Flexibility in the man–computer interface: a generative solution

This item was submitted to Loughborough University’s Institutional Repository by the/an author.

Additional Information:

- A Master’s Thesis. Submitted in partial fulfilment of the requirements for the award of Master of Philosophy at Loughborough University.

Metadata Record: https://dspace.lboro.ac.uk/2134/26908

Publisher: © Ossama K. Muslih

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 2.5 Generic (CC BY-NC-ND 2.5) licence. Full details of this licence are available at: http://creativecommons.org/licenses/by-nc-nd/2.5/

Please cite the published version.
This item was submitted to Loughborough University as an MPhil thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

![Creative Commons Licence](image)

**Attribution-NonCommercial-NoDerivs 2.5**

You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

- **Attribution.** You must attribute the work in the manner specified by the author or licensor.

- **Noncommercial.** You may not use this work for commercial purposes.

- **No Derivative Works.** You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the [Legal Code](http://creativecommons.org/licenses/by-nc-nd/2.5/)

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
<table>
<thead>
<tr>
<th>AUTHOR/FILING TITLE</th>
<th>MUSLIM, O</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESSION/COPY NO.</td>
<td>113692/02</td>
</tr>
<tr>
<td>VOL. NO.</td>
<td></td>
</tr>
<tr>
<td>CLASS MARK</td>
<td>LOAN COPY</td>
</tr>
</tbody>
</table>

- 1 JUL 1990
- 2 JUL 1992
- 1 JUL 1993
- 12 JAN 1993
- 16 JUN 1996
- 6 JUL 1990
- 8 DEC 1989
- 31 MAY 1990
- 9 JUL 1992
- 9 JUN 1996
FLEXIBILITY IN THE MAN-COMPUTER INTERFACE -

A GENERATIVE SOLUTION

by

OSSAMA K. MUSLIH

A Master's Thesis
Submitted in partial fulfilment of the requirements
for the award of Master of Philosophy
of the Loughborough University of Technology
December, 1979.

Director of Research: PROFESSOR D.J. EVANS, Ph.D.,D.Sc.,
Department of Computer Studies
Supervisor: MR. L.N. BLACKBURN, B.Sc.,P.G.C.E.,
Department of Computer Studies

During the past few years a growing demand has been observed for "humanised" interfaces between computers and their users. Unfortunately, the humanising process is expensive in programming and machine running terms. However, to achieve this objective such costs must be anticipated. The reduction of some of these costs can be achieved if a method involving automatic techniques can be developed.

To obtain a specification for an interface capable of meeting the wide range of demands of the people using it, is not easy. Even if such a specification could be agreed, developing it with any high-level language would still be unsatisfactory if the interface had to possess unique application dependent features, i.e., customising for the next application would be a deterrent.

An alternative is to write a program which could generate the source version of an interface which need possess less "intelligence". Also, a series of separate, complementary interfaces could be generated for the same target system, each possessing features required by a specific type of user. The log-in mechanism would automatically select the appropriate user interface. This technique is not strictly an adaptive one but would appear so to the casual user. Each interface is rigid but any variation of it is available to the user. In this way "complexity" and "intelligence" is removed from a single interface at the expense of multiple copies all ostensibly different.

This project has successfully accomplished the aims of developing and testing a program (the generator) which is capable of creating another
program (the interface) containing features determined by the user for any general purpose application system consisting of a number of functional programs. The interface achieves the objective of bringing together logically and functionally related separate programs in such a way that the user sees the whole system in a more meaningful way. Also, the generated interface allows the user to achieve a high degree of familiarity of the system immediately at minimal programming cost since no programs are written manually.
I would like to express my thanks to Professor D.J. Evans, the Director of Research, for his encouragement and advice throughout the course of my research.

My sincere thanks and gratitude to Mr. L.N. Blackburn, my supervisor, for his invaluable help, advice and guidance throughout the course of my research.

My greatest thanks to the IRAQI Government (Ministry of Higher Education and Scientific Research) for the financial support during my study.

I would like to extend my thanks to Miss J.M. Briers, for her expert typing of this thesis.
# CONTENTS

## Chapter 1: INTRODUCTION

1.1 Origin Of Interest
1.2 Other Factors Which Bear Upon The Man-Machine Interface
1.3 Feasibility And Flexibility
1.4 Summary

## Chapter 2: APPLICATION PACKAGE DESIGN AND HUMAN INTERFACE CONSIDERATIONS

2.1 Typical Application Package Design
2.2 User Requirement For The Interface To Interactive Systems
2.3 Techniques Used In Designing An Interface
   2.3.1 Question And Answer Technique
   2.3.2 Menu Selection Technique
   2.3.3 Command Language Technique
   2.3.4 Monitoring

## Chapter 3: INTERFACE DESIGN AND THE GENERATOR

3.1 Introduction To Interface Design Techniques
3.2 General Principles Of Terminal Dialogue Design
3.3 Interface Developing Methods
3.4 The Objectives Of The Interface
3.5 Structure Of The Interface
Chapter 4: DEVELOPMENT SYSTEM AND SPECIFICATION

4.1 System And Language Used For The Development 41
4.2 RT-11 Operating System 42
4.3 RPL Development System 42
4.4 Coding The Generator 46
4.5 The Specification Of The Generator 49
  4.5.1 The Generator System Description 49
  4.5.2 Acceptance Criteria 52

Chapter 5: TESTING AND RESULTS

5.1 Introduction And Test Plan 53
5.2 Use Of The Interface Generator 54
  5.2.1 Planning 54
  5.2.2 Execution Of The Generator 58
  5.2.3 Compilation Of The Interface and Modification of Application Source Code 56
  5.2.4 Exercising The System 56
5.3 Complete Test Example 57

Chapter 6: CONCLUSION AND FURTHER DEVELOPMENT

6.1 Conclusion 83
6.2 Further Development Work 84
6.3 Summary 85

APPENDICES: A1 - User Guide 90
A2 - Program Listing of INTGEN,RPL: The Generator 102
A3 - (i) Generator Dialogue 117
(ii) Interface Dialogue (Complete System Test) 122
A4 - Generated Interface Listing - LCS,RPL 147
A5 - Example of Planning Stage 153

REFERENCES 156
CHAPTER 1

INTRODUCTION
1.1 ORIGIN OF INTEREST

The future growth of the computer industry and the acceptance of computer methods will depend largely on the successful establishment of effective man-machine communications.

In recent years the data processing industry (with a few academic exceptions) paid little attention to effective man-machine dialogue. Technicians have concentrated primarily on the efficient use of the control processing unit and the storage media. This is hardly surprising in view of the computers expense and remarkable capabilities. Where they have been used at all, terminals and their languages have often been an adjunct to the computer and its languages. Systems have been designed from the inside, out. Man must become the prime focus of system design. The computer is there to serve him, to obtain information for him, and to help him to do his job. The ease which he communicates with it will determine the extent to which he uses it. Whether or not he uses it powerfully will depend upon the man-machine language available to him and how well he is able to understand it. To be effective, systems will have to be designed from the outside, in, see ref. [1].

For many types of computer users, remarkably little has been done to provide an efficient man-machine interface as yet. It is essential that we provide these people with information only when they need it, and that we present information in the clearest and most digestible form. We must protect computer users from the vast and overwhelming mass of data that computers generate, but we must enable them to ask questions, solve problems, and explore the likely effect of different courses of action.

In designing the structure of the dialogues that take place between man and his computer terminal it is essential to take into account
the shortcomings of both man and the computer. Man is limited in what he can achieve at the terminal and accomplishments will vary very widely between different persons. Some will be highly trained in this interaction, and some not. Some will be capable of programming, but most will not. All will have a limited capacity to remember, and to respond logically, see ref. [2].

To utilize the potential of this technology, it is necessary for these two different creatures - man and computer - to be able to communicate with one another. Because of their different talents, their dialogue will not normally be symmetrical. The man will make different types of statements in the dialogue to the machine. In fact, often, the worst approach the system designer could take is to attempt to design the dialogue as though the man simply is not intelligent enough for his approach; or, to phrase that another way, the quantity of sequential logic that would have to be programmed to enable the computer to follow all of man's variations in syntax, vocabulary, and idea structure would be so immense as to be impractical today, see ref.[3].

Natural English dialogue, then, while it would form the best possible man-machine interface for man, is not practicable in most commercial computer systems today. It might become practicable in decades hence, but here we must be concerned with advancing from the current state of the art. The alternatives all involve people with their wide differences in training and capability. In many cases we will be building man-machine systems in which a variety of man-machine configurations are possible for achieving a given result. The system designer has some measure of choice as to that type of man he uses at certain points in the system, see ref.[1].
To establish a definite line of interest in the human aspects of computer usage [6],[7],[21] ranging from the problems of machine independent job control languages through to testing adaptive user interfaces, the user efficiency was studied in the area of job control usage in batch processing system. One may well ask, why have there been so many shortcomings? In particular, in view of the many fine statements about harnessing the combined potential of human and computer intelligence, why at present is man-computer communication so poor? With the complexity inherent in people and computers there will clearly be no one simple answer to this equation. Amongst the general issues which may be relevant in various applications will be the general education and specific training of the people involved, and the organisation and procedures of both the computer and the working system (e.g. business office or industrial factory) within which it is to work. Equally an important factor in this question must be the man-computer interface; what are its main features, is it satisfactory, and can it be improved? The man-computer interaction (MCI) see ref.[4] is now generally accepted to refer to direct, close coupled, computer usage by users with a job to do, even if their primary work is in a non-computing field (e.g. banking or piloting). The phrase is not generally used to include the work of those in large computer installations providing data processing services to "customers" (in much the same way as a generating plant provides electricity). However, the latter will not be excluded from consideration here; indeed many aspects of man-computer interface will be similar for them, especially those ergonomic (human factor) aspects which are of a routine nature. It is evident, then, that the interface must be any hardware or software feature with which any human may have to interact during MCI. So the interface comprises not only obvious hardware elements affecting an operator but also such aspects as good
documentation to assist with maintenance and fault finding. It is useful to consider separately the hardware interface and the software interface.

The hardware interface comprises the displays, controls, terminals, consoles and similar equipment having a fixed physical form.

The software interface comprises those parts of the man-computer communication medium which are not hardware; they are often more transitory and are usually variable by program control, for example, the logical structure of content and procedures, and the format, layout, verbosity, etc. of sequences of man-machine messages. In this project we are concerned with the software interface.

1.2 OTHER FACTORS WHICH BEAR UPON THE MAN-MACHINE INTERFACE

The limitations of interface equipment are now more widely recognized. For example, a special Delphi study was commissioned by the AFIPS Social Implications Committee to identify the important issues in multi-national computer systems; see ref. [5] assembled an outstanding Delphi Panel of 57 corporate officers, government officials and computer experts. Concerning equipment, 97% of the panel agreed that "it would be necessary to provide improvements in portable, on-line and inexpensive terminals as well as smarter and more flexible terminals". Further, many (92%) focused upon the man-computer interface, suggesting that improvements were needed in interface support facilities; in cheap, high speed hard-copy printers with unlimited character sets; and in more sophisticated man-computer interaction possibilities.
In the years (1975-1978) there has been some improvement in new equipment, for example quieter teletype and similar printers, cheaper video display units with better editing facilities, several new hand-held terminals. But, in essence we are still very little advanced, that man-computer interaction consists of two sophisticated computers with very limited communication links between them see ref. [10]. Alongside these developments there are even more demanding areas for improvement which because they are software orientated will be achievable quite readily.

The potential for development, and the present very limited state of progress, is even more evident when we consider all the possible communication channels into and out of the human.

From the experience of most people in this field it would be generally agreed that only a rather small number of specialist users can in any sense be said to be in close-coupled communication with computers. For most users the interface is a space frontier and time barrier rather than an open door to communication. Some of the reasons for the slow progress are to be found in the nature and responsibilities of the computer system designer.

Computer designers are primarily, and quite rightly concerned to improve the performance of the computer hardware and software; they often forget that what matters most is the efficiency and performance of the total man-computer system, of which their computer is only a part. Efficient performance can only result from proper attention to the needs and problems of people as well as to the hardware and software aspects. But the complexity and sophistication of modern computer technology often results in the designer being so busy with his own technical problems that he has too little time, and often too little knowledge, to deal with human problems
adequately. Moreover, the computer boom was so rapid and strong that the training of computer system designers could not, and still usually does not, include ergonomic and human factors. Unless those human factor problems are foreseen in the design stage then a sub-optimum system is the inevitable result, see ref. [4].

It is hoped that the overview position taken in this chapter will help to show the whole range of reasons which have delayed the improvement of the interface and thus limited the growth of man-computer interaction. Likewise, it is hoped that the integrated approach to all the various issues which may influence man-computer communication will provide a useful basis both for those involved in equipment development and for those who specify and select equipment for their applications.

The structure of man-computer interaction is represented by the synoptic view in Figure 1.1. The framework is simple but the contents are complex.

1.3 FEASIBILITY AND FLEXIBILITY

The idea behind a man-computer interface is to present the user with a flexible human-oriented method of communicating with a computer system. Users tend to make fewer mistakes if allowed to input information in a more "conversational" form. System messages tend to be unrelated to the interchange of information which is in progress between user and machine. This is due to the internal workings of the system in which the user is not interested, and can therefore cause feelings of frustration. It can therefore be seen that the interface could lead to a reduction of errors and therefore
FIGURE 1.1: A Synoptic View of Man-Computer Interaction

(Redrawn with permission from Ref. [4])
an increase in productive computing. It could result in more satisfied users with a greater willingness to use the computer as a tool to help in their work, and therefore there should be less abuse of the system, see ref. [6],[23].

This project has tried to accomplish the aims of developing a software tool which is intended to improve the use of any packaged system of programs. These programs may have originated from a software supplier or produced through an in-house development system. In both cases it is often true that the system lacks coherence through the absence of an effective interface. This interface should allow the human operator to be made fully aware of the capability of the system and thereby encourage its efficient use. The interface represents a system overhead which could be avoided if a suitable generator existed for its production. This would encourage more interfaces to be produced since the effort involved would be negligible. One further aspect worth emphasising is the flexibility of the interface, which is due to the interactive method of generation. User-supplied descriptions become part of the interface which provides immediate user acceptance and should conditions change in time, re-generation is as simple as generation.

This work has followed the development of a generator which automated, to a useful extent, the production of the required JCL, see ref.[7], for any major mainframe operating system, in handling simple data processing functions (data capture, edit, update, report, sort, etc.). It was expected that some of these ideas and methods could be extended into the interactive environment.

A recent acquisition in the Department of Computer Studies - Loughborough University of Technology, has been a small business system
comprising a processor, dual flexible disc drive, keyboard and printer
typical of the systems which are currently being installed in large numbers.
Naturally we considered both the local requirements for this system as well
as the more general requirements for a very significant and growing
commercial market. The system is a hybrid system consisting of the system
hardware, system software and documentation. The system hardware comprise
a PDP 11-03 processor. The system is typical of many commercially available
in the price range £5000-£10,000 (1978 prices).

The system software is the DEC originated RT-11 (Version 3C) which
comprise a single job monitor (FDMNSJ) see ref.[8]. Within the operating
system a user may load a run-time system for developing application systems
in a language called Report Programming Language (RPL); this language is an
interactive strain of FILETAB which is in very common use, see ref.[9].
The interface generator was developed with this language.

This project has successfully accomplished the aims of developing
and testing a program (the generator) which is capable of creating another
program (the interface) containing features determined by the user for any
general purpose application system consisting of a number of functional
programs. The object here is to bring together logically and functionally
related separate programs in such a way that the user sees the whole system
in a more meaningful way. Also, it is to remove from the user his
dependence upon external documentation which in the early stages of usage
may be very high, and to achieve a high degree of familiarity of the system
immediately at minimal programming cost, since no programs are written
manually. It is believed that the method chosen by this research to
implement an interface using a flexible generative system satisfies both
objectives, i.e. can be generally applied to any commercial package and can
be particularly applied to generate an interface suitable for the local needs of project students using the small business system in the department.

1.4 SUMMARY

We propose to set out a description of a feasible method of achieving a flexible human interface to any computer system at relatively little cost to user. Since the interfaces can be generated locally it will be the user who determines the superficial aspects of the interfaces and who therefore will gain from the immediate familiarity of them.

Chapter 2 looks into existing application package design and human interface considerations.

Chapter 3 will consider in more detail the interface design technique and the general principles of terminal dialogue design.

Chapter 4 will consider in more detail the development of the system and specification. Also, the RT-11 operating system and the language used for the development are considered.

In Chapter 5, the state of the project, testing methods and methods of assessment are described. Throughout this development, testing was carried out on the selected applications and the results of these have been presented in the Appendix.
CHAPTER 2

APPLICATION PACKAGE DESIGN AND

HUMAN INTERFACE CONSIDERATIONS
2.1 TYPICAL APPLICATION PACKAGE DESIGN

For a computer to be used productively there must exist a program that will process data to achieve that required result. The objective of the application package must be to achieve programmed computer operation by a more efficient route than would be achieved using other techniques. Efficiency is a ratio of input to output and in this case the output (being the operation of the required computer program) must be assumed to be equivalent for each of the possible means of achieving the required end, see ref. [11].

Some years ago application packages provided by independent software houses were few in number, covering limited application areas, somewhat rigid in their design and usually produced by computer experts rather than application specialists. It was normal for firms acquiring these systems to change their commercial methods, whether this way was desirable or not, to suit the application package, which naturally gave the appearance to the uninitiated that the needs of the business were being adapted to fit the computer rather than hardware being a tool for furthering commercial activity. Maintenance was often available only for the correction of errors. In some cases, unless the user was prepared to wait for days to have a bug corrected, in-house expertise was necessary to look through the listing (if available) and dumps. Today, packages are flexible, often modular, and can be readily adapted to suit most business differences in the application area covered. Now packages and computers can be used to suit the business.

This is particularly important to analysts who can now concern themselves with user problems and therefore develop better information systems. Also, full documentation is usually available, ranging from user
manuals through to operation manuals, see ref. [12].

A common difficulty with the application package selection occurs when a solution is sought without the problem having been defined. It is possible to produce a formal functional specification of requirements based on adequate study. Such a specification will include relevant sections; for example, dealing with problem background, system objectives, related (or affected) systems, general approach (batch, on-line), I/O and file (volumes, sources), logic, hardware/software constraints, security, resource/cost/time requirements, etc. The specification can form the basis for estimating the relative costs and benefits to the installation. It can also help to determine what features a package must have and which are merely desirable. The understanding of features in this sense is important in the selection process.

Some aspects of the requirements specification may be unique to a particular envisaged package, for example, if the aim is to define what results were expected and what action will be taken in each eventuality.

The application is analysed into a set of fairly orthogonal problem areas. Each problem area, where possible, shares common features such as database or files and will resolve in solution in a small number of program modules. The total set of modules together with operational and reference documentation makes up a typical application package.

The operational features are of particular interest because it is here where the system is judged to be 'good' or 'bad' and where the interface is to be introduced. Typically a system will be driven by a command language which requires the support of a fairly heavy manual or with no help at all, in which case the user via the reference material is
able to schedule modules singly and in sequence to achieve the general function in progress, see ref. [13].

Packages designed to sit in an interactive environment may provide further assistance. However, the interfaces themselves require familiarisation and may contain features which are awkward or irrelevant to the user.

**USER REQUIREMENT FOR THE INTERFACE TO INTERACTIVE SYSTEMS**

The research reported by [14] gives a number of attributes of "naive" computer users.

a) They are only interested in those aspects of the system that serve their needs.

b) They neither have nor wish to have a deep knowledge of the system which they are using.

c) They wish to minimise both the time and effort required to use the system.

d) They need to be protected from unexpected events which do not relate to the problem they are trying to solve.

e) As they become more familiar with the system they will expect to be able to reduce the amount of information they need to supply to get a task done.

f) If they can bust it they will.

This list should apply to all computer users no matter what level of experience, since all users, however experienced, are knowledgeable in detail about only certain aspects of a system. Other, less used parts of a system will put heavier demands on the user. Typically the programmer
who is writing and testing a suite of programs will probably not be an expert on the detailed operation of the editor, the compiler, the link editor, or the operating system, all of which he will be using extensively, e.g. surveys show that users of job control languages tend to acquire a useful subset of commands and then keep to those which they know work rather than make an effort to learn new and perhaps better methods of achieving the results, see ref. [19]. The perceived effort, in finding out the necessary information to enable a new feature to be used, is substantial, and the perceived benefits to the user are negligible since he can manage to achieve the same results without needing to expend any special effort.

An additional attribute of people in general not mentioned directly by [14], but relevant to the use of computers, is the increase in the care people take and in the effort they expend when they feel that they have a positive choice and that their actions will affect the outcome.

Given these attributes a number of criteria, can be enumerated, see ref.[6]. These will represent the factors on which an interactive system will be judged by its users.

1. The ease with which a user can start using the system.
2. The degree to which the system is self-contained i.e. does not force the user to access manuals and other auxiliary material.
3. The amount of information that the user must submit to get the system to complete his task.
4. The protection the user has from unexpected system actions.
5. The flexibility with which the system interface becomes less verbose as the user becomes more knowledgeable.
6. The degree to which the user can control the progress of the interaction.
If it were to conform completely to these criteria an ideal interactive system would be fully self-contained, introducing itself, and all the facilities that were relevant to each new user. The commands, and responses would need to be exactly suited to the user, which implies dynamic changes in requirements and replies as the user becomes more experienced, and the introduction of new facilities when the user requires them, see ref. [16]. The system must, of course, do all of this without taking control of the interaction away from the user.

