AUTO equivalent).
GLOSDEFN	Glossary Definition	Requires a previously created Glossary Reference Table (GRT) and Lesson Control Record (LCR)
GLOSSARY	Glossary Reference Table (GRT)	Requires a previously defined Lesson Control Record (LCR) and results in an empty Glossary Reference Table being set up.
LESANAL	Lesson Analysis Record (LAR)	Requires a previously defined LCR and results in a zeroised Lesson Analysis Record.
LESSON	Lesson Control Record (LCR)	Requires an already defined subject Control Record (SCR). May imply creation of LAR and Syntax Information Block (SIB).
QUESTION	Primary Question Record (PQR) Hint Information Record (HIR) Primary Answer Record (PAR) Current Information Record (CIR) (i.e. complete Frame Structure.)	The actual range of logical records created within the Frame segment depends on the Author's requirements (e.g. HIR and CIR are optional). Under all circumstances however, an associated Lesson Control Record must already exist.

/continued

SUBCOMMAND	RELATED LOGICAL RECORD(S)	COMMENTS
RESET	-	Provides the ability to reset current subject name, lesson no. and question no.
SUBJECT	Subject Control Record (SCR)	Also creates an empty Student Reference Table (SRT).
SUBMSG	Subject Message Record (SMR)	Always created interactively (no AUTO equivalent). An associated SCR must have been previously defined.
SYNTAX	Syntax Inform-action Block (SIB)	Requires a previously defined Lesson Control Record (LCR).

(ii) as described above, several Data Creation subcommands cannot be actioned until certain prerequisite definitions have been completed (e.g. a Lesson cannot be defined until its parent subject is initialised). These prerequisites are normally self-evident and are defined to DIS by the Current Segment Block (CSB). This defines:

- current subject name
- current lesson number
- current question number.

As any Data Creation subcommand is issued, the CSB is checked to ensure that all prerequisite information is known - if not appropriate definitions will be requested before proceeding;

(iii) given 'prerequisite considerations', Data Creation subcommands can be activated in almost any order - although the RESET subcommand may be required to ensure that lessons and questions are subordinate to the appropriate parent(s). Consider the following subcommand sequences:

Example A: SUBJECT

LESSON (1)

QUESTION (repeat for 1,2,3,4,5)

LESSON (2)

QUESTION (1,2,3,4)

LESSON (3)

QUESTION (1,2,3,4,5,6)

END

Example B: SUBJECT

LESSON (1)

LESSON (2)

LESSON (3)

RESET (to lesson 1)

QUESTION (1,2,3,4,5)

RESET (to lesson 3)

QUESTION (1,2,3,4,5,6)

RESET (to lesson 2)

QUESTION (1,2,3,4)

END

These sequences have an identical effect, and this concept, provides the courseware Author with considerable session flexibility - even to the extent of creating different parts of his material at different times.

(iv) a final (and often complex) responsibility of module CREATOR is that of maintaining database cross-reference information. Typically this takes two forms:

- inserting a reference entry (e.g. lesson no.) in a D/B logical record;

- establishing logical parent/child relationships, normally in terms of physical DBTU numbers.

7.4.2 FORMAT modules

DIS FORMAT modules have the simple function of converting Author Document material into SCHOOL Database logical records. This is achieved in 3 stages:

(a) read in courseware data, either:

  (i) interactively on a prompt/response basis (or with the latest version of SCHOOL, via a full-screen input facility);

  (ii) automatically from an Author Document file as discussed in subsection 7.3.2, with messages being directed to the Author's terminal as processing is carried out;

(b) call Compression modules to process text. Subsequent processing is in two parts:

  (i) build Frame Data Block(s) i.e. encoded screen format definitions - see Chapter 8;

  (ii) compress display text;

(c) build a formatted logical record from the created Frame Data Blocks (FDB's) and other course control information. Once this is complete, control is passed back to CREATOR;

N.B. Database logical records are constructed within the FORMAT module Data Nucleus - visible to all FORMAT modules and CREATOR.

The range of FORMAT modules and associated logical records are as per Fig. 7.8:

Fig. 7.8

MODULE	CORRESPONDING LOGICAL RECORD(s)
FORMSDEF	SCR (Subject Control Record)
FORMLDEF	LCR (Lesson Control Record)
FORMLAR	LAR (Lesson Analysis Record)
FORMSMR	SMR (Subject Message Record)
FORMGDR	GDR (Glossary Definition Record)
FORMQDEF	PQR (Primary Question Record)
FORMHINT	HIR (Hint Information Record)
FORMADEF	PAR (Primary Answer Record)
FORMCOMM	CIR (Comment Information Record)
FORMSYN	SIB (Syntax Information Block)

7.4.3 Text Compression modules

Each block of text held on the SCHOOL Database is encoded to:

(i) remove blank lines;

(ii) remove leading and trailing spaces from each line;

(iii) compress text within each line;

This is known as Frame Data Block (FDB) format, and conversion from author-input to FDB data is carried out within DIS by 2 modules, viz:

BUILDFDB - constructs FDB's from incoming text (see Fig. 8.3, Chapter 8 for a full definition of FDB structure);

COMPRESS - compresses text strings within each FDB (as per algorithm described in Chapter 4);

FDB data assembled by BUILDFDB and COMPRESS is passed back to the calling FORMAT module, which then inserts this into the appropriate part of the logical record being processed.

N.B. Two complementary expansion modules (BREAKFDB and EXPAND) exist as part of DFS, the Display Format Subsystem - see Chapter 8.

chapter eight

- DISPLAY FORMAT SUBSYSTEM

8.1 INTRODUCTION

A major area of neglect in many previous CAI Systems has been that of presentation flexibility. All too often, the poor visual interface between System and Student has resulted in badly formatted instructional output, unprofessional appearance and rapid loss of student interest.

To a limited extent, this has been a result of the restrictions of the available hardware - teletypes, slow lines etc. Recent hardware advances have removed most of these but only a few CAI systems have utilised this extra freedom (e.g. IBM Interactive Training System), and often not as far as they might.

It was laid down therefore, that current terminal hardware - e.g. VDU's, high speed lines, satellite printers etc. should be used to the full within SCHOOL, placing as few constraints as possible on the creation of instructional material.

8.2 COMPOWER TERMINAL HARDWARE

As of August 1981 the major terminal Hardware implemented with Compower is as follows:-

VDU's - IBM 3270 range;
Mellordata; ITT 3280 range;
DEC VT100; IBM 8775.

Teletypes - IBM 2741;
IBM 3767;
Anderson-Jacobson 832;
DEC Decwriter;
Itel 1051;

These are online to a variety of host Systems running on a variety of mainframes:-

Systems:	VM/370 CMS	(IBM)
	IMS/DC	(IBM)
	VSPC	(IBM)
	TSO	(IBM)
	RSX-11M	(DEC)
Processors:	IBM 3081	
	IBM 3033	
	IBM 370/158	
	PDP 11/70	
	PDP 11/45	(several)
	IBM 8100	(")

It is eventually intended to employ a wideband network (using appropriate data communications protocols) whereby it will be possible to access any system on any mainframe from any physical terminal.

The projected number of terminals is in the region of one thousand, the majority being IBM 3270 VDU's (or equivalent).

8.3 DISPLAY FORMAT SUBSYSTEM DESIGN PHILOSOPHY

The design philosophy of SCHOOL's Display Format Subsystem (DFS) was established along the following lines:-

(i) DFS should be device dependent:

- SCHOOL will communicate with each user in the optimum manner for his type of terminal;

(ii) Author constraints should be minimised:

- subject authors should be able to use terminal devices to their full capabilities:

(iii) DFS should be extremely efficient

- points (i) and (ii) should be implemented in such a way that terminal display is as rapid as possible;

8.4 DFS DEVICE OPTIMISATION

Although there is a wide variety of terminal types online to Compower's Teleprocessing Systems, these are essentially split into two groups:

(i) 'Graphics devices' - a misnomer; this is IBM terminology for VDU's similar to the 3270 family. Most are not graphics VDU's in the normal sense;

(ii) 'Terminal devices' - all devices not in the former group, including some types of VDU and all Teletypes.

As soon as SCHOOL is invoked the Device Class is evaluated. This is done by module CYBER, which requests the information from the Real Device Block (RDEVBLOK) as held by the VM/370 Control Program.

Device Class codes returnable are:

X'40' - Graphics device

X'80' - Terminal device

N.B. Within each Device Class there are further Device characteristics (e.g. number of screen lines, device status, model number, etc.) but these are not currently of significance to DFS, as all operational terminals adhere to the descriptions given in Section 8.4.1.

CYBER then communicates this information to the remainder of SCHOOL by setting a flag (DEVTYPE) resident within CYBER's Data Nucleus of either 'GRAF' or 'TERM'.

Whenever SCHOOL subsequently displays information to the user, it uses the DEVTYPE setting to determine the Display technique required.

8.4.1 Device Characteristics:

Certain individual device characteristics are of significance:

3270 VDU's (i.e. 'Graphics' Devices)

(i) Screen size: 24 lines of 80 characters;

(ii) Under VM/370 the bottom 2 lines are designated the User Input Area, and as such should not be written to by program;

(iii) Under other systems, input may be in any screen position;

(iv) Current screen contents may be erased (either by programs or the user) prior to any write operation;

(v) Multiple lines may be written in one operation (only marginally slower than writing one line) and writing can begin at any screen line. This is applicable up to a maximum of 22 lines (1760 bytes) for VM/370 CMS, and 24 lines (1920 bytes) for other systems;

(vi) Extra (optional) facilities which can be supported are:-

Selector (light) pens

Audio alarm

Function buttons

Non - 3270 devices (i.e. 'Terminal' devices)

(i) Variable line sizes:

Mellordata VDU	-	72 (22 lines)
Anderson Jacobson	-	130 or 155
IBM 2741	-	130
Itel 1051	-	130

(ii) Input is always at the bottom line, the paper moving up accordingly, or in the instance of the Mellordata VDU, the complete screen contents move upwards (the top line being lost).

(iii) With hard copy terminals obviously no erase operation is possible, but with the Mellordata screen erase as for IBM 3270's is available. New input then starts at the bottom of the blank screen.

(iv) Only one line may be written at a time, although blank lines may be generated by imbedded Newline (NL) characters;

8.4.2 Display techniques

The process of displaying SCHOOL data at any terminal is split into two parts:

(a) construction of Screen Buffer from the message data;

(b) display (device optimised) of the Screen Buffer contents at the terminal.

Fig. 8.1 overleaf illustrates the overall organisation.

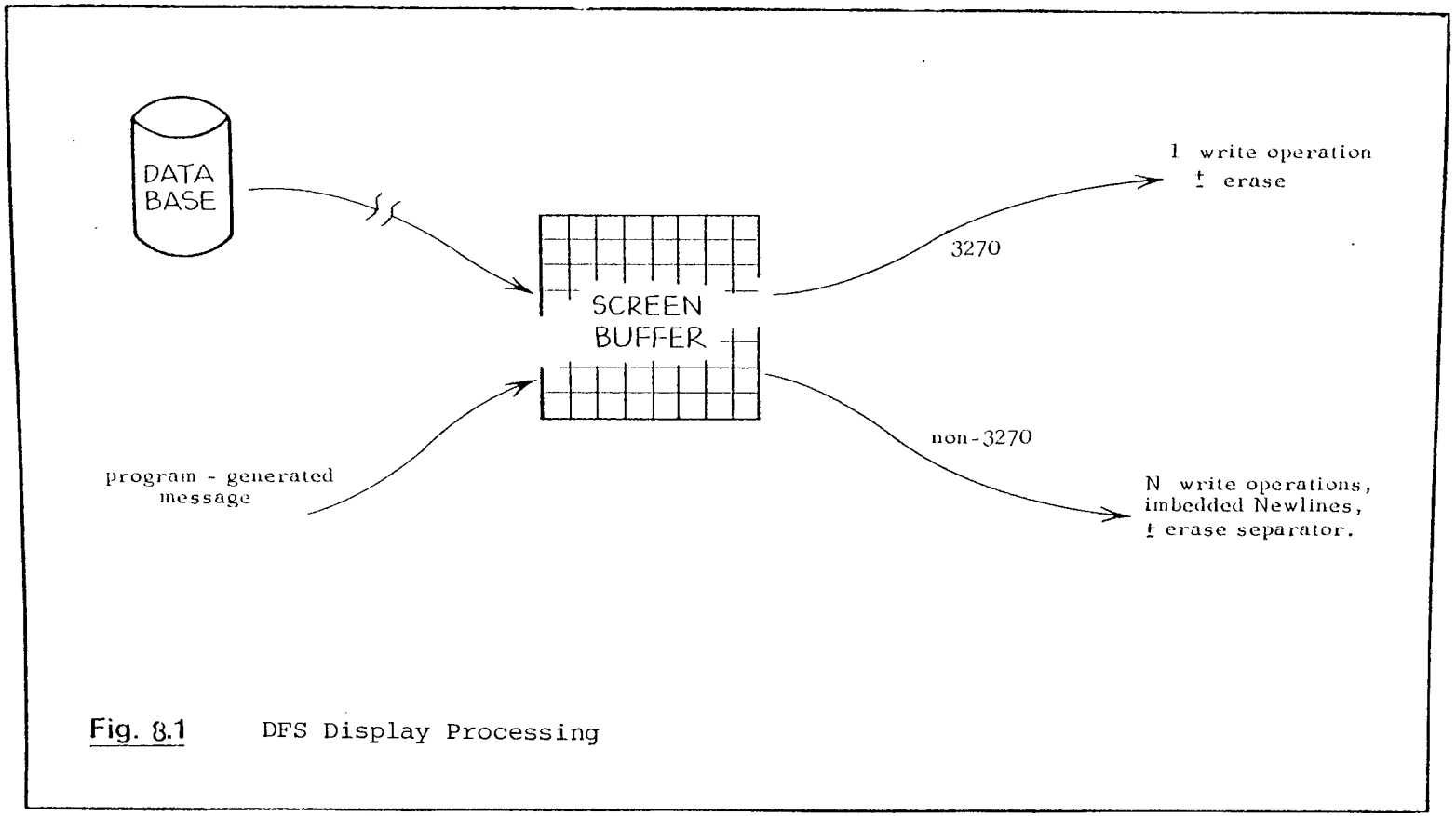


Fig. 8.1 DFS Display Processing

The Screen Buffer is currently 1760 bytes in size, representing 22 lines of 80 characters. This is a 'compromise' between the restrictions imposed by the various host systems under which SCHOOL can run. It can easily be altered.

Message display (handled entirely by module SCREEN) now depends on the characteristics of the active terminal device. The algorithm below (Fig. 8.2) illustrates the logic:

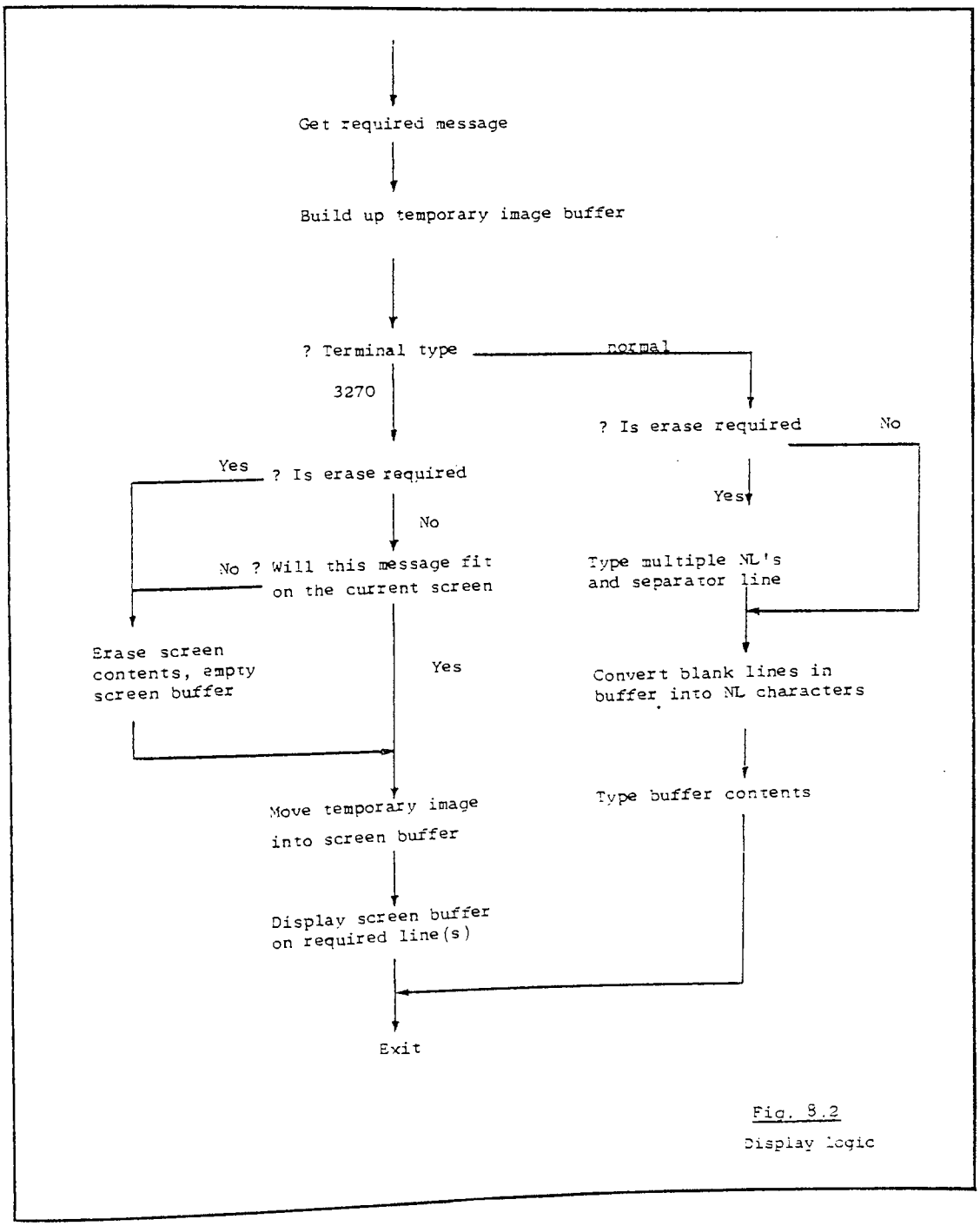


Fig. 8.2
Display logic

Notes on the preceding algorithm:

(i) All 3270 display operations are carried out via CALL's to IBM routine DMSGIO (Release 2). This requires the following parameter table:

Address of Screen Buffer

Start screen line number

Number of bytes to be displayed

Erase/Cancel flag (normally 0 - no erase)

(ii) 3270 erase operations are also carried out via DMSGIO, with the Erase/Cancel flag set to 2. Current screen contents are immediately cleared.

Whenever an erase operation is executed, the existing screen contents are stored in a backup buffer for subsequent recall (if requested via *RESCREEN Immediate Command - see Chapter 11).

(iii) Erase operations on non-3270 devices are simulated by typing 10 blank lines, a line of 50 asterisks and another 10 blank lines.

(iv) Whenever the Screen Buffer contents are to be typed at a non-3270 terminal, embedded blank lines within the buffer are generated on the device using NL characters.

No lines of spaces are typed.

8.4.3 Hard-copy facility

Each VM/370 user has configured within his Virtual Machine a Virtual Line Printer which can be used to take a hard-copy of all terminal I/O, normally unformatted.

DFS has built into it a mechanism to make use of this. At the discretion of the System Controller or Subject Manager/Author, a flag can be set by which all Terminal Input/Output is simultaneously printed on the virtual line printer.

The production of this hard-copy is controlled by the current setting of HCFLAG, a 1-byte field resident in module CYBER Data Nucleus. Settings are:-

00 - no hard-copy

01 - hard-copy produced

The HCFLAG setting may be defined at any level above that of student, i.e.

	ALL SUBJECTS	ALL LESSONS WITHIN SUBJECT	PARTICULAR LESSON WITHIN SUBJECT
SYSTEM CONTROLLER	✓	✓	✓
SUBJECT MANAGER		✓	✓
LESSON AUTHOR			✓

The exact format of the generated print depends on the type of terminal being monitored;

(i) 3270 devices:

- output is formatted into 'screen boxes', simulating the original appearance of each VDU screenful. These are generated whenever a screen erase is executed.

Examples overleaf:

Sample Hardcopy screen 'boxes':

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN   2

 0  *
 1  *
 2  *          COMPUTER POWER TRAINING SCHOOL  -  CAT SYSTEM  (VERSION ML/1.01)
 3  *
 4  *
 5  *
 6  *           @@@@@@     @@@@@@    @@    @@    @@@@@@     @@@@@@    @@
 7  *          @@@@@@@@   @@@@@@@@   @@    @@   @@@@@@@@   @@@@@@@@   @@
 8  *          @@    @@   @@    @@   @@    @@   @@    @@   @@    @@   @@
 9  *          @@@        @@         @@    @@   @@    @@   @@    @@   @@
10  *           @@@@@     @@         @@@@@@@@   @@    @@   @@    @@   @@
11  *             @@@@@   @@         @@@@@@@@   @@    @@   @@    @@   @@
12  *                @@@  @@         @@    @@   @@    @@   @@    @@   @@
13  *          @@    @@   @@    @@   @@    @@   @@    @@   @@    @@   @@
14  *          @@@@@@@@   @@@@@@@@   @@    @@   @@@@@@@@   @@@@@@@@   @@@@@@@@
15  *           @@@@@@     @@@@@@    @@    @@    @@@@@@     @@@@@@    @@@@@@@@
16  *
17  *
18  *  ENTER OPERATION MODE:-
19  *  AUTHOR
20  *  ENTER PASSWORD:
21  *  ELANS3
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN   6

