The development of on-line computing facilities for the KDF9 Part 1: COSEC—A single on-line console

By P. C. Poole and T. Lang*

This paper describes the first stage of a project to provide multi-access facilities in the Egdon system operating on a KDF9. Initially, on-line access was obtained through the single monitor console and a user could interrupt a background job to initiate such foreground tasks as program amendment and execution.

(First received April 1967, and in revised form December 1967)

Most people to-day are convinced of the benefits to be obtained from providing multi-access on-line facilities in a computer system. There is, however, still a great deal of debate about the price that should be paid for this mode of operation. In the early systems, which pointed computer development towards multi-access, a considerable overhead was incurred in swapping programs in and out of core to provide the quick response users expect from 'conversational' interaction. This overhead could be tolerated on the grounds that these experimental systems were dedicated to exploring the possibilities of on-line techniques, and would perhaps lead to the design of more efficient systems in the future. Such arguments, however, cannot be applied to justify the extension of a current system to include on-line access, particularly where the computer is already heavily committed in a batch processing mode of operation. Yet many groups have felt that such extensions should be made, and we, at Culham, share this view. Thus, the problem is to design a multi-access system in which overheads are kept to a minimum, but in which the user is provided with as many of the advantages of on-line access as possible. We are carrying out such a project for a KDF9 which operates under the Egdon system (Burns, Hawkins, Judd and Venn, 1966), and have adopted as the underlying philosophy the concept of providing interaction with a console only when absolutely necessary, i.e. the system will not be 'conversational' in all aspects of its operation but will interact with a user only when an error is detected. The first stage of the project is known as COSEC (Culham On-line Single Experimental Console) and it is the purpose of this paper to describe briefly the structure of this system, the techniques employed to implement it, and the way in which it has been used to develop multi-access facilities.

COSEC was developed primarily as a tool to assist in the implementation of the final system. It provides on-line access from the monitor console and enables a user to interrupt a background job to initiate such foreground tasks as program amendment and execution. There were a number of reasons for choosing to start with a single console, of which the following may be mentioned:

(*a*) it was the obvious first step in a 'bootstrap' operation to produce a multi-access system,
(*b*) the monitor console was already available and no additional hardware was required,
(*c*) much of the new software developed for the uni-access system would carry over directly to a multi-access environment.

No claims are made that COSEC is highly efficient in all aspects of its operation. Some of the problems encountered in adding on-line facilities to an existing system forced us to make decisions on the grounds of expediency only. Many of these difficulties have since been removed, and the fact that we have been able to make such changes has largely been due to the availability of COSEC itself. The first version of the multi-access system called COTAN (Culham On-Line Task Activation Network) is free of many of these inefficiencies, and represents a much closer approximation to our ultimate goal.

File storage system

On-line access usually implies the provision of facilities which allow console users to manipulate named data-sets or *files* held on a backing store. In COSEC, files are stored on the disc in the logical disc† assigned to the function of BLOCK SUBSTITUTION. Block (sometimes called data) substitution is a particularly powerful feature of the Egdon system which is equivalent to the macro facility found in some programming languages. If the supervisor of the system (hereafter called the Director) encounters a control card of the form *SUBSTITUTE <name> in the input deck, it will search through an index to the block substitution area. If the required name is found, the corresponding block of card images will be read from the disc and inserted in the input stream. The mechanism has been extended at Culham to allow control cards to be included in such a block, and it is therefore possible to store a complete Egdon job deck on the disc. The decision to use the

† The physical tracks on the disc are grouped into units called 'logical discs'. A particular function of the system is associated with each of these logical discs. The KDF9 disc has a capacity of 4×10^6 words and an average access time of 250 msec.

* *Computing and Applied Mathematics Group, Theory Division, Culham Laboratory, U.K.A.E.A., Abingdon, Berks.*

block substitution area for the storage of on-line files meant that no modifications to the existing system were required to allow such files to be accessed by a job in the background stream. However, this run-time access is read-only, as blocks may not be altered except during a special session when the systems program, DISC UPDATE, is in operation. Further, the access is unrestricted, since any program may read any block. This situation is unacceptable in an on-line system where a user must be able to alter the contents of a file from the console. Hence, it was first necessary to impose a structure on these blocks to define the relationship between users and files, and thereby to set up the appropriate mechanisms for file protection.