There is a wide variation in human performance and knowledge of a given system. Consequently, each user develops a very personal view or 'model' of how a given system works. This 'model' also evolves as the user becomes more acquainted with the system. Unfortunately, it is this 'model' i.e. a moving target, which provides the expectations of the user for the purposes of both the information supplied by him and the message received, see ref. [15]. Thus, every user will potentially need his own private interface to the system.

Furthermore, the package would need to be tolerant of mistakes in input, minimising the amount of information that needs to be re-input in the event of error (e.g. spelling mistake). It will also need to change its interface dynamically with time to accommodate the growth in knowledge and experience of the user. Shackel [4] has suggested that two alternative interfaces, one for the novice and one for the dedicated user, be provided. Although better than a single interface this clearly would not satisfactorily meet the criteria demands above (particularly items 5 and 6). Users frequently become expert on one aspect of a general purpose system after relatively few interactions, but can remain novices with respect to other aspects for many hundreds, and even be completely unaware
of some features throughout the period that they use the system. This feature illustrates the need to ensure that the system protects the user by restricting the range of facilities he can use, to the ones he is actually aware of (criterion 4). It also illustrates the difficulty of doing this since a feature which one user needs protecting from will be an essential part of the system as viewed by another user.

This same difficulty can be seen in terms of the other criteria. A novice will require a clear explanation of the facilities provided by the system before he starts (criterion 1) but this same explanation is an irritation for the experienced user (criterion 5). Similarly most occasional users could prefer to type in a date as '12/1/78' say or even '15 MARCH 1978' rather than 780206, or a colour as a word (e.g. 'green') rather than as a numeric code, which they must first look up, but an experienced user in a hurry may prefer to use the 'computer oriented' form since this reduces the amount of typing that must be done (criterion 3). However, expert and casual users would both argue that it should be unnecessary to repeat information to the 'system' as a way of making life easier for the 'system', see ref.[6].

2.3 TECHNIQUES USED IN DESIGNING AN INTERFACE

There is, of course, a wide range in the variety of interfaces provided by existing systems, they differ considerably in the amount of assistance that they give to the user, the flexibility with which he can vary the input he must provide, and the usefulness of the messages they produce in response to his efforts. As a general rule, it would appear that the more serious the purpose of the program, the more the effort that
has been put into ensuring that the program achieves its technical objectives, and the less that has been available to provide a suitable interface for the user.

Conversely, systems that have been written for fun, frequently have a relatively sophisticated user interface. Typically many game playing programs come into this latter category, while most operating systems and compilers come into the former category.

Separation of interface and system allows either to be changed or modified without altering or affecting the other. Once it has been decided to separate an interface structure from the system being used there are various techniques possible for use in building the interface.

2.3.1 Question and Answer Technique

Many of the earlier interactive systems intended for inexperienced users took this form, see refs. [13],[17]. This approach consists of questions being asked by the interface to which the user must reply. Only small amounts of information can be obtained in this way and it can be long and tedious for the user. It is most useful for users who have little experience of computers. Care must be taken not to ask questions which seem to have obvious answers to the user and which are too dictatorial e.g. "Answer Yes or No" which could lead to a frustrated user typing "yes or no".

If this technique is used with pattern matching to detect replies, then upper and lower case letters also variable numbers of spaces and obviously erroneous characters such as <> etc. must be dealt with or the interface will seem moronic. The interface must also be able to cope with misspelt replies, as this is common for words over seven characters in length.
As the user has no control over what happens he will soon become annoyed with the interface if he has much contact with it and it always asks the same questions, especially if this technique is used throughout.

2.3.2 Menu Selection Technique

When a limited set of responses are required by a program, menus provide an attractive mechanism that is simple to implement and clear to use. Menus are appropriate for routine task activities where the user wants to step through fixed procedures. A routine to do menu selection can be part of a general package, indeed it ought to be so to provide a consistent user interface for a given system, which may include automated formatting of menu entries [24] describe a flexible and portable screen library which includes menu functions as part of the standardised user interface. Most importantly menus are clear and flexible for the user.

Menus are useful on slow terminals and in time sharing environments, yet they are very effective with fast and sophisticated terminals. Menus provide useful features for both naive and experienced or specialist users especially when several location mechanisms are available together and the user can choose which suits.

This technique may be used where various alternative courses of action may be taken at a given point in the computer session and these are listed in a scheme where each alternative has a unique number or letter corresponding to it and displayed next to it in the list. Selection is achieved by the user typing the appropriate number or letter for the alternative he requires, (or on more sophisticated terminals a light pen may be used to indicate the choice). There needs to be an associated routine which checks the user response to ensure it lies within the correct range, see ref.[17].
This is most useful where the structure of the interface lends itself to a choice between a number of possibilities. It is a good scheme for minimising the typing required from the user, but it is quite verbose and if used indiscriminately can still be annoying for frequent users of the system.

The most useful application is where there are nested questions and each set of solutions depends on the choice at the previous set. Although a final choice could be allowed from a larger list, it is best to minimise the amount of reading required from the user who could miss the selection he requires in a larger set, see ref.[18].

This method is particularly suitable for enquiry systems which are providing information rather than gathering it, and which use a fast visual display unit as the interactive terminal; they can be used in other circumstances, but if the user has much information to input then the advantage of the menu selection method is lost, while waiting for a list of possible selections to be printed on a teletype is very tedious particularly if it is the first choice in the list that is to be selected.

The primary difficulty with this approach for the designer is structuring the sequence of menus such that all choices that required can be expressed as a linear combination of selections from each successive menu. This problem also appears as a flexible deficiency for a user who knows exactly what area he would like to receive information on, and yet has to go through the full menu selection process.

2.3.3 Command Language Technique

This approach requires the minimum of prompting, allowing the user to express his requirements more freely. It tends to require more
knowledge on the users behalf than the techniques explored in 2.3.1 and 2.3.2, however, it gives more flexibility of approach to user, see ref.[1]. It can be a language using all the tools which are already in use for handling programming languages. Tools such as those used for creating and maintaining a compiler could be used for a command language "compiler".

The major difference between programming and command languages is the amount of knowledge about particular subjects contained in them. Programming languages are multi-task whereas command languages are specialised to a particular task and therefore have knowledge about that task in-built, for instance in the vocabulary of commands and their parameters, see ref.[15].

2.3.4 Monitoring

When the protocol between interface and system is defined, message passing can easily be administered by special routines. These routines can therefore be made to logout all the messages they deal with (i.e. all valid instructions for the processing system). It is best to do this in the processing module then a number of different interface modules can have all their input and output recorded in the same way on the same file.

The information gained this way is of syntactically valid commands in a machine orientated form and is suited to machine processing to give statistical information on the use of the system. Also, the use made of the interface can be monitored in this way, see ref.[15].

It is best to call an "error" routine whenever an error is detected. This would give the user an appropriate message and record the time, type of error, user and input line on a file. This can be
supplemented by either a time ordered log of the total system interaction, or a session log kept separately for each user, which is in temporary storage and is deleted at the end of the session if no error has been detected, but is kept otherwise.

The designer may decide on the types of error to log and those to ignore, e.g. the inevitable ones such as providing insufficient information. This technique is very useful to monitor the interface, in which case the first few weeks of system life show what facilities the users really want (people are adaptable and learn to compromise to what they believe the system allows them to do after the first few weeks), and to monitor the system, which requires statistics to be taken throughout the system life and is most reliable some time after a change to the system has been made, i.e. when the users have settled down to a pattern of use once more.
CHAPTER 3

INTERFACE DESIGN AND THE GENERATOR
3.1 INTRODUCTION TO INTERFACE DESIGN TECHNIQUES

There is an underlying basic mismatch at the interface in the modes and rates of input and output between man and computer. This mismatch arises in hardware terms primarily because of the engineering problems involved; it is considerably more difficult and more costly to make reliable machinery which can match the flexibility of the human operator. In software terms the mismatch arises more through lack of knowledge of the capabilities of users and lack of realisation of difficulties users may have with logical organisation or information display which is not designed in accordance with the users needs.

The latter, in part, arises because we have too little knowledge yet about the basic characteristics of human cognitive processes in communications and especially about the relative importance of various characteristics (i.e. those which can be ignored and those which are vital). Much work still remains to be done both in computer and human science to redress the shortcomings mentioned above.

But even if enough knowledge were available from the human science, there is still a problem on the side of the computer designer and system analyst. Design to techniques are still evolving and are progressively taking more note of the human aspects of computer systems. However, designers still often do not appreciate the problems of the untrained or casual user and sometimes fail to appreciate the minute by minute problems connected with operating a computer terminal.

To overcome this last tendency, it is essential that users become much more insistent upon the hardware and software being made acceptable and usable by people. A change of approach to user-centred design may be the essential re-orientation in the system design process.
On the other hand, users must not expect the computer to be a panacea, for example, for managers' problems. No computer can solve difficult staff or other issues which require good management and not good computer design. The difficulty within the design process, is undoubtedly deep-rooted otherwise there would be less incidence of poor interfaces. Training given to designers on courses and undergraduate programmes may be a long term solution but even this may not prove successful if it is caused by the very training itself i.e. the training process may move the designer psychologically away from the user.

To assist any one who will design or influence the design of systems for others to use, perhaps it may be helpful to promote the following three, rather general, precepts, see ref.[4]:

1. The designer should change the orientation of system design process to be user-centred, and he should build flexibility into the system and software to enable development as users learn and improve. Never underestimate the potential of the user.

2. Before producing any interface or system for people as a final production model, always the designer should build a trial model or prototype for honest and thorough evaluation:
   * By impartial experimenters using samples of expected users.
   * By end user department staff or worst enemies, if the designer can get them to help.

3. The designer should remember that as a person he is unique, and cannot easily represent the other 99 unique people who make up the population of users for whom he is designing.
Before considering the design of the interface and its generator in detail, it might be appropriate to consider the structure of the interfaces in typical systems.

The generic use of the term "interface" might be applied to a group of related and structurally or syntactically similar exchanges with the system. The grouping might be related to terminal types or application areas. Within each "interface" we might have a number of "transactions" see Figure 3.1. A transaction is a self-contained exchange with the computer and will compromise one or more input and one or more output messages. A Sales Accounting Interface might include an Order-Input Transaction, a Credit Note Transaction, a Customer File Update Transaction and so on.

In turn, each message will consist of a number of data items and various formatting and control characters.

Many computer supplies provide software which causes the development of transaction-processing systems. Figure 3.2 illustrates how a user, once having logged-in, has access to a transaction handler which will enable him to select the transaction-processing program he wishes to run. Once in the program, he can get to another by returning to the handler or, in some cases, by going directly via a 'next transaction' field in the input form, see ref.[17].

The system designer's involvement with an interface dialogue may be relatively short-lived. On the other hand, many users may have to live with it for a number of years. Also, the designer's protection of 'ease of use', is likely to be quite different from that of ultimate end user. The message here is that the user must be involved in the development of the dialogue; experience shows that the approach pays dividends.
FIGURE 3.1: The Structure of User/Terminal Interface Dialogues
FIGURE 3.2: Profile of a Typical Interface Dialogue

- Name/user number
- Password
- Transaction ID/program ID
- System checks authorization
- Prompts or Screen formats
- Free format
- Forms mode
- Error handling
- Error indication
- Amendments/re-entry
- Re-verification
- Default to same transaction
- Choose to return to transaction handler
- Select new transaction and go direct
Figure 3.3 illustrates the recommended sequence in the design of dialogue. The approach is essentially 'topdown' (what else?) and corresponds to the general structure of interface dialogues illustrated in Figure 3.1.

3.2 GENERAL PRINCIPLES OF TERMINAL DIALOGUE DESIGN

It would be appropriate here to discuss principles to be used in designing effective interfaces (i.e. computer terminal dialogues). The terminal is the instrument through which a person, the operator or user, interfaces with the system. Each time information is entered via the terminal the operator expects the system to respond in a meaningful way. Thus, the prime objective of the interface is to communicate and provide a demanding dialogue thus avoiding operator frustration. A good dialogue design therefore strives to achieve the following, see ref.[20]:

1. Communication, messages both ways and clearly understood.
2. Minimum operator action, this allows the system to be directed by a minimum of operator input.
3. Unskilled operation. System assists positively to make system operation clear at all times.
4. Standardised operation. Abbreviations and terminology consistently used; method of response uniformly invoked across the system.
5. Stability. Mistakes in operation detected and suitable advice provided. Wrong entries cannot 'break' the system.
6. Satisfactory response times. Operators concentration maintained by continuity of action within the task undertaken.
FIGURE 3.3: A Structured Approach to Interface Dialogue Design
The degree to which the above objectives could be achieved within a given interface largely depends on:

1. Cost of the network required to support the dialogue - effective interfaces are more verbose and hence more costly.
2. Cost of producing the interface program - this will reduce through the wider use of interface generators, as proposed in this report.
3. Whether 'ad hoc' type reports will be required - this will necessitate the operator to state conditions and actions to apply in compiling the report.
4. The source of input to be entered from the terminal - here the mode of operation will strongly influence the character of the dialogue i.e. batch data entry should allow data to be entered in same sequence as it appears on the form; real-time enquiry should allow coded responses from the operator so that information is returned to enquirer with a minimum of delay; this latter technique will demand dedicated operators.
5. How much resources (usually money) are available.

The interface, or dialogue, must therefore provide the means by which the operator makes decisions and conveys these to the system.

If the system leads the operator through the dialogue the interface is system structured; alternatively, the operator may direct the system through the dialogue in which case the interface is operator structured. With system structured dialogues 'menu selection' methods are commonly adopted whereby the operator makes a choice of an actual function required from a list of options or (for larger systems) a selection of a sub-menu containing the functions.
With operator structured systems the system attempts to analyse anything that the operator enters. If analysis is possible the appropriate activity is initiated, if not a suitable error message is returned to the operator. Most operating system command language dialogues are in this class.

For casual users the system structure dialogues are essential since they 'present' the system options to the operator to select so that knowledge of the system need not be retained. Dedicated operators would find this technique cumbersome after a short period. When full knowledge of the system capabilities is understood an operator structured technique is more satisfying; it is also cheaper in line costs since very much less information gets sent along the communication network. A dedicated operator will need constant usage of the system to remain dedicated and would probably have undergone introductory training to understand the usage of the commands and action codes. Also, operators can be dedicated in one system and casual in another less familiar system; this leads to the question of the provision of a range of interfaces designed to control the use of any one given system in which the appropriate interface is selected automatically and given to the user. This is now feasible considering the cheapness of mass storage and availability of generators to create the interfaces.

The following design principles are introduced as a basis upon which the development of this project was established; the terminal characteristics available can vary widely from machine to machine with the result that some of these principles are easy to follow and others not, e.g. some terminals are provided with a comprehensive set of dedicated keys for STARTing a dialogue, RELEASing the buffer to the system (assuming a
buffered terminal), DELETing, INSERTing, CANCELling etc. while other terminals may only provide one or two of these functions, see ref. [20].

1. Messages from the system should be unambiguous and clear. To assist in this the messages should be grammatical and simple, jargon and abbreviation free and complete.

2. Each frame should relate to only one idea i.e. clear the screen before each new communication begins.

3. Each frame should be designed so that the amount of information required in the frame is small; this will ensure that principles 1 and 2 are achieved.

4. Operator actions, e.g. selection mode in a menu, should be confirmed by the system and sometimes even endorsed by the operator before application programs are started.

5. When multiple operator entries are required per frame the system should summarise the information presented before system action is triggered by the operator.

6. Screen formats should be improved by use of:
   - Justification and columnisation
   - Short messages
   - Screen zoning (asterisks, etc.)
   - Highlighting with intensity, flashing or colour.

7. Operator action and system messages should be clearly distinguished by:
   - Using separate columns, different colours
   - Underscoring system messages only
   - Using different cases (upper and lower).

8. System should anticipate operator action where input is standard, leaving operator only to modify where necessary before releasing the frame.
9. The operator should be able to type ahead i.e. anticipate the system even ahead of the next frame.

10. For batch type dialogue the screen formats should closely resemble the form layouts and should sequence from top to bottom and left to right.

11. Security - requirements of user, system and data protection.

3.3 INTERFACE DEVELOPING METHODS

Although a number of tools and techniques are available to the programmer of a man-computer interface, there is scope for much innovation. The program for most interfaces could be generated from a description of the interface flow if suitable compilers existed.

However, at present most commercial interfaces are still programmed using conventional or on-line programming languages. Commercial programmers often use a report program generator for generating the code necessary for producing reports. There could be equally well an interface program generator for generating the programs necessary for man-computer interface. Undoubtedly future levels of software will take much of the programming burden out of interface production.

Choosing the most appropriate style of interface is probably the most critical stage of the design process, the implications of getting it wrong can be worrying.

Both the interface and its generator are programs coded in RPL but since the interface is generated by the generator the structure of the interface will largely determine the functional nature of the generator; the preliminary design of the interface will therefore be considered first.
The whole project could span over eight major stages:

i) Determine objectives of the interface.

ii) Design the interface structure.

iii) Determine specification of the program (generator) to generate this.

iv) Code the generator in RPL.

v) Compile and test the generator.

vi) Compile and test its output (interface) for performance.

vii) Correct the generator and repeat stages v) and vi) until performance of the interface is acceptable.

viii) Apply the generator to a range of application packages including both well understood local systems and externally supplied relatively unfamiliar systems. See Figure 3.4.

3.4 THE OBJECTIVES OF THE INTERFACE

The interface is an entity situated between a set of programs which belong to a system and the user who attempts to use these programs. The object here is to bring together logically and functionally related separate programs in such a way that the user sees the whole system in a more meaningful way.

Another objective is to remove from the user his dependence upon external documentation which in the early stages of usage may be very high. A secondary objective is to achieve a high degree of familiarity of the system immediately at minimal programming cost since no programs are written manually. Also, to present the user with a flexible human-oriented
Determine objectives

Design interface

Draw up specification of programs to generate this interface and code it in RPL

Compile generator

Test generation

Compile and test the interface

Correct generator NO

Satisfactory YES

Repeat test a new user experience level

FIGURE 3.4: System Flow for the Development of the Generator
method of communicating with a computer system and a technique must be sought to achieve this if possible. These objectives are consistent with the principles announced in Section 3.2.

The first objective can be achieved by relating all the programs in the system through a main menu and sub-menu interface. The second objective can be achieved by providing an on-line manual text which is available via the menu system. The third objective can be achieved by allowing the user, when generating the interface, to enter details which are to appear. These data entries will originate in the user department and can be made brief or verbose. Also, regeneration will be possible to reflect major changes required in the interface.

3.5 STRUCTURE OF THE INTERFACE

A very simple structure was adopted so that the difficulties of generating a program reflecting this structure would be minimised. It was decided to provide a main menu, to be displayed after entering the password which prevents others from using the system, with any number of sub-menus. Further levels of menu were not permitted. No more than six categories were loaded in the main menu because the application package is assumed to have six basic categories of programs, and no more than five program modules per category (sub-menu). Program modules were loaded and executed via instruction with a given category; the program modules are listed in a scheme where each program has a unique number. Selection is achieved by the user typing the appropriate number for the program module. The system then checks with the user before calling the particular program to ensure that the user really wants to execute the appropriate program. The
advantage of this is that it reduces the errors caused by loading and running the wrong application program. Each program module returns control back to the interface program, thus keeping the user tied to an application until he requests to leave via the main menu, see Figure 3.5. An important implication is that the program module source must be available for amendment and re-compilation so that this control may be introduced.
FIGURE 3.5: Control Features of the Interface
CHAPTER 4

DEVELOPMENT SYSTEM AND SPECIFICATION
4.1 SYSTEM AND LANGUAGE USED FOR THE DEVELOPMENT

The system used in this project is a hybrid system consisting of the system hardware, system software and documentation. The system hardware comprises a DEC PDP 11-03 Processor with 48Kb of memory, the AED 6200P dual diskette drive, a VDU console keyboard and the OKIDATA 110 printer; such a system is typical of many commercially available in the price range £5,000-£10,000 (1978 prices).

The system software is the DEC originated RT-11 (version 3C) which comprises a single job monitor (FDMNSJ), device handlers (TT:, LP: and NL:), support for language processors e.g. RPL and range of system utilities (editor, librarian, etc.) see ref.[8]. Within the operating system a user may load a run time system for developing application systems in a language called Report Programming Language (RPL11); this language is an interactive strain of FILETAB which is in very common use, see ref.[9] and is a powerful programming language developed for commercial and business applications on PDP 11 computers operating under RSTS.

The RPL11 can be used to write most of the programs required including data input and validation, file creation, updating and report production. It incorporates many useful features and is constantly being enhanced to provide new facilities. RPL also supports a variety of file structures including ASCII, Record I/O, Sort Compatible, Virtual array, Indexed and database. Finally RPL11 Program Logic is specified using decision tables, a concise and flexible way of expressing logic, and is the major feature of the RPL11 language.
4.2 RT-11 OPERATING SYSTEM

As described above this system comprises of typical sub-systems which are essential for the efficient employment of a general purpose digital computer. However, although the command language represents a 'better than average' interface to the user, it is typical of many command languages available and lacks the features which the project is attempting to introduce. A comprehensive set of monitor operating commands allows the user to direct, from the console keyboard, those system operations that are required. Acquired with the RT-11 operating system was a development language RPL-11 which was implemented through its own run time system called R.SAV; this allows both the compiler (source is RPL) and user programs to be completely under the control of the run time system, see refs.[8],[22]. The hierarchy of control is as shown in Figure 4.1.