 0  *           AUTHOR FACILITY:    SUBJECT DEFINITION
 1  *           ****************
 2  *  ENTER SUBJECT NAME:
 3  *  TESTRUN
 4  *  ENTER ALLOCATED SUBJECT REFERENCE NO.
 5  *  94
 6  *  ENTER SUBJECT MANAGER DETAILS:
 7  *  NAME?
 8  *  MIKE LITTLE
 9  *  LOCATION?
10  *  COMPOWER
11  *  ENTER INTRODUCTORY TEXT - TYPE A COLON (:) WHEN A BLANK LINE IS REQUIRED:
12  *  :
13  *  DEMONSTRATION TEST SUBJECT -
14  *                     DESIGNED TO:
15  *                           (A) SHOW SUBJECT DEFINITION SEQUENCE
16  *                           (B) ILLUSTRATE HARD-COPY FACILITY
17  *
18  *
19  *
20  *
21  *
```

(ii) Non-3270 devices:

- each terminal line is output to the virtual line printer as it is created;
- Newline characters are replaced by special printer control characters:

0 NL's	-	x'40'	(next line)
1 NL	-	x'FO'	(one blank line)
⩾ 2 NL's	-	x'60'	(2 blank lines)

N.B. Many other examples of hard-copy facility output may be found throughout this thesis and various Appendices.

8.5 DFS AUTHOR CONTROL

The majority of devices on-line to Compower Teleprocessing Systems are VDU's, usually IBM 3270 type. For this reason the Author considerations within DFS have been orientated towards screen terminals.

As detailed in Chapter 7, SCHOOL courseware is created on special Author documents designed to provide much better layout control than would be possible using an Author Language.

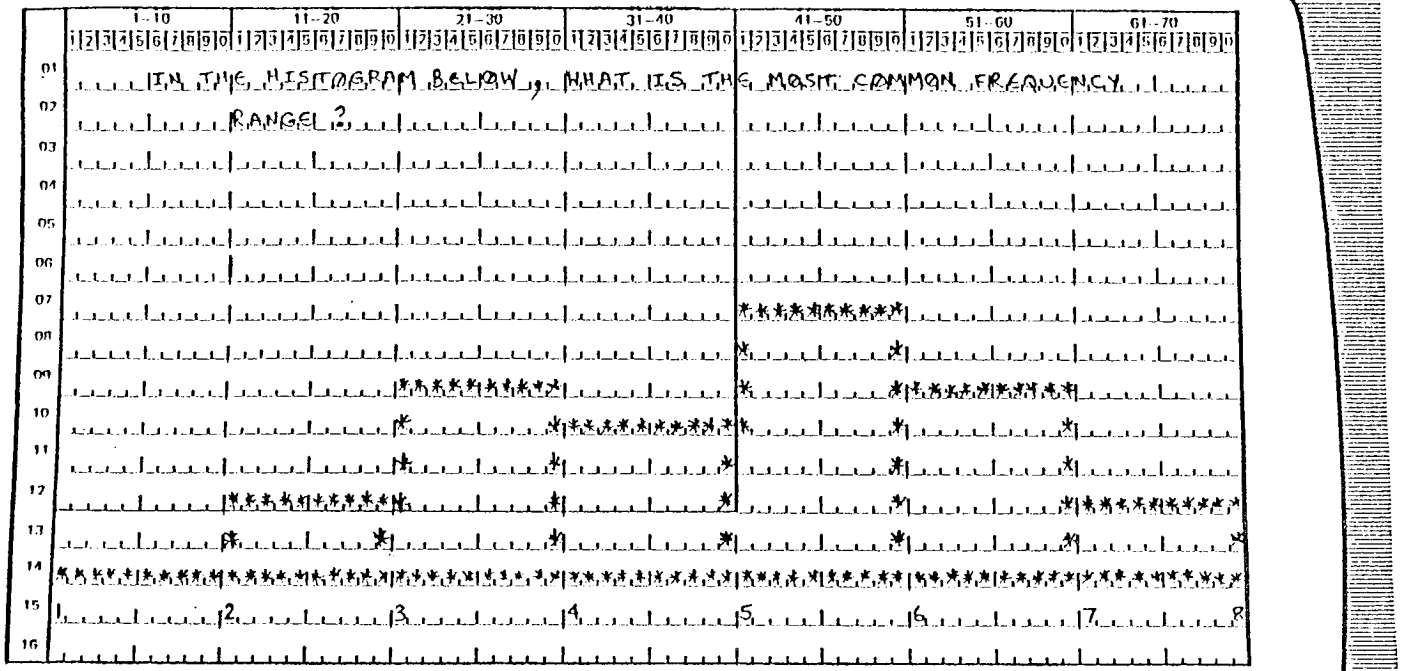
Various Author documents can define 'blocks' of text:

Document.	Number of lines (of 70 characters)
Question definition	16
Subject definition	5
Lesson definition	5
Hint Information	4
Comment definition	2
Glossary definition	3

and whatever is coded in the appropriate block is displayed by DFS in exactly the defined format - although it may be stored by the System differently. This gives Authors virtually complete control over the format in which data is presented to students, as the following examples show:

/see the following 2 pages.....

FORM NO. S/001



17 QUESTION No. [10 11]

HINT THRESHOLDS [20 21] [23 24] [26 27] A B C

HINT PENALTIES [30 31] [33 34] [36 36] A B C

71-80 [1 2 3 4 5 6 7 8 9 0]

18 SCREEN ERASE REQD. [10 11 12]

SEQUENCE: Normal [20 21] Remedial [28 29]

MAXIMUM SCORES [31 32 33] [35 36 37] easy hard

71-80 [1 2 3 4 5 6 7 8 9 0]

71-80 [1 2 3 4 5 6 7 8 9 0]

SCHOOL SYSTEM

- question definition

NB: Only punch lines coded within columns 1-70
Always punch cards 17 and 18

AUTHOR:

FORM NO. S/001

	1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80
01	IN THE FOLLOWING COBOL CODING, WHICH STATEMENT							
02	CONTAINS AN ERROR?							
03								
04								
05	0113 6900 3D.							
06	0137 00	IF WILN-CNT IS GREATER THAN 47	PERFORM D-HEADINGS.					
07	0138 00	PERFORM B-RITE-LINE						
08	0139 00	PERFORM E-FINAL-LINES.						
09	0140 00	STOP RUN.						
10								
11								
12	ENTER THE SEQUENCE NUMBER OF THE STATEMENT WHICH							
13	YOU THINK IS IN ERROR.							
14								
15								
16								

17 QUESTION No. [10 11]

HINT THRESHOLDS [20 21 | 23 24 | 26 27] A B C

HINT PENALTIES [30 31 | 33 34 | 35 36] A B C

71-80 [1 2 3 4 5 6 7 8 9 0]

18 SCREEN ERASE REQD. [10 11 12]

SEQUENCE: Normal [20 21] Remedial [28 29]

MAXIMUM SCORES [31 32 33 | 35 36 37] easy hard

71-80 [1 2 3 4 5 6 7 8 9 0]

SCHOOL SYSTEM

- question definition

NB: Only punch lines coded within columns 1 - 70
Always punch cards 17 and 18

AUTHOR:

8.6 DFS EFFICIENCY CONSIDERATIONS

SCHOOL's Display Format Subsystem is designed in such a way to ensure that the flexibility of display offered has minimal impact on the efficiency of the system - typically in terms of text storage.

Consider the screen definition as at Example A. In the format defined, 15 lines of 70 characters are used, i.e. 1050 bytes. Certain steps can be taken to reduce this:

(a) Compress text;

(b) Remove blank lines;

(c) Remove leading spaces;

(d) Remove trailing spaces.

Carrying out all of the above processing has a marked effect on the storage requirements:-

(a) Compressing text:

- text occupies 65 bytes (including punctuation and embedded blanks).
- assuming a 25% reduction, this could be held in 49 bytes (saving 16 bytes).

(b) Removing blank lines:

- 4 of the 15 lines of screen display are completely blank and can be removed - a saving of 280 bytes.

(c) Removing leading spaces:

- this can be carried out on lines 1, 2, 7, 8, 9, 10, 11, 12, 13 and represents a saving of 175 bytes.

(d) Removing trailing spaces:

- this can be carried out on lines 1, 2, 7, 8, 9, 10, 11 and represents a saving of 130 bytes.

The total amount of storage that could be saved therefore is 601 bytes - representing a Compression Factor of 57.24%. Applying the same process to Example B, reveals an even larger Compression factor of 695 bytes over 13 lines, i.e. 76.3%; (Operationally, control information will erode this slightly).

Although the two calculations above constitute only a small sample, they serve to illustrate the effectiveness of the four storage reduction techniques in respect of screen displays. These have been implemented within DFS via the concept of FRAME DATA BLOCKS.

8.6.1 Frame Data Blocks

All displayable data resident on the SCHOOL Database, irrespective of function, is held in FRAME DATA BLOCK (FDB) form. This is a device whereby any screen display can be stored with all previously discussed compression techniques applied, and a typical FDB structure is as per Fig. 8.3 overleaf:

FDB structure

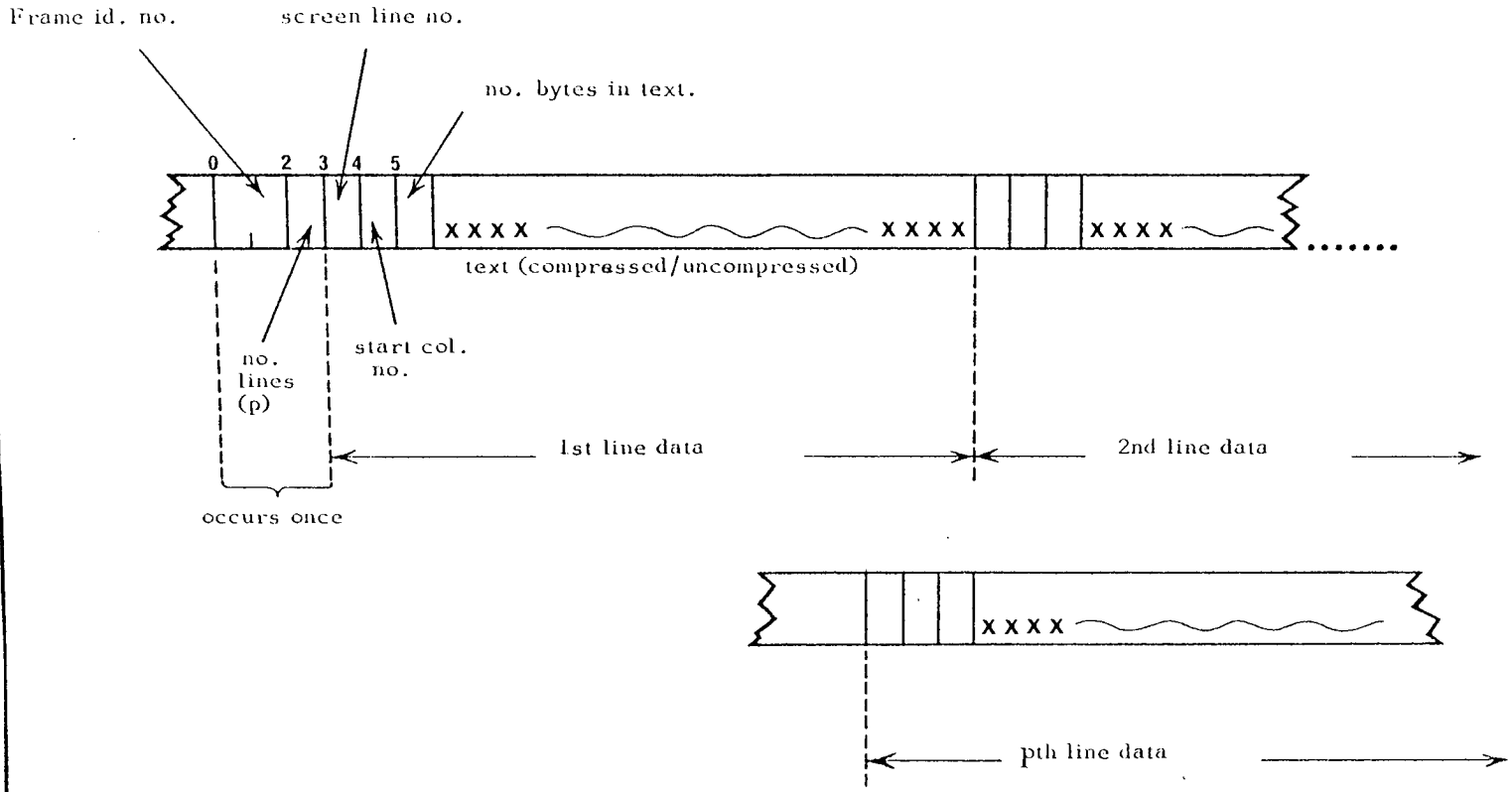


Fig. 8.3

Notes on Fig. 8.3

(i) Frame identification No:

- for Question text this will contain the question no. (e.g. for question 7, it will contain FO F7);

- for other types of data:

  byte1 contains X'FF'

  byte2 contains a numeric text identification code:

  00 - Subject Introduction

  01 - Lesson Introduction

  02 - Hint Information

  03 - Answer Comment

  04 - Global message

  05 - Subject message

  06 - Special Category 1 (e.g. SCHOOL header)

(ii) No. lines (p):

- the number of non-blank lines in the display,

- bit 0 is the 'Erase Bit', with settings:

  0 - no screen erase

  1 - erase current screen prior to displaying this frame.

N.B. The effect of the Erase Bit may be overidden by the System if necessary.

(iii) Screen line number: (1 byte, binary)

- line number on screen (0-21) on which the following text line is to be inserted - as defined on the Author document. This may be altered by the System if no screen erase is requested and if this message will fit on partially used screen;

(iv) Start column number: (1 byte, binary)

- character position (1-70) on the defined line of the first non-blank character;

(v) No. bytes in text (N): (1 byte, binary)

- number of bytes in the line text (either normal or compressed form). This will normally be no more than 70 characters, although certain special function messages may be up to 128 characters long (resulting in screen "wrap-round");

- bit 0 is the 'Compression bit', with settings

  0 - text in normal form

  1 - text compressed:

(vi) Text: (N bytes, EBCDIC)

- screen line text in either normal or compressed form (see Chapter 4).

N.B. Items (iii), (iv), (v) and (vi) are repeated for each text- containing line in the display.

8.6.2 Frame Data Block Encoding/Decoding:

To create FDB's from Author-provided text, and to break them back down, two interface modules are involved:

BUILDFDB - create FDB's from formatted text;

BREAKFDB - breaks down FDB's into screen buffer form.

Fig. 8.4 illustrates the relationship between these modules:

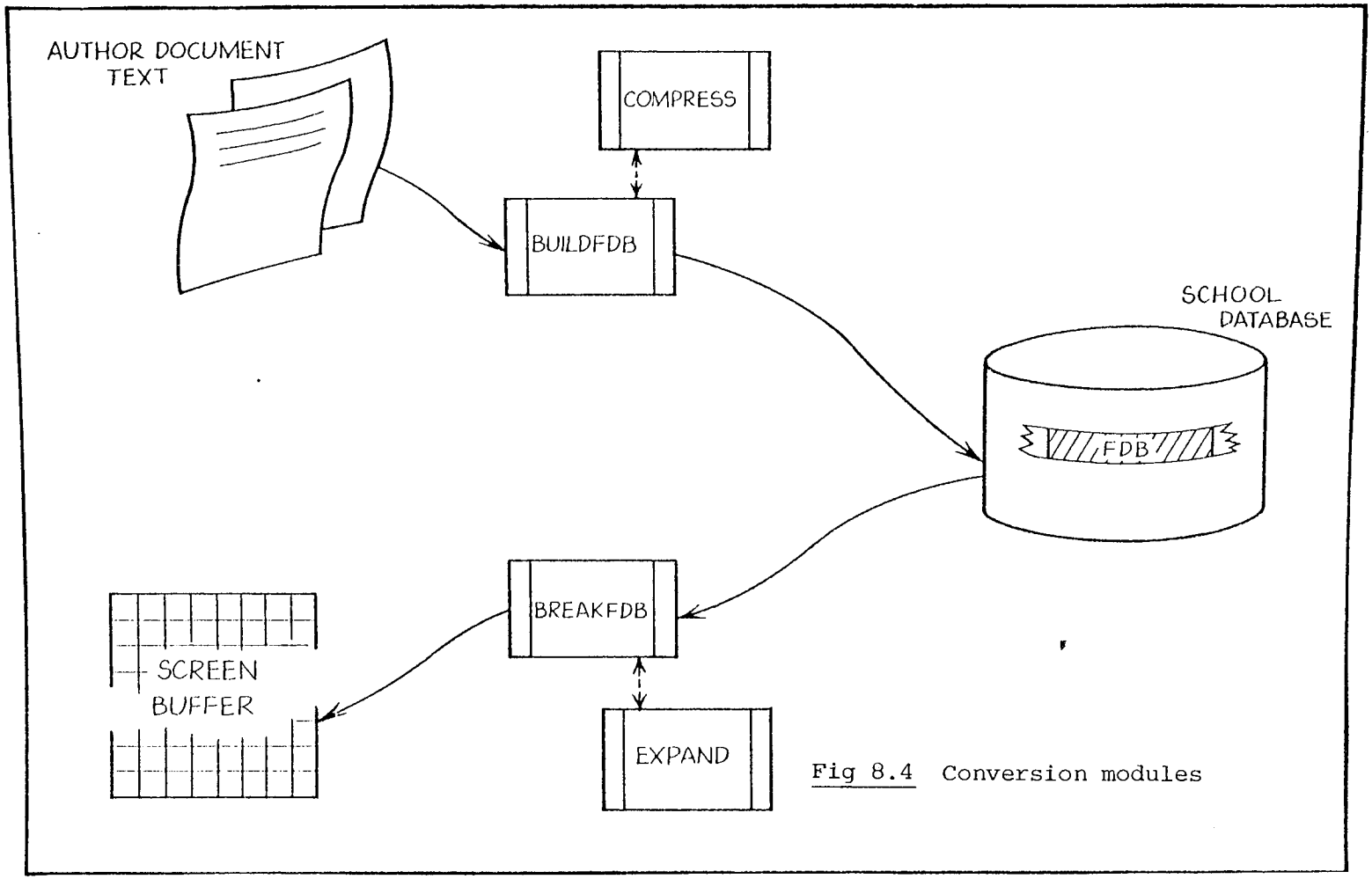


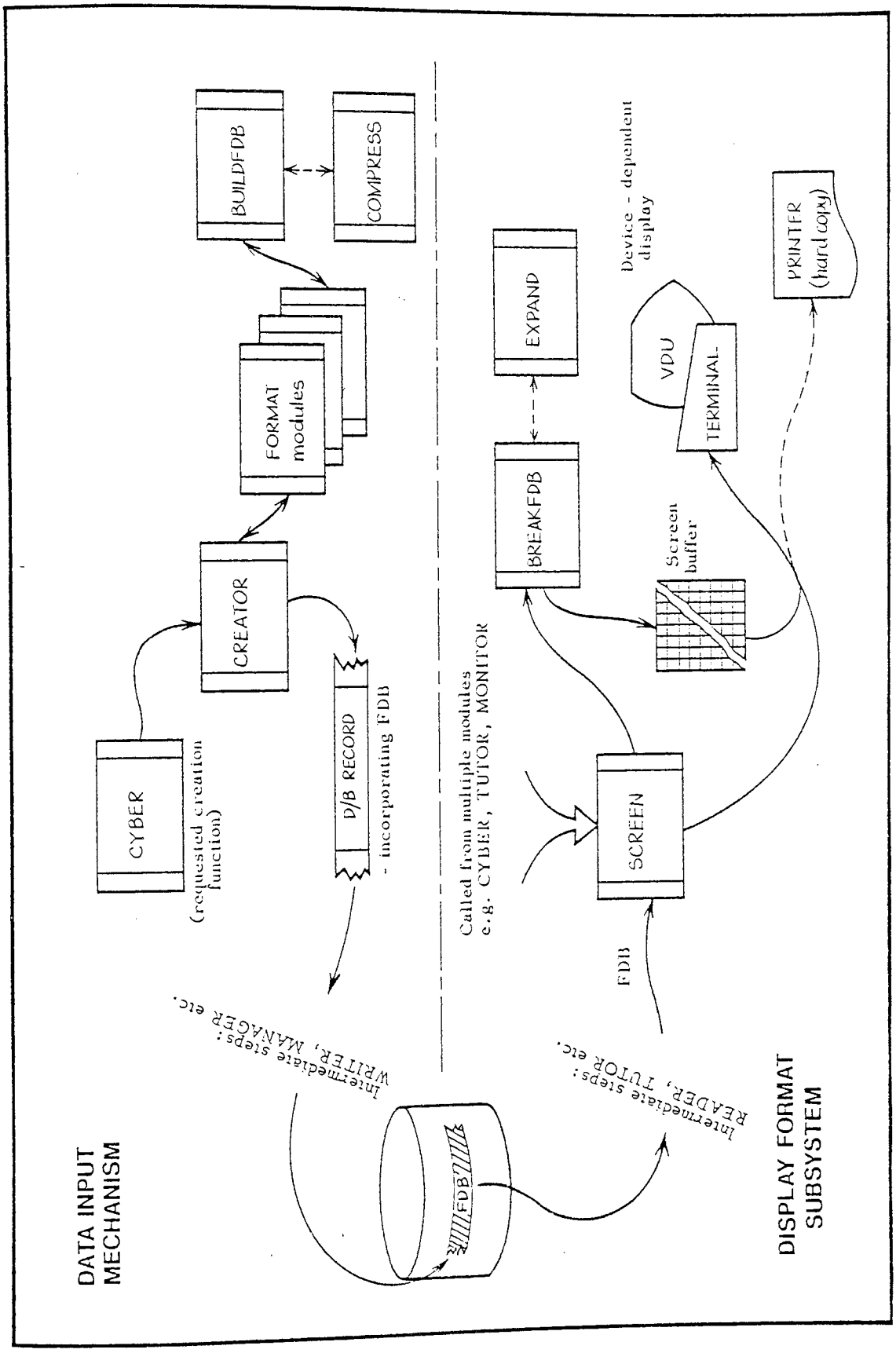
Fig 8.4 Conversion modules

N.B. Modules COMPRESS and EXPAND (Text Compression and expansion) are not always utilised - this is dependent on the type of text being processed.

8.7 DISPLAY FORMAT SUBSYSTEM STRUCTURE

It now remains to define the overall DFS structure, its relationship to the Data Input Subsystem (DIS) and the function of individual subsystem modules. Figure 8.5 should satisfy the first two points: see overleaf.

Fig 8.5 DFS/DIS relationship and component modules.



Notes on Fig 8.5:

(i) Database interface modules (i.e. WRITER, READER and MANAGER) are detailed in Chapter 3;

(ii) Full details of the Data Input Sybsystem (DIS) may be found in Chapter 7;

(iii) The organisation of modules EXPAND and COMPRESS, although part of both DFS and DIS, are discussed within Database Text compression/expansion techniques, Chapter 4.

8.7.1 DFS Modules

The constituent modules of DFS, and their functions are as follows:-

(a) SCREEN

Function: To output information to the terminal, optimised according to device type and to provide a simultaneous printer hard-copy if required.

Input-parameters:

(1) A(DISPLAY) - address of data to be displayed, held in eother:

(i) FDB format

(ii) non-FDB format - termed 'Information Block format' - This has the form:

Bytes 1/2 - length of text (max 1760)

Bytes 3... - text (uncompressed)

This latter form is commonly used for System messages as it involves no FDB-decoding, and is therefore slightly faster.

(2) DISPCODE - display code. This controls the way in which SCREEN interprets and displays the data. DISPCODE is a 1-byte binary field, individual bits being used as separate switches:

bit 0 - Erase Bit.

0 - no screen erase. display data on current screen.

1 - erase current screen contents (& store in backup buffer) prior to screening defined display.

N.B. The use of this bit is only necessary when displaying in non-FDB format.

bit 1 - FDB Bit. This may be set as follows:

0 - data in standard FDB format

1 - data in Information Block (non-FDB) format

bit 2 - Rescreen bit. Although this will work on printer terminals, it is only strictly useful on VDU's. The settings are:-

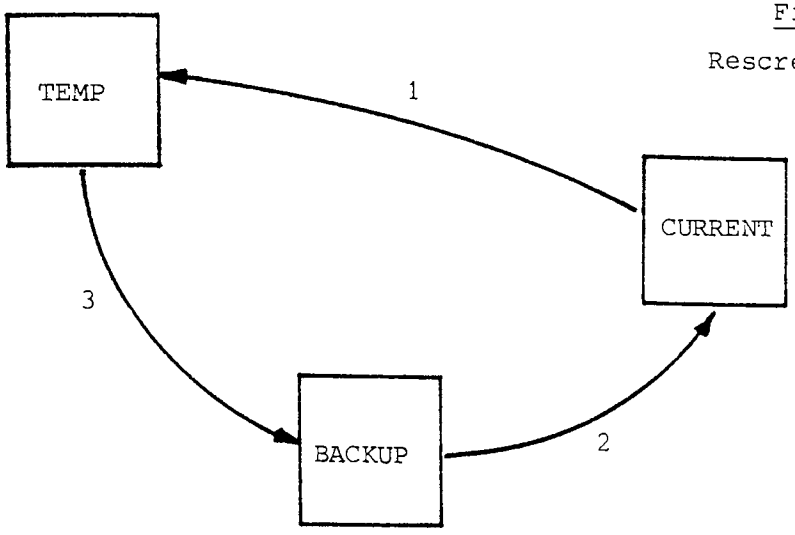
0 - normal SCREEN operation

1 - rescreen contents of backup, i.e. screen contents before last erase.

The rescreen function involves the use of backup and temporary screen buffers, and the current screen can be reinvoked. (See Fig 8.6 overleaf).

Fig. 8.6

Rescreen function



bits 3 thru 7 - not as yet used.

(3) HCFLAG - hard copy;

1 - byte field with settings;

00 - no hard-copy required

01 - produce hard-copy of all terminal I/O

(4) DEVTYPE - device type code; a 4 byte field set according to the active terminal type:

'GRAF' - VDU devices, typically IBM 3270's.

'TERM' - start/stop terminals.

Output parameters None

Notes: Further information on module SCREEN is detailed at Appendix 8.1

(b) BREAKFDB:

Function: To decode a database Frame Data Block into a terminal image buffer

Input parameters

A(FDB) - address of the FDB to be decoded.

Output parameters:

A(IMBUF) - address of the buffer containing the generated terminal image; normally returned via Register 1;

L - No. lines in generated display, including embedded blank lines.

X - Start line number of display. This is the line no. of the first non-blank line generated.

Logic BREAKFDB's logic hinges around the use of a Dope Vector to convert a line number L and column no. C (extracted from the FDB) into a relative displacement from the start of a Terminal Image Buffer.

Assuming that screen line numbers are in the range 0 - 21 and that column numbers are in the range 1 - 80, the displacement D of column C, line L may be calculated as:

$$D = L * 80 + (C - 1)$$
$$= 80 * L + C - 1$$

For further details on BREAKFDB, see Appendix 8.2

Appendix 8.3 illustrates SCHOOL session extracts on a non-3270 terminal.

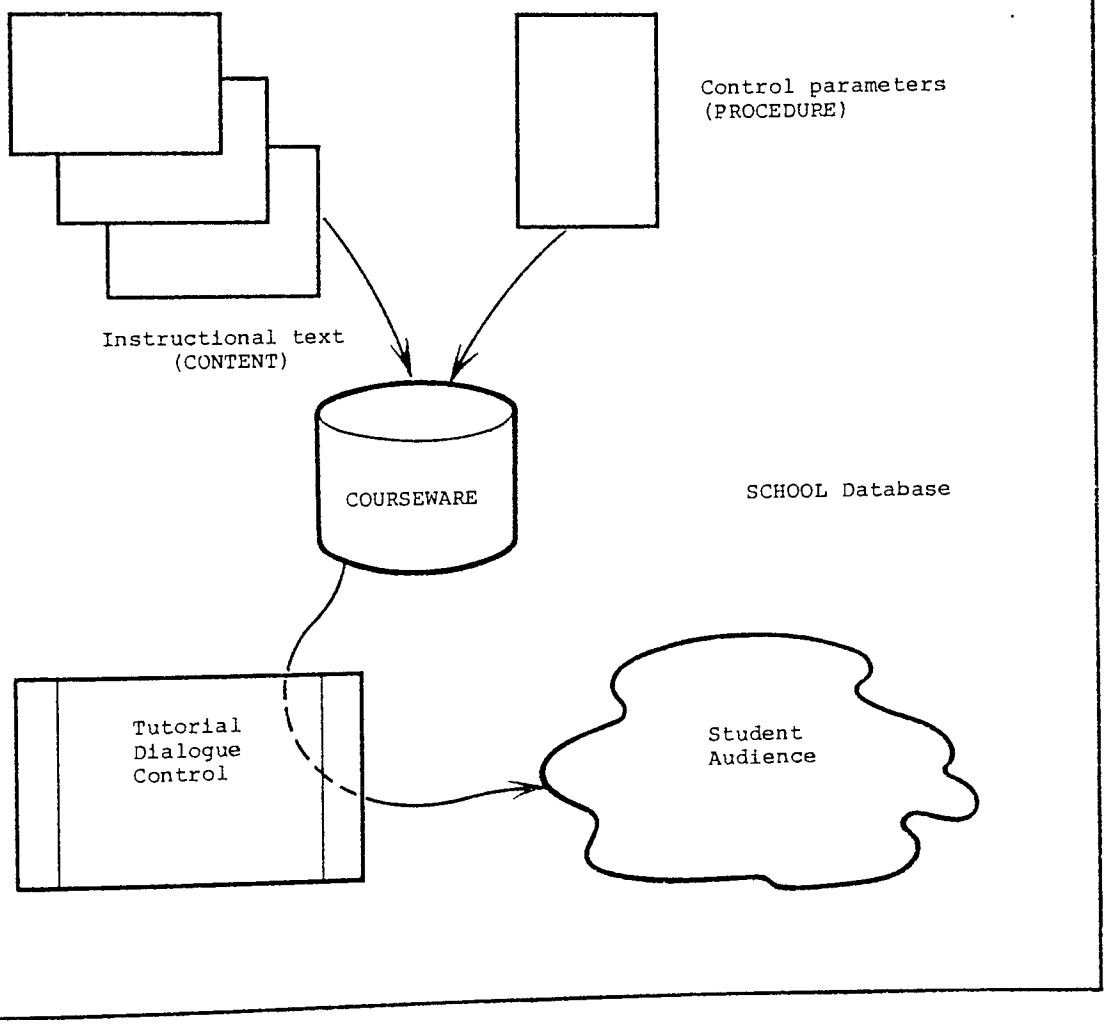
chapter nine

- TUTORIAL LOGIC CONTROL SUBSYSTEM

9.1 INTRODUCTION

The purpose of this Chapter is to discuss that part of SCHOOL which controls the 'tutorial dialogue'; Fig. 9.1 gives an overview of this:

Fig. 9.1 Tutorial component overview



Within SCHOOL this function is handled by the Tutorial Logic Control Subsystem, or TLCS.

9.2 TLCS FUNCTIONS

The Tutorial Logic Control Subsystem has six main functions:

(i) handling student registration;

(ii) processing SCHOOL Database courseware;

(iii) controlling the delivery of instructional text as prescribed by the Author;

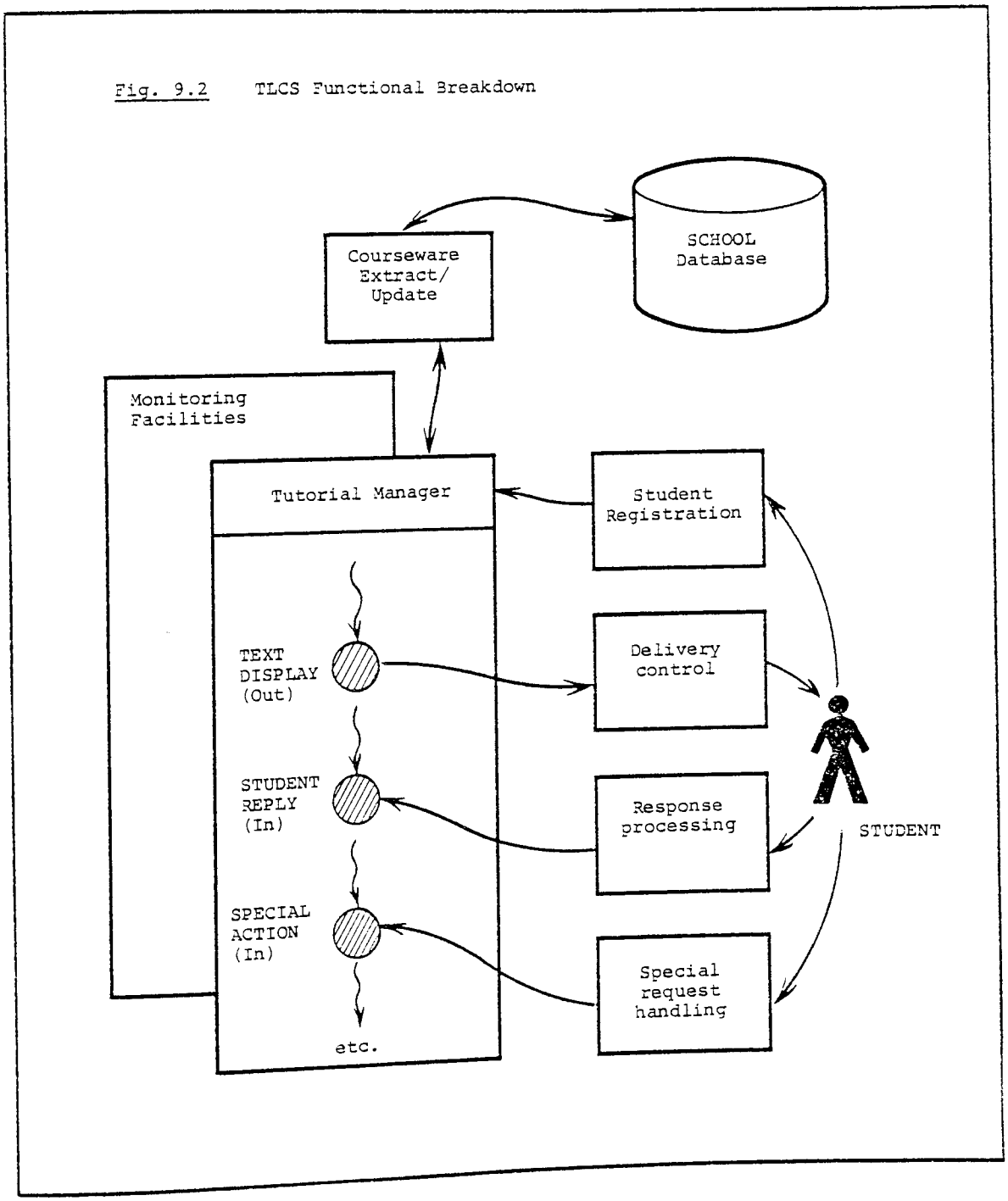
(iv) processing student reponses;

(v) providing and maintaining tutorial monitoring facilities;

(vi) controlling special student facilities;

The inter-relationship of these components is shown in Fig. 9.2:

Fig. 9.2 TLCS Functional Breakdown



Notes on Fig. 9.2:

(a) the Monitoring Facilities are effectively 'parallel' to the main processing of the Tutorial Manager, and are called as appropriate, depending on the monitoring features selected by the Lesson Author;

(b) several Service Function Subsystems are invoked by the various components of TLCS, i.e.

Courseware Extraction - DIOS

Delivery Control - DFS

Response Processing - KRES

Special Requests - ICES

(refer to Chapter 5, section 5.4.2 for an explanation of the above).

(c) Special Requests can be either:

(i) temporary interrupt - normally a request for information

or

(ii) permanent interrupt - a request for action which will have some effect on the session progress/sequence.

9.3 TLCS ORGANISATION AND LOGIC

The overall logic of TLCS can be summarised as in Fig. 9.3:

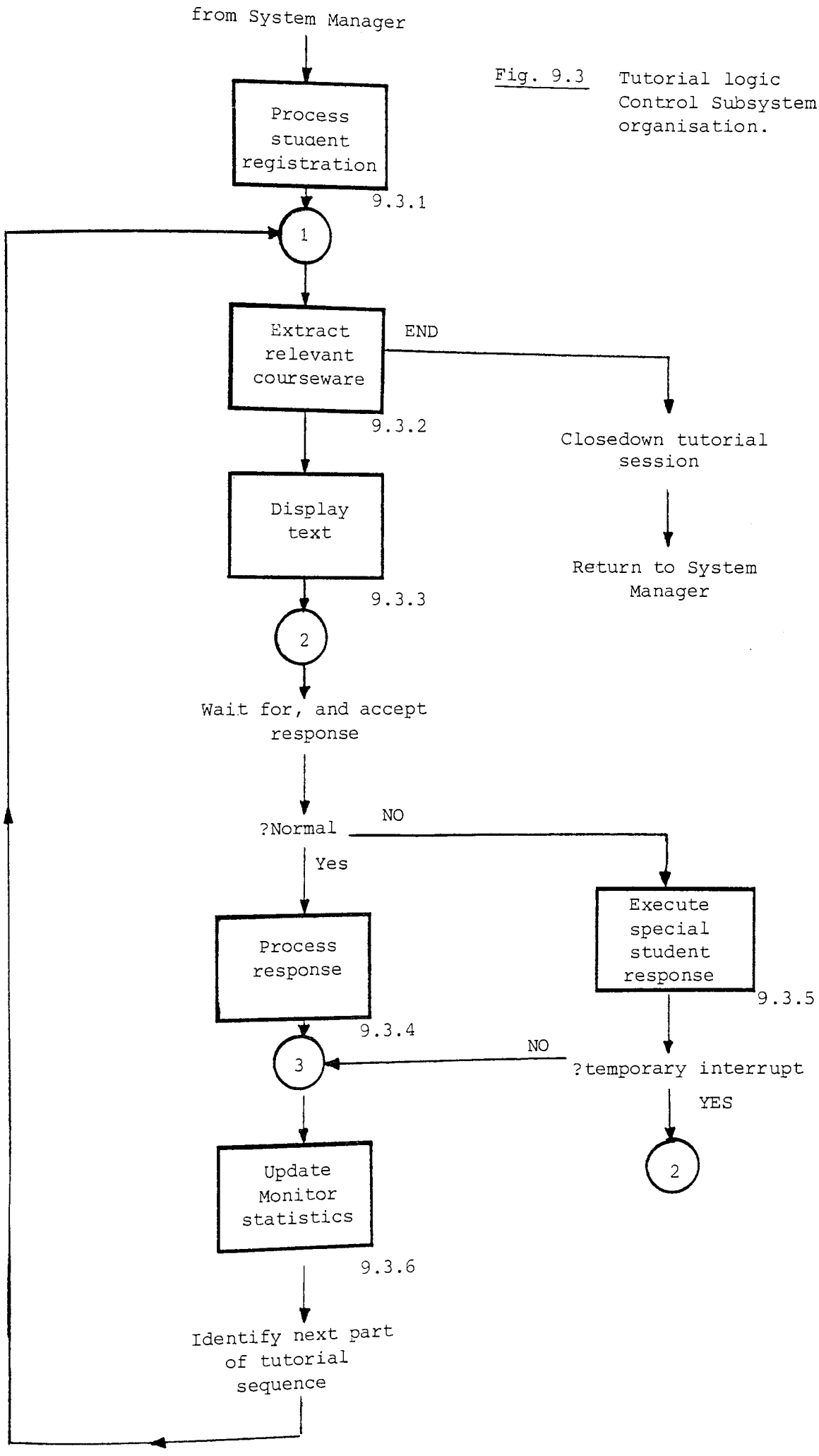


Fig. 9.3 Tutorial logic Control Subsystem organisation.

Within Fig. 9.3 the boxed items represent the six main functional components described in Section 9.2, annotated with the associated subsection number.

9.3.1 Student Registration

All students identify themselves to SCHOOL via 3 parameters:

(i) subject being studied;

(ii) name)
) both 15 characters long.
(iii) location)

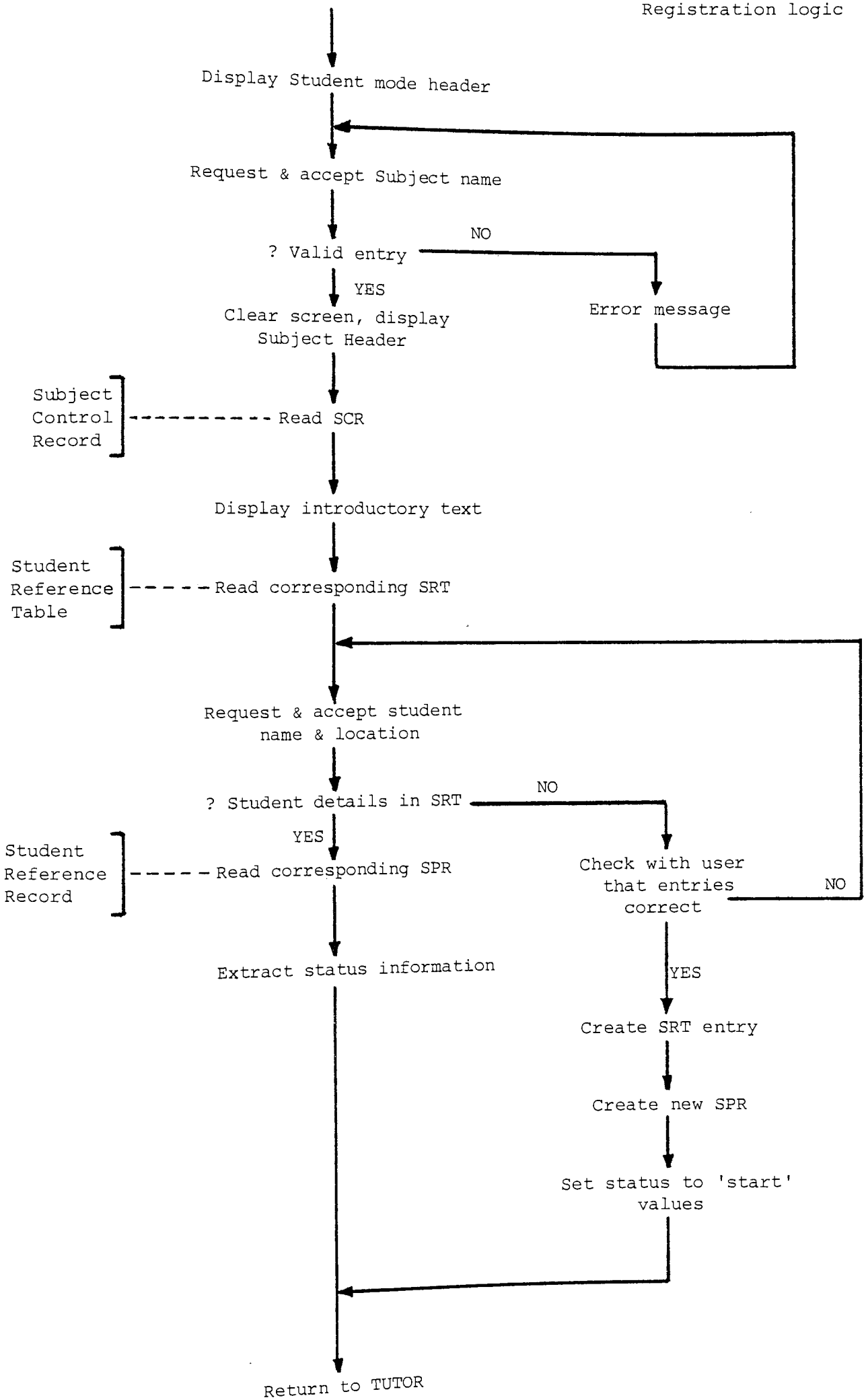
These are interpreted by the system into a unique 4-digit student id. XXYY (where XX is the subject reference number, and YY is the student number) which is used for internal identification purposes - in particular accessing student performance data, namely:

SRT - Student Reference Table

SPR - Student Performance Record

The outline logic of Student registration is illustrated by Fig. 9.4:

Fig. 9.4 Student Registration logic



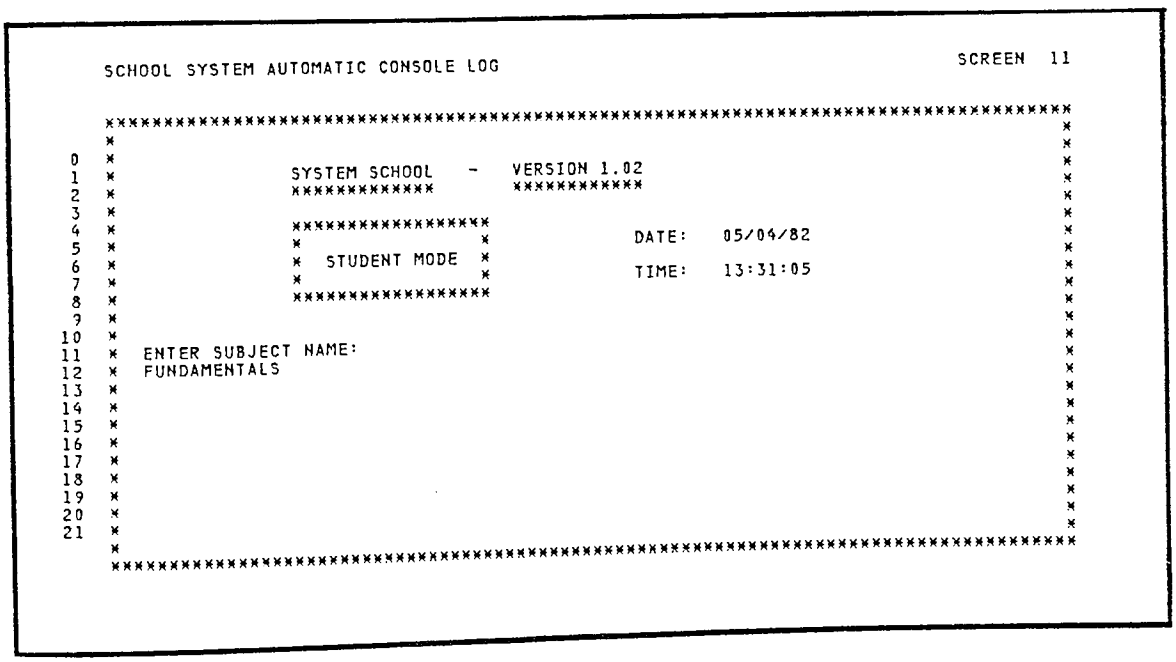
Notes on Fig. 9.4:

(i) Student status information (as extracted from the appropriate SPR, or newly-created) is held within the Student Status Block (SSBLOK). The format of this is identical to the SPR, plus Subject, Student and Location names;

(ii) Full details of SRT and SPR contents may be found at Appendix 3.1;

(iii) Some aspects of registration logic (e.g. checking for a full complement of students) have, for clarity, been omitted;

(iv) A typical registration sequence is as follows:-