An on-line file consists of two consecutive blocks in the block substitution area; the first, known as the *file head*, contains descriptive information about the file; the second, the *file body*, contains the actual information stored in the file. Both the file head and the file body may be manipulated from an on-line console, providing the user has the authority to do so. Part of the former is, however, protected by the system and contains the following information:

(*a*) the name of the file and of its owner, the date on which it was created;
(*b*) the format in which the information is stored in the file body;
(*c*) the size of the file body and the amount of space reserved for it. Since there is as yet no mechanism in the Egdon system for the dynamic allocation of disc space, sufficient space must be reserved for a file when it is created to allow for possible future expansion. A file may, however, be moved if it has outgrown its current allocation;
(*d*) the number of read accesses made to the file and the time at which the last read occurred;
(*e*) the number of times the file has been altered, the name of the user who last altered it, and the time at which this occurred.

The remainder of the file head, which may be amended by the file owner, determines who may access the file and in what manner. Read-only access is controlled by a READ status and read-write access by a WRITE status. If the value of a status is PUBLIC, then anyone is permitted to use the corresponding method of access; if PRIVATE, then only those users whose names appear in an associated list in the file head may address the file. This list also indicates the type of access permitted each user, and summarises the frequency and last occurrence of each such action. Absolute security is provided for the file by setting its WRITE status to PROTECT. Even the owner of such a file cannot alter its contents without first changing the status back to PRIVATE. Since the information defining file status is accessible to the file owner, any status value is interpreted in a 'fail-soft' manner; i.e. any unrecognisable value is assumed to be PROTECT.

One of the files owned by the Operations Manager contains a list of the names of persons authorised to use the system. Associated with each name is a password and an upper limit on the amount of disc space available to the user for permanent storage. This file, which is called the ONLINE INDEX, also holds statistics describing the use each person has made of the system (e.g. time at console, c.p.u. time, disc space in use, etc.) together with an overall summary. The Operations Manager can, at any time, examine the state of the system by listing the ONLINE INDEX on the console. By using the standard amendment facilities, he can alter this file to control the way in which the system is used. Some of the command programs also access the ONLINE INDEX to use the information stored in it or to update the statistics.

Any system providing on-line access for a number of users must contain back-up facilities for preserving files on some medium other than the disc. Initially the standard Egdon programs DISC UPDATE and DISC PRIME were used to carry out this operation. One of the actions of the former program is to copy the BLOCK SUBSTITUTION area on to magnetic tape. In the process, new files can be added from the card reader and existing files deleted. The second program merely copies this tape back to the disc.

User interface

The on-line user communicates with COSEC via the monitor typewriter. The system first types the query **'name;'** to which the user replies with the name of one of the command programs. If the command is not recognised, the user is informed and the query repeated; if the command is legal, the system responds with the query **'data;'** and the user supplies any information required by the program. When the data is complete the system calls in and enters the appropriate command program. Output is returned to the user at the console and the above cycle re-entered. Facilities are provided for cancelling excessive output or incorrect input and for editing the input character stream.

It should be noted that the background job stream continues to operate while the console is involved in any of the input/output transfers mentioned above. Considerable time could therefore be wasted if the console was required by the system while the user was typing in a large amount of data. For this reason, a small hardware modification was made to enable the Director to produce both audible and visual signals warning the user that it has output for the monitor. The user can then release the console, either by cancelling the input or by typing appropriate characters which cause the input to be preserved. When the output from the background system is complete, the current query is repeated and the user is then able to continue from where he left off. These interruption facilities may also be invoked if the machine operator wishes to use the console.

In addition to the mode described above, the COSEC system may also be operated by a sequence of com-

mands and data input through the paper tape reader. In this mode, output can be directed either to the monitor or to the paper tape punch. In the latter case, the output tape is passed through a mechanical reader and printed on a Flexowriter in the data preparation room. Hence the system may be used in an off-line manner to provide a high priority service on an interrupt basis for a number of users who require short turn-round times for program development.