The opportunity exists therefore to develop, using RPL, an interface generator which will run under R.SAV to generate an RPL source program which is the interface required. This will compile using the RPL compiler to produce an executable object. The steps involved in this process are shown in Figure 4.2.

4.3 RPL DEVELOPMENT SYSTEM

This is a system of programs which includes a compiler (RPLll), a text editor (ED), file utilities (DBUTIL,SFINIT,INRPL) and a run time system (R.SAV) which controls the execution of all the above. It was selected for student project development within the department from a wide range of alternative development systems because it is believed to enable
FIGURE 4.1: The Hierarchy of Control
Objective: to obtain USERIF.RPC automatically

Steps : 1 → 5

1. Hand written generator → INTGEN.RPL
2. Compile INTGEN.RPL → INTGEN.RPC
3. Run INTGEN.RPC → USERIF.RPL
4. Submit USERIF.RPL to RPL compiler
5. Install USERIF.RPC as interface and use

FIGURE 4.2: The Generation Process
students to write and develop programs very quickly and produce programs which are easily maintained. It can be used in two ways. As a programming language for writing validates, updates, merges, etc., and as a report generator. Used as a report generator it draws upon its FIXED LOGIC mechanism, see ref.[9] which relieves the programmer of many coding requirements essential with more conventional languages e.g. totalling, checking for control breaks, formatting and editing print lines, opening/closing of files, reading and printing data, input data validation checks. All of these may be defaulted and thus source versions of simple report programs are very short. However, such is the nature of the generator (and the interface) that it will be necessary to use the system as a programming language. As such, the program source code will be organised with the categories shown below:

Input and output file specification,
Data specification,
Program logic specification.

Both the interface and its generator will follow such a structure and in both cases the program logic specification details constitute the predominant group.

Source code is first entered (via the keyboard) through the utility INRPL. When complete the first compilation is attempted, source is corrected if necessary using ED and the cycle repeated. The object is then tested for run time errors and the source again adjusted as above until satisfactory. This is a conventional compiler-based development system.
4.4 CODING THE GENERATOR

The programming technique used was modular programming which consists of coding subroutines to do basic functions and then coding others to use the basic routines to perform a higher level task, and so on. The first few weeks of programming were therefore spent in mastering the use of these routines to examine full potential. Then, the next level of routines which would use these basic functions was written. These fundamental routines were then tested and once they were completely satisfactory they could be used to write the higher levels of the generator. The final version of the generator contained 630 RPLll lines made up of:

1 line - output file specification
6 lines - data specification
Remainder lines - program logic specification (decision tables), the source version was named INTGEN.RPL and the code version INTGEN.RPC.

The decision table is divided into four sections, see ref.[9]

<table>
<thead>
<tr>
<th>condition statements</th>
<th>condition entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>action statements</td>
<td>action entries</td>
</tr>
</tbody>
</table>

The condition and action statements consist respectively of a list of the conditions to be evaluated, and the actions to be carried out. These parts of the table are known as the condition stub and action stub. In the condition entry part of the table are listed the values of the conditions usually Yes and No or Y or N. In the action entry the particular actions to be carried out are indicated by placing an X by a particular action, listing 4.3 is a sample of the decision table from the generator program.
### Listing 4.3 - A sample decision table from the generator

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>262</td>
<td>DETAB SUB MENU GEN</td>
</tr>
<tr>
<td>263</td>
<td>I &amp; '</td>
</tr>
<tr>
<td>264</td>
<td>&amp; '</td>
</tr>
<tr>
<td>265</td>
<td>&amp; ' ENTER NUMBER OF SUBMENUS - '</td>
</tr>
<tr>
<td>266</td>
<td>INPUTNP = 82:2</td>
</tr>
<tr>
<td>267</td>
<td>C = 82:2 = ?</td>
</tr>
<tr>
<td>268</td>
<td>A &amp; '</td>
</tr>
<tr>
<td>269</td>
<td>&amp; '</td>
</tr>
<tr>
<td>270</td>
<td>&amp; ' ENTER FIRST MENU DETAILS</td>
</tr>
<tr>
<td>271</td>
<td>&amp; '</td>
</tr>
<tr>
<td>272</td>
<td>CALL MENU1.ENTRIES</td>
</tr>
<tr>
<td>273</td>
<td>CALL MENU1. GENERATOR</td>
</tr>
<tr>
<td>274</td>
<td>122:2 + 1</td>
</tr>
<tr>
<td>275</td>
<td>&amp; '</td>
</tr>
<tr>
<td>276</td>
<td>&amp; '</td>
</tr>
<tr>
<td>277</td>
<td>&amp; '</td>
</tr>
<tr>
<td>278</td>
<td>&amp; ' ENTER SECOND MENU DETAILS</td>
</tr>
<tr>
<td>279</td>
<td>&amp; '</td>
</tr>
<tr>
<td>280</td>
<td>CALL MENU1.ENTRIES</td>
</tr>
<tr>
<td>281</td>
<td>CALL MENU1. GENERATOR</td>
</tr>
<tr>
<td>282</td>
<td>122:2 + 1</td>
</tr>
<tr>
<td>283</td>
<td>&amp; '</td>
</tr>
<tr>
<td>284</td>
<td>&amp; '</td>
</tr>
<tr>
<td>285</td>
<td>&amp; ' ENTER THIRD MENU DETAILS</td>
</tr>
<tr>
<td>286</td>
<td>&amp; '</td>
</tr>
<tr>
<td>287</td>
<td>CALL MENU1.ENTRIES</td>
</tr>
<tr>
<td>288</td>
<td>CALL MENU1. GENERATOR</td>
</tr>
<tr>
<td>289</td>
<td>122:2 + 1</td>
</tr>
<tr>
<td>290</td>
<td>&amp; '</td>
</tr>
<tr>
<td>291</td>
<td>&amp; '</td>
</tr>
<tr>
<td>292</td>
<td>&amp; ' ENTER FOURTH MENU DETAILS</td>
</tr>
<tr>
<td>293</td>
<td>&amp; '</td>
</tr>
<tr>
<td>294</td>
<td>CALL MENU1.ENTRIES</td>
</tr>
<tr>
<td>295</td>
<td>CALL MENU1. GENERATOR</td>
</tr>
<tr>
<td>296</td>
<td>122:2 + 1</td>
</tr>
<tr>
<td>297</td>
<td>&amp; '</td>
</tr>
<tr>
<td>298</td>
<td>&amp; '</td>
</tr>
<tr>
<td>299</td>
<td>&amp; ' ENTER FIFTH MENU DETAILS</td>
</tr>
<tr>
<td>300</td>
<td>&amp; '</td>
</tr>
<tr>
<td>301</td>
<td>CALL MENU1.ENTRIES</td>
</tr>
<tr>
<td>302</td>
<td>CALL MENU1. GENERATOR</td>
</tr>
<tr>
<td>303</td>
<td>122:2 + 1</td>
</tr>
<tr>
<td>304</td>
<td>&amp; '</td>
</tr>
<tr>
<td>305</td>
<td>&amp; '</td>
</tr>
<tr>
<td>306</td>
<td>&amp; ' ENTER SIXTH MENU DETAILS</td>
</tr>
<tr>
<td>307</td>
<td>&amp; '</td>
</tr>
<tr>
<td>308</td>
<td>CALL MENU1.ENTRIES</td>
</tr>
<tr>
<td>309</td>
<td>CALL MENU1. GENERATOR</td>
</tr>
<tr>
<td>310</td>
<td>&amp; '</td>
</tr>
<tr>
<td>311</td>
<td>&amp; ' NOT MORE THAN SIX SUBMENUS '</td>
</tr>
<tr>
<td>312</td>
<td>EXIT T</td>
</tr>
</tbody>
</table>
4.5 THE SPECIFICATION OF THE GENERATOR

The generator program to be developed will have features determined by its output; this output is an interface which must meet the demands of a certain type of user wishing to use an application. It is considered too difficult to generate an adaptive interface at this stage of development and so a casual user has been selected as a "target" user. This user will require a system directed interface employing a menu structure. This is justified since a dedicated user can tolerate an easy interface whereas a casual user cannot tolerate a difficult interface.

Also, the operator will spend relatively little time in the interface compared with the time in the application program. The interface presented by the application program may (quite likely) be geared to the dedicated user since a given user frequently uses the same programs.

Following a conventional functional specification layout, two important areas will be considered:

i) System Description;

ii) Acceptance Criteria.

4.5.1 The Generator System Description

Output

This is a disk file containing an RPL program INTGEN.RPL. This will later be compiled and tested for correct performance as a casual user interface to an application system. The generator, being an interactive system, also outputs messages to the console VDU; the full specification of the VDU performance is documented in Appendix A1. The program contained in INTGEN.RPL will conform to the RPL language compiler specification as set out in the RPLll Reference Manual [9]. The program will have an outline structure as indicated below:
CONTROL

BANNER (Documentation)

QUESTIONNAIRE:

TEST (Input samples for user familiarization)

USER.DESCRIPTIONS

IDENTIFICATIONS (Input data)

MAIN.MENU (Input data)

MAIN.MENU GENERATOR (Output)

SUB.MENUS (Input data)

.MENUXGENERATOR (Output)

POSTSCRIPT. (Trailing documentation) (Output)

where the routines indicated have the following functions:

CONTROL - Performs the generator routines in a logical sequence.

BANNER - Insert (minimal) token documentation for identification purpose.

QUESTIONNAIRE - Starts up and continues the dialogue with the user.

TEST - Supports the Questionnaire Process by illustrations.

USER.DESCRIPTION - Obtains details from the user for inclusion in the interface.

POSTSCRIPT - Completes the program documentation.

In every case it is possible to call in subservient functions. This ensures very good modularity of the generator and enables rapid debugging.

The specification indicated above does not reflect a fine level of function e.g. routines for writing a current line of code.
Input

This is to be entered from the console keyboard in response to messages from the generator program, and so are completely defined in the Appendix A1 mentioned in the discussion of output. Here it is sufficient to summarise the input which comprises the data:

- Password: to be built into the interface
- Program numbers: for chaining purposes
- Program description: for menu inclusion
- System name: for chaining purposes
- File name for output: to lead generated program
- Number of sub-menus: to be generated

Processing

After the preliminary data has been accepted, the system generates file specification and initial documentation (in INTGEN.RPL). The generator then requests descriptions of the main menu of the interface before generating the decision table for it. This latter process is repeated for each of the sub-menus.

Including documentation completes the generation process. The processing sequence may be briefly summarised in a decision table.

*Main Processing Decision Table

<table>
<thead>
<tr>
<th>I</th>
<th>Enter Preliminary Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Preliminary Data Entered?</td>
</tr>
<tr>
<td></td>
<td>Main Menu Description Entered?</td>
</tr>
<tr>
<td></td>
<td>Sub Menu Description Entered?</td>
</tr>
<tr>
<td>A</td>
<td>Begin Main Processing</td>
</tr>
<tr>
<td></td>
<td>Enter Main Menu Description</td>
</tr>
<tr>
<td></td>
<td>Enter Sub Menu Description</td>
</tr>
<tr>
<td></td>
<td>End Main Processing</td>
</tr>
<tr>
<td></td>
<td>Repeat</td>
</tr>
</tbody>
</table>
Control

The interface, being generated, controls the use of an application system. Therefore, the control aspects preliminary relate to, firstly, the security check using the password for controlling access to the application, and secondly, the method used to maintain the interface as the driving mechanism i.e. the interface initiates the program required and the program returns control back to the interface upon completion. The generated interface must contain these minimum features required to achieve this level of control.

4.5.2 Acceptance Criteria

Tests will be made using operator-users of different backgrounds to assess the different reactions to both the generator dialogue and the interface dialogue when it itself is used. Note will be taken of comments made at the time of the tests. Particular note will be taken of critical remarks and observations and if an adequate connection is feasible this will be incorporated into the generator and/or the documentation. Where the criticisms are of a broader nature, note will be taken with respect to further development. The project's value will be judged by the relative ease at which an automatically generated interface program can be produced in comparison with a conventionally hand written equivalent interface.
CHAPTER 5

TESTING AND RESULTS
5.1 **INTRODUCTION AND TEST PLAN**

The main objective of this study is to prove the feasibility of a system which is designed to assist system development work at a high level. It is not intended to test a system having complete mechanisms which will ensure robustness and perfect operation. The chronology of development and testing has been described in the previous chapter but further details of the test plan is provided here.

The functional aspects of the generator cannot be proven until the output (the interface) is itself compiled and tested. Only then could processing errors be identified and fed back as corrections to the generator. Initially, the output was simply inspected for format compatibility i.e. syntactic correctness with respect to the RPL compiler. Gross errors were removed before a functional test of the interface was attempted.

The test plan for the whole project consisted of a number of stages which will allow an incremental approach to a fully tested system. This plan consisted of:

Stage 1: Visual test output for syntactic correctness using dummy data arguments. Correct the generator until output is satisfactory.

Stage 2: Prepare an existing well known system to be integrated via a generated interface i.e. Rename, modify the existing source and recompile. Prepare a system 'HELP' file. Use generator to compile the interface and use it on the prepared system. Note amendments and implement them in the generator.

Stage 3: Prepare an existing not so well known system and repeat the same steps in stage two.

Stage 4: Prepare an unfamiliar externally developed package (developed in RPL) and repeat the same steps in stage two.
5.2 USE OF THE INTERFACE GENERATOR

The interface generation process consists of four basic steps:
* The planning necessary to determine the desired components of the application and their descriptions.
* The execution of the generator.
* The compilation of the interface source code generated.
* Recompilation of the application source code with amendments necessary and a routine check that the new system is installed without error in the storage volume required.

5.2.1 Planning

Before attempting to use the generator the user should first investigate the target application and check for the following, some of which is required to respond intelligently to the generation dialogue:

a) Source code is available for every program module in the system.

b) Each program module has been assigned a name within the system, and a very brief function for each listed. The user should give each program a new name within the system by using the first three characters from the application name followed by three digits for the program number i.e. 'PAYO01' is the name of the first program from PAYROLL application. To change the names of all the programs, the user must use 'RENAME' command i.e.

```
.RENAME FD?:FILENAME1,EXT
TO FD?:FILENAME2,EXT
```

In which

- **FD?:** The device on which the file is stored
- **FILENAME1** - The old file name
- **FILENAME2** - The new file name
- **.EXT** - Either extension .RPL or .RPC ...etc.
c) The user should look at the end of each source program module and examine how it finishes. The user must then, if necessary, include the 'CHAIN' command (RPL) to connect all the programs to the interface program; The user must choose the name of the interface program at this stage to chain all the programs to it. The user should also include *CORCOM:2 field by setting it to 1 immediately before the 'CHAIN' command, whether it is in the middle of the program or in the *DETAB ENDOFJOB decision table at the end of the program. *CORCOM:2 field used to pass data from one program to another and it is necessary in this case to avoid the appearance of the password question every time if the user wishes to return to the main menu in the interface program; This is carried out using the editor; for example see lines 52 and 53 in listing 5.1.

d) A more detailed description of the function of each program prepared for entry into the help file which called 'HELP.TXT' which will be used for on-line interrogation.

The next stage is the execution of the generation program, but before this the user must divide the program of the system (application) into categories in the main menu. Because the application package is assumed to have six basic categories of programs, and no more than five program modules per category, the names the user gives to these categories will appear in the first control menu presented. Two other items will appear (namely - help and exit system). A separate menu (sub-menu) for each category will later be defined. See Appendix A1.

The programs in each category, as far as possible, deal with the same function within the system. For example, all the programs dealing with the master files should be included in the same category: See Appendix A5 for sample planning stage.
LISTING 5.1: Illustrating use of *CORCOM for inter program control

LINE SOURCE STATEMENT                COMPILED BY RPL11 MK28   31-OCT 4:00

1 * LOUGHBOROUGH UNIVERSITY OF TECHNOLOGY
2 * COMPUTER STUDIES DEPARTMENT
3 * THIS MODULE BELONGS TO LCS SYSTEM
4 *
5 * PROGRAM SOURCE FORMATTED BY FORMAT.RPL
6 * ***********************************************
7 *
8 * APPLICATION NAME - LCS
9 * PROGRAM NAME - LCS006
10 * PROGRAMMER NAME - OSSANA MUSLIM
11 * COMPLETION DATE - 15-MAY-79
12 * FUNCTION OF THE PROGRAM :
13 * THIS PROGRAM ALLOWS THE USER TO CREATE ANY SOURCE FILE
14 * NOT ONLY IN RPL. THE USER RESPONDS TO TERMINAL REQUESTS
15 * AND ENTERS THE DATA VIA THE KEYBOARD. "**" TERMINATE
16 * THE INPUT DATA AND A JOURNAL LISTING IS PROVIDED OF THE
17 * FILE. CONTROL IS THEN RETURNED TO THE MAIN MENU.
18 *
19 *PFILER LP:
20 *DFILE 3 A 'FILENAME.EXT'
21 *DICT
22 L -2:2
23 *LIST GENLINE
24 A 0:30
25 *PIC DATA 1.0
26 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
27 *DETAB LPLINEUP
28 A & / / 
29 & / / 
30 & 'PLEASE ENTER ** IF YOU WISH TO TERMINATE THE INPUT.' 
31 & / / 
32 0/80 $
(Continued Listing 5.1)

32 EDITMOVE *LIST *RECORD
33 PRINT DATA GENLINE
34 0/80 MV ' ASCII FILE LOAD DATA JOURNAL '
35 70/9 MVDATE$ 0
36 EDITMOVE *LIST *RECORD
37 PRINT DATA GENLINE
38 GOTO INPUT.DATA
39 *DETAB INPUT.DATA
40 I 0/512 S
41 INPUT 0/(L)
42 C 0/2 = '***' Y N
43 A CLOSE 3 X -
44 DELETE 'OK' X -
45 EDITMOVE *LIST *RECORD
46 PRINT DATA GENLINE
47 WRITE 3 0/(L)
48 L MV 512
49 GOTO INPUT.DATA
50 *DETAB ENDOFJOB
51 A *CORCOM:2 MV 1
52 CHAIN 'FD1:LC5'
53 *STOP

"ERRORS DETECTED: 0
RPL>"
5.2.2 Execution of the Generator

On completion of the planning stage the user should run the INTERFACE GENERATOR which is called 'INTGEN', in this case:

RPL>INTGEN

The 'INTGEN' asks a series of questions and requests for information, and uses a response made to direct further prompting and to construct the output file containing the user interface. Listing 5.2 shows a sample of the generation dialogue obtained when running INTGEN,RPC.

When the user finishes the entries for all the categories which are not necessarily the maximum number six, the execution of the generator is then terminated with the message 'END OF GENERATION PROGRAM'. At this stage the generation program is already chained to another program called 'DBTEST', the function of this program is to initialise a file called 'HELP.TXT' which will be used for on-line interrogation and will be 80 blocks and ready to use for entering a more detailed description of the function of each program in the system.

The execution of the 'DBTEST' program does not need any response from the user because it is already set up with the needs of the user, but when the initialisation process is finished the 'DBTEST' program is already chained to another program called CREATE HELP 'CRHELP', the function of this program is to create the help file which was initialised before by the 'DBTEST' program, the user should enter a serial number for each program in the system, and then he should enter a name (keyword name) which represents the name of the program. After the creation of help file and stored in the storage volume required, it will be ready to read by running READ HELP 'REHELP' program which is chained to the 'help' item in the main menu of the interface program. Listing 5.3 shows a sample of creation of the 'help' file and how to read it.
LISTING 5.2: Start of the generation dialogue  
(N.B. Upper and lower case is used when VDU is operated)

* **********************************************************
*  
* Loughborough University Software
* Personalised Interface Generator
*  
* **********************************************************

NOTE:

-----

If mistakes are made in entering data, please make a note and continue with the generation process. Correction to the interface generated may be made at the latest stage through using the editor.

Your application package is assumed to have six basic categories of programs and no more than five programs modules per category. The names you give to these categories will appear in the first control menu presented. Two other items will appear (namely: help and exit system). A separate menu for each category will later be defined.

Please press return key if you wish to continue:

Example of four basic categories

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

Initial Installation
Maintenance
Transaction Processing
Interrogation

Example of modules within Initial Installation -
**********************************************************

Create Vendor Master
Create Invoice Master
Append Vendor Master
Etc. (up to five allowed).

Please enter name of application :? LCS
Please enter valid password (six characters) :? OSSAMA

Which device you prefer your output file.
If on the line printer please enter LP:
If on the teletype please enter TT:
If on disk please enter which disk FD0: or FD1:
Followed by any name you like no more than six characters ended by . RPL

Which device :? FD0: LCS. RPL
DETAILS OF MAIN CONTROL MENU NOW REQUESTED, EACH CATEGORY NO MORE 20 CHARACTERS.

ENTER FIRST CATEGORY IN MENU : LANGUAGE SYSTEM.
ENTER NEXT CATEGORY IN MENU : FILE INITIALISATION.
ENTER NEXT CATEGORY IN MENU : FILE MANIPULATION.
ENTER NEXT CATEGORY IN MENU : SYSTEM FILES.
ENTER NEXT CATEGORY IN MENU : LIBRARY FILES.
ENTER LAST CATEGORY IN MENU : OTHERS.