```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  12

 0  *
 1  *                SYSTEM SCHOOL   -   VERSION 1.02
 2  *                **************      *************
 3  *
 4  *
 5  *                MODE:           STUDENT
 6  *                SUBJECT:        FUNDAMENTALS
 7  *
 8  *          THIS SUBJECT IS A SERIES OF 6 SHORT LESSONS, EACH AROUND
 9  *       15 MINS. LONG, DESIGNED TO INTRODUCE THE COMPUTER AND ITS
10  *       CAPABILITIES TO THE LAYMAN.
11  *
12  *
13  *   ENTER IDENTIFICATION:
14  *   NAME?
15  *   MIKE LITTLE
16  *   LOCATION?
17  *   COMPOWER
18  *
19  *
20  *
21  *
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  13

 0  *
 1  *          SYSTEM SCHOOL/1.02          SUBJECT:  FUNDAMENTALS
 2  *          ******************
 3  *
 4  *   SUBJECT RECORDS SHOW THAT YOUR STATUS IS:
 5  *
 6  *          NAME:               MIKE LITTLE
 7  *          LOCATION:           COMPOWER
 8  *          LESSON:             2
 9  *          FRAME:              6
10  *          TOTAL DURATION:     0:26   (HH:MM)
11  *          COMPLETION TYPE:    *SUSPEND
12  *
13  *   PRESS ENTER TO RESTART LESSON  2
14  *   OR ENTER '*RESUME' TO CONTINUE FROM FRAME  6:
15  *   ==>
16  *   *RESUME
17  *
18  *
19  *
20  *
21  *
```

9.3.2 Courseware Extraction

The mechanics of extracting course material from the SCHOOL database are fairly standard, i.e.

- establish the database address of the required logical record;
- pass this to DIOS, the Database Input/Output Subsystem;

- the appropriate contiguous logical record is then returned to the calling subsystem.

The important consideration is how these operations are controlled and sequenced. This is achieved via several tables resident in the Tutorial Status Control Block (TSCB). This contains:

(a) General status information:

- lesson number
- active scoring mode and passmark
- number of frames
- current frame number
- tutorial frame sequence (normal and remedial)

(b) Pointers for all relevant logical records:

- held as the Tutorial Pointer Table (TPT) and Frame Records Table (FRT). The TPT holds pointers to:

Subject Control Record	(SCR)
Student Reference Table	(SRT)*
Student Performance Record	(SPR)*
Lesson Control Record	(LCR)
Syntax Information Block	(SIB)
Lesson Analysis Record	(LAR)*
Glossary Reference Table	(GRT)*

N.B.* indicates those records which may be updated during a student session.

The FRT holds pointers to all frame logical records (i.e. Question, Hint, Answer and Comment) within the active lesson. This information is extracted from the corresponding Lesson Control Record.

(c) Frame Analysis Table (FAT):

- stores information on the performance of each frame within the lesson (if lesson analysis has been invoked). This information is extracted from the corresponding LAR.

(d) Syntax Definition Table (SDT):

- defines the syntax processing to be carried out on incoming student replies. This is a complete copy of the corresponding SIB logical record, if this exists.

The Tutorial Status Control Block is part of the TLCS Data Nucleus, along with an area used as the Frame Record Buffer. Logical records extracted by DIOS are transferred into this area for subsequent display and processing.

N.B. For an accurate definition of TSCB structure, see Appendix 9.1.

9.3.3 Text Display

Text displayed to a student during a tutorial session includes:

- question or frame text;
- additional (hint) information;
- comments on a student response;
- information returned in answer to a special request.

Actual display is under the control of DFS (Display Format Subsystem) in the normal way, although certain further considerations can apply:

(i) the Author can force a screen clear for each frame;

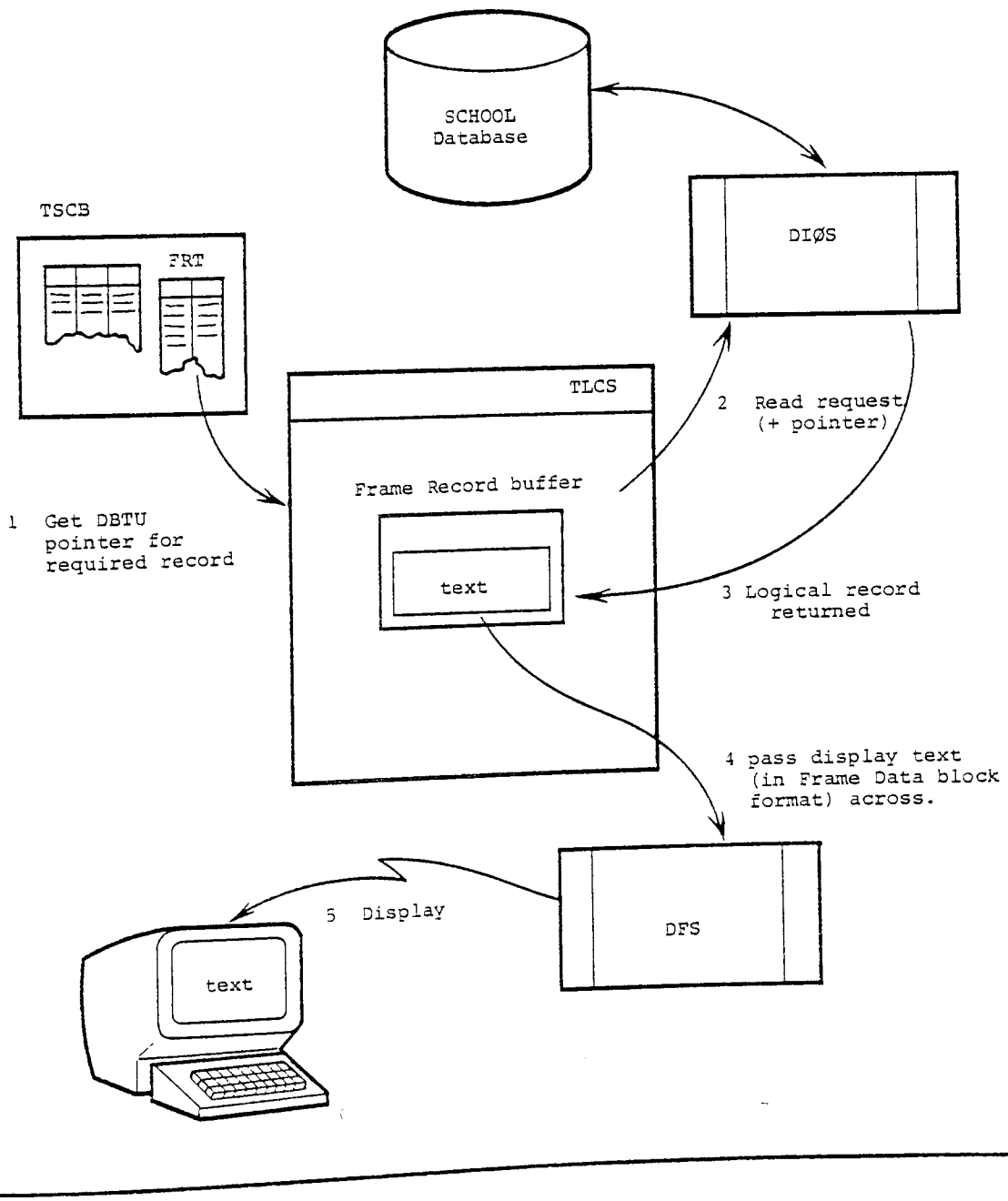
(ii) TLCS will subsequently attempt to fit all related material on the same screen;

(iii) special request responses can be either:

- on the same screen if appropriate (e.g. hint information);
- on a separate screen to avoid breaking session 'flow'.

Fig. 9.5 shows the sequence of events producing a display at the student's terminal:

Fig. 9.5 Tutorial text display sequence



Some samples of text displayed during test sessions follow:

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  31

0  *   SUBJECT:  SCHOOL TESTING     LESSON:   2/SCHOOL DATABASE     FRAME:   8
1  *
2  *    THE STRUCTURE OF THE SCHOOL DATABASE ROOT SEGMENT IS AS BELOW:
3  *
4  *                          ********************
5  *                          *  MASTER CONTROL  *
6  *                          *      RECORD      *
7  *                          ********************
8  *                                    |
9  *                  ------------------------------------------
10 *                  |                 |                       |
11 *                  |                 |                       |
12 *
13 *    ****************        ****************        ****************
14 *    *   BIT MAP    *        *    SUBJECT     *        *    GLOBAL      *
15 *    *    TABLE     *        *    SEGMENTS    *        *   MESSAGES     *
16 *    ****************        ****************        ****************
17 *
18 *
19 *                                /PRESS ENTER TO CONTINUE
20 *
21 *
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  35

0  *   SUBJECT:  FIRST AID          LESSON:   1/TYPES OF INJURY     FRAME:   2
1  *
2  *
3  *
4  *
5  *            BURNS AND SCALDS
6  *            ----------------
7  *
8  *            2 TYPES OF INJURY:
9  *
10 *           (1) SUPERFICIAL - ONLY THE OUTER LAYERS OF SKIN
11 *                             ARE DAMAGED;
12 *           (2) DEEP - THE WHOLE THICKNESS OF THE SKIN, INCLUDING
13 *                      THE NERVE ENDINGS, IS DESTROYED.
14 *
15 *           AN EXTENSIVE SUPERFICIAL BURN OR SCALD IS MORE
16 *           PAINFUL THAN A SMALL DEEP BURN.
17 *
18 *                                        /PRESS ENTER TO CONTINUE
19 *
20 *
21 *
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                  SCREEN  46

0  *   SUBJECT:  FIRST AID          LESSON:   2/SIGNS & SYMPTOMS    FRAME:   2
1  *
2  *                BURNS AND SCALDS:  SIGNS AND SYMPTOMS
3  *                -------------------------------------
4  *
5  *                PAIN - MAY BE INTENSE, ESPECIALLY WITH SUPERFICIAL
6  *                       BURNS;
7  *
8  *                - REDNESS AND LATER SWELLING AND SOMETIMES BLISTERING,
9  *                    IN SEVERE CASES, EVEN CHARRING;
10 *
11 *                - SHOCK; THERE IS A GREAT DANGER FROM SHOCK WHICH IS
12 *                         DIRECTLY RELATED TO THE EXTENT OF THE INJURY
13 *                         AND INCREASES RAPIDLY WITH THE LOSS OF FLUID
14 *                         OOZING FROM THE BURNT SURFACE, AND THE ESCAPE
15 *                         OF BLOOD OR PLASMA INTO THE TISSUES CAUSING
16 *                         SWELLING.
17 *                                        /PRESS ENTER TO PROCEED
18 *
19 *
20 *
21 *
```

9.3.4 Response Processing

Student input at any stage after registration can be either:

(a) a normal reply to a prompt or question;

or

(b) a special facility request (see subsection 9.3.5).

Normal replies can vary in format depending on the Author's requirements - e.g. a null entry (where no actual data is needed to proceed) or a character string (to be matched against author - defined entries).

To facilitate matching, a system of answer-string pre-processing is available to the author at 2 distinct levels.

(i) Trivial formatting:

- delete common punctuation;
- remove leading, trailing and non-significant blanks;

(this does not require the presence of a Syntax Information Block (SIB) within the database Lesson segment).

(ii) Lexical analysis:

- as defined by the corresponding SIB. This includes:

  Removing
  - leading, trailing, non-significant or all spaces;
  - common punctuation;
  - leading zeros;
  - redundant brackets;
  - listed unwanted characters;
  - listed unwanted words.

Converting - upper case to lower;

- lower case to upper;

Substituting - defined characters for others.

These preprocessing steps are built-in features of KRES, the Keyboard Response Evaluation subsystem, and are activated via a system of Lexical Analysis Flags (LEXFLAGS) - this is fully described in Chapter 10, subsection 10.3.1. The active requirements for the current session are held in the TSCB Syntax Definition Table (SDT), created from the corresponding SIB when the lesson is invoked.

The lexically processed character string returned by KRES is then matched against an author-defined range of answers, within three distinct categories:

(i) most commonly expected correct reply:

- 'Prime' answer;

(ii) seven other valid, but less common correct replies:

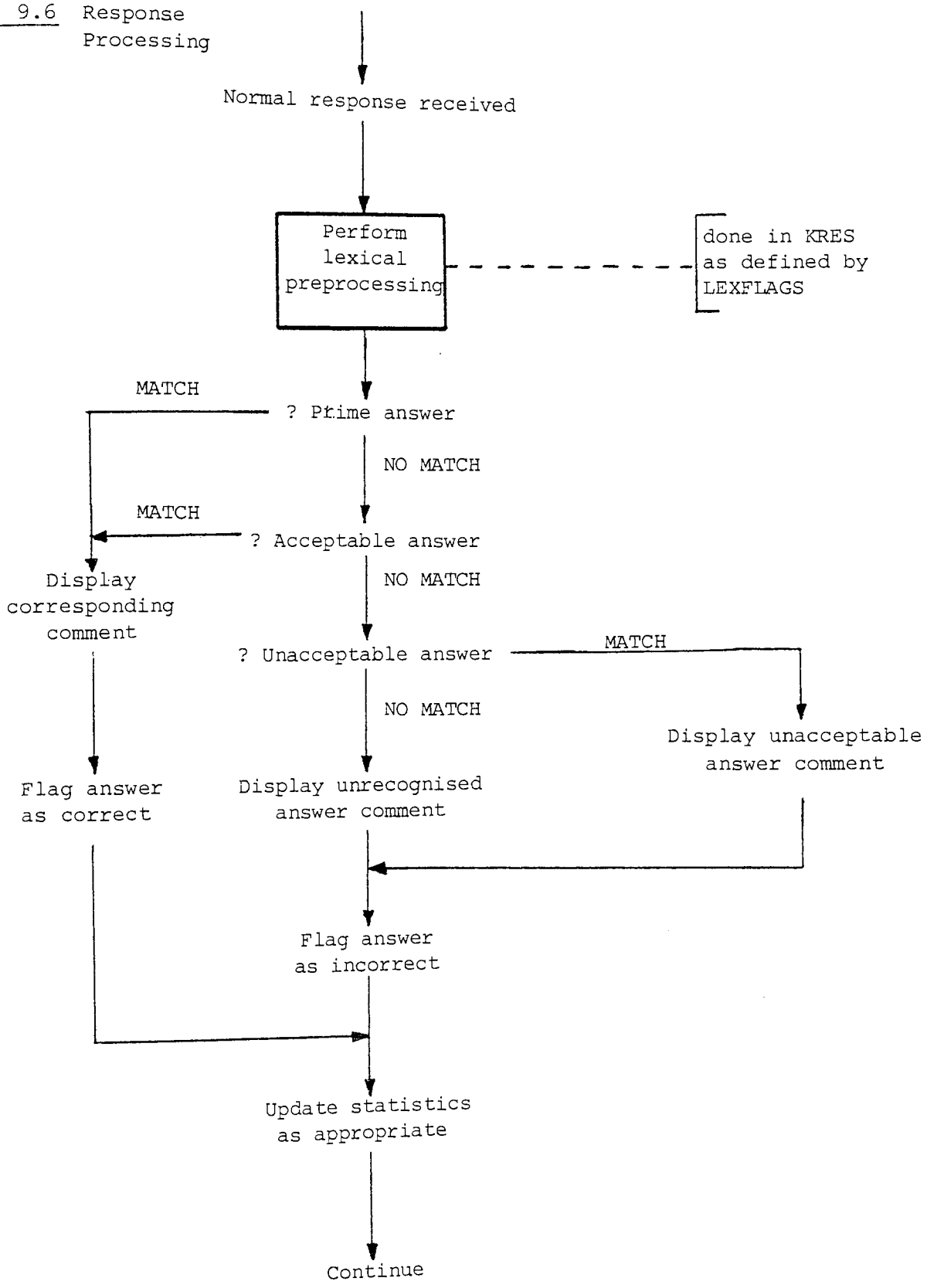
- 'Acceptable' answers;

(iii) four expected, common, but incorrect replies:

- 'Non-acceptable' answers.

Anything other than these is considered 'Unrecognised'. The twelve 'Recognised' answers can be cross-matched to a selection of suitable comments, and the complete procedure then becomes as shown in Fig. 9.6:

Fig. 9.6 Response Processing



N.B. If the author has not explicitly defined a range of comments, SCHOOL will automatically generate suitable alternatives, selected randomly from a list of similar phrases.

Answer matching can itself be done in different ways:

(i) character-string matching:

- lexically processed student input is compared character-by-character with author entries (as extracted from the appropriate database Primary Answer Record (PAR)).

(ii) keyboard structure matching:

- student input is compared to a 'structure' of keywords and variable words. This allows the Author to define:
  - (a) a range of answer-string keywords and
  - (b) the relationships between them without having to explicitly state all valid combinations.

Keywords are defined by the Author on a Syntax Information author document and subsequently within an SIB logical record, relationships between them being defined on each corresponding Answer Definition author document and database PAR. Fig. 9.7 illustrates the types of relational definitions available:

Fig. 9.7 Keyword relational operators

OPERATOR	SYMBOL	EXAMPLES	COMMENTS
AND	+	LONDON + EDINBURGH	Any order acceptable.
ADJACENT	.	NATIONAL.COAL.BOARD	Must be in order specified separated by only spaces.
SYNONYM	=	CAR=AUTOMOBILE	Either acceptable.
OR	|	THIS|THAT	Either acceptable, can apply to phrases

OPERATOR	SYMBOL	EXAMPLES	COMMENTS
NOT	¬	¬SECTION	Answer will be incorrect if defined word present.
Exclusive OR	#	CAT#DOG#PIG	Only one is acceptable.
THEN	;	JAN;FEB;MAR	Words must appear in defined order, can have text between keywords which will be ignored.

Consider the following examples:

Example 1

Q: Name the capitals of the 4 countries of the United Kingdom.

Answer definition:

KW/LONDON+EDINBURGH+CARDIFF+BELFAST

where KW/indicates that the answer definition is in keyword notation.

N.B. This answer can have 24 valid combinations.

Example 2

Q: Name the 4 largest countries in the world in descending order.

Answer definition:

KW/USSR=RUSSIA; CANADA; CHINA; USA=US=UNITED STATES

Example 3

Q: In COBOL, how could a variable WSUB be set to zero (other than by using the VALUE clause)?

Keywords (on Syntax Information document):

ZERO ZEROS ZEROES 0 +0

Answer definitions:

Prime: KW/MOVE.KW*.TO.WSUB..

Acceptable: KW/COMPUTE.WSUB.==.KW*..

N.B. KW* means 'any keyword entry'

Duplicate operators signify that the specific character (. or =) appears in the answer string.

For a full description of the use of keyword Notation and its possibilities, refer to the SCHOOL Author Manual.

9.3.5 Special Request Handling

Because of difficulty in differentiating between normal student responses and special requests, the convention adopted has been to use the Immediate Command concept, whereby any input string preceded by an asterisk is interpreted as a special requirement and is processed immediately. A full description of this can be found at Chapter 11.

The interruption to the normal sequence of events can be one of two types:

(i) transient - the student's request is satisfied, and TLCS then returns to await the original response;

(ii) permanent - the student's request is satisfied and has some effect on the session sequence;

The facilities available to the student are as listed in Fig. 9.8:

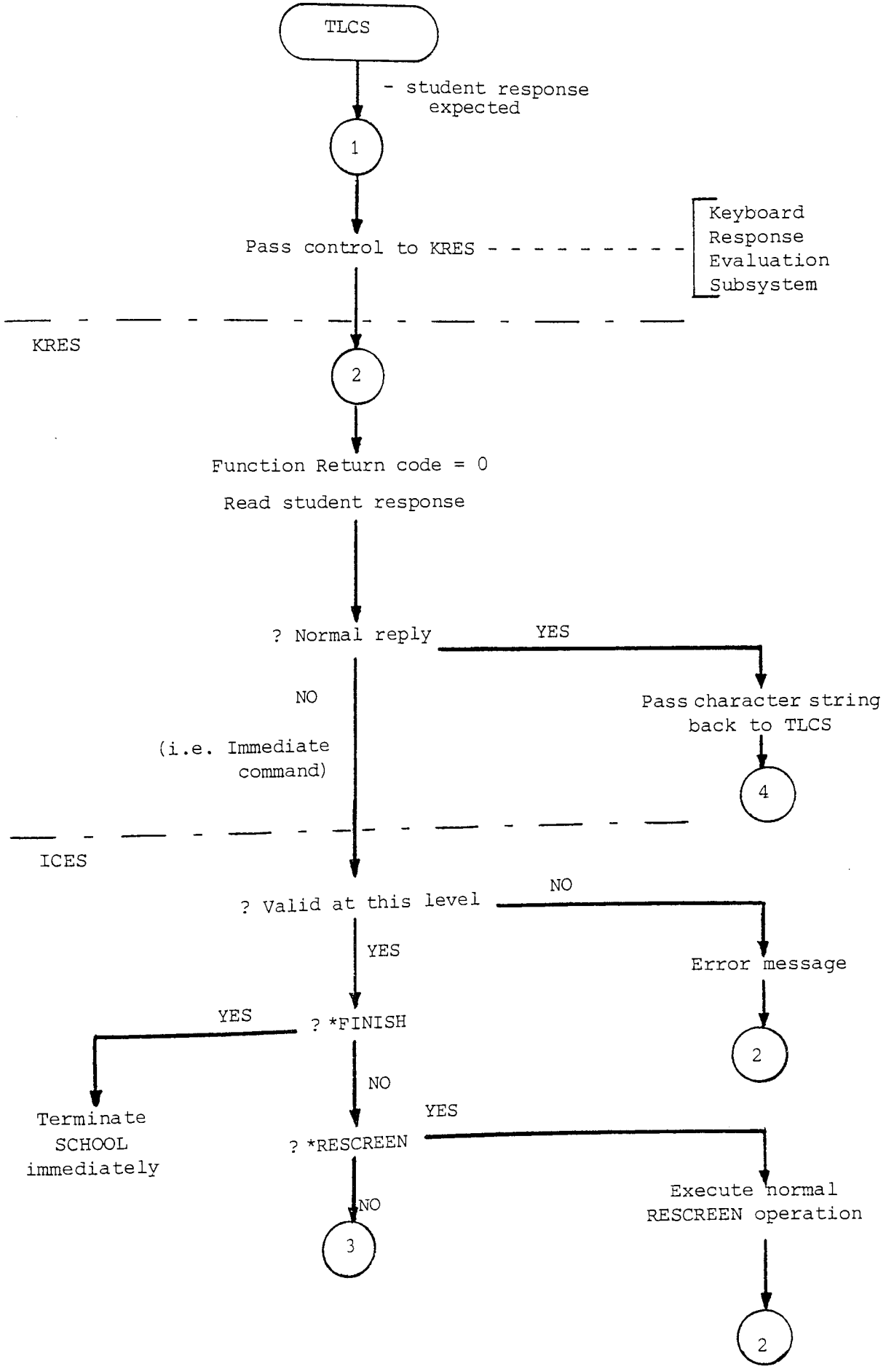
Type	Command	Code (Hex)
Transient	*AUTHOR	10
	*COMMENT	11
	*CURRENT	12
	*DETAILS	13
	*GLOSSARY	14
	*LIST	15
	*REPEAT	16
	*RESCREEN	17
	*SCORE	18
Permanent	*ABANDON	41
	*ANSWER	42
	*FINISH	43
	*HINT	44
	*MESSAGES	45
	*RESUME	46
	*SUSPEND	47

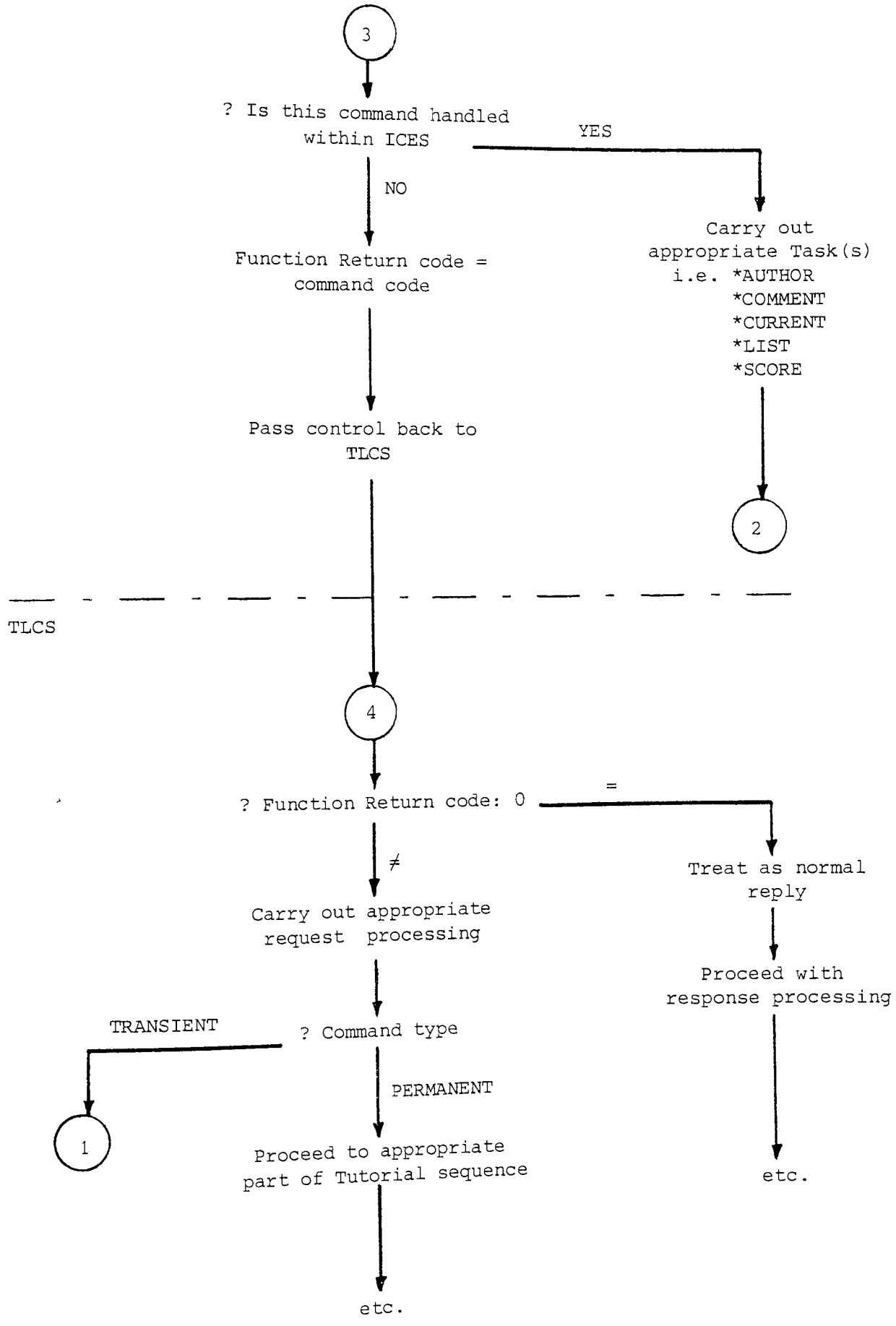
Fig. 9.8
Special student facilities

A full description of the function of each of the above may be found in Chapter 11, Section 11.4. In the interests of brevity, these have not been included here.

Module PLEASIR and the Immediate Command Execution Subsystem (ICES) are responsible for processing these requests, and Fig. 9.9 illustrates the associated logic:

Fig. 9.9 Student special request logic.




3
? Is this command handled within ICES
YES
NO
Carry out appropriate Task(s) i.e. *AUTHOR *COMMENT *CURRENT *LIST *SCORE
Function Return code = command code
Pass control back to TLCS
2
TLCS
4
? Function Return code: 0
=
≠
Treat as normal reply
Carry out appropriate request processing
Proceed with response processing
TRANSIENT
? Command type
PERMANENT
1
Proceed to appropriate part of Tutorial sequence
etc.
etc.

9.3.6 Statistics Monitoring and Update

Two types of statistics are monitored by TLCS:

(i) student performance;

(ii) courseware performance, this being further broken down into 2 optional components:

- lesson frame analysis;
- glossary request analysis.

Student performance statistics are maintained within the Student Status Block (SSBLOK) plus other variables, and are subdivided into 2 types of information:

(i) ACTIVE
- session start time;
- time of last student entry;
- current position (lesson/frame numbers);
- active hint number;
- number of attempts at answer;
- marks obtainable on this frame;
- total marks obtained in session so far.

(ii) HISTORICAL
- subject message status;
- start and latest dates;
- accumulated session durations (including the current one);
- marks/durations etc. for all previous sessions.

This information is being constantly updated as the session proceeds, for instance:

- each time a student response is received, the time of day (HH:MM:SS) is stored;
- each time a Hint is requested, the maximum mark obtainable on this frame is adjusted and the active Hint number re-assigned;

- at the end of each Frame, the Historical data is updated from the Active values (total duration, completion code, lesson/frame numbers, marks, duration, active scoring mode) and the appropriate SPR (Student Performance Record) is rewritten. Active values are then reset and control passes to the appropriate part of the session, as defined by TSCB parameters.