Commands

A list of the commands available together with their respective functions is given in **Table 1.** The number of commands has been kept to a minimum and many of those which one normally associates with file control are implied in the ability of a file owner to amend a file head.

After successfully logging in, a user will normally open a file, amend it, cause it to be executed, and selectively scan the output stored on the disc. He may then continue to cycle through the commands AMEND, RUN and PRINT until satisfied that the program is working. (At any time, he can issue a variation of the RUN command which causes the system to produce hard copy output on the line printer and card punch.) At the end of the session he will usually save the file before logging out. Needless to say, the commands have a hierarchal structure with the necessary protective interlocks, e.g. a file cannot be amended until it has been opened. An example of a demonstration session at the console is shown in **Table 2.** The background job in operation at the time did not use the monitor and all console messages in this table relate to the on-line system.

The complementary commands, OPEN, and SAVE, are two of the most important in the available set. OPEN, after first checking the user's right of access, copies the named file into another area of the disc called the file nesting store (FNS). In analogy with a push-down stack, the only directly accessible file in the FNS is the one in the top cell. The concept of a file nesting store has proved extremely useful in the situation where on-line facilities are being provided within the framework of an existing system. Information to be accessed on-line must still remain available to parts of the system already in existence. The format of this information is therefore fixed and any change might necessitate extensive modifications to the current system. Further, the information may exist in a number of different formats. Thus, the commands which move information between the FNS and other areas of the disc can provide the necessary mechanisms for transforming all the different formats into a standard one. Hence, any command which operates on the FNS need only be designed to accept files in the standard format. Further, these commands are address-less in the sense that once a user has 'opened' a file, he can continue operating on it with various commands until he decides to change to another file. If the file in the top cell of the nesting store is inadvertently destroyed, it can always be recovered from the permanent storage area by means of the OPEN command. Conversely, the 'opened' file may be saved from time to time either in its original position or in another temporary file thereby ensuring that an up-to-date copy is always available. Another advantage of the concept is that space in the region of the disc assigned to the file nesting store may be allocated dynamically, and hence economically, in a very simple manner.

The RUN command enables the foreground user to interrupt a background job and initiate execution of the 'opened' file. All the facilities of the EGDON system are available to a foreground job since it is processed by the standard system programs. However, these normally produce a considerable quantity of output on the line printer, and it would have required a major effort to modify them so that a selection of this information could be sent directly to the console. Hence, the solution adopted in COSEC is to file all output on the disc in the top cell of the FNS. The user is informed when the on-line run has finished and can then use the PRINT command to scan this output file as many times as required. The parameters supplied as data to this command specify the type of output and the basis on which it is to be selected. For example, a parameter of the form

$$\langle\text{character strings}\rangle/n/m$$

indicates that all lines which commence with a specified character string are to be output together with the n preceding and the m succeeding lines. Thus to obtain all failure messages from the compilers at the console, it is only necessary to issue the PRINT command with parameters:

$$\langle\text{FAIL}\rangle/1/1$$

The results obtained from this process are illustrated in Table 2. Since the PRINT command labels all output with an absolute line number, it is a simple matter to extract any piece of information from the output file.

Implementation

As mentioned previously, COSEC was the first stage of a project to provide multi-access to an Egdon KDF9 and is currently being used in the development of the final system. However, COSEC itself was implemented by a bootstrap technique. Using the background job stream, we expended about 3 man-months of effort to produce the following:

(*a*) simplified versions of OPEN and SAVE which ignored file structure and ownership:
(*b*) a RUN command with direct output facilities only;
(*c*) the commands UPDATE, CHECK, RESET, CRDEDIT;
(*d*) the necessary extensions to the existing Director to control the system.

We then used the primitive form of on-line access afforded by these facilities to develop the remainder of the system. This resulted both in a saving of effort and an increase in efficiency, since job turn around time was reduced from hours to a few minutes.