ENTER NUMBER OF SUBMENUS - 6

ENTER FIRST MENU DETAILS :

PLEASE ENTER DETAILS OF THE CATEGORY X MENU. ENTER DESCRIPTION OF EACH PROGRAM IN THIS CATEGORY, NO MORE THAN 38 CHARACTERS, AND THEN ENTER THREE DIGITS FOR THE PROGRAM NUMBER.

ENTER FIRST MODULE DETAILS : CREATE A SOURCE RPL FILE.
ENTER PROGRAM NUMBER : 001
ENTER NEXT MODULE DETAILS : FORMATTED SOURCE FILE.
ENTER PROGRAM NUMBER : 002
ENTER NEXT MODULE DETAILS : LISTS *DICT WORKING OUT START POST.
ENTER PROGRAM NUMBER : 003
ENTER NEXT MODULE DETAILS : RENAME PAGES ON LIBRARY PAGE FILE.
ENTER PROGRAM NUMBER : 004
ENTER LAST MODULE DETAILS : CHECK LIBRARY.
ENTER PROGRAM NUMBER : 005

ENTER SECOND MENU DETAILS :

PLEASE ENTER DETAILS OF THE CATEGORY X MENU. ENTER DESCRIPTION OF EACH PROGRAM IN THIS CATEGORY, NO MORE THAN 38 CHARACTERS, AND THEN ENTER THREE DIGITS FOR THE PROGRAM NUMBER.

ENTER FIRST MODULE DETAILS : CREATE A SOURCE FILE NOT ONLY IN RPL.
ENTER PROGRAM NUMBER : 006
ENTER NEXT MODULE DETAILS : DATABASE UTILITY PROGRAM.
ENTER PROGRAM NUMBER : 007
ENTER NEXT MODULE DETAILS : SORT FILE INITIALISER.
ENTER PROGRAM NUMBER : 008
ENTER NEXT MODULE DETAILS : PRINT SORT FILE HEADER.
ENTER PROGRAM NUMBER : 009
ENTER LAST MODULE DETAILS : DATABASE UTIL. PROG. (ALPH. NUMERIC).
ENTER PROGRAM NUMBER : 031
LISTING 5.3: Start of the creation of 'help' file and how to read it
(N.B. Upper and lower case is used when VDU is operated)

ENTER SYSTEM DEFINITION DETAILS TO BE STORED
IN A HELP FILE CALLED "HELP.TXT".

PLEASE ENTER THE KEYWORD NUMBER STARTING FROM 1
AND THEN INCREASE BY 1 (FOR EXAMPLE 1, 2, 3 ...).

PLEASE TERMINATE YOUR ENTERING THE INFORMATION BY
CREATE A DUMMY RECORD BY GIVEN A NUMBER AND THEN
ENTER *** FOR THE KEYWORD NAME.

DO YOU WISH TO CONTINUE PLEASE ENTER YES OR NO ? YES

PLEASE ENTER KEYWORD NUMBER (NO MORE THAN TWO DIGITS) : 1

PLEASE ENTER KEYWORD NAME (NO MORE THAN 20 CHARACTERS) : LIST

PLEASE ENTER KEYWORD DETAILS AND THEN TERMINATE BY "STOP"
ON OF IT IS OWN :

THIS PROGRAM ALLOWS THE USER TO GET A LIST
OF DICTIONARY. THE USER RESPONDS TO TERMINAL
REQUESTS AND ENTER THE DATA VIA THE KEYBOARD.
STOP

DO YOU WISH TO ENTER MORE KEYWORDS PLEASE ENTER YES OR NO ? YES

PLEASE ENTER KEYWORD NUMBER (NO MORE THAN TWO DIGITS) : 2

PLEASE ENTER KEYWORD NAME (NO MORE THAN 20 CHARACTERS) : RENAME

PLEASE ENTER KEYWORD DETAILS AND THEN TERMINATE BY "STOP"
ON OF IT IS OWN :

THIS PROGRAM ALLOWS THE USER TO RENAME THE PAGES
ON LIBRARY CONTROL FILE. THE USER MUST ENTER THE
DATA VIA THE KEYBOARD WHICH ARE THE OLD NAME, OLD
TYPE, NEW NAME AND NEW TYPE.
STOP

DO YOU WISH TO ENTER MORE KEYWORDS PLEASE ENTER YES OR NO ? YES

PLEASE ENTER KEYWORD NUMBER (NO MORE THAN TWO DIGITS) : 3

PLEASE ENTER KEYWORD NAME (NO MORE THAN 20 CHARACTERS) : REPLACE

PLEASE ENTER KEYWORD DETAILS AND THEN TERMINATE BY "STOP"
ON OF IT IS OWN :
(Continued 5.3)

THIS PROGRAM ALLOWS THE USER TO REPLACE SPACES BY TABS IN ANY ASCII FILE. THE USER MUST ENTER THE INPUT FILE NAME AND OUTPUT FILE NAME ONLY.
STOP

DO YOU WISH TO ENTER MORE KEYWORDS PLEASE ENTER YES OR NO ? YES
PLEASE ENTER KEYWORD NUMBER (NO MORE THAN TWO DIGITS) : 4
PLEASE ENTER KEYWORD NAME (NO MORE THAN 20 CHARACTERS) : ***
PLEASE ENTER KEYWORD DETAILS AND THEN TERMINATE BY "STOP"
ON OF IT IS OWN :
STOP

DO YOU WISH TO ENTER MORE KEYWORDS PLEASE ENTER YES OR NO ? NO

THE HELP FILE NOW IS READY TO USE IN YOUR APPLICATION.
*******************************************************************************

PLEASE COMPILE THE INTERFACE PROGRAM NOW TO MAKE IT READY TO USE AND THIS IS THE END OF THE WHOLE PROCESS.

THANK YOU AND GOOD LUCK.
*******************************************************************************
PFL>REHELP
THE KEYWORDS IN THE HELP FILE ARE :
01 LIST 02 RENAME 03 REPLACE

PLEASE ENTER KEYWORD NUMBER REQUIRED : 2

RENAME

THIS PROGRAM ALLOWS THE USER TO RENAME THE PAGES ON LIBRARY CONTROL FILE. THE USER MUST ENTER THE DATA VIA THE KEYBOARD WHICH ARE THE OLD NAME, OLD TYPE, NEW NAME AND NEW TYPE.

DO YOU REQUIRE FURTHER INFORMATION, PLEASE ENTER YES OR NO : ? YES
THE KEYWORDS IN THE HELP FILE ARE :
01 LIST 02 RENAME 03 REPLACE
PLEASE ENTER KEYWORD NUMBER REQUIRED : 1

LIST

THIS PROGRAM ALLOWS THE USER TO GET A LIST OF DICTIONARY. THE USER RESPONDS TO TERMINAL REQUESTS AND ENTER THE DATA VIA THE KEYBOARD.

DO YOU REQUIRE FURTHER INFORMATION, PLEASE ENTER YES OR NO :? NO
THANK YOU
RPL>
5.2.3 Compilation of the Interface and Modification of Application Source Code

The output file from the previous stage can now be compiled by the RPL compiler (RPL.RPC). To produce an executable interface program, the compilation process is as follows:

RPL>RPL
#,TT:=PROGRAM NAME

The program name in this case is the interface program name which the user created in the previous stage.

All the application programs are now examined for logical exist and changed, so that the return control to (CHAIN) the interface program, these are compiled and recompiled and objects renamed as part of an application system consistent with decisions taken in section 5.2.1.

For general purpose applications, the interface and application programs are stored on the same user volume (DRIVE 1). For special applications (e.g. local student development systems), the interface program should be located on the system volume (DRIVE 0). This requires that two versions of the generator are provided to handle the appropriate drive to be used.

5.2.4 Exercising the System

The application system and interface should be exercised for obvious flaws before back up copy is taken. The system is then ready for use. This stage is vital to ensure that each amendment carried out in section 5.2.3 has resulted in a consistent set of controls which originate from the interface program.
5.3 COMPLETE TEST EXAMPLE

The first live test of the generator followed the routine format tests; this attempted to integrate the local system of programs used in the student project development system. Except for a small number of programs, the source code was available and hence it was possible to connect them so that they all 'chained' back to the interface program on exit. This system was given the name 'Local Control System' (LCS), and its prime function was to supply the functions available to the user 'student' via the interface. Each program (or function) was given an identification having LCS as a family application name (below).

Sample (from 32 programs);

LCS001 - Source RPL input (INRPL)
LCS002 - Source RPL FORMATTER
LCS003 - Dictionary entry converter for start position
LCS004 - Renames pages in Library Control file
LCS005 - Checks contents of Library Control file
LCS006 - Generalised data entry program
LCS007 - Database utility program (DBUTIL)
LCS008 - Sort file type - initialise (SFINIT)
LCS032 - Test scale factor in compiler (TSF)

These programs were then grouped into 6 categories (sub-menus) by type of function.

Having now identified what programs are to exist in the system the generator was run and requested data supplied. Within a few minutes the generated interface was ready for compilation using RPL11 compiler. The interface produced contained 243 RPL11 lines (Appendix A4).
It, of course, compiled without error since format corrections had been dealt with earlier. More gratifying was that it performed reliably first time too, which indicated reliability of the design.

As well as staging the test to develop reliably it is proposed to repeat stage four testing with different classes of user namely:

- Very experienced - the developers
- Experienced - computer knowledgeable staff
- Inexperienced - end user department staff

This level of testing is considered highly important since the 'developers' view of the system may possess highly subjective elements.

The second test was carried out by the experienced user (computer knowledgeable staff) by repeating the whole process mentioned in 5.2 to provide a time for each step so that relative performance measures could be taken. The third test was carried out by the inexperienced user, also by repeating the whole process mentioned in 5.2 noting a time for each step.

Two examples follow (Examples 5.4, 5.5) illustrating the use of the generated interfaces. Also, observations were made concerning attitudes relating to the use of the generator and the interface generated.

In practice every item in the menu system would be exercised to confirm the proper installation of the interface. However, these two (incomplete) examples fully illustrate the uniformity of operation and general control provided by the generated interface when in use.
Ex 5.4: Sample dialogue using the generated interface (run in hard copy terminal for listing purposes)

R R
RPL> LCS
PLEASE ENTER PASSWORD:

DO YOU KNOW WHICH PROGRAM YOU WANT TO USE.
PLEASE ENTER YES OR NO: NO

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED: 1

11 - CREATE A SOURCE RPL FILE.
12 - FORMATTED SOURCE FILE.
13 - LISTS *DICT WORKING OUT START POST.
14 - RENAME PAGES ON LIBRARY PAGE FILE.
15 - CHECK LIBRARY.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED: 11

ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N: Y
PROGRAM NAME? CHECK RPL
*IFILE 3 R ARPL LPF
*
*DICT
L /512, 8 ;2, P :2, V /34
*
*DETAB ONE
I B MY 8.
(Continued Ex.5.4)

GOTO TWO

*DETAB TWO
I GET 3 L B
P MV A'L
C P = A'L+510
(P):2 = 0
A V MV (P)/34+
PRINT P L
REPEAT
B MV (P);2
GOTO TWO

*LI  L
A V, B V+32:2
*PI  P
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA 8BBBB
+50
END OF RUN

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED: 6

11 - RUN A PROGRAM.
12 - ALLOWS PATCHES TO FILES IN ASC/DEC.
13 - PRINTS ASCII FILES ON TT: HANDLING PP.
14 - LISTS ASCII FILES ON TT: HANDLING FED.
15 - INTERROGATE LIBRARY CONTROL FILE.
16 - RETURN TO THE MAIN MENU.
PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 11

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y

PLEASE ENTER DEVICE:FILENAME CONTAINING THE PROGRAM
NAME REQUIRED TO RUN WITH OR WITHOUT EXTENTION : ? FD1:LC5002

PLEASE ENTER APPLICATION NAME : ? LIBRARY

0/6 IS TOO LONG. PLEASE RE-INPUT

? LIBRARY

PLEASE ENTER PROGRAM NAME : ? CHECK

PLEASE ENTER PROGRAMMER NAME : ? OSSAMA MUSLIM

PLEASE ENTER COMPLETION DATE : ? 30-OCT-79

OUTPUT RPL NAME? CHECK.RPL

FUNCTION OF THE PROGRAM : <PLEASE ENTER FIFTY CHARACTER
PER LINE BUT TERMINATE BY FULL STOP ON LINE OF IT IS OWN.>

THIS PROGRAM ALLOWS THE USER TO CHECK THE LIBRARY FILE
WHICH IS STORED IN ARPL.LPF.

SOURCE RPL NAME? CHECK.RPL

PROGRAM READY.

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 7

THE KEYWORDS IN THE HELP FILE ARE :

| 01 LANGUAGE | 02 INITIALISE | 03 MANIPULATE | 04 SYSTEM |
| 05 PAGE     | 06 OTHERS    | 07 CREATE     | 08 FORMAT |
| 09 LIST     | 10 RENAME    | 11 CHECK      | 12 SOURCE |
| 13 DATABASE | 14 PRINTSORT | 15 HEADER     | 16 REPLACE|
| 17 COPY     | 18 CONTROL   | 19 IDENTITY   | 20 DATE   |
| 21 LOG      | 22 DISKETTE  | 23 USAGE      | 24 ADDITIONS|
| 25 LIBRARY  | 26 RUN       | 27 PATCHES    | 28 PAUSES |
| 29 FEEDS    | 30 UTILITY   | 31 SCALE      |          |

PLEASE ENTER KEYWORD NUMBER REQUIRED : 17
COPY

THIS PROGRAM ALLOWS THE USER TO COPY ANY ASCII FILE TO ANOTHER FILE WITH A NEW NAME. THE USER RESPONDS TO TERMINAL REQUESTS AND ENTERS THE DATA VIA THE KEYBOARD WHICH IS THE OLD FILE NAME OR DEVICE NAME AND THE NEW NAME OF THE FILE OR DEVICE. CONTROL IS THEN RETURNED TO THE MAIN MENU.

PLEASE USE SYSTEM FUNCTION COPY (UNDER FILE MANIPULATION) TO LIST FILE LCLS011.RPL FOR PROGRAM DETAILS.

DO YOU REQUIRE FURTHER INFORMATION. PLEASE ENTER YES OR NO ? NO

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 3

11 - REPLACE SPACES BY TABS IN ASCII FILE.
12 - COPY A FILE.
13 - RENAME A FILE.
14 - KILL A FILE.
15 - SORT A FILE.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 12

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
(Continued Ex.5.4)

PLEASE ENTER FILE NAME WITH EXTENSION FOR
THE INPUT AND OUTPUT FILES.

FROM ? FDO: CHECK.RPL
TO ? IT

* LOUGHBOURGH UNIVERSITY OF TECHNOLOGY
* COMPUTER STUDIES DEPARTMENT
* THIS MODULE BELONGS TO LIBRARY SYSTEM
* PROGRAM SOURCE FORMATTED BY FORMAT.RPL
* **********************************************
* APPLICATION NAME - LIBRARY
* PROGRAM NAME - CHECK
* PROGRAMMER NAME - OSSAMA MUSLIH
* COMPLETION DATE - 20-OCT-79
* FUNCTION OF THE PROGRAM:
* THIS PROGRAM ALLOWS THE USER TO CHECK THE LIBRARY FILE
* WHICH IS STORED IN ARPL.LPF.
*IFILE 3 R ARPL.LPF
*DICT
L /512, B ;2, P ;2, Y /34
*DETAB ONE
I B MV S
GOTO TWO
*
*DETAB TWO
I GET 3 L B
P MV A;L
C P = A;L+510 N Y
(P);2 = 0 N N
A V MV (P);/34+ X -
PRINT P L X -
REPEAT X -
B MV (P);2 - X
GOTO TWO - X
*
*LI L
A V, B V+32:2
*PI P
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA BBBBB
*GO
FILE COPIED

1 - LANGUAGE SYSTEM.
2 - FILE Initialisation.
3 - FILE Manipulation.
4 - SYSTEM FILES.
(Continued Ex. 5.4)

5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 8

END OF APPLICATION
RPL>
Ex. 5.5: Sample dialogue using the generated interface (run in hard copy terminal for listing purposes)

```
.R R
RPL> PAY
PLEASE ENTER PASSWORD:

DO YOU KNOW WHICH PROGRAM YOU WANT TO USE.
PLEASE ENTER YES OR NO : NO

1 - MANIPULATE MASTER.
2 - REVIEW PREV. PAYROLL.
3 - RUN PAYROLL & AMEND.
4 - DATA INPUT.
5 - PRELIST EMP. & KILL.
6 - NONE.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 1

11 - CREATION AND EXTENSION MASTER FILE.
12 - INTERROGATION OF THE MASTER FILE.
13 - NONE.
14 - NONE.
15 - NONE.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 11

ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N : Y
```
THIS PROGRAM ENABLES THE CREATION OR ADDITION OF EMPLOYEE RECORDS TO THE PAYROLL MASTER FILE

PLEASE ENTER FIRST 3 DIGITS OF EMPLOYEE NUMBER . ... ? 010

PLEASE ENTER DATA

LAST 2 DIGITS OF EMPLOYEE NUMBER . ... ? 10

DEPARTMENT CODES ARE:

<table>
<thead>
<tr>
<th>CODE</th>
<th>DEPARTMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>MARKETING</td>
</tr>
<tr>
<td>02</td>
<td>RESEARCH</td>
</tr>
<tr>
<td>03</td>
<td>MANUFACTURING</td>
</tr>
<tr>
<td>04</td>
<td>DATA PROCESSING</td>
</tr>
<tr>
<td>05</td>
<td>MAINTENANCE</td>
</tr>
<tr>
<td>06</td>
<td>PERSONNEL AND ADMINISTRATION</td>
</tr>
<tr>
<td>07</td>
<td>DISTRIBUTION</td>
</tr>
<tr>
<td>08</td>
<td>NON-CLASSIFIED</td>
</tr>
</tbody>
</table>

ENTER DEPARTMENT CODE . . . ? 02
SURNAME . . . ? MUSLHI
FORENAME . . . ? OSSAMA
STATUS (I.E MR, MRS, MISS, DR ETC.) . . . ? MR
INITIALS . . . . ? O. M.
(Continued Ex. 5.5)

MARITAL STATUS CODES ARE
----------------------------------

<table>
<thead>
<tr>
<th>CODE</th>
<th>MARITAL STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MARRIED</td>
</tr>
<tr>
<td>2</td>
<td>SINGLE</td>
</tr>
<tr>
<td>3</td>
<td>WIDOW / WIDOWER</td>
</tr>
<tr>
<td>4</td>
<td>DIVORCED</td>
</tr>
</tbody>
</table>

MARITAL STATUS CODE . . . . ? 2
ADDRESS . . . . ? 71 LOWESWATER DRIVE, LOUGHBOROUGH
TELEPHONE NUMBER . . . . ? 61255
DATE OF BIRTH ( xx-xxx-xx ) . . . . ? 23-AUG-48
DEPARTMENT OF HEALTH AND SOCIAL SECURITY NUMBER . . . . ? 367

COMPANY POSITION CODES ARE

<table>
<thead>
<tr>
<th>CODE</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DIRECTOR</td>
</tr>
<tr>
<td>2</td>
<td>MANAGERIAL</td>
</tr>
<tr>
<td>3</td>
<td>SUPERVISORY</td>
</tr>
<tr>
<td>4</td>
<td>TECHNOLOGIST / SCIENTISTS</td>
</tr>
<tr>
<td>5</td>
<td>SALES REPRESENTATIVE</td>
</tr>
<tr>
<td>6</td>
<td>COMPUTER STAFF</td>
</tr>
<tr>
<td>7</td>
<td>OTHER PROFESSIONAL AND ADMINISTRATIVE STAFF</td>
</tr>
<tr>
<td>8</td>
<td>TECHNICIAN</td>
</tr>
<tr>
<td>9</td>
<td>CRAFTSMAN</td>
</tr>
<tr>
<td>10</td>
<td>OPERATOR</td>
</tr>
<tr>
<td>11</td>
<td>DRIVER</td>
</tr>
<tr>
<td>12</td>
<td>OFFICE SUPPORT STAFF</td>
</tr>
<tr>
<td>13</td>
<td>NON-CLASSIFIED</td>
</tr>
<tr>
<td>14</td>
<td>PERSONNEL</td>
</tr>
<tr>
<td>15</td>
<td>MAINTENANCE</td>
</tr>
<tr>
<td>16</td>
<td>CLEANING STAFF</td>
</tr>
<tr>
<td>17</td>
<td>SECURITY</td>
</tr>
</tbody>
</table>

COMPANY POSITION CODE . . . . ? 7
HOURLY OR SALARY PAID ( H OR S ) . . . . ? 5

PAY RATE CODES ARE:

<table>
<thead>
<tr>
<th>SALARY PAID</th>
<th>HOURLY PAID</th>
</tr>
</thead>
<tbody>
<tr>
<td>A - $24,700</td>
<td>J - 250P</td>
</tr>
<tr>
<td>B - $13,000</td>
<td>K - 200P</td>
</tr>
<tr>
<td>C - $10,000</td>
<td>L - 175P</td>
</tr>
<tr>
<td>D - $8,200</td>
<td>M - 150P</td>
</tr>
<tr>
<td>E - $6,000</td>
<td>N - 125P</td>
</tr>
<tr>
<td>F - $5,500</td>
<td>O - 100P</td>
</tr>
<tr>
<td>G - $4,500</td>
<td></td>
</tr>
<tr>
<td>H - $4,000</td>
<td></td>
</tr>
<tr>
<td>I - $3,200</td>
<td></td>
</tr>
</tbody>
</table>
(Continued Ex.5,5)