Lesson Analysis statistics are only collected if the author has requested this - notified to TLCS by having a valid Lesson Analysis Record (LAR) pointer. All relevant data is held and maintained within the Frame Analysis Table (FAT) part of the TSCB. Each frame is monitored according to:

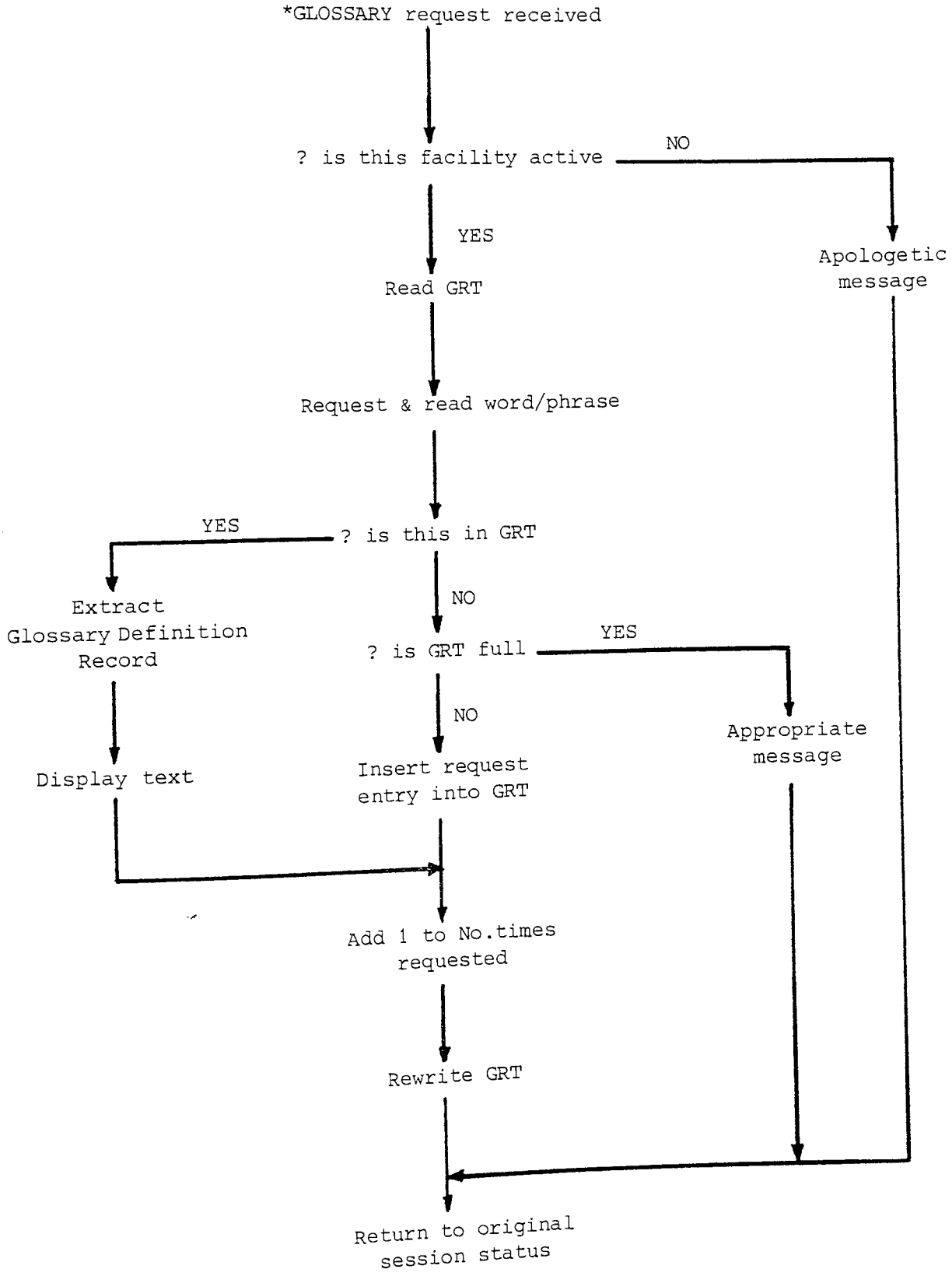
- number of times used;
- number of attempts at an answer;
- number of incorrect attempts;
- number of hint requests;
- number of answer requests;
- total marks obtained.

As a frame is completed, all FAT values are updated, and the appropriate database LAR is then rewritten.

N.B. Rewriting both SPR and LAR logical records at the end of each frame ensures that any host system failure does not involve the student repeating a complete lesson.

Glossary Analysis is also an author-option, and any glossary request emanating from a student is processed as in Fig 9.10:

Fig. 9.10 Glossary Request Processing.



A typical Glossary request sequence is shown below:

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN  27

 0  *   SUBJECT:  LOTUS CARS          LESSON:   2/ELANS & EUROPAS     FRAME:   8
 1  *
 2  *      BOTH ELAN AND EUROPA RANGES USED A FRONT SUSPENSION SYSTEM
 3  *   BASED ON UNEQUAL LENGTH WISHBONES, COIL SPRINGS AND TELESCOPIC
 4  *   DAMPERS.
 5  *
 6  *      THIS WAS MANUFACTURED BY ALFORD AND ALDER AND WAS USED ON
 7  *   A COMMON BRITISH SALOON CAR OF THE SAME PERIOD,
 8  *
 9  *                          - WHICH CAR WAS THIS?
10  *  *GLOSSARY
11  *
12  *
13  *
14  *
15  *
16  *
17  *
18  *
19  *
20  *
21  *
```