Table 1

Commands currently available in COSEC

CLASS	NAME	FUNCTION
User Identification	LOGIN	Checks that the person at the console is an authorised user and, if so, initialises the system.
	LOGOUT	Updates statistics in the ONLINE INDEX, outputs a summary of the on-line session and of the state of the user's files, resets system in readiness for next user.
File Maintenance	CREATE	Allocates space for a new file provided user's allotment of disc space is not exceeded.
	INITIAL	Informs on-line system of the existence of a new file, if format is correct. The file may have been established via the CREATE command or loaded to disc during a DISC UPDATE run.
	DELETE	Deletes file from on-line system and marks it to be removed from disc during next DISC UPDATE run.
File Manipulation	OPEN	Locates a named file, checks that user is allowed read access and, if so, copies file into first cell of file nesting store. Also provides facilities for concatenating files.
	SAVE	Copies the 'opened' file from the top cell of the FNS, over-writing a named file after first checking that user has write access to this file.
	RESET	Empties the file nesting store.
	AMEND	Edits the 'opened' file on the basis of character strings. Also provides facilities for listing the file or part thereof.
	CRDEDIT	Edits the 'opened' file on the basis of card numbers. Also provides facilities for renumbering a file and for merging or comparing two files.
Job Execution	RUN	Causes the 'opened' file, if it constitutes a job deck, to be processed by the Egdon system. Output is filed on the disc in second cell of FNS.
	NJCRUN	A special command, similar to RUN, used for compiling system programs.
	PRINT	Interrogates output file selectively on the basis of line numbers and character strings and prints results on the console.
System Maintenance	UPDATE	Informs on-line system about command programs currently available.
	CHECK	Used for system debugging since it activates and tests all facilities in the on-line supervisor.

Many of the decisions made during the implementation of COSEC were dictated by the existing structure of the EGDON system rather than by considerations of efficient design. The main problems occurred with the Director, which is a large (6K) non-segmented program residing permanently in core store. Its size could only be increased by a few hundred words before existing users with large programs were seriously inconvenienced. Further, the way in which the system addressed the disc complicated the problem of interrupting a background job to initiate one in the foreground.

COSEC consists of 4 main elements and these will now be described briefly.

(*a*) *Console Control Package* (*CCP*)

This is a small subprogram added to the existing Director, which is multi-programmed along with other Director subprograms and the background job. Its main functions are to control the flow of messages to and from the console and to call the On-Line Supervisor into core when required. Whenever the CCP is held up waiting for an I/O transfer to terminate, the processing