<table>
<thead>
<tr>
<th>Description</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pay Rate Code</td>
<td>? I</td>
</tr>
<tr>
<td>Tax Code</td>
<td>? TXCD</td>
</tr>
<tr>
<td>National Insurance (Weekly)</td>
<td>? 20</td>
</tr>
<tr>
<td>National Insurance (Year)</td>
<td>? 60</td>
</tr>
<tr>
<td>Superannuation Percentage?</td>
<td>15</td>
</tr>
<tr>
<td>Superannuation (Week)</td>
<td>? 52</td>
</tr>
<tr>
<td>Superannuation (Year)</td>
<td>? 200</td>
</tr>
<tr>
<td>DEDUCTIONS</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
<tr>
<td>*** NOTE *** PLEASE INPUT AMOUNT IN $</td>
<td></td>
</tr>
<tr>
<td>PER WEEK FOR HOURLY PAID EMPLOYEES</td>
<td></td>
</tr>
<tr>
<td>PER MONTH FOR SALARY PAID EMPLOYEES</td>
<td></td>
</tr>
<tr>
<td>UNION FEES</td>
<td>? 10</td>
</tr>
<tr>
<td>Life Insurance</td>
<td>? 2</td>
</tr>
<tr>
<td>Social Club Membership</td>
<td>? 022</td>
</tr>
<tr>
<td>Loan Interest</td>
<td>? 2</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>? 000</td>
</tr>
<tr>
<td>Travelling Allowance</td>
<td>? 12</td>
</tr>
<tr>
<td>HOURS WORKED</td>
<td>? 39</td>
</tr>
<tr>
<td>STOP PAY FLAG (1 ELSE 0)</td>
<td>? 1</td>
</tr>
<tr>
<td>DATE PAY STOPPED (XX-XXX-XX)</td>
<td>? 5-JUL-79</td>
</tr>
<tr>
<td>GROSS PAY (WEEK)</td>
<td>? 58</td>
</tr>
<tr>
<td>GROSS PAY (YEAR)</td>
<td>? 330</td>
</tr>
<tr>
<td>TAX PAID (WEEK)</td>
<td>? 2</td>
</tr>
<tr>
<td>TAX PAID (YEAR)</td>
<td>? 200</td>
</tr>
<tr>
<td>PAY AFTER TAX (WEEK)</td>
<td>? 48</td>
</tr>
<tr>
<td>PAY AFTER TAX (YEAR)</td>
<td>? 29</td>
</tr>
<tr>
<td>TAKE HOME PAY (WEEK)</td>
<td>? 71</td>
</tr>
<tr>
<td>TAKE HOME PAY (YEAR)</td>
<td>? 103</td>
</tr>
<tr>
<td>WEEKS OFF WORK THIS YEAR</td>
<td>? 4</td>
</tr>
<tr>
<td>COMMENT FIELD. (18 CHARACTERS LONG)</td>
<td></td>
</tr>
<tr>
<td>THANK YOU. DO YOU WISH TO CREATE MORE EMPLOYEE RECORD, PLEASE TYPE Y OR N :</td>
<td>? N</td>
</tr>
</tbody>
</table>

1 - MANIPULATE MASTER.
2 - REVIEW PREV. PAYROLL.
3 - RUN PAYROLL & AMEND.
4 - DATA INPUT.
5 - PRELIST EMP. & KILL.
6 - NONE.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 2

11 - GROSS WEEKLY EARNING REG. REPORT.
12 - TAX WITHHOLDING REGISTER REPORT.
13 - DEDUCTION REGISTER REPORT.
14 - PAYROLL REGISTER REPORT.
15 - CHEQUES & CHEQUE STABS PRODUCTION.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 12

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
PLEASE INPUT YOUR REPORT OPTION . . . ? 2
DO YOU REQUIRE REPORT ON (LP: OR IT:). . . . ? IT:

ABC COMPANY LIMITED

TAX WITHHOLDING REGISTER REPORT

15-MAR-79

<table>
<thead>
<tr>
<th>EMPLOYEE NUMBER</th>
<th>SURNAME</th>
<th>INITIALS</th>
<th>HOURS WORKED</th>
<th>OVERTIME HOURS</th>
<th>GROSS PAY</th>
<th>P.A.Y.E TAX</th>
<th>PAY AFTER TAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>03 99</td>
<td>SMITHE</td>
<td>S. S.</td>
<td>0</td>
<td>0</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>01131</td>
<td>FAILEY</td>
<td>D. A</td>
<td>49</td>
<td>9</td>
<td>$108.00</td>
<td>$29.42</td>
<td>$78.58</td>
</tr>
<tr>
<td>01328</td>
<td>STAFFORD</td>
<td>J. S</td>
<td>55</td>
<td>15</td>
<td>$165.00</td>
<td>$44.74</td>
<td>$120.26</td>
</tr>
<tr>
<td>01852</td>
<td>NORTON</td>
<td>R.</td>
<td>39</td>
<td>0</td>
<td>$53.50</td>
<td>$8.90</td>
<td>$49.60</td>
</tr>
</tbody>
</table>
(Continued Ex. 5.5)

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>M.O</th>
<th>Hrs</th>
<th>Ohr</th>
<th>P/H</th>
<th>O/P</th>
<th>G/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>02543</td>
<td>REDFORD</td>
<td>M.O</td>
<td>45</td>
<td>0</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>02737</td>
<td>RHODES</td>
<td>P.G</td>
<td>51</td>
<td>11</td>
<td>$58.00</td>
<td>$9.43</td>
<td>$48.57</td>
</tr>
<tr>
<td>04219</td>
<td>PARKET</td>
<td>C.W</td>
<td>31</td>
<td>0</td>
<td>$54.25</td>
<td>$11.68</td>
<td>$42.57</td>
</tr>
<tr>
<td>05210</td>
<td>FARRIS</td>
<td>H.G</td>
<td>45</td>
<td>5</td>
<td>$71.25</td>
<td>$17.29</td>
<td>$53.96</td>
</tr>
<tr>
<td>05321</td>
<td>BURGESS</td>
<td>E.J</td>
<td>0</td>
<td>0</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>05454</td>
<td>BURGESS</td>
<td>F.</td>
<td>0</td>
<td>0</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>07642</td>
<td>MAASSIAH</td>
<td>W.B</td>
<td>0</td>
<td>0</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>07854</td>
<td>SLEE</td>
<td>F.A</td>
<td>0</td>
<td>0</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>08047</td>
<td>SEARLE</td>
<td>T.A</td>
<td>0</td>
<td>0</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>09554</td>
<td>BUERRY</td>
<td>S.I</td>
<td>44</td>
<td>4</td>
<td>$63.00</td>
<td>$16.55</td>
<td>$52.45</td>
</tr>
</tbody>
</table>

TOTAL HOURS WORKED = 359 HOURS
TOTAL OVERTIME WORKED = 44 HOURS
TOTAL GROSS PAY = $584.00
TOTAL P.A.Y.E TAX = $445.98
TOTAL PAY AFTER TAX = $445.98

1 - MANIPULATE MASTER.
2 - REVIEW PREV. PAYROLL.
3 - RUN PAYROLL & AMEND.
4 - DATA INPUT.
5 - PRELIST EMP. & KILL.
6 - NONE.
7 - HELP.
8 - EXIT SYSTEM.
(Continued Ex.5.5)

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 3

11 - RUN PAYROLL.
12 - AMENDMENT.
13 - NONE.
14 - NONE.
15 - NONE.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 12

ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N : Y

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

THIS PROGRAM CARRIES OUT AMENDMENTS

TO THE PAYROLL MASTER FILE

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

PLEASE ENTER FIRST 3 DIGITS OF THE EMPLOYEE
RECORD THAT YOU WISH TO CHANGE . . . . ? 11

RECORD CATEGORIES FOR AMENDMENT

1: NAME, ADDRESS AND PERSONAL DETAILS
2: EMPLOYMENT DETAILS
3: DEDUCTIONS AND EXPENSES
4: TAX CODE AND DHSSNO
(Continued Ex.5.5)

5: HOURS WORKED AND STOP PAY DETAILS
6: WEEKLY TOTALS
7: YEARLY TOTALS
8: COMMENT FIELD

PLEASE INPUT BY NUMBER, THE AREA TO BE AMENDED . . . . . . . ? 4

TAX CODE AND DHSS NUMBER
=================================

1. - TAX CODE.
2. - DEPARTMENT OF HEALTH AND SOCIAL SECURITY NUMBER.
3. - EXIT TO UPPER LEVEL.

PLEASE SELECT BY NUMBER, WHICH FIELD YOU WISH TO CHANGE . . . . ? 1

FIELD CONTENTS = 098L

INPUT NEW CONTENTS ? 096L

DO YOU WISH TO MAKE ANY MORE AMENDMENTS (Y OR N) . . . ? N

1 - MANIPULATE MASTER.
2 - REVIEW PREV. PAYROLL.
3 - RUN PAYROLL & AMEND.
(Continued Ex.5.5)

4 - DATA INPUT.
5 - PRELIST EMP. & KILL.
6 - NONE.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 8.

END OF APPLICATION
RPL>
CHAPTER 6

CONCLUSION AND FURTHER DEVELOPMENT
6.1 CONCLUSION

It may be concluded that it is feasible to obtain a short term solution to the problem of generating a man-computer interface for controlling commercial applications. Results indicate that this project has accomplished the aims of developing and testing a program which is the generator described in this report. This is capable of producing another program which is a terminal dialogue interface that is comfortable for the casual user. This terminal dialogue contains features determined by the user for any general purpose application system consisting of a number of functional programs. It has already been tested using application programs as a target and produced satisfactory results. These impressions are, of course, highly subjective and the tests must be broadened to include less dedicated operators. The interface achieves the objectives of bringing together logically and functionally related separate programs in such a way that the user sees the whole system in a more meaningful way. Also, the generated interface allows the user to achieve a high degree of familiarity of the system immediately at minimal programming cost since no programs are written manually. However, it is believed that such generators will be essential and in common use in the future, if only due to economic factors.

Restrictions are bound to be introduced in the performance of such standardised dialogues and will be tolerated for the same reason that it is desirable to use a high level language and tolerate its restrictions.

At present the generator and its interface is written in FILETAB (RPL) which is found to be an excellent development language for initial feasibility exercises of this nature, since it allows for very rapid development. Unfortunately, it has not the portable aspects of a more internationally standardised language such as COBOL or FORTRAN.
This project has considered the interface problems relating to a single user working under the control of a single-job operating system. An obvious extension to this would be to consider the interface problems relating to many users working in a multiprogramming environment; in this case further research would be made to improve development methods to handle security and accounting aspects.

6.2 FURTHER DEVELOPMENT WORK

An appraisal has been made and a decision taken to develop an enhanced system using COBOL to improve its commercial attractiveness. The generator can be developed in future to expand its capability to produce for example three different interfaces which will conform to three different techniques as follows:

1. Menu selection technique - for casual users
2. Question and answer technique - for intermediate users
3. Command language technique - for dedicated users.

A login program could also be generated to control the whole system usage; this login program could be used to check the information which is already stored in a file to ensure that usage is authorised for example. The information in the login file could include such things as:

a. User code number
b. Start date
c. Last login date
d. The terminal used
e. Violation black marks

The system could also contain a feature which maintains a profile for each user which accesses data items, these profiles are set up by a special privileged program and usually can be represented by an authorisation matrix.
The security requirements conflict to some extent with the requirement that the system should be as easy as possible to use; the designer wants his dialogue structure to encourage everybody to use terminals; on the other hand he must prevent them seeing or modifying information they are not entitled to. He wants to avoid making the sign-on procedure an obstacle course that will daunt the would-be user (as often happens today), and so the security procedures, tight though they must be, should be built in with attention to user psychology not merely with the mentality of the user.

The disadvantage of the password or security code techniques is that the code can be given to another person without any physical loss by the giver and without anything having to be duplicated. There is no physical evidence of the other person's possession of it. This technique must, therefore, be accompanied by rigorous controls and serious attempts to detect quickly and automatically. If the terminal users think that there is a high probability that they will be caught if the attempt to enter the system with another person code, then they may be deterred psychologically from making an invalid entry.

Below are listed some of the techniques which could be used in security:

1. For each application there is a password.
2. Time out mechanism for a certain period after login.
3. A history be kept in which all changes that were made to these security records are logged, indicating who made the change and where it was made.
4. For each user there is a password.
5. Authentication (personal questions).
6. Identifying the terminal.

7. User authorisation file which consists of all the information which belong to his access for any application.

6.3 SUMMARY

This project has attempted to contribute to the techniques and methods applied in the area of user support. Current technology is quite capable of ensuring that the computer aspects of a given system are self-evident or transparent to the user. It is quite clear that most systems in use require a fair amount of support by the way of 'program advisory services' or 'consultants' or 'user groups' etc. which attempt to compensate for support lacking in the interface provided, i.e. user involvement and user support are closely linked - both influence the effectiveness of any computer application. The main purpose of user support is to compensate for the lack of 'transparency' in the man-computer interface; interaction with a computer usually requires the human user to compensate for the limitations and short-comings of the technological system.

It is important to be able to define the scale and type of user support. It is also useful to identify the minimal user support which is required to sustain man-computer interaction. This minimum requirement appears to be that which ensures:

1. That essential data are provided to the system by users.

2. That system output is received, understood and assimilated.

Another purpose of user support is to promote continued viability of man-computer interaction. This evolutionary user support function is complex and varied since it is concerned with an evolving computer system applied in a changing environment through the activities of a user who is also
changing as a result of his learning from experience [3]. Successful computer applications are characterised by the number and diversity of mechanisms which have arisen to meet these needs. The importance of supporting an evolving man-computer relationship lies in the fact that unless a given system can continue to develop and to match the needs of the environment in which it functions, it ceases to have any value.

The generative approach developed here is considered a valuable contribution to the set of tools enabling the technological system to evolve to meet the demands referred to above.
APPENDICES
APPENDIX A1

User Guide
1. **INTRODUCTION**

This document is the user guide for testing the interface generator. The aim was to explore the concept of providing the users of a computer system with a man-computer interface generated by a generator program. The feasibility of such an interface and its usefulness as a tool were investigated.

The system was predominantly a design concept and the research work carried out was basically to bring together existing knowledge about interfaces, human behaviour with respect to computers, ideas about the facilities that a human-oriented interface should provide, and the techniques currently available for structuring man-computer interfaces.

This project has accomplished the aims of developing and testing a program (the generator) which is capable of creating another program (the interface) containing features determined by the user for any general purpose application system consisting of a number of functional programs. The interface achieves the objective of bringing together logically and functionally related separate programs in such a way that the user sees the whole system in a more meaningful way, also the generated interface allows the user to achieve a high degree of familiarity of the system immediately at minimal programming cost since no programs are written manually.

2. **USE OF THE INTERFACE GENERATOR**

The interface generation process consists of four basic steps:

* The planning necessary to determine the desired components of the application and their descriptions.
* The execution of the generator.
* The compilation of the interface source code generated.
Recompilation of the application source code with amendments necessary and a routine check that the new system is installed without error in the storage volume required.

2.1 **Planning**

Before attempting to use the generator the user should first investigate the target application and check for the following, some of which is required to respond intelligently to the generation dialogue:

1. Source code is available for every program module in the system.
2. Each program module has been assigned a name within the system, and a very brief function for each listed.

The user should give each program a new name within the system by using the first three characters from the application name followed by three digits for the program number, for example 'PAY001' is the name of the first program from PAYROLL application.

To change the names of all the programs, the user must use 'RENAME' command, for example:

```
.RENAME FD?:FILENAME1,EXT TO FD?:FILENAME2,EXT
```

in which

- **FD?:** - The device on which the file is stored
- **FILENAME1** - The old file name
- **FILENAME2** - The new file name
- **.EXT** - Either extension RPL or RPC

3. The user should look at the end of each source program, examine how it finishes, and draw a simple diagram showing the connection (if any) with the other programs. The user must then include the 'CHAIN' command (RPL) to connect all the programs to the interface program; the user must choose the name of the interface
program at this stage to chain all the programs to it; this is carried out using the editor.

The user should also include "*CORCOM:2" field by setting it to 1 immediately before the 'CHAIN' command, whether it is in the middle of the program or in the *DETAB ENDOFJOB decision table at the end of the program.

*CORCOM:2 field is used to pass data from one program to another, and it is necessary in this case to avoid the appearance of the password question every time if the user wishes to return to the main menu in the interface program, for example, if the name of the interface program is PAY (from PAYROLL application) then the user should use the "*CORCOM:2" field and 'CHAIN' command as follows:

*DETAB ENDOFJOB
A *CORCOM:2 MV 1
      CHAIN 'PAY'
*STOP

4. A more detailed description of the function of each program prepared for entry into the help file 'HELP.TXT' which will be used for on-line interrogation.

The next stage is the execution of the generation program, but before this execution the user must divide the programs of the system into categories in the main menu. Because the application package is assumed to have six basic categories of programs, and no more than five program modules per category, the names the user gives to these categories will appear in the first control menu presented. Two other items will appear (namely, Help and Exit system), a separate menu for each category will later be defined. The programs in each category should as far as possible deal with the same function within the system, for example, all
the programs dealing with the master files should be included in
the same category. At the end of this part the feature of the
system as shown below:

```
   Login
   A
   category one  category two  category three  category four  . . .
     |        |        |        |
   prog 1 prog 2 prog 3 prog 4 prog 5 prog 1 prog 2 prog 3 prog 4 prog 5
```

2.2 Execution of the Generator

On completion of the planning stage the user should run the
INTERFACE GENERATOR program which is called 'INTGEN' but before that the
user must run the run time system by:

```
  .RR
```
The system will reply by:

```
RPL>
```
At this stage the user should enter the program name which is wanted to
run, 'INTGEN' in this case:

```
RPL>INTGEN
```
The INTGEN asks a series of questions and requests for information, and uses
the response made to direct further prompting and construct the output file
containing the user interface. The operating instructions which follow
define the dialogue to be expected.
**OPERATING INSTRUCTIONS**

<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>INTGEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGRAM NO:</td>
<td></td>
</tr>
<tr>
<td>SYSTEM:</td>
<td></td>
</tr>
</tbody>
</table>

Program to generate an interface program (USERIF.RPL) for a package containing less than 36 program modules. See NOTES below.

<table>
<thead>
<tr>
<th>STEP</th>
<th>VDU PROMPT</th>
<th>OPERATOR ACTION/COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RPL&gt;</td>
<td>Enter INTGEN to start the generator program.</td>
</tr>
<tr>
<td>2</td>
<td>Please press return key if you want to continue</td>
<td>The user should press return key to receive next screen of information.</td>
</tr>
<tr>
<td>3</td>
<td>Please enter name of application (no more 3 characters)</td>
<td>Enter 3 characters. These become the family name of the programs in the package.</td>
</tr>
<tr>
<td>4</td>
<td>Please enter valid password (no more 6 characters)</td>
<td>The user should enter 6 characters. These characters are incorporated into the generated program as a password for the package.</td>
</tr>
</tbody>
</table>

**NOTES:**

The package is an application system of programs for which an interface is required. Details of the package should be completely familiar before attempting to run INTGEN.
## OPERATING INSTRUCTIONS

### PROGRAM: INTGEN

<table>
<thead>
<tr>
<th>STEP</th>
<th>VDU PROMPT</th>
<th>OPERATOR ACTION/COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Which device you prefer your output file?</td>
<td>If the user prefers his output on the lineprinter he should enter LP: If on the teletype he should enter TT: If on disk he should enter which disk (FDO: or FD1:) followed by the interface program name which no more 6 characters ended by .RPL</td>
</tr>
<tr>
<td>6</td>
<td>Enter first category in menu (no more 20 characters)</td>
<td>The user should enter the details of main control menu, which are the names of all the categories of sub-menu.</td>
</tr>
<tr>
<td>7</td>
<td>Enter number of sub-menus</td>
<td>The user should enter the number of sub-menus which is equal to the number of categories entered before in Step 6,</td>
</tr>
</tbody>
</table>

### NOTES:

- **Step 5** - If the interface is to be compiled, the 3rd option (disk) must be chosen
- **Step 6** - Will be repeated until the last category in the main menu will be entered.
## OPERATING INSTRUCTIONS

<table>
<thead>
<tr>
<th>PROGRAM: INTGEN</th>
<th>PROGRAM NO:</th>
<th>SYSTEM:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### OPERATOR ACTION/COMMENT

<table>
<thead>
<tr>
<th>STEP</th>
<th>VDU PROMPT</th>
<th>OPERATOR ACTION/COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Enter first module details:</td>
<td>The user should enter a brief meaning no more 38 characters about the function of the first program in the first category.</td>
</tr>
<tr>
<td>9</td>
<td>Enter program number (no more 3 digits):</td>
<td>The user should enter the number of the program associated with the function entered at step 8.</td>
</tr>
<tr>
<td>10</td>
<td>Enter next module details:</td>
<td>The user should enter a brief meaning no more 38 characters about the function of the second program in the first category.</td>
</tr>
<tr>
<td>11</td>
<td>Enter program number (no more 3 digits)</td>
<td>The user should enter the number of the second program entered its function in Step 10.</td>
</tr>
</tbody>
</table>

### NOTES:
Step 10 and 11 will be repeated until the last module details and its number will be entered. The whole sequence repeats in this manner for each category entered at step 6.
The execution of the generator is then terminated with the message:

"End of generation program"

At this stage the generation program is already chained to another program called 'DBTEST', the function of this program is to initialise a file called 'HELP.TXT' which will be used for on-line interrogation and it will be 80 blocks and ready to use for entering more detailed description of the function of each program in the system.