```
SCHOOL SYSTEM AUTOMATIC CONSOLE LOG                                    SCREEN  28

 0  *   SUBJECT:  LOTUS CARS          LESSON:   2/ELANS & EUROPAS
 1  *
 2  *  ********** G L O S S A R Y   F A C I L I T Y **********
 3  *
 4  *  ENTER REQUIRED WORD/PHRASE:
 5  *  WISHBONE
 6  *
 7  *  EXPLANATION:          WISHBONE
 8  *
 9  *  A SUSPENSION LOCATION SYSTEM WHEREBY THE WHEEL HUB UPRIGHT IS
10  *  LOCATED BY TWIN CONVERGING (USUALLY HORIZONTAL) STRUTS.
11  *
12  *
13  *
14  *                                        /PRESS ENTER TO CONTINUE
15  *
16  *
17  *
18  *
19  *
20  *
21  *
```

9.4 TUTORIAL LOGIC CONTROL SUBSYSTEM - SOFTWARE

TLCS consists of seven modules, with responsibilities as described in Fig. 9.11:

Module	Function
TUTOR	Overall subsystem control
REGISTER	Student sign-on and registration
LOOKUP	Courseware access and extraction
BLAKBORD	Text display control
CHECKER	Student Response processing
PLEASIR	Processing special requests not capable of being handled by ICES.
MARKER	Performance monitoring control (student and courseware).

Fig. 9.11 TLCS Modules

For further details of the above, refer to Appendix 9.2.

chapter ten

- KEYBOARD RESPONSE EVALUATION SUBSYSTEM

10.1 INTRODUCTION

All complex on-line systems have many different instances where information is read from the Terminal keyboard; often using entirely different components in the System coding.

The situation is further complicated when each Read is extended and enhanced with special functions:

- intercept non-standard replies;
- perform lexical analysis on the input stream;

To alleviate this situation, SCHOOL was structured to provide a standard Keyboard/System interface, termed the Keyboard Response Evaluation Subsystem (KRES). This provides:

(a) a single, standard method of reading information from the keyboard and delivering the input data to any module of the SCHOOL system;

(b) the ability to perform a wide range of lexical analysis functions on the input character string;

(c) the ability to intercept non-standard messages, react accordingly and then return to satisfying the original pending request;

Incorporating a Keyboard Monitor of this type permits a wider range of input control facilities than is normally found within existing CAI Systems.

For instance:

(a) Authors can select from a spectrum of lexical analysis procedures:

- remove leading spaces
- remove trailing spaces
- remove all spaces

- remove all common punctuation
- convert answer to upper or lower case
- remove non-significant leading zeros
- remove redundant parentheses
- remove specified characters

The coding to achieve the above can be standardised and then held within 1 module, as opposed to being duplicated wherever required;

(b) As all normal System input goes through the KRES interface, all stages of system execution may use the Lexical Analysis facilities provided - e.g. replies to System requests, Author/Supervisor control commands, etc;

(c) Again, as KRES constitutes the only 'door' into SCHOOL, non-standard input messages are immediately interceptable. Typical of the facilities that this can provide are:

- requests for assistance/answers
- requests for information
- execution suspension/resumption
- ability to revise earlier work without losing the current 'place' .
- requests for glossary definitions.

This facility is termed the Immediate Command Execution subsystem (ICES) and complete details of its scope may be found in Chapter 11.

10.2 KEYBOARD RESPONSE EVALUATION SUBSYSTEM - STRUCTURE

All normal SCHOOL terminal input, no matter what its function or module/subsystem initiating the request, is processed via the Keyboard Response Evaluation Subsystem (KRES). The relationship between KRES and the Display Format Subsystem (DFS) can be seen at Fig 10.1:

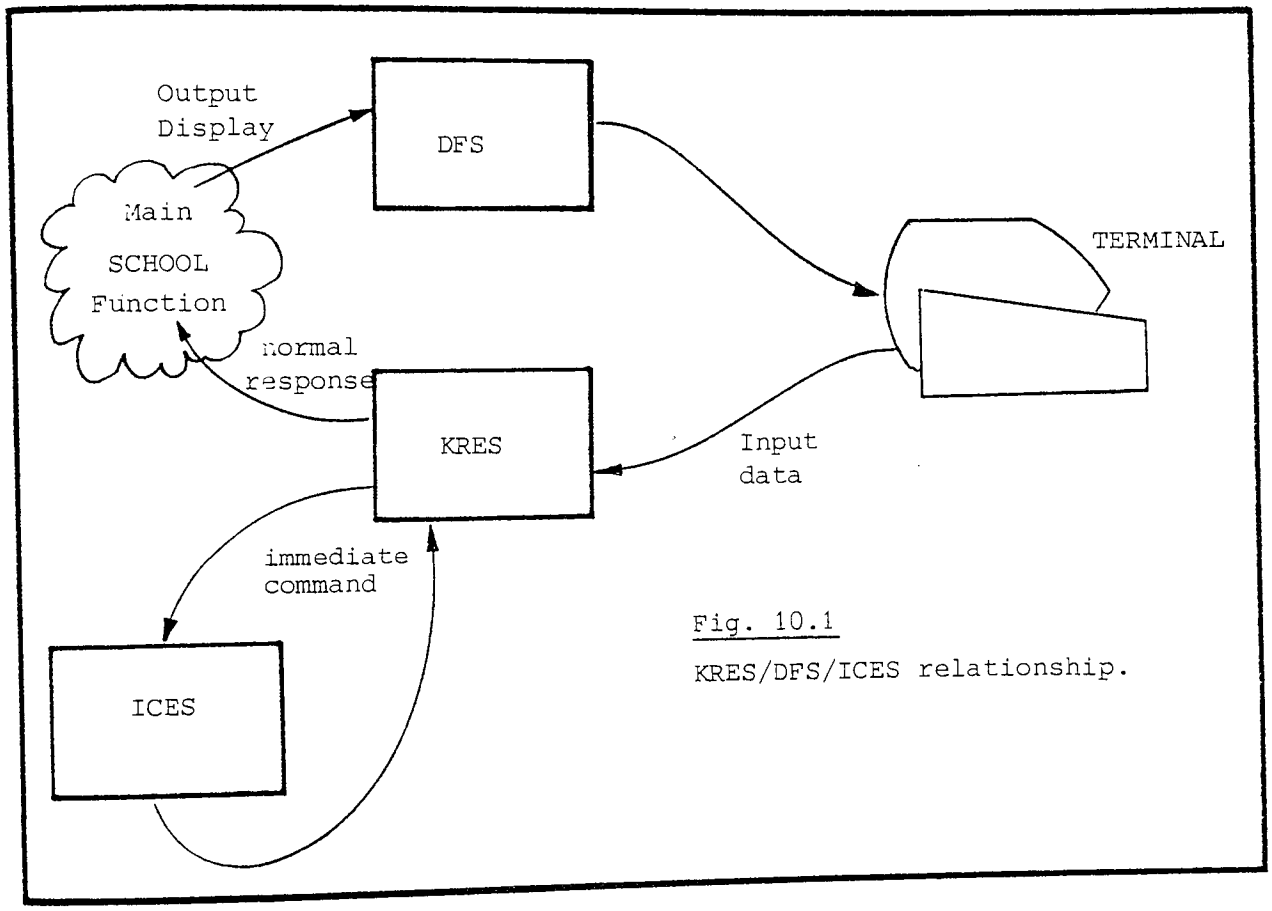


Fig. 10.1

KRES/DFS/ICES relationship.

KRES has in fact two functions, depending on the incoming information:

(i) Normal replies - returned (after lexical analysis) to the calling module;

(ii) Immediate commands - identified and passed intact to the Immediate Command Execution Subsystem. This decodes and executes the command. Once completed, control returns to KRES which then re-awaits the originally requested reply;

N.B. The convention adopted to differentiate between normal replies and immediate commands is that the latter are always prefixed with an asterisk, e.g.

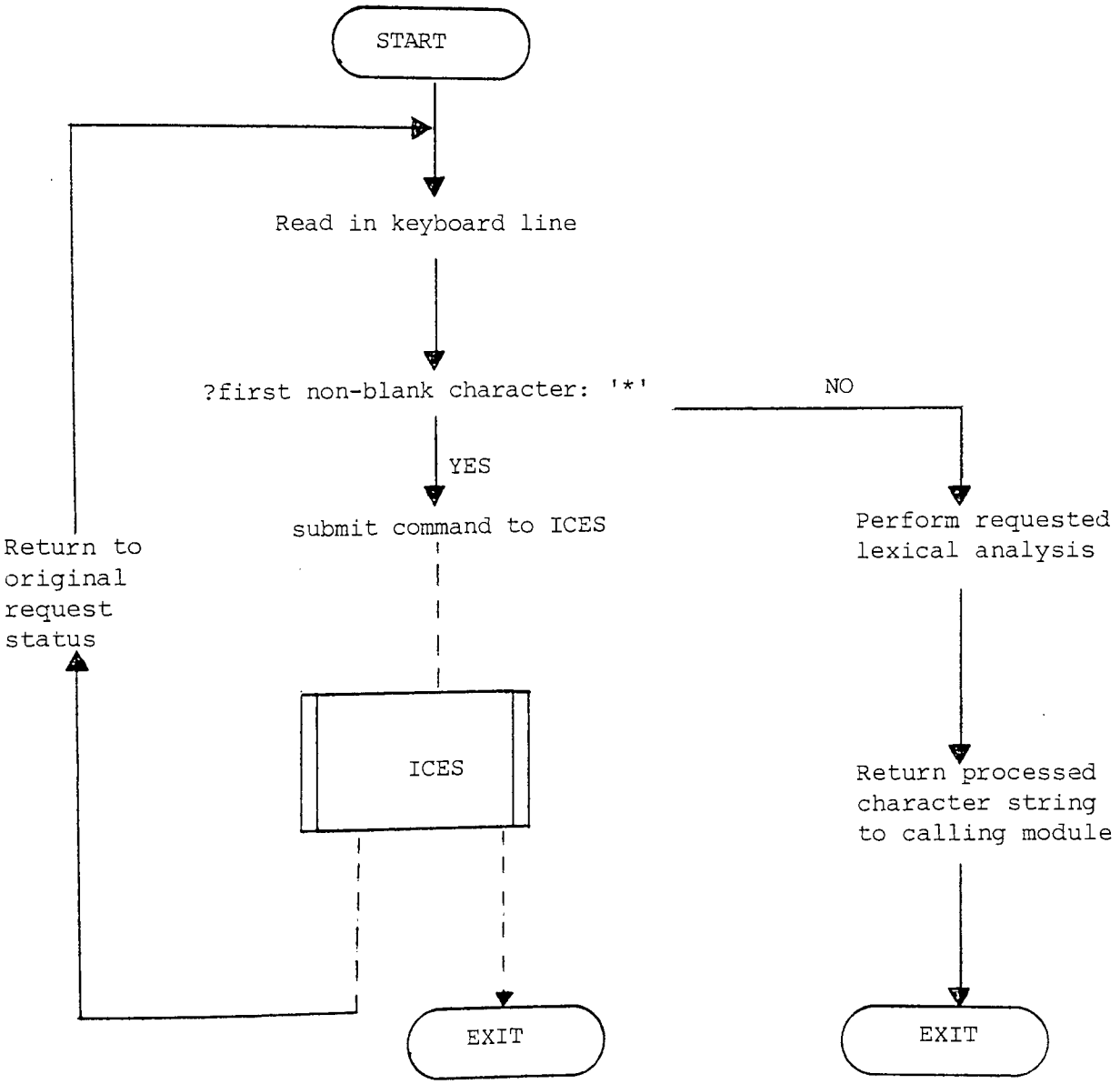
REPEAT - normal reply

*REPEAT - immediate command to repeat an earlier frame.

10.3 MODULE MONITOR

One module, MONITOR, performs all functions of KRES plus the main control function of the Immediate Command Execution Subsystems. It can be called from any point within SCHOOL and its outline logic is as follows:

Fig. 10.2 KRES functional components



N.B. In certain instances Immediate commands terminate system execution, and as such no return to KRES takes place.

10.3.1 Module MONITOR Input parameters

MONITOR has in fact 5 input parameters, all of which are resident in module CYBER's Data Nucleus (see Chapter 5). These are:

ANSWER - an initially blank field into which the incoming reply is inserted (after lexical analysis). Maximum length 130 characters.

LANSL - logical answer length (1 byte, binary). This can have two initial values:

- 0 - take length from processed character string
- non 0 - truncate processed answer string at this length.

FALEVEL - function authorisation level
SPLEVEL - system progress level

) see ICES, Chapter 11

LEXFLAGS - lexical analysis flags. (2 bytes, binary) Each of the 16 bits in this field initiate a particular analysis function if set to 1. Bit allocations are:-

Bit No.	Significance when set
0	Remove leading spaces
1	Remove non-delimiting spaces, e.g. THE␣␣␣CAT␣␣␣ SAT␣␣.␣␣ → THE␣CAT␣SAT␣.␣

2 Remove all spaces

3 Remove trailing spaces

4 Remove all common punctuation, i.e.

, . : ; " ! ' ?

5 Suppress automatic conversion of input into upper case (i.e. a 'mixture' such as 'The Answer.....' will remain unaltered)

6 Convert all alphabetic characters to lower case

7 Switch off Immediate Command facility (i.e. '*' as the first non-blank character is treated as normal input)

8 Remove non-significant leading zeros in any numeric string, e.g.

0010.03 - 10.03

0.004 - unaffected

0 - unaffected

9 Remove redundant parentheses in any algebraic expression, e.g.

Y = -(X + (3) + ((Y)))

becomes

Y = X + 3 + Y

10 Ignore characters specified by the author in the Redundancy Table

11-15 Not as yet in use.

An example of a LEXFLAGS setting is

X'8180' = 1000 0001 1000 0000

= Bits 0, 7, 8 set;

- remove leading spaces;

- suppress ICE facility;

- remove non-significant leading zeros.

10.3.2 Module MONITOR Output parameters

MONITOR does not normally return any output parameters, but in the instance where ICES has been invoked 2 parameters may be utilised. They are:-

FNRETCDE - Function return code (1 - byte, binary)

RRADDR - Recall Return address (1 fullword)

)))))) See Chapter 11.

10.4 KRES LEXICAL ANALYSIS PROCEDURES

Lexical Analysis in its simplest form (Hopgood , 37) is the removal of redundant information from any data stream prior to further processing - i.e. it is a formative procedure, the meaning of individual data items being of no significance, only the overall 'appearance'.

The lexical analysis phase of KRES attempts therefore to put any incoming keyboard line in a suitable base form, tailored to suit individual author's requirements.

This is akin to the author stating to KRES:

'Incoming messages for my lessons ought to have no leading spaces, only one space between words and contain no common punctuation characters. Furthermore, all characters should be capitals. Please convert all messages into this format for me'.

10.4.1 General approach

KRES lexical analysis procedures are split into 2 main groups:

(i) those which remove characters;

(ii) those which convert characters.

10.4.2 Character removal techniques

Nearly all character removals are carried out using a 'Mark/Sweep' technique. Individual searches are carried out and whenever the appropriate character is encountered, it is marked - the convention adopted within SCHOOL is to replace it with X'FF'. Once all searches have been completed, the entire input string is scanned and all marks (X'FF') deleted with the string being compacted accordingly.

N.B. There are two departures from this technique:

(i) removal of redundant parentheses - see Appendix 10.1;

(ii) removal of trailing spaces - this being done by reducing the string length as appropriate.

The 'marking' of redundant characters may be done at different stages depending on the range of removal required:

(a) Local removals (e.g. leading zeros, leading spaces) - these are searched for individually and marked when found;

(b) Global removals (e.g. all spaces, all common punctuation characters, all of particular characters specified by the author)

- these are removed using the Redundancy Table.

The Redundancy Table consists of up to 30 single - character entries. If all occurrences of a particular character are to be deleted, the character is inserted into the Redundancy Table. When all Local removal searches are complete, each input string character is compared with all entries in the Redundancy Table. If any match is found, a mark is inserted.

10.4.3 Character conversion techniques

It is often useful to the Author to have the ability to 'standardise' input into either upper or lower case (typically for answer matching). Normally all alphabetic input is immediately converted to upper case, but the Author has the ability to suppress this or convert to lower case as desired.

N.B. Full details of the logic associated with all lexical analysis techniques may be found at Appendix 10.1.

chapter eleven

- IMMEDIATE COMMAND EXECUTION SUBSYSTEM

11.1 INTRODUCTION

All interactive computing systems have a mechanism whereby processing can be interrupted by the user. This can be permanent, or (as in more sophisticated systems) temporary - the original task being re-invoked after another task has been completed.

This facility has been designed into SCHOOL at two distinct levels:

(i) Host system - adherence to the convention(s) of the software under which SCHOOL runs, i.e. normal ATTENTION or BREAK key actions are not altered;

(ii) SCHOOL software - a user can interrupt SCHOOL, carry out a particular function and then resume or stop, as required.

This latter facility is termed the Immediate Command Execution Subsystem (ICES) and it is available to any SCHOOL user at any point during system execution although the scope and range of options obviously vary.

11.2 INVOKING ICES

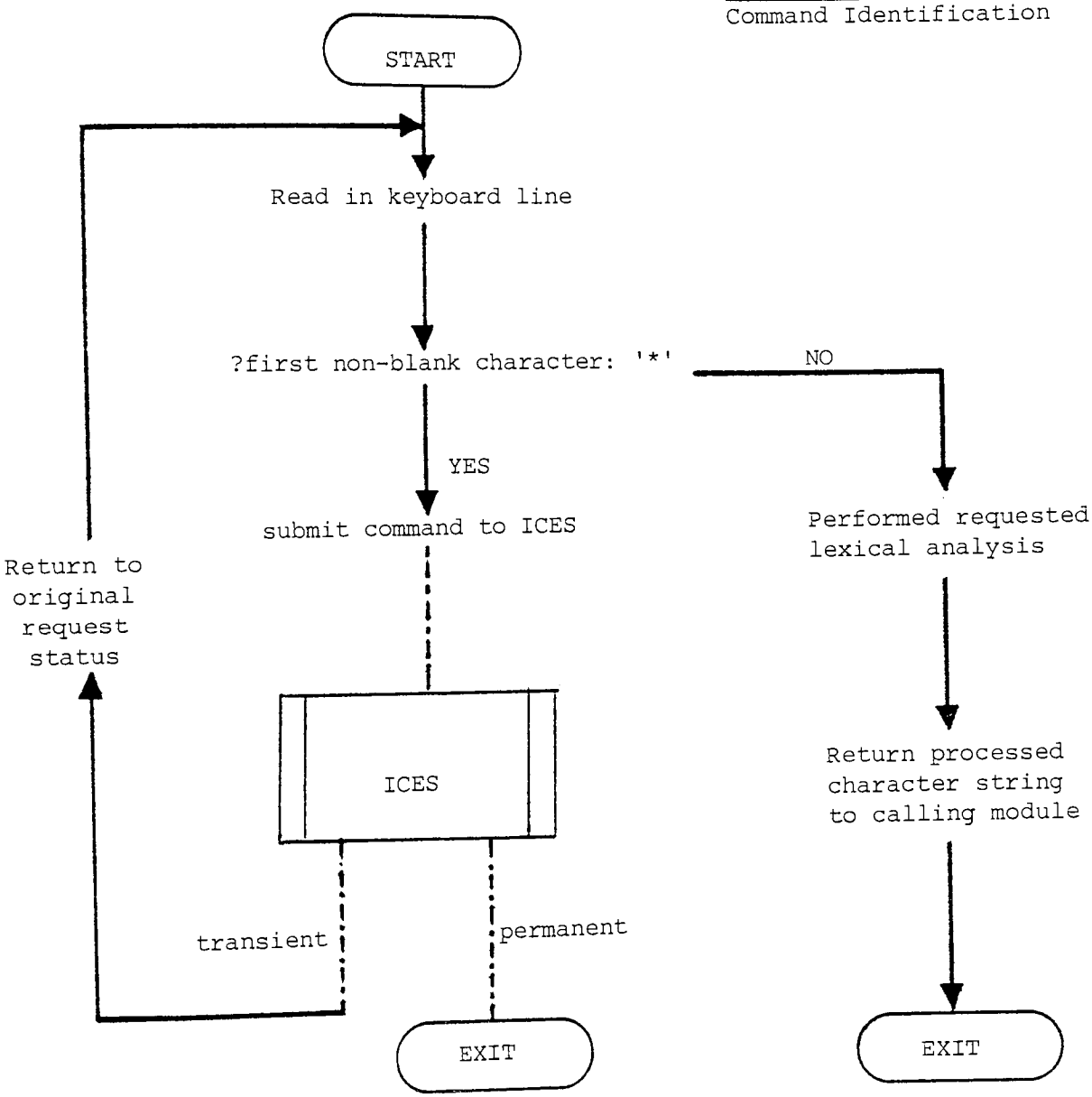
The mechanism for this has already been described in Chapter 10, but briefly any reply which has as its first non-blank character an asterisk, is treated as a call to ICES. The remainder of the entered line is then translated and the appropriate action (one of two types) taken:

(i) transient - the Immediate Command is executed and control returns to the original point;

(ii) permanent - the Immediate Command is executed and control is then transferred elsewhere;

Fig. 11.1 below illustrates the associated logic:

Fig. 11.1 Immediate Command Identification



This logic is embedded into module MONITOR, as is the control function of the ICE subsystem.

11.3 IMMEDIATE COMMAND EXECUTION AND CONTROL

Two important parameters control the processing of Immediate Commands. These are:

(i) Function Authorisation Level (FALEVEL)

- i.e. who is allowed to do what;

(ii) System Progress Level (SPLEVEL)

- i.e. when it can be carried out.

The first of these is established as the user logs on, and is set to:

0 - Student

1 - Author/Subject Manager)
) a password will be requested.
2 - System Manager)

Each Immediate Command has an FALEVEL threshold associated with it, and is then available to any user with an individual FALEVEL $\geq$ the Command FALEVEL.

A similar technique is used to control the stages of SCHOOL progress at which particular Immediate Commands can be actioned. SPLEVEL is the device used to achieve this, and it can take the following values:

0 - Introductory phase, no Immediate Commands available;

1 - Overall control level; mode of execution known (i.e. author/student/etc.), but not subject or user identification;

2 - Subject level; Current subject has been requested but user identification still pending;

3 - Subject level; user identification complete;

4 - Lesson level; required lesson has been defined and execution is in progress;

SPLEVEL is then set and reset as execution proceeds. Each Immediate Command has an equivalent SPLEVEL threshold associated with it, and becomes actionable when:

System SPLEVEL > = Immediate Command SPLEVEL

Identification and control of all components within ICES is achieved via the Control Command Table (CCT), organised as per Fig. 11.2:

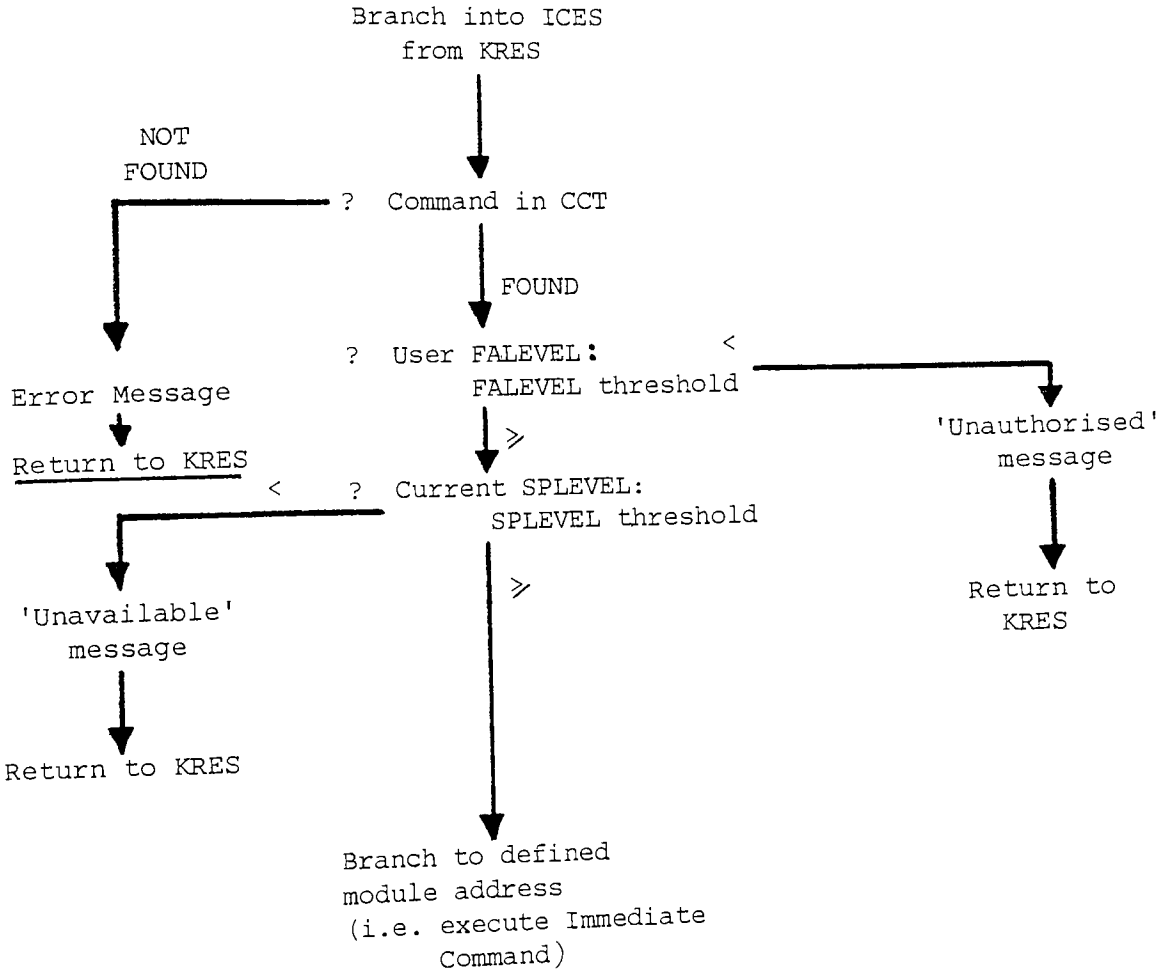
Command Name	Threshold		Module Address
	FALEVEL	SPLEVEL	
RESUME	0	2	A(RESUME)
SCORE	0	3	A(SCORE)
etc.			

Fig. 11.2

Control Command Table

CCT control logic is as follows:-

Fig. 11.3 Control command table processing logic.



Entries within the CCT are ordered:

(i) Student commands

(ii) Author commands

(iii) System Manager commands

This saves search time as Student Commands will be used most frequently - it will also be possible to both add new CCT entries and 'tune' the look-up facility (by putting the more frequently used commands at the front of their category).

Once a command has been identified and execution sanctioned, it may be necessary to display significant amounts of information to the user. ICES has been designed so that this is done

without spoiling session 'flow' - immediately relevant information (e.g. current score) will be displayed on the same screen if there is sufficient room but more general information (e.g. list of subjects) will be displayed on a new screenful, the original display being reactivated once the user has assimilated the data returned.

11.4 IMMEDIATE COMMANDS

The following table lists the facilities provided by ICES as at SCHOOL Release 1.1 (June 1981):

Fig. 11.4

Command	FALEVEL	SPLEVEL	Effect
*ABANDON	0	4	Current lesson abandoned. All records of the attempt are erased, and control returns to SPLEVEL 1.
*ANSWER	0	4	Answer to current frame (if any) is supplied. No score is awarded and control passes to the next remedial sequence.
*AUTHOR	0	4	Details of lesson Author and System Supervisor (names, addresses etc.), are output. Normal progress then continues
*COMMENT	0	3	The user may enter up to 50 lines of comment which is subsequently printed, prefixed by the name of the originator + address of the relevant Subject Manager (to whom it is then forwarded).
*CURRENT	0	2	Details of system progress level are returned, e.g. user type, subject name, lesson no., frame within lesson (if known).

Command	FALEVEL	SPLEVEL	Effect
*DETAILS	0	3	Details of the currently active subject are returned. Includes subject name, subject Manager details, no. subordinate lessons. For each lesson details are given of author, no. frames and score obtained by the user (if any).
*FINISH	0	1	System execution is immediately terminated; all current progress levels, scores, etc. are saved. Anything created or altered during the session is retained.
*GLOSSARY	0	4	Glossary Enquiry Mode is entered. The user is prompted to define the word or phrase requiring explanation. Once entered, the appropriate glossary definition is returned, or, if none exists, a 'Glossary Request' entry is created on the Database and a suitable message displayed.
*HINT	0	4	Next allowable Hint (if any) is displayed and user's score is adjusted accordingly.
* LIST	0	1	A list of the currently active and available subjects on SCHOOL is returned.
*MESSAGES	0	3	All of the currently active Subject Messages are displayed to the user, one at a time. The user is subsequently registered as having seen all Subject Messages.
*REPEAT	0	4	The user is prompted to define an earlier frame number. The current frame is then suspended and the defined frame reinvoked. Once completed, execution returns to the normal sequence.

Command	FALEVEL	SPLEVEL	Effect
*RESCREEN	0	1	Current screen contents are stored and the previous display re-screened. To return to the current display, another *RESCREEN is required.
*RESUME	0	3	A previously suspended session (see *SUSPEND) is re-invoked, all status and progress parameters being restored.
*SCORE	0	4	Full details of the user's performance within the current lesson are displayed, including no. questions attempted, score on each and total score.
*SUSPEND	0	4	The current session is suspended, and progress so far recorded (for a subsequent *RESUME). Control then passes back to SPLEVEL 1.
*ANALYSIS	1	2	A performance analysis of the complete Subject or selected Lessons is displayed and/or printed. It is advisable to use the print option for complete Subject analysis, or for multiple lessons.
*COPYON	1	1	The hard copy flag is switched ON, and all subsequent screen contents (including the current one) are printed. This will continue until session end or an *COPYOFF command is executed.
*COPYOFF	1	1	The hard copy flag is switched OFF.
*HISTORY	1	2	A performance history of the defined student(s) within the current subject is produced. This may be displayed or printed or both.

Command	FALEVEL	SPLEVEL	Effect
*PROCEED	1	4	Used by Authors when testing courseware. The current frame is abandoned and the system proceeds to the next in the Lesson sequence (either normal or remedial, as specified).

11.5 ICES SOFTWARE

In order to provide this 'Immediate Command' facility in a controlled manner, a rigid protocol for software construction was defined, encompassing:

- coding of ICES modules internal or external to keyboard monitoring software;
- register utilisation conventions;
- data and system progress integrity;

Full details of the protocols adopted within ICES software may be found in Appendix 11.

chapter twelve

- SUPERVISOR CONTROL SUBSYSTEM

12.1 INTRODUCTION

SCHOOL can be accessed in 3 discreet modes:

(i) Student - TLCS (Tutorial Logic Control Subsystem);

(ii) Author - ACS (Author Control Subsystem);

(iii) Control - SCS (Supervisor Control Subsystem);

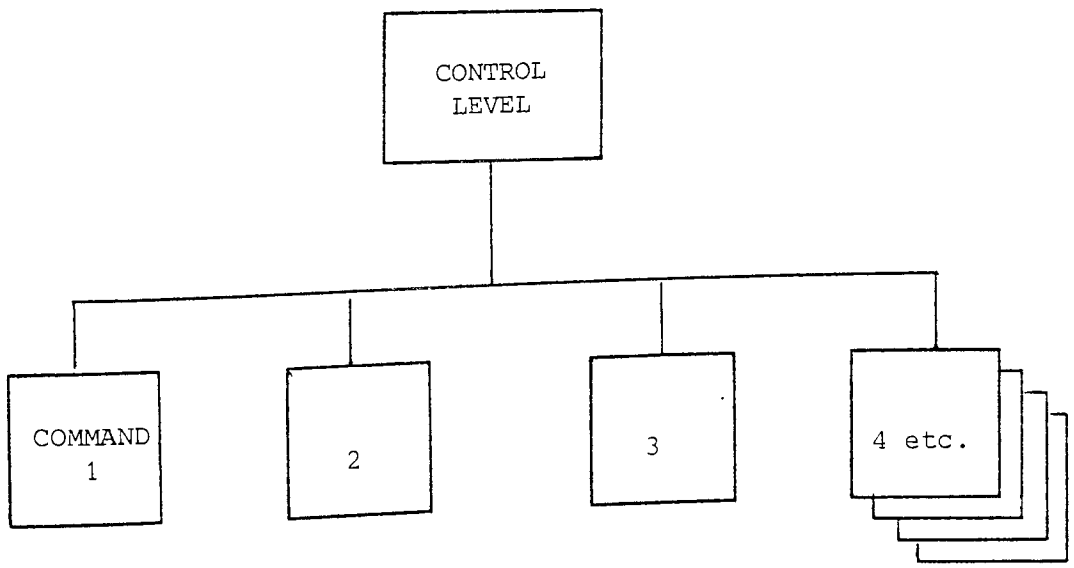
of which TLCS and ACS have already been described.

The philosophy of SCS is to provide an on-line facility for the control, monitoring and maintenance of the SCHOOL System, at a level higher than that of Author. Most other CBL systems (e.g. IIS, IBM) provide something similar but generally the on-line components have less power than SCHOOL SCS, and the important functions need to be done as off-line batch jobs, often requiring systems programming staff.

12.2 SUPERVISOR CONTROL SUBSYSTEM - ORGANISATION

SCS is organised at two levels; as Fig. 12.1 illustrates:

Fig. 12.1 SCS Organisation



Entry is always at the control level, and is achieved by invoking SCHOOL, specifying CONTROL as the Operation mode and supplying the correct primary password (4 attempts allowed).

CONTROL mode is a high security environment, and as such two important considerations apply:

(i) only one or two people should be capable of invoking it (frequent password changes help here);

(ii) certain commands have further individual password protection;

Once the control level of SCS is entered successfully, the user (termed the 'System Supervisor') is prompted to enter an SCS Command, the associated secondary password (if any), and an appropriate range of parameters - the number and type of which vary from command to command.

12.3 SCS COMMAND GROUPS

Control mode commands have been subdivided into 6 groups:-

12.3.1 Database Control Commands

- designed to interrogate the status of the physical SCHOOL database, backup and restore its contents and reorganise physical block allocations (DBTU's - see Chapter 3) to avoid excessive fragmentation. Commands within this group are:-

BACKUP *
DBPRINT
DBTUMAP
REORG *
RESTORE *

(asterisk signifies secondary password protection)

To carry out any of the above, the System Supervisor must temporarily suspend SCHOOL operations (via the TIMESET command - see below)

12.3.2 System Control commands:

- provide the ability to suspend (and resume) SCHOOL processing until a particular time and date, alter passwords, system version no. and System Supervisor details. Also included within this group is the error recovery procedure, used to checkout system abend information. Commands within this group are:-

ERRORS
PASSWORD *
SYSBOSS *
TIMESET *
VERSION *

12.3.3 Monitoring Commands

- essentially interrogation commands, this group gives the System Supervisor the facility of listing currently active users (at any point in time), and all registered students, authors and subject managers. Commands are:-

ACTIVE
AUTHORS
MANAGERS
STUDENTS

12.3.4 Global Message Control commands:

- System Global Messages may only be manipulated by the System Supervisor. Facilities encompass creation, deletion, listing and editing, and are provided by the following commands:-

GMRDEL *

GMREDIT *

GMRGEN *

GMRLIST

12.3.5 Subject Control commands:

- whilst subject registration is an automatic procedure (under the jurisdiction of a Subject Manager), the System Supervisor is responsible for the allocation of subject reference nos. (each of which must be unique within the SCHOOL system). The following commands enable the Supervisor to cross-reference allocated reference nos. with subject titles, interrogate the status of any subject and delete any unwanted ones:

SCRDEL *

SUBJECTS

SUBREFNO

12.3.6 Miscellaneous commands:

- commands not otherwise classified. Only one command currently exists:

END

The next section gives further details of the use and function of all SCS commands, listed alphabetically:

12.4 SCS COMMANDS

Fig. 12.2

Command	Secondary Password?	Function
ACTIVE	N	Returns a list of students within each current subject, active at the instant the command is issued.
AUTHORS	N	Returns a list of registered lesson authors, listed by subject and lesson within subject. This data may be displayed at the Supervisor terminal or printer.

Command	Secondary Password?	Function
BACKUP	Y	A backup copy of the SCHOOL database is dumped to the file defined. As at the current release of SCHOOL (1.1), this must be a CMS disk file.