Table 2

Demonstration session of the COSEC system

CONSOLE MESSAGES	COMMENTS
name;LOGIN.→ data;AD1 /ADEMON/ADHJPP/.→ <o> 21/01/67 11.15 SPACE RESERVED - 000100 BSU SPACE ALLOCATED - 000096 BSU→ name;OPEN.→ data;COMTAB/AD.→ <o>	USER A.DEMON LOGS IN ,SUPPLYING PASSWORD CONTAINING NON-PRINTING CHARACTERS, HIS NAME AND JOB CODE. ALL USER INPUT FOLLOWS THE ; CHARACTER AND IS TERMINATED BY → (END MESSAGE). THE REMAINDER OF THE CHARACTERS ARE OUTPUT BY THE SYSTEM.
OPEN CLASS A RESERVED SPACE 000020 BSU FILE SIZE 000013 CARDS LAST UPDATED 21/01/67 11.09→ name;AMEND.→ data;P .→	FILE COMTAB/AD IS OPENED. (1 BSU = 10 WORDS)
<o> CONTROL/WV95/WO/* * JOB090901/PPSPAA/SYSTEMS/POOLE/20 * XEQ * DISC PROGRAM ONLINEDEMON * CHAIN 1 * FORTRAN SUBROUTINE TAB(X,Y,XMIN,XMAX,NPTS) XMIN=0.0 XMAX3.0 NPTS=4 Y=X** RETURN END new size 13 → name;RUN.→ data;.→	FILE IS LISTED VIA AMEND COMMAND. FILE IS AN EODON JOB DECK WHICH CONTAINS A FORTRAN SUBROUTINE TAB. AT EXECUTION, THIS WILL BE COMPILED AND COMBINED WITH A PROGRAM ONLINEDEMON WHICH IS STORED IN RELOCATABLE BINARY ON THE DISC. PROGRAM TABULATES Y = F(X) AT EQUAL INTERVALS WITHIN A RANGE AS DEFINED BY XMIN, XMAX AND NPTS.
-→-→-→-→-→-→-→ TINT WV95→ UNPAUSE	BACKGROUND JOB IS SUSPENDED AND OPENED FILE IS PROCESSED IN THE FOREGROUND STREAM.
*11.17START JOBORGANISER *11.17 BEGIN JOB090901/PPSPAA/SYSTEMS/POOLE/20 UNCOMPILED ROUTINE TAB PPSPAA END RAN/EL/000M06S/000M14S TINT WO→ PAUSE -→-→-→-→-→-→-→	SUBROUTINE TAB FAILS TO COMPILE AS IT CONTAINS ERRORS IN SYNTAX.
<o> ON LINE RUN COMPLETE→ name;PRINT.→ data;L2 <FAIL>/1/1/.→	BACKGROUND JOB IS RESTARTED.
<o> 0026 XMAX3.0 0027 FAILURE 333, 0 0028 333 UNIDENTIFIABLE KEYWORD 0030 Y=X** 0031 FAILURE 200, 6 0032 200 SYNTAX ERROR (E.G. *+) →	COMPILER FAILURES OBTAINED FROM THE OUTPUT FILE STORED BY THE RUN COMMAND ON THE DISC.
name;AMEND.→ data;BXMAX KAXI= CN = X**5 .→	FILE IS AMENDED. UNDERLINED CHARACTERS ARE DIRECTIVES. E.G. B COPIES LINES TO BEFORE LINE COMMENCING WITH SPECIFIED CHARACTER STRING X COPIES CHARACTERS IN CURRENT LINE TO AFTER SPECIFIED CHARACTER STRING I INSERTS SPECIFIED CHARACTER STRING AT CURRENT POSITION IN CURRENT LINE
<o> XMAX=3.0 Y=X**5 new size 13 → name;RUN.→ data;.→	
-→-→-→-→-→-→-→ TINT WV95→ UNPAUSE	
*11.20START JOBORGANISER *11.20 BEGIN JOB090901/PPSPAA/SYSTEMS/POOLE/20 PPSPAA END RAN/EL/000M06S/000M15S TINT WO→ PAUSE -→-→-→-→-→-→-→	FOREGROUND JOB PROCESSED SUCCESSFULLY.
<o> ON LINE RUN COMPLETE→ name;PRINT.→ data;L2 <BEGINDEM><ENDDEM> .→	PARAMETER CAUSES PRINT COMMAND TO OUTPUT ALL LINES FROM LINE COMMENCING WITH CHARACTERS BEGINDEM TO LINE WHICH STARTS WITH ENDDEM.
<o> 0066 BEGIN DEMO 0067 I X Y 0068 1 0. 0. 0069 2 0.1000$_{10}$ 01 0.1000$_{10}$ 01 0070 3 0.2000$_{10}$ 01 0.3200$_{10}$ 02 0071 4 0.3000$_{10}$ 01 0.2430$_{10}$ 03 0072 END DEMO →	RESULTS OF TABULATION. Y = X↑5 FOR X = 0.0(1.0)3.0
name;LOGOUT.→ data;.→ <o> ADEMON LOGGED OUT AT 11.24.02 TIME ONLINE 00526S→	USER LOGS OUT. COMPLETE SUMMARY OF SESSION IS SENT TO THE LINE PRINTER.

of jobs in the background stream continues. At the end of an output transfer, the CCP either fetches the next section of output from the fixed head disc* for transmission to the console, or returns to the input state; when an input transfer finishes, the CCP either writes the input to the fixed head disc and calls for more input, or fetches the on-line supervisor into core. The latter action involves suspending the background job and preserving a section of it on the fixed head disc, since the on-line supervisor and the command programs operate in the user area of core.