The execution of the 'DBTEST' program does not need any response from the user, but when the initialisation process is finished the 'DBTEST' program is already chained to another program called 'CRHELP', the function of this program is to create the help file which is initialised before by 'DBTEST'. The CRHELP program asks a series of questions and requests for information, and uses the response made to direct further prompting, Operating dialogue is as follows:
**OPERATING INSTRUCTIONS**

<table>
<thead>
<tr>
<th>PROGRAM: CRHELP</th>
<th>PROGRAM NO:</th>
<th>SYSTEM:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Program to create a file called (HELP.TXT) containing information about the function of each program in the application package.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STEP</th>
<th>VDU PROMPT</th>
<th>OPERATOR ACTION/COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Do you wish to continue, please enter Yes or NO?</td>
<td>If the user answer is 'NO' it means that the user does not need to continue with the creation process of (HELP.TXT), and the whole process will terminate and it will return to the control system. If the user answer is 'YES' the creation process carried out to the next step.</td>
</tr>
<tr>
<td>2</td>
<td>Please enter keyword number:</td>
<td>Enter no more than 2 digits. These become the serial number for the first program.</td>
</tr>
<tr>
<td>3</td>
<td>Please enter keyword name:</td>
<td>Enter no more than 20 characters. These become the name of the first program mentioned in step 2.</td>
</tr>
</tbody>
</table>

**NOTES:**
OPERATING INSTRUCTIONS

<table>
<thead>
<tr>
<th>STEP</th>
<th>VDU PROMPT</th>
<th>OPERATOR ACTION/COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Please enter keyword details and terminate by 'STOP' on line of its own:</td>
<td>The user should enter information about the function of the program mentioned in step 3, and terminate this information by the 'STOP' on line of its own.</td>
</tr>
<tr>
<td>5</td>
<td>Do you wish to enter more keywords, please enter YES or NO.</td>
<td>If the user answer is 'YES' it will return to step 2 and the process repeated for the second program. If the user answer is 'NO', it means that the whole process will terminate by creating a new record with '****' for the keyword name.</td>
</tr>
</tbody>
</table>

NOTES: The Help item in the main menu chained to a program called 'REHELP', the function of this program to read the 'HELP.TXT' when the user needs it during the use of the interface program.
2.3 Compilation of the Interface and Modification of Application Source Code

The output file from the previous stage can now be compiled by the RPL compiler (RPL.RPC) to produce an executable interface program, the compilation process as follows after running the run time system:

\[ RPL \rightarrow RPL \]
\[ \# , TT := \text{PROGRAM NAME} \]

The program name in this case is the interface program which the user created in the previous stages, the compilation will appear on the teletype. If the user wishes to obtain the compilation list, he should type LP instead of TT above.

All the application programs are now examined for logical exits and changed, so that they return control to (CHAIN) the interface program, these are compiled and objects renamed as part of an application system consistent with decisions taken in Section 2.1 of this Appendix.

Finally, the user should test the system just generated. To do this the interface program is executed and a systematic use of each item in the main and sub-menu made for correct performance checking [Appendix A3(ii)]. Omissions or errors in previous stages will be discovered in this test. Corrective action is then taken and re-checked before the system is released to the user community.
APPENDIX A2

Program Listing of INTGEN.RPL: The Generator
1 *
2 *
3 *
4 *
5 *FILE "WHICH DEVICE :"
6 *DICT
7 L -1000;2
8 *LI GEN
9 A -80/80
10 *PICTURE GEN.LINE 1,0
11 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
12 *DETAB CONTROL
13 A 300/15 MV 'TEST RECORD
14 CALL BANNER
15 CALL QUESTIONNAIRE
16 CALL DUMMY.MENU GEN
17 CALL POST.SCRIPT
18 DELETE 'END OF GENERATION PROGRAM'
19 *DETAB BANNER
20 A CALL STARS
21 CALL SPACELINE
22 CALL SPACELINE
23 CALL TITLELINE1
24 CALL TITLELINE2
25 CALL DATELINE
26 CALL SPACELINE
27 CALL STARS
28 EXIT T
29 *DETAB STARS
30 A -80/80 S
31 -80/80 FILL *'
32 -79/1 S
33 CALL DISPLAYLINE
34 EXIT T
35 *DETAB DISPLAYLINE
36 A DISPLAY -80/80
37 -80/80 S
38 EXIT T
39 *DETAB SPACELINE
40 A -80/80 S
41 -80/1 MV '*'
42 -1/1 MV '*'
43 CALL DISPLAYLINE
44 EXIT T
45 *DETAB TITLELINE1
46 A -80/80 S
47 -80/1 MV '*'
48 -1/1 MV '*'
49 -60/40 MV ' Loughborough University Software
50 CALL DISPLAYLINE
51 EXIT T
52 *DETAB TITLELINE2
53 A -80/80 S
54 -80/1 MV '*"
PERSONALISED INTERFACE GENERATOR

CALL DISPLAYLINE

EXIT T

*DETAB DATELINE

A -80/80 S

-80/1 MV '*'

-1/1 MV '*'

-48/9 MYDATE$ 0

CALL DISPLAYLINE

EXIT T

*DETAB QUESTIONNAIRE

A CALL TEXT

CALL USER. DESCRIPTIONS

EXIT T

*DETAB TEXT

A & '

' NOTE:

'------

' IF MISTAKES ARE MADE IN ENTERING DATA, PLEASE MAKE A NOTE AND

' CONTINUE WITH THE GENERATION PROCESS. CORRECTION TO THE INTERFACE GENERATED MAY BE MADE AT THE LATEST STAGE THROUGH USING

' THE EDITOR.

YOUR APPLICATION PACKAGE IS ASSUMED TO HAVE SIX BASIC CATEGORIES

OF PROGRAMS AND NO MORE THAN FIVE PROGRAMS MODULES PER CATEGORY.

' THE NAMES YOU GIVE TO THESE CATEGORIES WILL APPEAR IN THE FIRST

CONTROL MENU PRESENTED. TWO OTHER ITEMS WILL APPEAR (NAMELY: HELP

AND EXIT SYSTEM). A SEPARATE MENU FOR EACH CATEGORY WILL LATER BE

DEFINED.

PLEASE PRESS RETURN KEY IF YOU WISH TO CONTINUE :

EXAMPLE OF FOUR BASIC CATEGORIES

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

INITIAL INSTALLATION

MAINTENANCE

TRANSACTION PROCESSING

INTERROGATION

EXAMPLE OF MODULES WITHIN INITIAL INSTALLATION

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

CREATE VENDOR MASTER

CREATE INVOICE MASTER

APPEND VENDOR MASTER

ECT. ( UP TO FIVE ALLOWED ).

EXIT T

*DETAB EXPLAIN

A & '

WHICH DEVICE YOU PREFER YOUR OUTPUT FILE.

IF ON THE LINE PRINTER PLEASE ENTER LP:

IF ON THE TELETYPE PLEASE ENTER TT:

IF ON DISK PLEASE ENTER WHICH DISK FD0: OR FD1:

FOLLOWED BY ANY NAME YOU LIKE NO MORE THAN SIX

CHARACTERS ENDED BY .RPL
& " "
EXIT T
*DETAB STARS1
A -30/30 S
-30/2 MV ' '*
-78/76 FILL ' '*
-2/2 MV ' '*
CALL EXPLAIN
CALL WRITELINE1
EXIT T
*DETAB STARS2
A -30/30 S
-30/2 MV ' '*
-78/76 FILL ' '*
CALL WRITELINE1
EXIT T
*DETAB TRADE MARK
A CALL STARS1
-80/1 MV ' '*
-1/1 MV ' '*
CALL WRITELINE1
-80/30 S
-30/1 MV ' '*
-1/1 MV ' '*
-78/45 MV ' LOUGHBOROUGH UNIVERSITY OF TECHNOLOGY'
CALL WRITELINE1
-80/1 MV ' '*
-1/1 MV ' '*
-78/40 MV ' DEPARTEMENT OF COMPUTER STUDIES'
CALL WRITELINE1
-30/1 MV ' '*
-1/1 MV ' '*
-78/25 MV ' GENERATED SOFTWARE'
CALL WRITELINE1
-80/1 MV ' '*
-1/1 MV ' '*
-78/40 MV ' BY PERSONALISED INTERFACE GENERATOR ON'
-30/9 MVDATE$ 0
CALL WRITELINE1
-80/1 MV ' '*
-1/1 MV ' '*
CALL WRITELINE1
EXIT T
*DETAB USER DESCRIPTIONS
A CALL APPLICATION NAME
CALL TRADE MARK
CALL CORCOM GEN
CALL PASSWORD CHECK GEN
CALL MAIN MENU ENTRIES
CALL MENU GENERATOR
CALL SUB MENU GEN
EXIT T
*DETAB WRITELINE1
A & -30/30
EDITMOVE *LI *RECORD
169 PRINT GEN. LINE GEN
170 -80/80 S
171 EXIT T
172 *DETAB APPLICATION. NAME
173 A & " "
174 & " "
175 & " PLEASE ENTER NAME OF APPLICATION :";
176 INPUT -200/30
177 & " PLEASE ENTER VALID PASSWORD ( SIX CHARACTERS ) :";
178 INPUT -170/6
179 EXIT T
180 *DETAB CORCOM. GEN
181 I -80/20 MV "*DETAB CHECK"
192 CALL WRITELINE1
193 ' -80/45 MV 'C *CORCOM:2 = 1' Y N'
194 CALL WRITELINE1
195 -80/45 MV 'A *CORCOM:2 MV 1 - X'
196 CALL WRITELINE1
197 -80/45 MV ' GOTO PASSWORD. CHECK - X'
198 CALL WRITELINE1
199 -80/45 MV ' GOTO MAIN. MENU X -'
200 CALL WRITELINE1
201 *DETAB PASSWORD. CHECK. GEN
202 A -80/21 MV "*DETAB PASSWORD. CHECK"
203 CALL WRITELINE1
204 -80/39 MV 'I & "PLEASE ENTER PASSWORD :"'
205 CALL WRITELINE1
206 -80/15 MV ' INPUTSC 4/1'.
207 CALL WRITELINE1
208 -80/15 MV ' INPUTSC 5/1'.
209 CALL WRITELINE1
210 -80/15 MV ' INPUTSC 6/1'.
211 CALL WRITELINE1
212 -80/15 MV ' INPUTSC 7/1'.
213 CALL WRITELINE1
214 -80/15 MV ' INPUTSC 8/1'.
215 CALL WRITELINE1
216 -80/15 MV ' INPUTSC 9/1'.
217 CALL WRITELINE1
218 -80/9 MV 'C 4/6 = "'
219 -71/6 MY -170/6
220 -65/4 MY ""'
221 -64/20 MV ' Y N'
222 CALL WRITELINE1
223 -80/33 MV 'A & " " X X'
224 CALL WRITELINE1
225 -80/33 MV ' & " " X X'
226 CALL WRITELINE1
227 -80/33 MV ' GOTO USER. CHECK X -'
228 CALL WRITELINE1
229 -80/33 MV ' & "WRONG PASSWORD." - X'
230 CALL WRITELINE1
231 -80/33 MV ' & " " - X'
232 CALL WRITELINE1
233 -80/33 MV ' GOTO PASSWORD. CHECK - X'
234 CALL WRITELINE1
235 -80/17 MV "*DETAB USER. CHECK" - 226 CALL WRITELINE1
-80/35 MY 'I & " "'
228 CALL WRITELINE1
229 -80/49 MY ' & "DO YOU KNOW WHICH PROGRAM YOU WANT TO USE."
230 CALL WRITELINE1
231 -80/35 MY ' & "PLEASE ENTER YES OR NO :";
232 CALL WRITELINE1
233 -80/17 MY ' INPUTNP -164/3'
234 CALL WRITELINE1
235 -80/12 MY 'C -164/1 = "'
236 -69/1 MY O'077'
237 -67/1 MY '"
238 -62/6 MY ' Y N'
239 CALL WRITELINE1
240 -80/25 MY 'A GOTO MAIN.MENU X X'
241 CALL WRITELINE1
242 EXIT T
243 *DETAILS MAIN.MENU. ENTRIES
244 I & '
245 & " 
246 & " DETAILS OF MAIN CONTROL MENU NOW REQUESTED, EACH CATEGORY
247 & " NO MORE 20 CHARACTERS."
248 & " 
249 & " ENTER FIRST CATEGORY IN MENU :'
250 INPUTNP 0/20
251 & " ENTER NEXT CATEGORY IN MENU :'
252 INPUTNP 20/20
253 & " ENTER NEXT CATEGORY IN MENU :'
254 INPUTNP 40/20
255 & " ENTER NEXT CATEGORY IN MENU :'
256 INPUTNP 60/20
257 & " ENTER NEXT CATEGORY IN MENU :'
258 INPUTNP 80/20
259 & " ENTER LAST CATEGORY IN MENU :'
260 INPUTNP 100/20
261 EXIT T
262 *DETAIL SUB. MENU. GEN
263 I & '
264 & " 
265 & " ENTER NUMBER OF SUBMENUS - ;
266 INPUTNP -82:2
267 C -82:2 = ?
268 1 2 3 4 5 6 ELSE
269 A & ' X X X X X X -
270 & ' X X X X X X -
271 & ' *********************** X X X X X X -
272 122:2 MY 1 X X X X X X -
273 CALL MENU1.ENTER
274 CALL MENU1.GEN
275 122:2 + 1 X X X X X X -
276 & ' - X X X X X -
277 & ' - X X X X X -
278 & ' ENTER SECOND MENU DETAILS : X X X X X X -
279 & ' *********************** - X X X X X X -
280 CALL MENU1.ENTER
281 CALL MENU1.GEN
282 122:2 + 1 - X X X X X -
283 & ' - - X X X X -
**ENTER THIRD MENU DETAILS**

**ENTER FOURTH MENU DETAILS**

**ENTER FIFTH MENU DETAILS**

**ENTER SIXTH MENU DETAILS**

*NOT MORE THAN SIX SUB/MENU.*

**EXIT T**

**DETAB DUMMY. MENU. GEN**

**DETAB CATEGOR2. MENU**

**DETAB CATEGOR3. MENU**

**DETAB CATEGOR4. MENU**

**DETAB CATEGOR5. MENU**

**DETAB CATEGOR6. MENU**

**DETAB MENU1. ENTRIES**

**DETAB MENU1. GENERATOR**

**EXIT T**
109

339 & " PLEASE ENTER DETAILS OF THE CATEGORY X MENU. ENTER"
340 & " DESCRIPTION OF EACH PROGRAM IN THIS CATEGORY, NO MORE"
341 & " THAN 38 CHARACTERS, AND THEN ENTER THREE DIGITS FOR"
342 & " THE PROGRAM NUMBER."
343 & " "
344 & " ENTER FIRST MODULE DETAILS : "
345 INPUTNP 140/38
346 & " ENTER PROGRAM NUMBER : "
347 INPUTNP -860/3
348 & " ENTER NEXT MODULE DETAILS : "
349 INPUTNP 178/38
350 & " ENTER PROGRAM NUMBER : "
351 INPUTNP -846/3
352 & " ENTER NEXT MODULE DETAILS : "
353 INPUTNP 216/38
354 & " ENTER PROGRAM NUMBER : "
355 INPUTNP -832/3
356 & " ENTER NEXT MODULE DETAILS : "
357 INPUTNP 254/38
358 & " ENTER PROGRAM NUMBER : "
359 INPUTNP -818/3
360 & " ENTER LAST MODULE DETAILS : "
361 INPUTNP 292/38
362 & " ENTER PROGRAM NUMBER : "
363 INPUTNP -804/3
364 EXIT T
365 *DETAB INTERFACE. GENERATOR
366 A CALL PRELUDE
367 CALL MENU. GENERATOR
368 CALL MENU1. GENERATOR
369 EXIT T
370 *DETAB PRELUDE
371 A CALL OPTLINE
372 CALL BANNER. GEN
373 EXIT T
374 *DETAB OPTLINE
375 A -80/9 S
376 L MV 9
377 -80/9 MV "*OPT P,PE"
378 CALL WRITELINE
379 EXIT T
380 *DETAB WRITELINE
381 A & -80/80
382 EDITMOVE *LI *RECORD
383 PRINT GEN. LINE GEN
384 -80/80 S
385 EXIT T
386 *DETAB BANNER. GEN
387 A CALL STARS. GEN
388 CALL SPACELINE. GEN
389 CALL TITLELINE1. GEN
390 CALL TITLELINE2. GEN
391 CALL DATELINE. GEN
392 CALL SPACELINE. GEN
393 CALL SPACELINE. GEN
394 CALL STARS. GEN
395 EXIT T
110

& " PLEASE ENTER DETAILS OF THE CATEGORY X MENU. ENTER"
& " DESCRIPTION OF EACH PROGRAM IN THIS CATEGORY, NO MORE"
& " THAN 38 CHARACTERS, AND THEN ENTER THREE DIGITS FOR"
& " THE PROGRAM NUMBER."

& " ENTER FIRST MODULE DETAILS : "
INPUTP 140/38
& " ENTER PROGRAM NUMBER : "
INPUTP -860/3
& " ENTER NEXT MODULE DETAILS : "
INPUTP 178/38
& " ENTER PROGRAM NUMBER : "
INPUTP -846/3
& " ENTER NEXT MODULE DETAILS : "
INPUTP 216/38
& " ENTER PROGRAM NUMBER : "
INPUTP -832/3
& " ENTER NEXT MODULE DETAILS : "
INPUTP 254/38
& " ENTER PROGRAM NUMBER : "
INPUTP -818/3
& " ENTER LAST MODULE DETAILS : "
INPUTP 292/38
& " ENTER PROGRAM NUMBER : "
INPUTP -804/3
EXIT T