DBPRINT	N	Produces a formatted print (hexadecimal and character equivalent) of the SCHOOL Database contents. The Supervisor has the option of the range of physical DBTU's scanned, and the output can be directed to the terminal or printed.
DBTUMAP	N	Produces a definition of physical DBTU allocation within the SCHOOL database. This is extracted from the Bit Map Table (BMT) at the instant the command is issued, and may be displayed at the supervisor terminal or printed. The main uses of this command are in pre-empting excessive data-base fragmentation, and checking on current database size.
END	N	Leave CONTROL mode.
ERRORS	N	Displays current system error status (held in the database Master Control Record. MCR).
GMRDEL	Y	Deletes one, several or all Global Message Records from the database. An option "TIMEUP" exists whereby all GMR's with overdue expiry dates will be deleted.
GMREDIT	Y	Alters the status of a GMR. This may be in terms of expiry date or message text (both are displayed in full prior to Supervisor alteration).
GMRGEN	Y	Creates a new Global Message. Details required are expiry date and message text (max. 16 lines).

Command	Secondary Password?	Function
GMRLIST	N	Displays one, several or all current Global Messages. Details produced are message no., expiry date and message text (one per screen). The data may be displayed at the terminal or printed.
MANAGERS	N	Returns a list of registered subject managers, either at the terminal or printed.
PASSWORD	Y	The current system control and author control passwords are displayed. The Supervisor is then given the option of changing either or both.
REORG	Y	The current SCHOOL database contents are reorganised to eliminate as much fragmentation as possible. Each logical record is read from the database in turn and written (as a physically contiguous block) to the defined copy database with Control statistics being displayed for the Supervisor as the processing is done. It is recommended that prior to this operation a database BACKUP is taken.
RESTORE	Y	The current SCHOOL database is overwritten with the contents of the defined file. This file will have been previously created via a BACKUP or REORG operation.
SCRDEL	Y	Deletes a defined subject in its entirety. The subject name is requested, and the subject control record plus all subordinate logical records are deleted from the database (this is done via bit reset at the BMT level).
STUDENTS	N	Returns a list of all registered students, either at the terminal or printed. The list is in subject order.

Command	Secondary Password?	Function
SUBJECTS	N	A list of all registered SCHOOL subjects is produced, including a structural breakdown of each. This is in terms of lessons within the subject, frames within each lesson plus other control information. Information may be either displayed or printed.
SUBREFNO	N	Produces a cross-reference list of all allocated subject reference nos. and corresponding titles, either at the terminal or printed.
SYSBOSS	Y	The current System Supervisor (or Manager) definition is returned, in terms of name, address and telephone no. The Supervisor then has the option of altering all or any part of this.
TIMESET	Y	Provides the Supervisor with the ability to suspend SCHOOL operations at student or author levels (or both), until a specified date and time. This is used typically as a prerequisite to database control functions, and the same command can be used with an option 'OFF' to resume activity.
VERSION	Y	The current system version no. (format 9.99) is displayed, and the Supervisor may then update this if required.

A SCHOOL Supervisor Guide is available (on a restricted distribution) to describe the processing and operand requirements of the above commands in more detail. Extracts of this are included at Appendix 12.1.

12.5 SCHOOL SYSTEM UTILITIES

In addition to the above SCS commands, the system Supervisor has at his disposal a small range of SCHOOL System Utilities. These are controlled externally to the main SCHOOL system via a control module named SCHUTL. Great care must be taken when using these (even though they are heavily password protected) as they incorporate operations such as:

- delete the entire SCHOOL database;
- reset the database, leaving only an initial Master Control Record and Bit Map Table;
- access (and deletion) by relative record no. of physical database records;

Details may be found in the SCHOOL Supervisor Guide (and Appendix 12.1).

12.6 SUPERVISOR CONTROL SUBSYSTEM - SOFTWARE

The control level of SCS (see Fig. 12.1) is coded as module SUPERVYS. This handles the initial operations of SCS, i.e.:

- command entry and recognition;
- password request (if necessary) and matching;
- branching to the appropriate SCS command coding;

SUPERVYS also contains coding to provide some of the more trivial SCS functions (i.e. END, ERRORS) as these either involve no logical record processing, or only reference the Master Control Record.

All other SCS functions are supported by separate program modules, named SCS..... For a detailed description of these, see Appendix 12.2.

chapter thirteen

- THE SCHOOL SYSTEM IN OPERATION

13.1 INTRODUCTION

A major objective in the design and development of the Compower CBL system was that the final product should be as efficient as possible. This Chapter assesses the effectiveness of the SCHOOL system towards this end.

13.2 CURRENT HARDWARE/SOFTWARE ENVIRONMENT

The current release of the system (as at March 1982) runs on an IBM 3033 mainframe at Compower's Cannock Computer Centre. The characteristics of this machine are:-

Real storage size: 8 megabytes

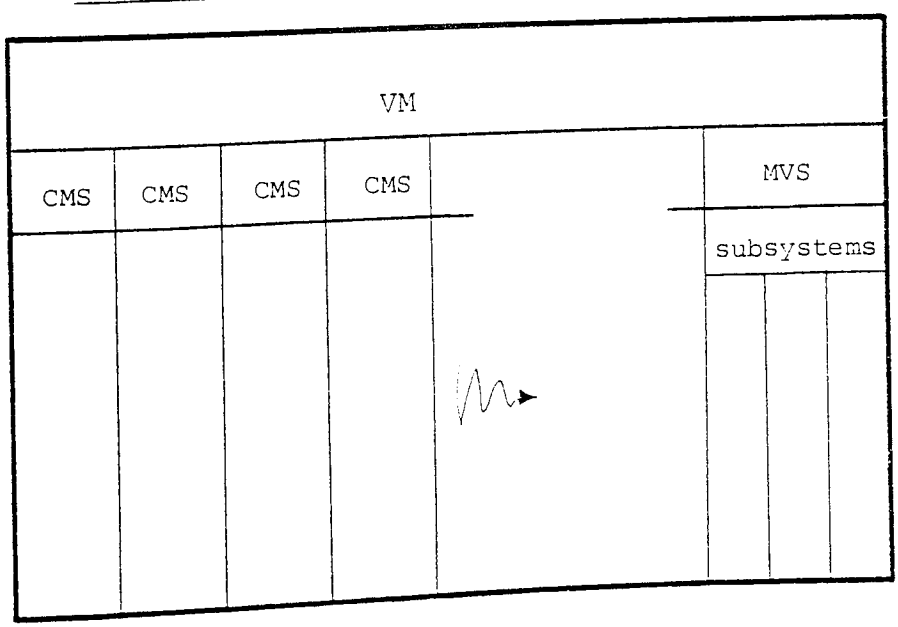
Processing performance rating: 5 mips

(where mips = million instructions per second).

The machine operating system is VM/370 (Virtual Machine Facility/ System 370 - (IBM,4)), the installed release level being VM/SP service level 0101.

As described in Chapter 5, VM/370 provides within a single real machine an emulation of multiple 'virtual' machines, each with a single conventional Operating System. A typical breakdown of the Cannock 3033 is shown in Fig. 13.1:

Fig. 13.1 Compower 3033 System



MVS (Multiple Virtual Storage; (IBM,41)) is a conventional multi-task operating system which can run several application systems concurrently (including on-line facilities, each supporting several users). CMS (Conversational Monitoring System; (IBM,4)) is a single-task supervisor controlling the resources of one virtual machine.

A typical breakdown of 3033 user level is:

55 CMS virtual machines

1 MVS virtual machine
- normally running batch work and interfacing with other mainframes

5 miscellaneous virtual machines

An average CMS machine's virtual storage requirements are 200-300K (Max. 768K), of which 128K is reserved for system use.

N.B. As described in Chapter 5, each SCHOOL user, irrespective of type, occupies a separate CMS virtual machine. The following sections will discuss the resource requirements of such a user.

13.3 SCHOOL SYSTEM STATUS

The SCHOOL system software currently comprises 70+ program modules, grouped into the various subsystems already discussed. A breakdown of this structure is shown in Fig. 13.2

Fig. 13.2 SCHOOL Software breakdown

Subsystem	No. Modules	Total size.	Data Nuclei sizes.
SYSTEM CONTROL MANAGER	1	3081	1424
SUPERVISOR CONTROL SUBSYSTEM	21	18026	2473
AUTHOR CONTROL SUBSYSTEM	13	12462	1268

... cont.

Subsystem	No. Modules	Total size	Data Nuclei sizes
TUTORIAL LOGIC CONTROL SUBSYSTEM	7	14008	3376
DATA INPUT SUBSYTEM	13	33301	3779
KEYBOARD RESPONSE EVALUATION SUBSYSTEM	1	5044	-
IMMEDIATE COMMAND EXECUTION SUBSYSTEM	12	6820	-
DISPLAY FORMAT SUBSYSTEM	3	9765	-
DATABASE INPUT/OUTPUT SUBSYSTEM	3	7305	(4000 (4216
ERROR MANAGEMENT SUBSYSTEM	1	1656	102
T O T A L	75	111468	20638

N.B. These statistics refer to the testing version of SCHOOL active as at 14th March 1982. They are likely to change.

It is reasonable to compare SCHOOL in these terms with the IBM Interactive Instructional System (IIS). A typical IIS configuration running under VM/CMS requires 280K (286720 bytes) virtual storage for each transaction type (i.e. author, student or system administrator - (IBM,42)) compared to the SCHOOL maximum requirement of 132106 bytes (129K) per user (i.e.111468 +20638)

13.4 SCHOOL SYSTEM PERFORMANCE

The performance of the system has been considered from 3 specific viewpoints:

(i) reponse times;

(ii) machine resource utilisation.

(iii) cost.

13.4.1 Response times

Most aspects of SCHOOL system operation (except some of the more complex Supervisor Control operations) produce almost instantaneous response - the majority of test

sessions involved response times of between 0.5 and 2 seconds even at the busiest times of a normal working day. The longest operation monitored during testing was the automatic generation of a 'medium-sized' course from an Author Document file, taking 3 mins. 38 secs. from start to finish. However, this is still a very 'interactive' operation with several messages appearing at the terminal as processing proceeds.

N.B. All formal monitoring sessions were conducted using IBM 3270 VDU equipment. These use much faster telecommunications lines than conventional terminals, with corresponding benefits to response times. For instance an Anderson Jacobson 'daisy-wheel' terminal can take up to 12 times as long for identical operations, depending on the amount of text being transmitted.

13.4.2 Machine Resource Utilisation

This was measured in terms of 4 components across a variety of terminal sessions:

(i) Session durations;

(ii) CPU utilisation;

(iii) Input/Output traffic (both terminal and disk);

(iv) Courseware disk storage.

It should be noted that items (i), (ii) and (iii) are applicable to one individual user, but that courseware disk storage is shared between all users.

Fig. 13.3 overleaf tabulates some typical session statistics.

Fig. 13.3 SCHOOL session statistics

NO.	DATE	DURATION (HH:MM:SS)	CPU (MM:SS)	I/O (ITEM & DISK)	COMMENTS
1.	18. 6.81	0: 8:48	0:06	264	)
2.	18. 6.81	0:10:35	0:10	754	) Author sessions)
3.	2. 9.81	0:10:50	0:13	736	Supervisor mode
4.	11. 9.81	0: 7:22	0:32	895	Supervisor mode (BACKUP operation)
5.	20.10.81	0: 6:03	0:03	202	)
6.	19.12.81	0:11:47	0:01	639	)) Various Author sessions)
7.	27. 1.82	0: 8:10	0:01	610	)
8.	10. 2.82	1:15:49	0:10	1253	)
9.	16. 2.82	0:11:50	0:01	299	)) Student sessions
10.	20. 3.82	1: 0:44	0:05	1253	))
11.	20. 3.82	2:39:15	0:01	245	)
12.	20. 3.82	0: 5:31	0:01	270	Supervisor (TIMESET)
13.	24. 3.82	0:20:42	0:01	482	)
14.	25. 3.82	0:27:48	0:03	587	)) Student sessions
15.	25. 3.82	0: 9:40	0:01	389	))
16.	25. 3.82	0:12:52	0:02	453	)
TOTALS:		7:27:46	1:31	10051	

Whilst no directly comparable CBL system in a similar hardware environment has been benchmarked, the statistics tabled in Fig 13.3 bear comparison with almost any type of on-line activity (as will be discussed in greater detail in section 13.4.3).

Disk storage requirements for the SCHOOL database will of course vary with the number and complexity of subjects stored. The effective maximum is dictated by the number of bits in the Bit Map Table (BMT). Two considerations apply:

(i) test system BMT allocation:

- set at 4000 bytes;
- limits D/B to 32000 DBTU's (Data Base Transfer Units), or 6,400,000 bytes (6.1 megabytes);

(ii) absolute limit of database:

- determined by chain pointer system;
- limited to 2^{21} - 1 DBTU's (i.e. 419,430,200 bytes, or 400 megabytes);

N.B. For a detailed discussion on the above, see Chapter 3, section 3.6).

The test system database has supported up to 25 subjects concurrently, albeit of limited size, but has also been used with fewer, much larger subjects. It is for instance capable of holding:

5 Subjects, each with:

- 20 students
- 4 subject messages
- 10 lessons, each with:

- Lesson Analysis
- Syntax Information
- 10 Glossary entries
- 20 Frames, each with:
    - Question)
    - Hints) all max. size
    - Answers)
    - Comments)

(plus overheads such as BMT, 3 Global messages).

This structure (5 subjects, 100 registered students, 1000 frames) requires 25417 DBTU's (4.848Mb).

An absolute maximum Database would involve the following:

50 Subjects, each with
- 100 students
- 10 subject messages
- 40 Lessons, each with
    - Lesson Analysis
    - Syntax Information
    - 40 Glossary entries
    - 40 Frames, each with
        - Question)
        - Hints) all max. size
        - Answers)
        - Comments)

(plus overheads such as BMT, 10 Global messages).

This represents 50 subjects, 5000 registered students, 80000 frames and would require 2058012 DBTU's (392.5 Mb).

N.B. Using existing disk hardware (IBM 3350), it would probably be convenient to restrict the above to 316Mb, the capacity of 1 disk volume. Planned devices (IBM 3380) will have no such restriction.

In comparison to the above, the IIS system has an overhead of 8.3 Mb disk storage as delivered (for Messages, student history file, formats etc.), plus 13 Mb for sample courseware, (IBM,42). User developed material is additional to this, and would typically require 0.63Mb for a course of similar dimensions to each subject of the SCHOOL 10 lessons/20 frames example quoted earlier. Total disk requirements (not including sample courseware) are therefore:

5 courses @ 0.63Mb	=	3.15 Mb
Overheads (for above with 20 students max. per course)	=	8.3 Mb
		11.45 Mb

(Sources: IBM(47,48)) (Comparative SCHOOL size: 4.848Mb)

13.4.3 Costs - Processing

Processing at Compower is charged out to clients on the basis of Computer Resource Unit (CRU) utilisation, these being calculated from a combination of:

CPU time,

I/O (disk and terminal),

Virtual storage,

Connect time.

Different algorithms are then applied to these components (depending on host environment) to produce a CRU value for the job or session, and a charge is made to the user at the appropriate rate of £x per CRU.

Using the same test sessions as Fig. 13.3, typical SCHOOL costs are as follows:-

Fig. 13.4 SCHOOL Session costs

No.		DURATION (HH:MM:SS)	CRU's	COST @ 65p/CRU	COST/HOUR
1.	A	0: 8:48	4.12	2.80	19.09
2.	A	0:10:35	6.78	4.61	26.14
3.		0:10:50	7.60	5.17	28.63
4.		0: 7:22	13.47	9.16	74.61
5.	A	0: 6:03	2.40	1.63	16.17
6.	A	0:11:47	5.10	3.32	16.91
7.	A	0: 8:10	4.34	2.82	20.72
8.	S	1:15:49	29:11	18:92	14:97
9.	S	0:11:50	4.35	2.83	14.35
10.	S	1: 0:44	18.55	12.06	11.91
11.	S	2:39:15	27.86	18.11	6.82
12.		0: 5:31	2.22	1.44	15.66
13.	S	0:20:42	6.01	3.91	11.33
14.	S	0:27:48	9.11	5.92	12.78
15.	S	0: 9:40	3.89	2.53	15.70
16.	S	0:12:52	5.31	3.45	16.09
TOTAL		7:27:46	150.22	98.68	13.22 *

* overall average

(N.B. 'A' in Column 1 indicates an Author session,
'S' indicates a Student session)

Average cost/hour figures for the test SCHOOL sessions as derived from the previous tables are therefore:

Author	£20.07
Student	£10.73

N.B. System Supervisor operations have not been included in the above cost assessment. Such a wide range of functions is available within this mode (most of which are used infrequently) that it is difficult to be representative.

Equivalent costs for other systems are:-

PLATO	- £10 per hour at a CDC Learning Centre (Box,27) November 1980;
	- To carry out user authoring and study in-house, cost of £22,000 per annum (or £22 per hour for 4 hours use per day, 250 working days per annum);
IIS	- not benchmarked at Compower, but likely to be between £18 and £24 per hour, based on existing evidence;
CMS (Compower)	- normal on-line development work averages £15 to £25 per hour, APL work £29 to £49 (averaged over 8 months);
VSPC (Compower)	- normal on-line work averages £11.26 per hour, APL £17.51;

Costs evaluated for SCHOOL sessions are therefore most favourable, and very much in keeping with other commercially available on-line systems (CBL or otherwise). They are not however anywhere near the forecasts of Bitzer & Skaperdas (10) of a few cents per hour or the claims of Flockhart (17) of 4p per student hour (15p if hardware is included). These figures are simply not realistic if formal commercial accounting procedures are adhered to.

13.4.4 Costs - Hardware

Within Compower and the National Coal Board SCHOOL hardware costs will be negligible as it is intended to run the system alongside existing facilities using existing terminal equipment. Should a client wish to install a dedicated terminal however, entirely standard devices will be suitable with appropriate costs as follows:

	£.	
IBM 3278 VDU (will require access to appropriate control unit)	55	per month rental
Teletype terminals (e.g. AJ832, Mellordata, DECscope etc.)	1000-2500	purchase

Specialised CBL terminals similar to those used on the PLATO system, whilst having many extra functions, tend to be very much more expensive - PLATO terminals currently cost £960 per month.

Conclusions

- AND PROPOSED EXTENSIONS

14.1 INTRODUCTION

The purpose of this final chapter is to assess and review the prototype SCHOOL system from a number of viewpoints:

- firstly, to what extent the original objectives were achieved and how simple the system is to use;
- secondly, to highlight some of the specific techniques which have been evolved;
- thirdly, to address the continued development and extension of the system.

It is not proposed to discuss the effectiveness of Computer-Based Learning in general within this Chapter. Considerable space was devoted to this in Chapter 2 and the current limited success of the concept is seen not as a result of CBL being an inherently and irreversibly poor teaching medium, but of having failed to reach its full potential for a variety of reasons:

- expense and inefficiency;
- difficulty of courseware authoring;
- special hardware requirements;
- inflexibility and complexity.

SCHOOL has attempted to address all of these problem areas in order to produce a much more usable form of Computer-Based Learning.

14.2 ORIGINAL SYSTEM OBJECTIVES

The basic philosophy of SCHOOL, outlined in Chapter 2 should now be restated:

(i) the package should have efficient system software and optimum data storage utilisation;

(ii) existing terminal and telecommunications equipment should be used;

(ii) course authoring should be straight-forward, and no Author language should be involved;

(iv) the entire system should be easy, attractive and interesting to use.

As the project has developed, two further objectives have been added to this list:

(v) the software should be portable;

(vi) both software and database should be readily capable of expansion;

14.3 ACHIEVEMENT OF ORIGINAL OBJECTIVES

14.3.1 Efficiency

Considerations within this area have been discussed in some detail in Chapter 13, and it appears that SCHOOL is significantly more efficient than similar existing systems, e.g. comparing SCHOOL and IBM's IIS:

	Ratio (SCHOOL : IIS)	
Software size	1	2.17
Course Storage	1	2.36
Processing costs	1	1.58

14.3.2 Use Existing Hardware

Entirely standard terminal and telecommunications equipment is used to access SCHOOL. Also 'device optimisation' techniques ensure that these are used as efficiently as possible.

14.3.3 Easy and Flexible Authoring

The removal of any dependency on a CBL Author Language was considered most important, and this has been entirely achieved whilst still retaining most, if not all of a

language's flexibility and power. Certainly an important trend in computing has got to be towards more flexible and agreeable man/machine interfaces, as many current commentators agree (Peltu, 46). Computing languages as we understand them now do not fit well into this concept.

N.B. An even more powerful author interface is currently being examined for SCHOOL - see 14.5.3.

Within the small sample of Authors who have produced SCHOOL Courseware (none with CBL experience and one with no computing experience at all), favourable reaction was aroused by:

(i) the 'structuring' of courses;

(ii) use of Author documents, particularly for planning;

(iii) clarity of prompts and error messages;

(iv) screen layout control;

(v) the hardcopy facility;

(vi) automatic input of course material;

(vii) the glossary concept;

(viii) author monitoring aids (lesson analysis, student performance);

(ix) general 'user-friendliness'.

Criticisms were expressed however on the following:

(i) Syntax Information facility. Some aspects of this were considered too complex, and doubts were expressed on its usefulness;

(ii) Author Document volume. Complex courses could involve a considerable number of input documents, and it was felt (by one author) that this could be intimidating;

(iii) Graphics. Some authors felt constrained by the lack of graphics within SCHOOL, although a student handout

containing related diagrams was considered an acceptable alternative;

(iv) High brightness/underscoring. Two authors noted that they would have found it useful if they could have highlighted parts of their text;

(v) Text width. The restriction to 70 characters (caused by Author Document control fields) caused some irritation;

(vi) Size of glossary text. All authors who used the glossary facility noted that sometimes they felt constrained by having only 3 lines for a glossary explanation.

14.3.4 User-friendliness

Considerable effort has been put into making SCHOOL attractive and straightforward, as can be appreciated from the many session extracts used in this text. It is certainly more usable than a system such as IIS, although it must be acknowledged that some aspects of of PLATO (e.g. dynamic graphics, touch activated VDU screen) are most impressive, albeit involving special CBL hardware. Planned extensions to the SCHOOL user interface will be discussed in Section 14.5

14.3.5 Portability

It was hoped that SCHOOL would be capable of running without modification on different machines. This has not been achieved, although CMS host environment dependence has been kept to a minimum (even though using specific VM/CMS facilities was at times very attractive and could have saved considerable design and development). True portability would inevitably compromise efficiency - probably more than would be acceptable.

14.3.6 Expandability

There are two aspects to consider in this context:

(i) D/B size
- current limit 6.1 mb;
- can be extended to 400 mb very simply;
- multiple databases could also be configured.

(ii) Software facilities
- the hierarchical subsystem/module structure lends itself particularly well to the addition of new functions.

14.4 SPECIFIC TECHNIQUES EVOLVED

In addressing the objectives of this project it has become necessary to develop and apply a variety of new software techniques, in particular:

14.4.1 Using a Hierarchical Database for Courseware

This has not been encountered by the author in any other CBL context, but the benefits that have resulted from the SCHOOL implementation are many:

(i) rapid access to any part of any course;

(ii) one database maintains and controls all system information (course contents, procedure, messages, performance data etc.);

(iii) flexible and efficient storage control, without constant manual manipulation;

(iv) expandability;

(v) database logical record structure makes update/deletion easy and efficient (all course components are effectively separate entities within the overall course structure).

14.4.2 Text Compression

Any CBL system will involve considerable volumes of text, and even though hardware costs will continue to decline,

there are significant benefits to be gained from concise text storage - especially where compression of the text is augmented by compression of its layout.

The investigations carried out on this within the SCHOOL project may not necessarily represent the best possible compression, but the reductions achieved are certainly most useful and, in conjunction with the rapid table searching algorithms evolved, efficiency of storage has been achieved without introducing noticeable processing overheads.

14.4.3 Author Language Elimination

Whilst the absence of an authoring language is not entirely unique within CBL systems, SCHOOL has taken this philosophy much further than previous products - e.g. ACATS (Flockhart,17). The comprehensive range of features within SCHOOL's Author Control Subsystem include:

(i) Author Document facility;

(ii) Two input protocols: - automatic
- interactive;

(iii) Course performance monitoring;

(iv) Interactive courseware maintenance;

(v) Interactive student control.

N.B. It should be noted that the implementations of many of the above have been made much of easier by the hierarchical organisation of the SCHOOL Database.

14.4.4 Device Control/Output Optimisation

Access to the SCHOOL system will probably be from a number of different types of terminal - although the majority will be IBM 3270-Series VDU's.

The Display Format Subsystem (DFS) was designed to use these different devices in different ways, taking into account the optimum characteristics of each.

Furthermore, it is almost certain that new terminal types will come into use in the future - possibly implying further unique characteristics. The organisation of SCHOOL is eminently suited to handle these as and when they are introduced.

14.5 PLANNED ENHANCEMENTS

For any software product to have a significant lifetime, it must be flexible and able to 'change with the times'. These changees can be provoked in a number of ways:

(i) a recognition of current shortfalls;

(ii) new ideas producing major or minor system enhancements;

(iii) the exploitation of new technology.

Items from each of the above categories have been identified, and it is appropriate to add some brief notes about them at this point:

14.5.1 Current Shortfalls and consequent Revisions

(a) Data and time on Hardcopy boxes:

- could prove useful for administrative purposes, and is simple to implement.

(b) Hardcopy switching for students:

- currently this is available only at Author and Supervisor levels, but test session students have noted how useful it could be. It would be extremely easy

to activate via *COPYØN and *COPYØFF immediate commands (reset command FALEVEL - see Chapter 11, section 11.3).

(c) More back screens:

- test user reaction to this facility (activated via *RESCREEN) has been most favourable, but it was often felt that having more than one previous screenful would be useful. Implementation of this concept would involve the creation of several 'backscreen' buffers, as shown in Fig. 14.1:

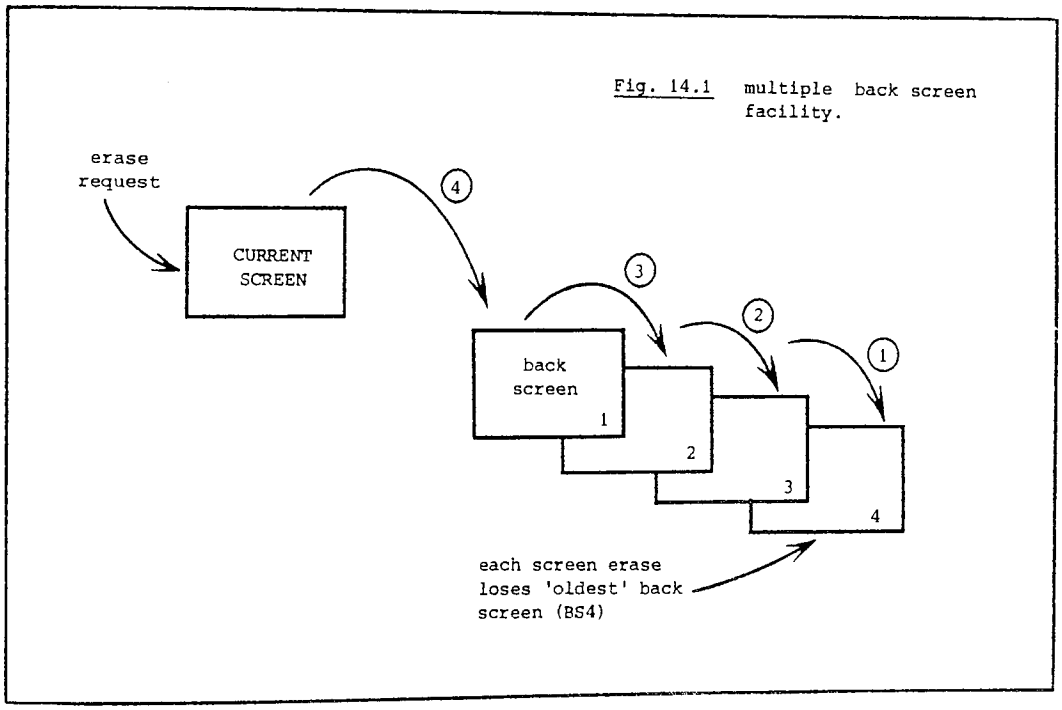


Fig. 14.1 multiple back screen facility.

The number of buffers manipulated would need careful consideration as each requires

approximately 2K bytes. A new *RETURN command would be required for returning to the current screen.

14.5.2 Minor Enhancements

(a) Use of Program Function (PF) keys:

- many current terminals (including 3270 screens) have PF key facilities, whereby various host system/application commands can be activated by a single key depression. This could be applied to great effect within SCHOOL by assigning various common Immediate commands to PF keys - a typical list would be:

PF Key	Associated command
1	*RESCREEN
2	*LIST
3	*GLØSSARY
4	*HINT
5	*ANSWER
6	*SCØRE
7	*CURRENT
8	*DETAILS
9	*CØMMENT
10	*MESSAGES
11	*CØPYØN
12	*CØPYØFF

Fig.14.2 PF key assign-ments.

N.B. 'Termination' commands such as *ABANDON would not be allocated to safeguard against accidental key depression.

(b) Command abbreviations:

- as user expertise increases, entering system commands in full can become irritating, and 'user-friendliness' would benefit from being able to abbreviate these commands down to a minimum unique character string. For example:

Fig. 14.3 Command abbreviations

COMMAND TYPE	COMMAND	MIN. ABBREVIATION
AUTHOR	ALTER	AL
	ANALYSIS	AN
	REMOVE	REM
	RESETLAR	RES
SUPERVISOR	SCRDEL	SC
	STUDENTS	ST
	SUBJECTS	SUBJ
	SUBREFNO	SUBR
IMMEDIATE	*ABANDON	*AB
	*AUTHOR	*AU
	*ANSWER	*AND
	*ANALYSIS	*ANA

14.5.3 Major Enhancements

(a) 'Black box' simulations:

- some CBL applications can be significantly improved if some form of dynamic model (or 'Black box') is incorporated into the courseware, e.g. mathematical formulae, physical process model. A possible approach for this has been devised for SCHOOL, along the following lines:

(i) the course author designs the function of his 'Black Box', typically in terms of Input/Procedure/Output;

(ii) an emulation program is produced for it, written in a compatible programming language (e.g. COBOL, ASSEMBLER, FORTRAN), possibly by a professional programmer;

(iii) a link between the Black Box program and the corresponding part of the SCHOOL course is set up by specifying

BLACKBOX (....programname....)

as the appropriate Prime Answer entry;

(b) Help facility:

- a further improvement to SCHOOL's man/machine interface would be the introduction of a Help facility, implemented at various levels:

(i) *HELP Immediate command

- would temporarily enter Help mode, whereby the user could list commands, request explanations, etc.;

(ii) '?command':

- available on an 'Immediate' basis, this would return an explanation of the specified command;

(iii) '?':

- would return an explanation of what is required from the user at this point of system progress.

Various techniques would be required to effect the above, but almost certainly a new SCHOOL subsystem would be involved.

(c) Full Screen Management:

- recent improvements to CMS and VSPC systems have made available a facility whereby a complete screenful of information can now be read as well as written. This product is the Full Screen Manager (FSM) and it will have major implications within the SCHOOL Author and Supervisor Control Subsystems. For example, instead of prompt/response interactive input, consider the following sequence:

  (i) display 'Author Document' screen format;

  (ii) position cursor at start of first input field;

  (iii) the user then enters information into each field as required, tabbing from one field to the next;

  (iv) ENTER is pressed on completion of the last field, and the complete screen is then transmitted.

This technique is also applicable to courseware alteration - existing material can be displayed on the screen and the user simply moves the cursor around, making whatever alterations that are required. Pressing ENTER then freezes these alterations.

The use of FSM could represent a significant improvement to SCHOOL's user interface, as well as reducing terminal I/O operations.

14.5.4 New Technology

During the development of SCHOOL a number of important technological advances have taken place which are expected to be incorporated in the near future. These are discussed below:

(a) Multiple screen sizes:

- IBM 3270-type devices are now available with different screen sizes. e.g.

IBM 3278 VDU:

Model	Rows	Columns
2	24	80
3	32	80
4	43	80
5	27	132

SCHOOL Display Format Subsystem (DFS) was originally tailored to 24 x 80 screens, but could quite readily cater for more rows. Note that:

(i) the terminal model number can be extracted automatically from the host system (as is already done for terminal type);

(ii) most operational screens will continue to be 24 x 80.

(b) Upper and lower case:

- when development work on SCHOOL started, terminals capable of accessing the system were upper case only, and whilst allowance for upper and lower case I/O was made, no special emphasis was put on its use. Modern terminal devices now handle mixed case, and this is much more suited to CBL applications than all capitals. SCHOOL is capable of processing upper/lower case mix with certain minor considerations:

(i) alterations to the text compression algorithm will be necessary to retain a definition of original word format:

- all lower case
- first character capital, remainder lower case
- all capitals

(compression code bit settings could be used to define the above combinations);

(ii) compression/expansion will be slightly slower;

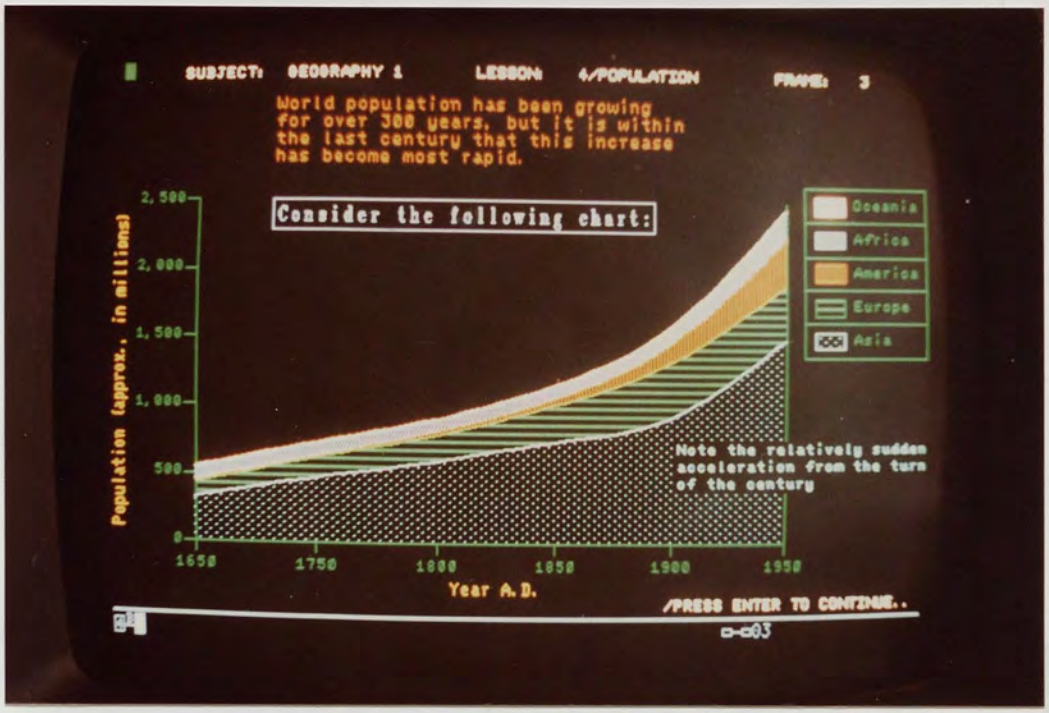
(iii) hardcopy control software will require modification to use upper and lower case character fonts on the mainframe printer.

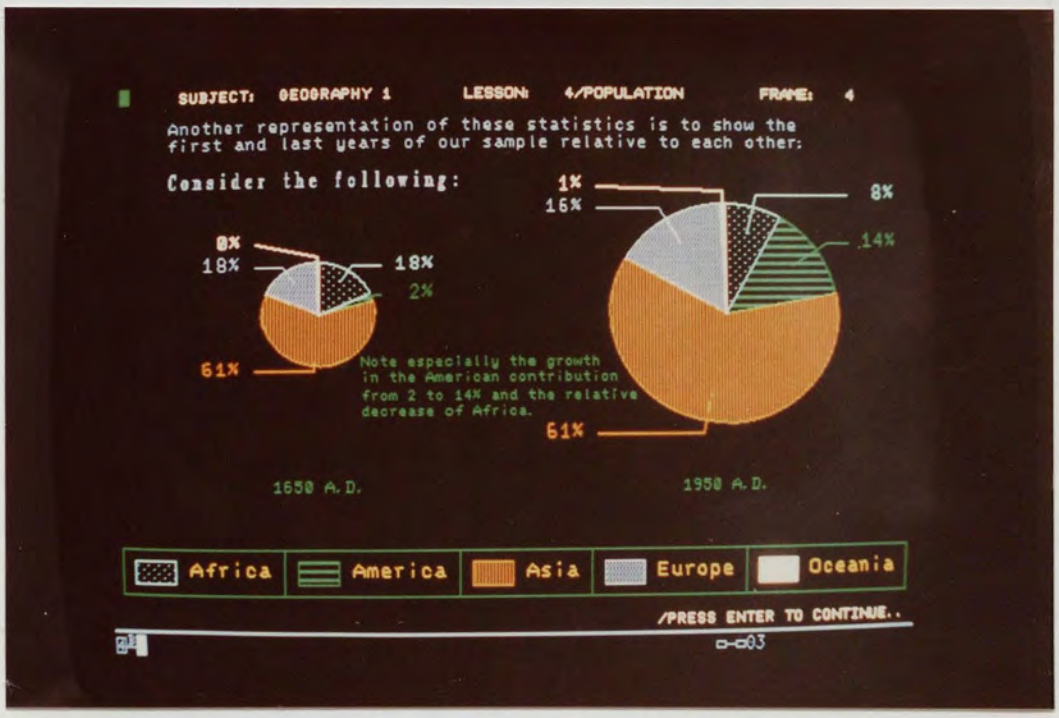
(c) Colour graphics:

- the use of graphics within CBL, whilst a major enhancement, has normally involved the use of special purpose terminals (e.g. PLATØ) or been microprocessor based (e.g. APPLE II PILOT system), with the associated storage restrictions. Recently however examples of general purpose VDU's with high-resolution graphics capabilities (sometimes in colour) have become available. Typical of these is the IBM 3279, a 32 x 80 character screen with 7 base colours and with associated software GDDM (Graphical Data Display Manager - IBM,43). A function within GDDM is the Interactive Chart Utility (IBM,44) which provides an ability to create graphs, charts, histograms, etc. totally interactively - no user programming being involved. Once created, a display can be saved and subsequently recalled either interactively or by program.

  This interface could quite readily be incorporated into SCHOOL, although courseware Authors would need to be familiar with the Interactive Chart Utility (a

very simple task). Some examples of how colour graphics could be employed follow:





Photographs taken with Minolta SRT101;
$\frac{1}{4}$ sec @ f 1.4 using Kodacolor II, ASA 100.

More sophisticated graphics are also possible, but these need to be program-driven and an interface along the lines of the 'Black Box' concept described earlier would be involved.

(d) Distributed data processing - minis and micros;

- in the mid 1970's, a project was started at Compower to implement a network of Digital Equipment Corp-pration PDP minicomputers which would link into the central IBM mainframe(s), as shown in Fig. 14.4:

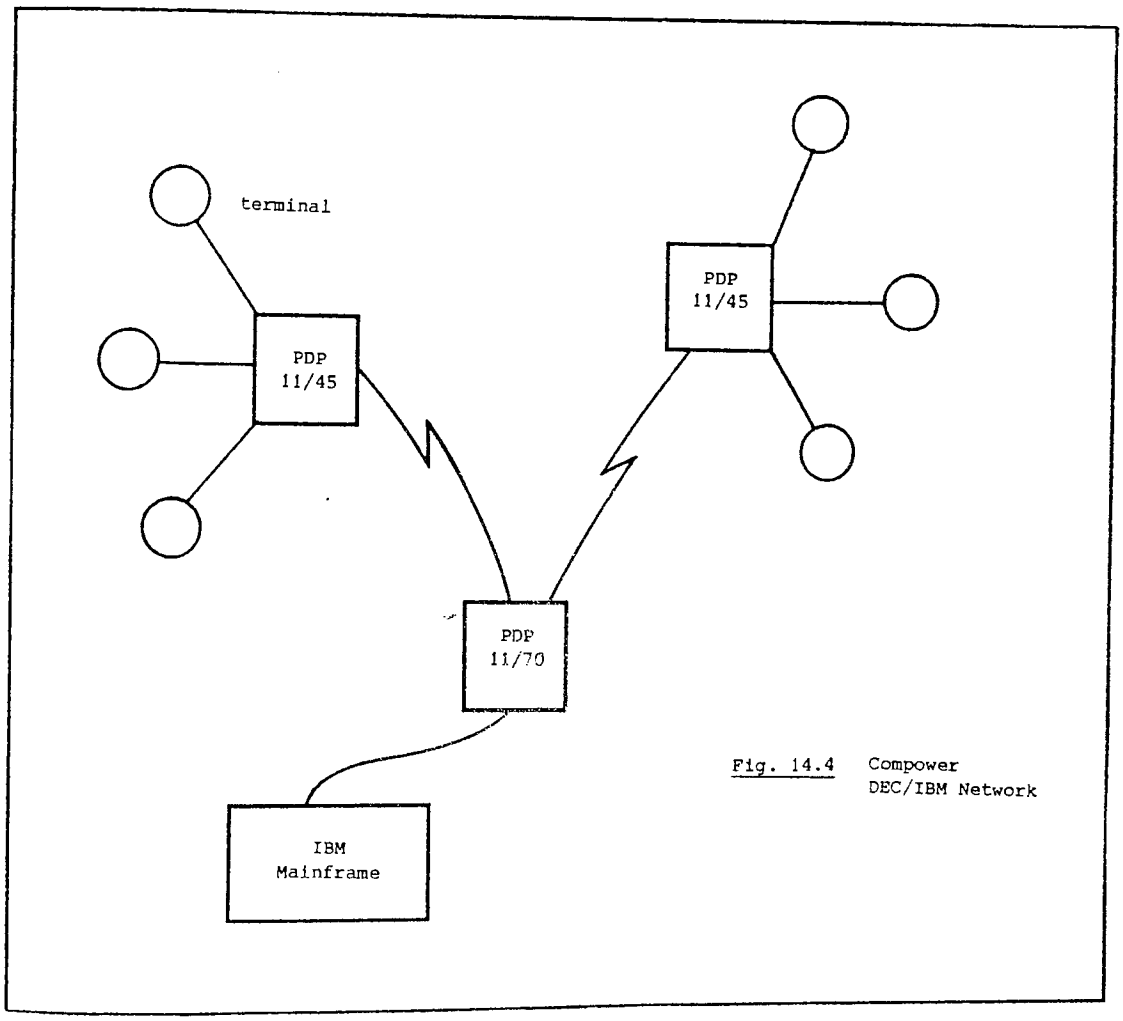


Fig. 14.4 Compower DEC/IBM Network

The minicomputers were to be installed in a number of NCB locations, with processing being broken down into local and central components, and certain aspects of SCHOOL were designed to reflect this.

For a variety of reasons, this concept has had only limited success and the network has been restricted to NCB Scientific installations.

Since March 1981 however a more powerful approach has been under development using multiple IBM 8100 processors and System Network Architecture protocols (IBM , 5, 7, 8), as shown by Fig. 14.5:

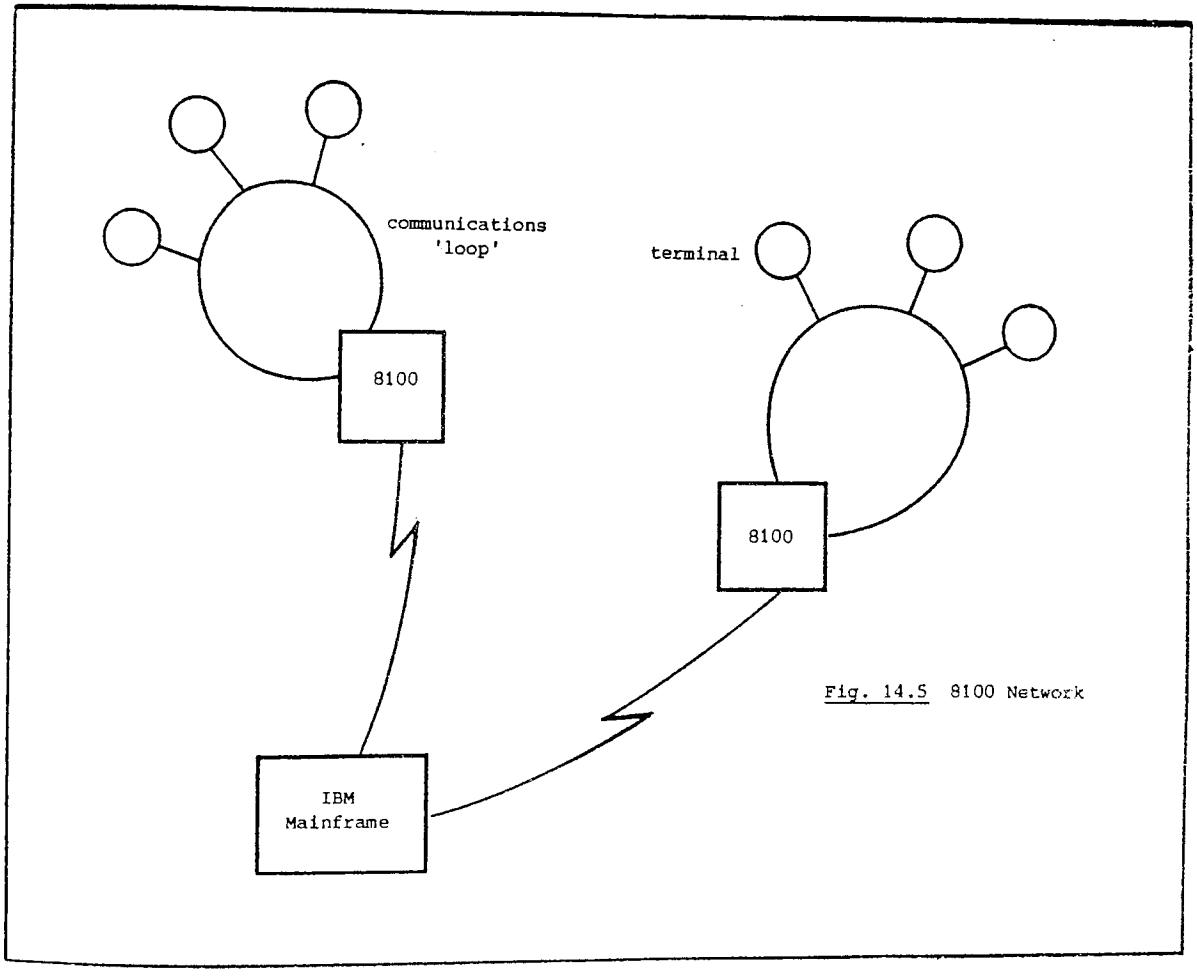


Fig. 14.5 8100 Network

Within this structure, any terminal on an 8100 loop can access systems resident locally, or centrally on the mainframe. Furthermore under DPPX, the 8100 operating system (IBM,45), locally-resident software can interface automatically with central on-line systems. This could be used within the SCHOOL context as follows:-

Local 8100 would support:

- Keyboard Response Evaluation Subsystem
- some immediate commands

  (e.g. *RESCREEN, *CURRENT)
- text compression (for local authoring)
- Display Format Subsystem

  (text transmitted compressed, expanded, formatted and displayed locally. This would produce large transmission reductions).
- some aspects of Tutorial Logic Control Subsystem.

Central Mainframe would support:

- a full SCHOOL system which would provide the facilities not supported locally for 8100-based users, but would cater for normal access also.

A conceptually similar approach could be applied to communicating microprocessors, and experiments have shown that VM/CMS and SCHOOL can be accessed from a remote APPLE II system, and that central files can be down-loaded to APPLE diskettes. This could be applied as follows:-

(i) log microprocessor on to IBM mainframe;

(ii) access SCHOOL;

(iii) specify down-load operation plus appropriate parameters;

(iv) once down-loading is complete, exit SCHOOL and loggoff mainframe;

(v) study courseware locally under micro-based Tutorial Logic Control Subsystem.

There are certain reservations associated with this approach, not least of which is the higher skill requirement on the part of the user. Also the microprocessor will need to have an 80-column display, but the possibilities seem most attractive.

14.6 CONCLUSION

'Civilisation is not a collection of finished artefacts, it is the elaboration of processes'.

Jacob Bronowski.

Glossary

- OF ABBREVIATIONS

ACATS	Aston Computer-Assisted Teaching System
ACS	Author Control Subsystem
AP	Auxiliary Processor
APL	A Programming Language
ASCII	American National Standard Code for Information Interchange
BASIC	Beginners All-purpose Symbolic Instruction Code
BMT	Bit Map Table
CAI	Computer Assisted Instruction
CAL	Computer Assisted Learning
CAT	Computer Assisted Training
CBE	Computer Based Education
CBL	Computer Based Learning
CBT	Computer Based Training
CCT	Control Command Table
CDC	Control Data Corporation
CIR	Comment Information Record
CML	Computer Managed Learning
CMS	Conversational Monitoring System
CMT	Computer Managed Training
COBOL	Commercial Business Oriented Language
CPS	Conversational Programming System
CPU	Central Processing Unit
CRU	Computer Resource Unit
CSB	Current Segment Block
CWD	Common Word Dictionary
D/B	Database
DBTU	Data Base Transfer Unit

DCB	Data Control Block
DEC	Digitial Equipment Corporation
DECAL	Digital Equipment Corporation Author Language
DFS	Display Format Subsystem
DIOS	Database Input/Output Subsystem
DIS	Data Input Subsystem
DP	Data Processing
DPPX	Distributed Processing Program Executive
EBCDIC	Extended Binary Coded Decimal Interchange Code
EMS	Error Management Subsystem
FAT	Frame Analysis Table
FDB	Frame Data Block
FIS	Field Instructional System
FLR	Formatted Logical Record
FP	Foreground Processor
FRT	Frame Records Table
FSM	Full Screen Manager
FSCB	File System Control Block
GDDM	Graphical Data Display Manager
GDR	Glossary Definition Recprd
GMR	Global Message Record
GRT	Glossary Reference Table
HIR	Hint Information Record
IBM	International Business Machines
ICES	Immediate Command Execution Subsystem

ICL	International Computers Limited
IIS	Interactive Instructional System
IMS	Information Management System
I/O	Input/Output
ITS	Interactive Training System
ITT	International Telephone & Telegraph
KRES	Keyboard Response Evaluation Subsystem
LAR	Lesson Analysis Record
LCR	Lesson Control Record
MCR	Master Control Record
MVS	Multiple Virtual Storage
NCB	National Coal Board
PAR	Primary Answer Record
PDP	Programmable Data Processor
PF	Program Function
PL/1	Program Language 1
PLATO	Programmed Logic for Automatic Teaching Operations
PQR	Primary Question Record
RDEVBLOK	Real Device Block
RSX	Real-time System Executive
SCHOOL	System Controlling Hierarchical Organisation of Online Lessons
SCR	Subject Control Record

SCS	Supervisor Control Subsystem
SDT	Syntax Definition Table
SIB	Syntax Information Block
SMR	Subject Message Record
SNA	System Network Architecture
SPR	Student Performance Record
SRT	Student Reference Table
SSBLOK	Student Status Block
SSM	Shared Storage Management
STAIRS	Storage and Information Retrieval System
TICCIT	Time-shared Interactive Computer-Controlled Information Television
TLCS	Tutorial Logic Control Subsystem
TPT	Tutorial Pointer Table
TSCB	Tutorial Status Control Block
TSO	Time-Sharing Option
VDU	Visual Display Unit
VM	Virtual Machine
VM/SP	Virtual Machine/System Product
VSPC	Virtual Storage Personal Computing
XSCF	Extended Course Structuring Feature

references

REFERENCES

1. 'Information Management System - General Information.
IBM Systems Library, No. GH20-1260.

2. 'Storage and Information Retrieval System - General Information'.
IBM Systems Library, No. GH12-5114.

3. 'Virtual Storage, Personal Computing - General Information'.
IBM Systems Library, No. GH20-9070.

4. 'Virtual Machine Facility/System 370 - General Information'.
IBM Systems Library, No. GC20-1800.

5. J. H. McFadyen.
'Systems Network Architecture: An Overview'.
IBM Systems Journal 15, No.1, 4-23 (1976).

6. C. R. Blair and J. P. Gray.
'IBM Systems Network Architecture'.
Datamation 21, No. 3, 51-56 (April 1975).

7. 'Systems Network Architecture - General Information'.
IBM Systems Library, No. GA27-3102.

8. H. Lorin.
'Distributed processing : an assessment'.
IBM Systems Journal 18,.No.4. 582-602 (1979)

9. J. Fielden.
'The Financial Evaluation of NDPCAL'.
British Journal of Educational Technology,
Vol. 8. No.3. October 1977.

10. D. Bitzer and D. Skaperdas.
'The Economics of a large-scale Computer-based Education System: PLATO IV' Computer-Assisted Instruction, Testing and Guidance. (Editor W. H. Holtzman) Harper & Row, 1970.

11. H. T. Lippert.
'Computer support of instruction and student services in a college or university'.
Educational Technology, 11, 5, May 1971.

12. R, Hooper.

'An Introduction to the National Development Programme in Computer Aasisted Learning'.

British Journal of Educational Technology, Vol.8, No.3. October 1977.

13. G. Beech.

'Training Computers to Train'.

Business Computing, November 1980.

14. G. M. Silvern and L. C. Silvern.

'Computer-assisted instruction : specification of attributes for CAI programs and programmers'.

Proceedings of ACM National Conference, 1966.

15. P. Suppes and E. Macken.

'The Historical Path from Research and Development to Operations Use of CAI'.

Educational Technology, April 1978.

16. R. Hooper.

'The Learn Machine'

Personal Computer World, April 1979.

17. A. D. Flockhart.

'The Design and Implementation of a Computer-Assisted Instruction system in a University environment'.

University of Aston in Birmingham, M. Phil Thesis, November 1973.

18. D. L. Alderman, L. R. Appel and R. T. Murphy.

'PLATØ and TICCIT : An evaluation of CAI in the Community College'

Educational Technology, April 1978.

19. J. C. Baker.

'Corporate Involvement in CAI'

Educational Technology, April 1978.

20. Zinn K. L.

'Comparative Study of Languages for Programming Interactive Use of Computers in Education'

EDUCOM (microfiche), Boston, Mass.,February 1969.

21. Zinn, K. L.

'Requirements for Programming Languages in CAI Systems'

Educational Yearbook 1971/2, British Computer Society.

22. Mills & Allen Communications Ltd.

'Computer Based Training - COMBAT System Overview' Spring 1980.

23. 'Interactive Training System - Expanded Course Structuring Feature : Course Authoring Guide'.

IBM Systems Library, No. SH20-1472.

24. Tagg, W.

'Computer Managed Learning in Hertfordshire'

British Journal of Educational Technology, Vol. 8. No.3. October 1977.

25. Miles, R.

'Computers in Military Training'

British Journal of Educational Technology, Vol. 8. No.3. October 1977.

26. Denenberg, S. A.

'A Personal Evaluation of the PLATO System'

ACM Sigcue Bulletin, Volume 12, No.2. April 1978.

27. Box, C.

'Evaluating PLATO'

Business Computing, November 1980.

28. 'IBM System/370 - Principles of Operation'

IBM Systems Library, No. GA22-7000.

29. 'IBM OS Full American National Standard CØBØL'

IBM Systems Library, No. GC28-6396.

30. 'OS PL/1 Checkout and Optimizing Compilers : Language Reference Manual'

IBM Systems Library, No. GC33-0009.

31. 'VS Personal Computing (VSPC) :Writing Processors'.

IBM Systems Library, No.SH20-9074.

32. D. A. Huffman.
'A Method for the Construction of Minimum-Redundancy Codes'
Proc. I.R.E., Vol. 4D, No.9. 1952.

33. M. Wells.
'File compression using variable length string encodings'
The Computer Journal, Vol.15, No.4. Novermber 1972.

34. M. P. West
'A General Service List of English Words with Semantic Frequencies'.
Longmans, 1953.

35. G. Dewey.
'Relative Frequency of English Speech Sounds'
Oxford University Press, 1923.

36. Thorndike & Lorge.
'The Teacher's word book of 30000 words'
Columbia Press, 1944.

37. F.R.A. Hopgood.
'Compiling Techniques'
London : Macdonald, 1969,

38, F.R.A. Hopgood & J. Davenport.
'The Quadratic Hash method when the table size is a power of 2'
The Computer Journal, Vol.15, No.4. November 1972.

39. A. Mayne & E. B. James
'Information Compression by Factorising Common Strings'
The Computer Journal, Vol.18, No.2. May 1975.

40. 'Storage and Information Retrieval System/Virtual Storage (STAIRS/VS): Program Reference Manual'
IBM Systems Library, No. SH12-5400

41. 'OS/VS2 MVS Supervisor Services and Macro Instructions'
IBM Systems Library, No. GC28-0683

42. 'Interactive Instructional System : General Information Manual'.
IBM Systems Library, No. GH20-2446

43. 'Graphical Data Display Manager and Presentation Graphics Feature : General Information'

IBM Systems Library No. GC33-0100

44. 'Presentation Graphics Feature : Interactive Chart Utility : Introductory Course'

IBM Systems Library, No. SC33-0111

45. 'Distributed Processing Programming Executive (DPPX): General Information'

IBM Systems Library, No. GC27-0400

46. M. Peltu.

'Artificial Intelligence : the Key to the fifth Generation'

Datamation, January 1982.

47. 'Interactive Instructional Presentation System : Interactive Instructional Authoring System : Operations Guide (VM/370 - CMS)'

IBM Systems Library, No. SH20-2459.

48. 'Interactive Instructional Presentation System: Interactive Instructional Authoring System : Administrators Guide'

IBM Systems Library, No. SH20-2449.

49. J. Bronowski.

'The Ascent of Man'

Book Club Associates, 1976.