(*b*) *On-line Supervisor* (*ONSUP*)

The on-line supervisor which is compiled with the Director, is written away to the fixed head disc whenever the Director is loaded into the machine. ONSUP is the chief executive program of the COSEC system and carries out the following functions:

(i) checks the validity of the command input by the user;
(ii) edits the data input from the console;
(iii) calls down from the disc the appropriate command program or, if a foreground run has been requested, the On-Line Director;
(iv) supervises the running of a command program.

(*c*) *Command programs*

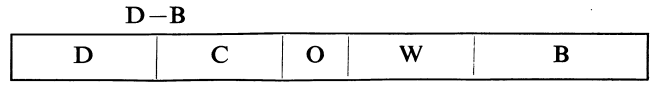
The command programs function in supervisor mode and are essentially segments of the Director. Each program is, however, independently compiled, and it is therefore a simple task to add new commands to the system as required. Communication between a command program and the remainder of the system is carried out via a software interface. At entry, a command program is supplied by ONSUP with the base address of a list of addresses. Each entry in this list may be the address of a subroutine, the address of an I/O buffer or the base address of a list of parameters. The subroutines carry out functions common to all command programs, e.g. disc transfers, clock update, message output, allocation of work space; the parameter lists provide the permanent storage for information which has to be transmitted from one command program to another. **Fig. 1** illustrates the map of core store when a command program is being obeyed.

When a new command program is to be added to the system, it is first checked-out on-line in user mode with a set of routines which simulate the action of the interface and the on-line supervisor. This technique has proved to be very effective, as few errors have been encountered when these programs were eventually incorporated in the system.

(*d*) *On-line Director*

Whenever a RUN command is detected, ONSUP dumps the background job and Director on the disc. It then reads in a second copy of the Director which has been modified to accept input from the disc (the opened file), and to return output to the top cell of the file nesting store. This Director also contains a different allocation of space on the disc. Hence, job-dependent logical discs may be duplicated, and the background job can be interrupted at any time. Once the On-Line Director has been provided with the date and the current state of the clock, it processes the foreground job automatically, requiring no manual intervention unless there is an error in the control information supplied with the job. When the foreground run is completed, ONSUP is recalled and it, in turn, restores the background job stream.

In implementing COSEC, we used the technique of calling in a second version of the Director, because it enabled us to provide facilities for interrupting a background job to initiate one in the foreground with a minimum of effort. However, the process is very inefficient, with a poor response, since a large amount of core must be swapped. Further, it would be unacceptable in a multi-access situation since it prevents commands from other consoles being serviced.



D : Director
C : command program
O : on-line supervisor
W : working space for command program
B : residual part of suspended background job
D−B : normal boundary between Director and background job.

Fig. 1. Map of store when a command program is being obeyed

Operational experience

COSEC has been fully operational since April 1966 and represents about 12 man-months of effort. Its use has been restricted mainly to the Systems Programming Section, although it has been made available to a limited number of application users on an experimental basis. It is generally felt by users that the system is somewhat confusing to operate when the background job stream is making heavy demands on the console. The continual interruptions make concentration difficult, and users tend to choose periods when long jobs are being processed. However, as an example of the advantages offered by the system, we can compare the times required to check out two of the command programs. CRDEDIT, which was developed in the background stream, required 2 weeks of testing; LOGIN, a program of similar complexity which was developed on-line, required only 2 days.

The response time of COSEC is very short, as would be expected with only one console. For all commands except RUN, the overhead due to core swaps is about 2 sec per command. Thereafter, the response time is a function of the total size of the disc transfer involved,

* The fixed head disc has a capacity of 3840 words and an average access time of 30 msec. Part of it contains the on-line supervisor while the remainder is used as an I/O buffer and for core dumps.

since all these commands are disc limited; e.g. to open at a file of 500 cards takes about 2 sec at the average disc transfer rate of 5000 words/sec. No response is obtained from the RUN command until about 10 sec have elapsed. There is a similar delay at the end of the foreground run when return to the background system occurs.