*DETAB INTERFACE. GENERATOR
A CALL PRELUDE
CALL MENU. GENERATOR
CALL MENU1. GENERATOR
EXIT T

*DETAB PRELUDE
A CALL OPTLINE
CALL BANNER. GEN
EXIT T

*DETAB OPTLINE
A -80/9 S
L MY 9
-80/9 MY "OPT P, PE"
CALL WRITELINE
EXIT T

*DETAB WRITELINE
A & -80/80
EDITMOVE *LI *RECORD
PRINT GEN. LINE GEN
-80/80 S
EXIT T

*DETAB BANNER. GEN
A CALL STARS. GEN
CALL SPACELINE. GEN
CALL TITELINE1. GEN
CALL TITELINE2. GEN
CALL DATELINE. GEN
CALL SPACELINE. GEN
CALL SPACELINE. GEN
CALL STARS. GEN
EXIT T
396 *DETAB TITLELINE1. GEN
397 A -80/80 S
398 -80/1 MV '*'
399 -1/1 MV '*'
400 L MV 80
401 -60/30 MV -200/30
402 CALL WRITELINE
403 EXIT T
404 *DETAB STARS. GEN
405 A -80/80 S
406 -80/30 FILL '*'
407 -79/1 S
408 L MV 80
409 CALL WRITELINE
410 EXIT T
411 *DETAB SPACELINE. GEN
412 A -80/80 S
413 -80/1 MV '*'
414 -1/1 MV '*'
415 L MV 80
416 CALL WRITELINE
417 EXIT T
418 *DETAB TITLELINE2. GEN
419 A -80/80 S
420 -80/1 MV '*'
421 -1/1 MV '*'
422 L MV 80
423 -60/30 MV 'INTERFACE GENERATED BY PIG'
424 CALL WRITELINE
425 EXIT T
426 *DETAB DATELINE. GEN
427 A CALL LINESETUP
428 -60/3 MV 'ON'
429 -57/9 MV -172/9
430 CALL WRITELINE
431 EXIT T
432 *DETAB LINESETUP
433 A -80/80 S
434 -80/1 MV '*'
435 -1/1 MV '*'
436 L MV 80
437 EXIT T
438 *DETAB MENU. GENERATOR
439 A CALL MENU. DETAB. HEAD
440 CALL MENU. DETAB. BODY
441 EXIT T
442 *DETAB MENU. DETAB. HEAD
443 A -80/80 S
444 -80/16 MV '*DETAB MAIN. MENU'
445 CALL WRITELINE
446 EXIT T
447 *DETAB MENU. DETAB. BODY
448 A -80/80 S
449 -80/55 MV 'A &'
450 CALL WRITELINE
451 -80/90 S
452 -80/55 MV ' &'
453 CALL WRITELINE
CALL DISPLAY. SPACE
CALL DISPLAY. BIT
-80/5 MV ' 4 "'
-75/4 MV ' 1 - ''
-71/20 MV 0/20
CALL WRITELINE
CALL DISPLAY. SPACE
CALL DISPLAY. BIT
-75/4 MV ' 2 - ''
-71/20 MV 20/20
CALL WRITELINE
CALL DISPLAY. SPACE
CALL DISPLAY. BIT
-75/4 MV ' 3 - ''
-71/20 MV 40/20
CALL WRITELINE
CALL DISPLAY. SPACE
CALL DISPLAY. BIT
-75/4 MV ' 4 - ''
-71/20 MV 60/20
CALL WRITELINE
CALL DISPLAY. SPACE
CALL DISPLAY. BIT
-75/4 MV ' 5 - ''
-71/20 MV 80/20
CALL WRITELINE
CALL DISPLAY. SPACE
CALL DISPLAY. BIT
-75/4 MV ' 6 - ''
-71/20 MV 100/20
CALL WRITELINE
CALL DISPLAY. SPACE
CALL DISPLAY. BIT
-75/9 MV ' 7 - HELP. '
CALL WRITELINE2
-75/29 MV ' 8 - EXIT SYSTEM. '
CALL WRITELINE2
-80/5 MV ' 9 - 'HELP. '
CALL WRITELINE3
-80/20 MV ' INPUTNP 0:1 '
CALL WRITELINE3
-80/19 MV ' GOTO GET.FUNCTION'
CALL WRITELINE3
-80/19 MV ' *DETAB GET.FUNCTION'
CALL WRITELINE3
-80/8 MV ' C 0:1 = ';
-71/1 MV 0:077''
-54/38 MV ' 1 2 3 4 5 6 7 8 ELSE'
CALL WRITELINE3
-80/15 MV ' A GOTO CATEGORY'
-65/1 MV 0:077''
-64/5 MV ' MENU'
-55/38 MV ' 1 2 3 4 5 6 - - '
CALL WRITELINE3
-80/32 MV ' & "FUNCTION NOT ON THE MENU!"'
-48/27 MV ' - - - - - - - X'
CALL WRITELINE3
-80/20 MV ' GOTO MAIN MENU'
-48/27 MV ' - - - - - - X'
CALL WRITELINE3
-80/25 MV ' CHAIN "REHELP"'
-48/27 MV ' - - - - - X'
CALL WRITELINE3
-80/20 MV ' & "'
-48/27 MV ' - - - - - X'
CALL WRITELINE3
-80/20 MV ' DELETE "END OF APPLICATION"'
-48/27 MV ' - - - - - X'
CALL WRITELINE3
EXIT T
*DATAB DISPLAY. BIT
-80/5 MV ' & ""
-30/1 MV '"
EXIT T
*DATAB DISPLAY. SPACE
A CALL DISPLAY. BIT
CALL WRITELINE3
EXIT T
*DATAB WRITELINE2
A CALL DISPLAY. SPACE
CALL DISPLAY. BIT
EDITMOVE LI *RECORD
PRINT GEN. LINE GEN
-80/30 S
EXIT T
*DATAB MENU1. GENERATOR
A CALL MENU1. DATAB. HEAD
CALL MENU1. DATAB. BODY
EXIT T
*DATAB WRITELINE3
A & -80/30
EDITMOVE *LI *RECORD
PRINT GEN. LINE GEN
-80/30 S
EXIT T
*DATAB MENU1. DATAB. HEAD
A -80/30 S
-65/1 MV 122:2
-80/15 MV "*DATAB CATEGORY"
-64/5 MV ' MENU'
CALL WRITELINE3
EXIT T
*DATAB MENU1. DATAB. BODY
A -80/80 S
-80/60 MV 'I & "
CALL WRITELINE3
CALL WRITELINE2
-75/5 MV '11 - '
-70/38 MV '140/38
CALL WRITELINE2
CALL ENTRY.CHECK

CALL WRITELINE3
-30/16 MV 'GOTO MAIN.MENU'
-55/38 MV '-' - - - - - X - -'
CALL WRITELINE3
CALL DISPLAY.BIT
-75/28 MV 'FUNCTION NOT ON THE MENU!''
-47/38 MV '- - - - - - - - - - X - -'
CALL WRITELINE3
-80/15 MV 'GOTO CATEGORY'
-65/1 MV 122:2
-64/5 MV '.MENU'
-55/38 MV '- - - - - - - - - - X X'
CALL WRITELINE3
EXIT T

*DETAB ENTRY.CHECK
C -?/3 = '47 43 39 25 31 27 23
A -?/3 MV '- '47 43 39 35 31 27 23
EXIT T

*DETAB POST.SCRIPT
A -80/80 S
-80/5 MV '*STOP'
-40/21 MV 'C COPYRIGHT LUT 1979'
CALL WRITELINE3
EXIT T
EXIT T

*DETAB ENDOFJOB
A & '
&'
&'PLEASE WAIT HERE BECAUSE THE GENERATION PROGRAM IS CHAINED'
&'TO OTHER PROGRAM CALLED DBTEST TO INITIALISE THE HELP'
&'FILE AND IT WILL BE 90 BLOCKS AND READY TO USE.'
&'
CHAIN 'DBTEST'
*STOP

^ERRORS DETECTED: 0
RPL>
APPENDIX A3

SAMPLE USER DIALOGUES
APPENDIX A3

(i) GENERATOR DIALOGUE
NOTE:

IF MISTAKES ARE MADE IN ENTERING DATA, PLEASE MAKE A NOTE AND CONTINUE WITH THE GENERATION PROCESS. CORRECTION TO THE INTERFACE GENERATED MAY BE MADE AT THE LATEST STAGE THROUGH USING THE EDITOR.

YOUR APPLICATION PACKAGE IS ASSUMED TO HAVE SIX BASIC CATEGORIES OF PROGRAMS AND NO MORE THAN FIVE PROGRAMS MODULES PER CATEGORY. THE NAMES YOU GIVE TO THESE CATEGORIES WILL APPEAR IN THE FIRST CONTROL MENU PRESENTED. TWO OTHER ITEMS WILL APPEAR (NAMELY: HELP AND EXIT SYSTEM). A SEPARATE MENU FOR EACH CATEGORY WILL LATER BE DEFINED.

PLEASE PRESS RETURN KEY IF YOU WISH TO CONTINUE:

EXAMPLE OF FOUR BASIC CATEGORIES

INITIAL INSTALLATION
MAINTENANCE
TRANSACTION PROCESSING
INTERROGATION

EXAMPLE OF MODULES WITHIN INITIAL INSTALLATION -

CREATE VENDOR MASTER
CREATE INVOICE MASTER
APPEND VENDOR MASTER
ECT. (UP TO FIVE ALLOWED).

PLEASE ENTER NAME OF APPLICATION :? LCS
PLEASE ENTER VALID PASSWORD (SIX CHARACTERS) :? OSSAMA

WHICH DEVICE YOU PREFER YOUR OUTPUT FILE:
IF ON THE LINE PRINTER PLEASE ENTER LP:
IF ON THE TECHTYPE PLEASE ENTER TT:
IF ON DISK PLEASE ENTER WHICH DISK FD0: OR FD1:
FOLLOWED BY ANY NAME YOU LIKE NO MORE THAN SIX CHARACTERS ENDED BY .RPL

WHICH DEVICE :? FD0:LCS.RPL
DETAILS OF MAIN CONTROL MENU NOW REQUESTED. EACH CATEGORY NO MORE 20 CHARACTERS.

ENTER FIRST CATEGORY IN MENU : LANGUAGE SYSTEM.
ENTER NEXT CATEGORY IN MENU : FILE INITIALISATION.
ENTER NEXT CATEGORY IN MENU : FILE MANIPULATION.
ENTER NEXT CATEGORY IN MENU : SYSTEM FILES.
ENTER NEXT CATEGORY IN MENU : LIBRARY FILES.
ENTER LAST CATEGORY IN MENU : OTHERS.

ENTER NUMBER OF SUBMENUS - 6

ENTER FIRST MENU DETAILS:
********************************************************************************

PLEASE ENTER DETAILS OF THE CATEGORY X MENU. ENTER DESCRIPTION OF EACH PROGRAM IN THIS CATEGORY, NO MORE THAN 38 CHARACTERS, AND THEN ENTER THREE DIGITS FOR THE PROGRAM NUMBER.

ENTER FIRST MODULE DETAILS: CREATE A SOURCE RPL FILE.
ENTER PROGRAM NUMBER: 001
ENTER NEXT MODULE DETAILS: FORMATTED SOURCE FILE.
ENTER PROGRAM NUMBER: 002
ENTER NEXT MODULE DETAILS: LISTS *DICT WORKING OUT START POST.
ENTER PROGRAM NUMBER: 003
ENTER NEXT MODULE DETAILS: RENAME PAGES ON LIBRARY PAGE FILE.
ENTER PROGRAM NUMBER: 004
ENTER LAST MODULE DETAILS: CHECK LIBRARY.
ENTER PROGRAM NUMBER: 005

ENTER SECOND MENU DETAILS:
********************************************************************************

PLEASE ENTER DETAILS OF THE CATEGORY X MENU. ENTER DESCRIPTION OF EACH PROGRAM IN THIS CATEGORY, NO MORE THAN 38 CHARACTERS, AND THEN ENTER THREE DIGITS FOR THE PROGRAM NUMBER.

ENTER FIRST MODULE DETAILS: CREATE A SOURCE FILE NOT ONLY IN RPL.
ENTER PROGRAM NUMBER: 006
ENTER NEXT MODULE DETAILS: DATABASE UTILITY PROGRAM.
ENTER PROGRAM NUMBER: 007
ENTER NEXT MODULE DETAILS: SORT FILE INITIALISER.
ENTER PROGRAM NUMBER: 008
ENTER NEXT MODULE DETAILS: PRINT SORT FILE HEADER.
ENTER PROGRAM NUMBER: 009
ENTER LAST MODULE DETAILS: DATABASE UTIL. PROG. (ALPH. NUMERIC).
ENTER PROGRAM NUMBER: 031
ENTER THIRD MENU DETAILS:

PLEASE ENTER DETAILS OF THE CATEGORY X MENU. ENTER DESCRIPTION OF EACH PROGRAM IN THIS CATEGORY, NO MORE THAN 38 CHARACTERS, AND THEN ENTER THREE DIGITS FOR THE PROGRAM NUMBER.

ENTER FIRST MODULE DETAILS: REPLACE SPACES BY TABS IN ASCII FILE.
ENTER PROGRAM NUMBER: 010
ENTER NEXT MODULE DETAILS: COPY A FILE.
ENTER PROGRAM NUMBER: 011
ENTER NEXT MODULE DETAILS: RENAME A FILE.
ENTER PROGRAM NUMBER: 022
ENTER NEXT MODULE DETAILS: KILL A FILE.
ENTER PROGRAM NUMBER: 023
ENTER LAST MODULE DETAILS: SORT A FILE.
ENTER PROGRAM NUMBER: 024

ENTER FOURTH MENU DETAILS:

PLEASE ENTER DETAILS OF THE CATEGORY X MENU. ENTER DESCRIPTION OF EACH PROGRAM IN THIS CATEGORY, NO MORE THAN 38 CHARACTERS, AND THEN ENTER THREE DIGITS FOR THE PROGRAM NUMBER.

ENTER FIRST MODULE DETAILS: PRINTS CONTROL BLOCK OF A FILE.
ENTER PROGRAM NUMBER: 012
ENTER NEXT MODULE DETAILS: CREATES IDENTITY LABEL ON USER DISK.
ENTER PROGRAM NUMBER: 015
ENTER NEXT MODULE DETAILS: SET DATE AND GET TODAYS DATE.
ENTER PROGRAM NUMBER: 016
ENTER NEXT MODULE DETAILS: LISTS LOG OF USER.
ENTER PROGRAM NUMBER: 017
ENTER LAST MODULE DETAILS: TEST SCALE FACTOR IN COMPILER.
ENTER PROGRAM NUMBER: 032

ENTER FIFTH MENU DETAILS:
PLEASE ENTER DETAILS OF THE CATEGORY X MENU. ENTER DESCRIPTION OF EACH PROGRAM IN THIS CATEGORY, NO MORE THAN 38 CHARACTERS, AND THEN ENTER THREE DIGITS FOR THE PROGRAM NUMBER.

ENTER FIRST MODULE DETAILS: CREATE AND MAINTAIN DISK LIBRARY.
ENTER PROGRAM NUMBER: 013
ENTER NEXT MODULE DETAILS: LISTS DISKETTE LIBRARY USAGE.
ENTER PROGRAM NUMBER: 014
ENTER NEXT MODULE DETAILS: SYSTEM SOFTWARE UTILS. LIB. ADDITIONS.
ENTER PROGRAM NUMBER: 018
ENTER NEXT MODULE DETAILS: LISTS SYSTEM PROGRAM LIBRARY.
ENTER PROGRAM NUMBER: 019
ENTER LAST MODULE DETAILS: MAINTAIN LIBRARY CONTROL FILE.
ENTER PROGRAM NUMBER: 021

ENTER SIXTH MENU DETAILS:

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

PLEASE ENTER DETAILS OF THE CATEGORY X MENU. ENTER DESCRIPTION OF EACH PROGRAM IN THIS CATEGORY, NO MORE THAN 38 CHARACTERS, AND THEN ENTER THREE DIGITS FOR THE PROGRAM NUMBER.

ENTER FIRST MODULE DETAILS: RUN A PROGRAM.
ENTER PROGRAM NUMBER: 027
ENTER NEXT MODULE DETAILS: ALLOWS PATCHES TO FILES IN ASC/DEC.
ENTER PROGRAM NUMBER: 028
ENTER NEXT MODULE DETAILS: PRINTS ASCII FILES ON TT: HANDLING PP.
ENTER PROGRAM NUMBER: 029
ENTER NEXT MODULE DETAILS: LISTS ASCII FILES ON TT: HANDLING FED.
ENTER PROGRAM NUMBER: 030
ENTER LAST MODULE DETAILS: INTERROGATE LIBRARY CONTROL FILE.
ENTER PROGRAM NUMBER: 020
END OF GENERATION PROGRAM
APPENDIX A3

(ii) INTERFACE DIALOGUE (COMPLETE SYSTEM TEST)
R R
RPL > LCS
PLEASE ENTER PASSWORD :

DO YOU KNOW WHICH PROGRAM YOU WANT TO USE?
PLEASE ENTER YES OR NO : NO

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 1

11 - CREATE A SOURCE RPL FILE.
12 - FORMATTED SOURCE FILE.
13 - LISTS *DICT WORKING OUT START POST.
14 - RENAME PAGES ON LIBRARY PAGE FILE.
15 - CHECK LIBRARY.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 11

ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N : Y
PROGRAM NAME? CREATE.RPL
+0FILE 3 A 'PROGRAM NAME'.RPL
+DI
  L -52;2, LINE 0/(L)
+DETAB ONE
OPEN 3
GOTO TWO
+DETAB TWO
I L MV 240
INPUTNP LINE
C 0/3 = '*?*
GO ST ** ELSE
A WRITE 3 LINE X X - X
GOTO TWO - - X
DELETE "END OF RUN"
+STOP
END OF RUN

1 - LANGUAGE SYSTEM.
2 - FILE INITIALIZATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 2

11 - CREATE A SOURCE FILE NOT ONLY IN RPL.
12 - DATABASE UTILITY PROGRAM.
13 - SORT FILE INITIALIZER.
14 - PRINT SORT FILE HEADER.
15 - DATABASE UTILITY PROG. (ALPH. NUMERIC).
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 11
ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y

PLEASE ENTER ** IF YOU WISH TO TERMINATE THE INPUT:
**TSF.RPL - TEST SCALE FACTOR IN COMPILER FILENAME: EXT? TSF.RPL**

? *DETAB ONE
? I  O:8 MY 1234.56789
? PRINT P L
? *LI L
? O  O:3
? *PI P
?
? *DETAB ENDOFJOB
? A DELETE 'OK'
? *DO
? **

OK

1 - LANGUAGE SYSTEM.

2 - FILE INITIALISATION.

3 - FILE MANIPULATION.

4 - SYSTEM FILES.

5 - LIBRARY FILES.

6 - OTHERS.

7 - HELP.

8 - EXIT SYSTEM.

Please enter number of function required: 1

11 - CREATE A SOURCE RPL FILE.

12 - FORMATTED SOURCE FILE.

13 - LISTS *DICT WORKING OUT START POST.

14 - RENAME PAGES ON LIBRARY PAGE FILE.

15 - CHECK LIBRARY.

16 - RETURN TO THE MAIN MENU.

Please enter number of function required: 12
ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N: Y
PLEASE ENTER APPLICATION NAME: ? LCS
PLEASE ENTER PROGRAM NAME: ? CREATE RPL
PLEASE ENTER PROGRAMMER NAME: ? OSSAMA K. MUSLIH
PLEASE ENTER COMPLETION DATE: ? 24-JUL-79
OUTPUT RPL NAME? CREATE RPL
FUNCTION OF THE PROGRAM: (PLEASE ENTER FIFTY CHARACTER
PER LINE BUT TERMINATE BY FULL STOP ON LINE OF IT IS OWN.)

THIS PROGRAM ALLOWS THE USER TO CREATE ANY FILE CONTAINING
VARIABLE ASCII CHARACTERS. THE USER RESPONDS TO TERMINAL
REQUESTS, AND ENTER THE DATA VIA THE KEYBOARD. "***" OR
*STOP OR *GO TERMINATE THE INPUT.

SOURCE RPL NAME?: CREATE RPL
PROGRAM READY.

1 - LANGUAGE SYSTEM.
2 - FILE INITIAlISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED: 3

11 - REPLACE SPACES BY TABS IN ASCII FILE.
12 - COPY A FILE.
13 - RENAME A FILE.
14 - KILL A FILE.
15 - SORT A FILE.
16 - RETURN TO THE MAIN MENU.
PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 12

ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N : Y

PLEASE ENTER FILE NAME WITH EXTENSION FOR THE INPUT AND OUTPUT FILES.

FROM ? FD1:LCS006.RPL
TO ? TT:
* LOUGHBOURGH UNIVERSITY OF TECHNOLOGY
* COMPUTER STUDIES DEPARTMENT
* THIS MODULE BELONGS TO LCS SYSTEM

* PROGRAM SOURCE FORMATTED BY FORMAT.RPL
* ************************************************************
* APPLICATION NAME - LCS
* PROGRAM NAME - LCS006
* PROGRAMMER NAME - OSSAMA MUSLIH
* COMPLETION DATE - 15-MAY-79
* FUNCTION OF THE PROGRAM:
* THIS PROGRAM ALLOWS THE USER TO CREATE ANY SOURCE FILE
* NOT ONLY IN RPL, THE USER RESPONDS TO TERMINAL REQUESTS
* AND ENTERS THE DATA VIA THE KEYBOARD. ** TERMINATE
* THE INPUT DATA AND A JOURNAL LISTING IS PROVIDED OF THE
* FILE. CONTROL IS THEN RETURNED TO THE MAIN MENU.

*PFILE LP:
*DFILE 3 A 'FILENAME.EXT'
*DICT
L -2; 2
+LIST GENLINE
A 0/80
+PIC DATA 1; 0
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
+DETAB LPLINEUP
A & ' 
& 'PLEASE ENTER ** IF YOU WISH TO TERMINATE THE INPUT. '
& ' 
0/80 S
EDITMOVE *LIST *RECORD
PRINT DATA GENLINE
0/80 MV ' ASCII FILE LOAD DATA JOURNAL'
70/9 MVDATE= 0
EDITMOVE *LIST *RECORD
PRINT DATA GENLINE
GOTO INPUT. DATA
+DETAB INPUT. DATA
I Q/'512 S
 INPUT Q/(L)
C 0.2 = '***' Y N
A CLOSE 3
X
.. DELETE 'OK'
X
EDITMOVY *LIST +RECORD
PRINT DATA GENLINE
WRITE 3 0/(L)
L MV 512
GOTO INPUT DATA
+DETB ENOFJOB.
A *CORCON.2 NV 1
CHAIN 'LCS'
*STOP
FILE COPIED

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 2

11 - CREATE A SOURCE FILE NOT ONLY IN RPL.
12 - DATABASE UTILITY PROGRAM.
13 - SORT FILE INITIALISER.
14 - PRINT SORT FILE HEADER.
15 - DATABASE UTIL. PROG. (ALPH. NUMERIC).
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 12

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
ACTION - I.C.E.EI.CI.R.S.HELP? I
FILENAME? DBASE.DAT
+LOWKEY? 1
HIKEY? 10
FILESIZE? 1
END OF RUN

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED: 1

11 - CREATE A SOURCE RPL FILE.
12 - FORMATTED SOURCE FILE.
13 - LISTS *DICT WORKING OUT START POST.
14 - RENAME PAGES ON LIBRARY PAGE FILE.
15 - CHECK LIBRARY.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED: 13

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
DICTIOARY NAME? GENDIC. TIO
0 *
0 *DICTIOARY
0 NAME 0/20
0 DEPT 20/4
0 PAYCODE 24/1
1 NATIN 25/9
10 PAYRATE 34/2
12 TAXCODE 36/1
13 TAXB 37/1
14  TOTPAY 38:4
18  TOTTAX 42:4
22  SUPAN 46:4
26  EMPNO 50:8
34  WEEKNO 100:1
35  OPTION 120/3
38  EDOPT 126/2
40  *

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 4

11 - PRINTS CONTROL BLOCK OF A FILE.
12 - CREATES IDENTITY LABEL ON USER DISK.
13 - SET DATE AND GET TODAY'S DATE.
14 - LISTS LOG OF USER.
15 - TEST SCALE FACTOR IN COMPILER.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 11

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
FILE NAME? CREATE.RPL
26-NOV-79 24:00  HEADER BLOCK FOR FILE:/-MAT.RPL
KL = 32  LOK = 9224  HIB = 19488  KLP3= 30063
TLI = 26727  EXB = 28514  LEP = 28530
1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED: 1

11 - CREATE A SOURCE RPL FILE.
12 - FORMATTED SOURCE FILE.
13 - LISTS *DICT WORKING OUT START POST.
14 - RENAME PAGES ON LIBRARY PAGE FILE.
15 - CHECK LIBRARY.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED: 15

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
#RPLERR1 1 2
#RPLERR2 1 4
**SORT 4 6
#RPLSYS2 4 8
#2 0 10
#RPLSYS4 4 12
#RPLSYS5 4 14
#RFLSYS6 4 16
ENDOFJOB 4 18
*CURSOR 4 20
*SORT 4 22
SQUASH 4 24
UNSQUASH 4 26
INVALID. STOCK. CODE 3 28
ECINPUT 4 30
END OF RUN
1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 6

11 - RUN A PROGRAM.
12 - ALLOWS PATCHES TO FILES IN ASC/DEC.
13 - PRINTS ASCII FILES ON TT: HANDLING PP.
14 - LISTS ASCII FILES ON TT: HANDLING FED.
15 - INTERROGATE LIBRARY CONTROL FILE.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 11

ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N : Y
PLEASE ENTER DEVICE:FILENAME CONTAINING THE PROGRAM
NAME REQUIRED TO RUN WITH OR WITHOUT EXTENTION : ? FD1:LCS002
PLEASE ENTER APPLICATION NAME : ? LCS
PLEASE ENTER PROGRAM NAME : ? TSF.RPL
PLEASE ENTER PROGRAMMER NAME : ? OSSAMA K. MUSLIH
PLEASE ENTER COMPLETION DATE : ? 20-JUN-79
OUTPUT RPL NAME? TSF.RPL
FUNCTION OF THE PROGRAM : (PLEASE ENTER FIFTY CHARACTER
PER LINE BUT TERMINATE BY FULL STOP ON LINE OF IT IS OWN.)