When the first COSEC commands were written, it was felt that users would require as much output information as possible—e.g. for OPEN, the name of the file, the reserved and occupied sizes, the format of the contents, and the time and date when the file was last updated. However, a little experience showed that such verbosity was extremely irritating particularly to the more experienced users. In COTAN, output of such information has been greatly reduced and may be suppressed entirely if specifically requested by the user. On the other hand, during the input phase the user does feel the need for more frequent prompting than just the **'name'** and **'data'** queries of COSEC, and this feature has been considerably enhanced in COTAN.

It is of interest to compare the use of the two editors available in COSEC. The line number editor was available first, but as soon as the context editor was written, users quickly showed their preference for this mode of operation. This command has proved very attractive and convenient to use, and was well worth the labour involved in writing it. The line number editor still finds some use for large scale modification of files, e.g. merging of two files.

It should be noted that COSEC does not embody a full implementation of the philosophy 'interaction only when absolutely necessary'. It is a step in this direction, since a user may type in a large amount of data before actually calling on the system to obey the command. However, if the data is incorrect, the only interaction currently provided is to ask the user for another command. Any data following the offending section is lost. Thus, users tend to break up lengthy input sequences into a number of short sections to avoid the possibility of having to retype. However, this situation has been corrected in COTAN, and the facilities available in this system which allow a user to input a large amount of data with confidence will be described in a later paper.

COSEC does not attempt to handle the problem of interaction between a user and his own program. Clearly this is possible, since there is only one console, but it is obviously a very inefficient use of the machine. The problem is still with us in COTAN, but interaction is allowed if the program is small (<8K).

Conclusion

In a computer with a comparatively slow backing store, the cost of allowing complete user interaction in all phases of on-line operation is probably prohibitive. However, if such interaction is restricted mainly to file manipulation and then provided only when absolutely necessary, the resulting system can be attractive and convenient to use while still being reasonably efficient. COSEC has illustrated the feasibility of such an approach to on-line access within a batch processing system, in addition to providing a very powerful tool for future development.

Acknowledgements

The authors wish to thank Dr. K. V. Roberts and Dr. K. W. Morton for many helpful discussions and suggestions. We are also indebted to Mr. G. Bernau and Mr. D. C. Carpenter who coded and tested some of the command programs.

Reference

BURNS, D., HAWKINS, E. N., JUDD, D. R., and VENN, J. L. (1966). The Egdon System for the KDF9, *Computer Journal*, Vol. 8, p. 297.

Book Review

The Design and Analysis of Scientific Experiments, by K. C. PENG, May 1967; 217 pp. of text, 29 pp. of appendices. (New York: *Addison-Wesley Publishing Co.*, 72s.)

This is a theoretician's book, being mainly concerned with the analysis of variance in factorial experiments. Described by the author as 'designed for statisticians, computer programmers and persons engaged in experimental work who have some background in mathematics and statistics', this book does not seriously compete with the well-known texts with similar titles.

Statisticians will find little in this book which is not discussed more thoroughly elsewhere, nor will they find references to recent work in this field. Students of statistics may, however, obtain profit from studying the middle chapters on factorial design where the author is at his best.

For computer programmers the main interest will be in the use of the operator calculus for partitioning sums of squares in balanced factorial schemes. Three FORTRAN programs are supplied in the appendices (i) for complete factorial designs (ii) for Latin squares and (iii) for 2^n fractional replicates. By modern standards these programs are very limited in their conception of the users' needs (see e.g. Yates, F. and Anderson, A. J. B. (1966) *Biometrics* Vol. 22, pp. 503–24), and would require considerable extension to be of much practical use. Real users want to present data in a way that suits themselves and not the programmer, and want a readable output over which they have some control, ideally an output which could be published without further explanation. A general program should be able to derive combinations of variates, to estimate missing data automatically, to compute nominated orthogonal effects etc.

Experimentalists will not be greatly helped by this book as there is little discussion of the reasons for choice of design or of the meaning of the analysis. The numerical examples are all fictitious, and the multiple summation formulae are not well suited for instructing laymen in the computation of analyses.

G. J. S. Ross (Rothamsted)