THIS PROGRAM ALLOWS THE USER TO TEST THE SCALE FACTOR IN THE
COMPILER. THE USER RESPONDS TO TERMINAL REQUESTS AND ENTER
THE DATA VIA THE KEYBOARD.

SOURCE RPL NAME? TSF.RPL
PROGRAM READY.
1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 2

11 - CREATE A SOURCE FILE NOT ONLY IN RPL.
12 - DATABASE UTILITY PROGRAM.
13 - SORT FILE INITIALISER.
14 - PRINT SORT FILE HEADER.
15 - DATABASE UTIL. PROG. (ALPH. NUMERIC).
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 13

ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N : Y
FILENAME? BASEDA.DAT
RECORD SIZE, FILE SIZE
? 5.1
FILE INITIALISED

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.
PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 1

11 - CREATE A SOURCE RPL FILE.
12 - FORMATTED SOURCE FILE.
13 - LISTS *DICT WORKING OUT START POST.
14 - RENAME PAGES ON LIBRARY PAGE FILE.
15 - CHECK LIBRARY.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 14

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
OLD NAME, OLD TYPE? INVALID. STOCK CODE, 3
NEW NAME, NEW TYPE? VALID. STOCK CODE, 4
PAGE RENAMED
END OF RUN

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 4

11 - PRINTS CONTROL BLOCK OF A FILE.
12 - CREATES IDENTITY LABEL ON USER DISK.
13 - SET DATE AND GET TODAY'S DATE.
14 - LISTS LOG OF USER.
15 - TEST SCALE FACTOR IN COMPILER.
16 - RETURN TO THE MAIN MENU.
PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 12

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y

NAME: COURSE: OLABANA K. NULIH. 11A
YEAR: ACCOUNT NO: 2.5234
DATEISSUED, DISKETTE NO: 13-AUG-78, 200

1 - LANGUAGE SYSTEM.
2 - FILE INITIALIZATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 5

11 - CREATE AND MAINTAIN DISK. LIBRARY.
12 - LISTS DISKETTE LIBRARY USAGE.
13 - SYSTEM SOFTWARE UTILS. LIB. ADDITIONS.
14 - LISTS SYSTEM PROGRAM LIBRARY.
15 - MAINTAIN LIBRARY CONTROL FILE.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 11

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y

DISKETTE NUMBER
? 12
USER NUMBER, USER NAME
? 5234, JOHN OLDSHAM
COURSE, YEAR
? DP. 2
DATE ISSUED, DATE OF EXPIRY
? 25-SEP-79, 24-NOV-79
USAGE, ISSUING STAFF
? DEVELOPING, LNB
RECORD LOADED
DISKETTE NUMBER
1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 4

11 - PRINTS CONTROL BLOCK OF A FILE.
12 - CREATES IDENTITY LABEL ON USER DISK.
13 - SET DATE AND GET TODAY'S DATE.
14 - LISTS LOG OF USER.
15 - TEST SCALE FACTOR IN COMPILER.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 13

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
PLEASE INPUT TODAY'S DATE IN THE FORM DDMMYY
TODAY'S DATE? 26-NOV-79
END OF RUN

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.
PLEASE ENTER NUMBER OF FUNCTION REQUIRED: 2

11 - CREATE A SOURCE FILE NOT ONLY IN RPL.
12 - DATABASE UTILITY PROGRAM.
13 - SORT FILE INITIALLSER.
14 - PRINT SORT FILE HEADER.
15 - DATABASE UTIL. PROG. (ALPH. NUMERIC).
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED: 14

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N :Y
FILE NAME? CREATE RPL
BLOCK NUMBER OF LAST BLOCK...... 10784
NUMBER OF RECORDS IN LAST BLOCK: 8224
RECORD SIZE.............................. 8268
NUMBER OF RECORDS IN FILE....... 8224
END OF RUN

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED: 5

11 - CREATE AND MAINTAIN DISK. LIBRARY.
12 - LISTS DISKETTE LIBRARY USAGE.
13 - SYSTEM SOFTWARE UTILS. LIB. ADDITIONS.
14 - LISTS SYSTEM PROGRAM LIBRARY.
15 - MAINTAIN LIBRARY CONTROL FILE.
16 - RETURN TO THE MAIN MENU.
PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 12

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y

USER NAME ORS Y INITIAL ISSUED EXPIRY USAGE

C S345L NB D F5
2 C S444M AGED 2 03
2 C S224J HON OLCHAM D.F2
3 C S300L NB 1 34
2 C S1230 SSAMA M P-

1 - LANGUAGE SYSTEM.
2 - FILE INITIALIZATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 4

11 - PRINTS CONTROL BLOCK OF A FILE.
12 - CREATES IDENTITY LABEL ON USER DISK.
13 - SET DATE AND GET TODAY'S DATE.
14 - LISTS LOG OF USER.
15 - TEST SCALE FACTOR IN COMPILER.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 14

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y

NO DATA SELECTED
1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 3

11 - REPLACE SPACES BY TABS IN ASCII FILE.
12 - COPY A FILE.
13 - RENAME A FILE.
14 - KILL A FILE.
15 - SORT A FILE.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 11

ARE YOU SURE YOU WISH THIS PROGRAM: PLEASE ENTER Y OR N : Y
INPUT FILE NAME : ? CREATE.RPL
OUTPUT FILE NAME : ? REPTAB.RPL
END OF RUN

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 6

11 - RUN A PROGRAM.
12 - ALLows PATCHES TO FILES IN ASC/DEC.
13 - PRINTS ASCII FILES ON TT: HANDLING PP.
14 - LISTS ASCII FILES ON TT: HANDLING FEO.
15 - INTERROGATE LIBRARY CONTROL FILE.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 13

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
INPUT FILE? FDL: LC5005.RPL
*FILE ? TT:
* LOUGHBOROUGH UNIVERSITY OF TECHNOLOGY
* COMPUTER STUDIES DEPARTMENT
* THIS MODULE BELONGS TO LCS SYSTEM
* PROGRAM SOURCE FORMATTED BY FORMAT.RPL
* *******************************
* APPLICATION NAME - LCS
* PROGRAM NAME - LC5005
* PROGRAMMER NAME - OSSAMA MUSLIM
* COMPLETION DATE - 15-MAY-73
* FUNCTION OF THE PROGRAM :
* THIS PROGRAM ALLOWS THE USER TO CHECK THE LIBRARY FILE
* WHICH IS STORED IN ARPL.LPF. CONTROL IS THEN RETURNED
* TO THE MAIN MENU.
*IFILE 3 R ARPL.LPF
* +DICT
L /512. 8 ;2, P ;2, V /34
* +DETAB ONE
I E MV 8
   GOTO TWO
* +DETAB TWO
I GET 3 L B
   P MV A' L
C P = A' L+510   N Y
   (P) ;2 = 3   N N
   A V MV (F) ;14+   X
1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 4

11 - PRINTS CONTROL BLOCK OF A FILE.
12 - CREATES IDENTITY LABEL ON USER DISK.
13 - SET DATE AND GET TODAY'S DATE.
14 - LISTS LOG OF USER.
15 - TEST SCALE FACTOR IN COMPILER.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 15

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
1234. 567890
END OF RUN
1 - LANGUAGE SYSTEM.
2 - FILE INITI ALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 6

11 - RUN A PROGRAM.
12 - ALLOWS PATCHES TO FILES IN ASCII/DEC.
13 - PRINTS ASCII FILES ON TT: HANDLING PP.
14 - LISTS ASCII FILES ON TT: HANDLING FED.
15 - INTERROGATE LIBRARY CONTROL FILE.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 14

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
INPUT FILE? FD1: LCS028.RPL
*FILE ? TT:
* LOUGHBOROUGH UNIVERSITY OF TECHNOLOGY
* COMPUTER STUDIES DEPARTMENT
* THIS MODULE BELONGS TO LCS SYSTEM
* PROGRAM SOURCE FORMATTED BY FORMAT RPL
* *****************************************************
* APPLICATION NAME - LCS
* PROGRAM NAME - LCS028
* PROGRAMMER NAME - OSSAMA MUSLIH
* COMPLETION DATE - 15-MAY-79
* FUNCTION OF THE PROGRAM :
MORE TO LIST? N

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 5

11 - CREATE AND MAINTAIN DISK LIBRARY.
12 - LISTS DISKETTE LIBRARY USAGE.
12 - SYSTEM SOFTWARE UTILS. LIB. ADDITIONS.
14 - LISTS SYSTEM PROGRAM LIBRARY.
15 - MAINTAIN LIBRARY CONTROL FILE.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 14

ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N : Y  
NO DATA SELECTED

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 6

11 - RUN A PROGRAM.
12 - ALLOWS PATCHES TO FILES IN ASC/DEC.
13 - PRINTS ASCII FILES ON TT: HANDLING PP.
14 - LISTS ASCII FILES ON TT: HANDLING FED.
15 - INTERROGATE LIBRARY CONTROL FILE.
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 12

ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N : Y  
FILE NAME? CREATE RPL
BLOCK NO, BYTE POSN? 2, 6
TYPE OF PATCH A OR D? : A  
PRESENT CONTENTS :- I  
ALTER TO ::=? ABCD
END OF RUN
1 - LANGUAGE SYSTEM.
2 - FILE INITIALLISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 7
THE KEYWORDS IN THE HELP FILE ARE :

01 LANGUAGE 02 INITIALISE 03 MANIPULATE 04 SYSTEM
05 PAGE 06 OTHERS 07 CREATE 08 FORMAT
09 LIST 10 RENAME 11 CHECK 12 SOURCE
12 DATABASE 14 PRINTSORT 15 HEADER 16 REPLACE
17 COPY 18 CONTROL 19 IDENTITY 20 DATE
21 LOG 22 DISKETTE 23 USAGE 24 ADDITION
25 LIBRARY 26 RUN 27 PATCHES 28 PAUSES
29 FEEDS 30 UTILITY 31 SCALE

PLEASE ENTER KEYWORD NUMBER REQUIRED : 30

UTILITY
THIS PROGRAM ALLOWS THE USER TO INITIALISE ANY
DATABASE FILE (INDEXED FILE). THE USER RESPONDS
TO TERMINAL REQUESTS AND ENTERS THE DATA VIA THE
KEYCARD I.E : THE FILE NAME, LOW KEY, HIGH KEY,
NUMBER OF DATA BLOCKS, NUMBER OF KEYS, CONTROL
IS THEN RETURNED TO THE MAIN MENU.

PLEASE USE SYSTEM FUNCTION COPY (UNDER FILE MANIPULATION)
TO LIST FILE LCS031.RFL FOR PROGRAM DETAILS.

DO YOU REQUIRE FURTHER INFORMATION. PLEASE ENTER YES OR NO : ? N
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 2

11 - CREATE A SOURCE FILE NOT ONLY IN RPL.
12 - DATABASE UTILITY PROGRAM.
13 - SORT FILE INITIALISER.
14 - PRINT SORT FILE HEADER.
15 - DATABASE UTIL. PROG. (ALPH. NUMERIC).
16 - RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 15

ARE YOU SURE YOU WISH THIS PROGRAM. PLEASE ENTER Y OR N : Y
LOW KEY, HI KEY? A.L
NO OF DATA BLOCKS? 20
NUMBER OF KEYS? 15
FILE NAME? DBASE.DAT
END OF RUN

1 - LANGUAGE SYSTEM.
2 - FILE INITIALISATION.
3 - FILE MANIPULATION.
4 - SYSTEM FILES.
5 - LIBRARY FILES.
6 - OTHERS.
7 - HELP.
8 - EXIT SYSTEM.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 8
END OF APPLICATION
APPENDIX A4

Generated Interface Listing - LCS.RPL
LINE SOURCE STATEMENT

1 *                                                        
2 *                                                        
3 * LOUGHBOURGH UNIVERSITY OF TECHNOLOGY
4 * DEPARTEMENT OF COMPUTER STUDIES
5 * GENERATED SOFTWARE -
6 * BY PERSONALISED INTERFACE GENERATOR ON 09-OCT-79
7 *                                                        
8 *                                                        
9 *DETAB CHECK
10 C *CORCOM:2 = 1 Y N
11 A *CORCOM:2 MV 1 - X
12 GOTO PASSWORD.CHECK - X
13 GOTO MAIN.MENU X -
14 *DETAB PASSWORD.CHECK
15 I & "PLEASE ENTER PASSWORD :"
16 INPUTSC 4/1
17 INPUTSC 5/1
18 INPUTSC 6/1
19 INPUTSC 7/1
20 INPUTSC 8/1
21 INPUTSC 9/1
22 C 4/6 = "OSSAMA" Y N
23 A & " " X X
24 & " " X X
25 GOTO USER.CHECK X -
26 & "WRONG PASSWORD." - X
27 & " " - X
28 GOTO PASSWORD.CHECK - X
29 *DETAB USER.CHECK
30 I & " "
31 & "DO YOU KNOW WHICH PROGRAM YOU WANT TO USE."
32 & "PLEASE ENTER YES OR NO :"
33 INPUTNP -164/3
34 C -164/1 = "?" Y N
35 A GOTO MAIN.MENU X X
36 *DETAB MAIN.MENU
37 A & ""
38 & ""
39 & ""
40 & "1 - LANGUAGE SYSTEM."
41 & ""
42 & "2 - FILE INITIALISATION."
43 & ""
44 & "3 - FILE MANIPULATION."
45 & ""
46 & "4 - SYSTEM FILES."
47 & ""
48 & "5 - LIBRARY FILES."
49 & ""
50 & "6 - OTHERS."

LANGUAGE SYSTEM.
FILE INITIALISATION.
FILE MANIPULATION.
SYSTEM FILES.
LIBRARY FILES.
OTHERS.
& "7 - HELP.
52 & "8 - EXIT SYSTEM.
55 & ".
56 & ".
57 & ".
58 & "PLEASE ENTER NUMBER OF FUNCTION REQUIRED : ";
59 INPUTNP 0:1
60 GOTO GET. FUNCTION
61 *DETAB GET. FUNCTION
62 C 0:1 = ? 1 2 3 4 5 6 7 8 ELSE
63 A GOTO CATEGORY?. MENU 1 2 3 4 5 6 7 8
64 & "FUNCTION NOT ON THE MENU!" - - - - - - - - X
65 GOTO MAIN. MENU - - - - - - - - X
66 CHAIN "REHELP" - - - - - - X -
67 & " " - - - - - - - - X -
68 DELETE "END OF APPLICATION" - - - - - - - - X -
69 *DETAB CATEGORY1. MENU
70 I & "
71 & "
72 & "
73 & "11 - CREATE A SOURCE RPL FILE.
74 & "
75 & "12 - FORMATTED SOURCE FILE.
76 & "
77 & "13 - LISTS *DICT WORKING OUT START POST.
78 & "
79 & "14 - RENAME PAGES ON LIBRARY PAGE FILE.
80 & "
81 & "15 - CHECK LIBRARY.
82 & "
83 & "16 - RETURN TO THE MAIN MENU.
84 & "
85 & "
86 & "
87 & "PLEASE ENTER NUMBER OF FUNCTION REQUIRED : ";
88 INPUTNP 0:1
89 & " "
90 & "ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N : ";
91 INPUTNP -100/1
92 C -100/1 = "?" Y Y Y Y Y Y Y N
93 0:1 = ? 11 12 13 14 15 16 - -
94 A CHAIN "FD1:LCS?" 001 002 003 004 005 - - -
95 GOTO MAIN. MENU - - - - - - X -
96 & "FUNCTION NOT ON THE MENU!" - - - - - - X -
97 GOTO CATEGORY1. MENU - - - - - - X X
98 *DETAB CATEGORY2. MENU
99 I & "
100 & "
101 & "

CREATE A SOURCE FILE NOT ONLY IN RPL.
12 DATABASE UTILITY PROGRAM.
13 SORT FILE INITIALISER.
14 PRINT SORT FILE HEADER.
15 DATABASE UTIL. PROG. (ALPH. NUMERIC).
16 RETURN TO THE MAIN MENU.

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 

"11 - REPLACE SPACES BY TABS IN ASCII FILE."
"12 - COPY A FILE."
"13 - RENAME A FILE."
"14 - KILL A FILE."
"15 - SORT A FILE."
"16 - RETURN TO THE MAIN MENU."

PLEASE ENTER NUMBER OF FUNCTION REQUIRED : 

ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N : 

FUNCTION NOT ON THE MENU !

FUNCTION NOT ON THE MENU !
156 *DETAB CATEGORY4. MENU
157 I & "
159 & "
161 & "
163 & "
165 & "
167 & "
169 & "
171 & "
173 & "
175 INPUTNP O:1
177 & " ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N :";
179 C -100/1 = "?" 
180 0:1 = ? 11 12 13 14 15 16 - - 
181 A CHAIN "FD1:LC5?" 012 015 016 017 032 - - - - 
183 & "FUNCTION NOT ON THE MENU !" - - - - - - X - 
185 *DETAB CATEGORY5. MENU
186 I & "
188 & "
190 & "
192 & "
194 & "
196 & "
198 & "
200 & "
202 & "
204 INPUTNP O:1
206 & " ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N :";
207 INPUTNP -100/1
208 C -100/1 = "?"
209 0:1 = ? 11 12 13 14 15 16 - -
210 A CHAIN "FD1:LCS?"
211 GOTO MAIN.MENU
212 & "FUNCTION NOT ON THE MENU !" - - - - - - X -
213 GOTO CATEGORY5.MENU
214 *DETAB CATEGORY6.MENU
215 I & "
216 & "
217 & "
218 & "11 - RUN A PROGRAM.
219 & "
220 & "12 - ALLOWS PATCHES TO FILES IN ASC/DEC.
221 & "
222 & "13 - PRINTS ASCII FILES ON TT: HANDLING PP.
223 & "
224 & "14 - LISTS ASCII FILES ON TT:HANDLING FEED.
225 & "
226 & "15 - INTERROGATE LIBRARY CONTROL FILE.
227 & "
228 & "16 - RETURN TO THE MAIN MENU.
229 & "
230 & "
231 & "
232 & "PLEASE ENTER NUMBER OF FUNCTION REQUIRED : ";
233 INPUTNP 0:1
234 & "
235 & "ARE YOU SURE YOU WISH THIS PROGRAM, PLEASE ENTER Y OR N : ";
236 INPUTNP -100/1
237 C -100/1 = "?"
238 0:1 = ? 11 12 13 14 15 16 - -
239 A CHAIN "FD1:LCS?"
240 GOTO MAIN.MENU
241 & "FUNCTION NOT ON THE MENU !" - - - - - - X -
242 GOTO CATEGORY6.MENU
243 *STOP

XENTRY

*ERRORS DETECTED: 0
RPL}
APPENDIX A5

Example of Planning Stage (see 5.2.1)
Target System Preparation

In preparation for the use of the generator a thorough examination of the application (target system) is required so that grouping and program naming can be done in a meaningful way. In one of the tests a **LOCAL CONTROL SYSTEM (LCS)** was used as a target application and the table below is a result of this preparatory examination.

<table>
<thead>
<tr>
<th>Original Program name</th>
<th>Re-named</th>
<th>Function of the program</th>
</tr>
</thead>
<tbody>
<tr>
<td>INRPL</td>
<td>LCS001</td>
<td>Create a source RPL file</td>
</tr>
<tr>
<td>FORMAT</td>
<td>LCS002</td>
<td>Formatted source file</td>
</tr>
<tr>
<td>DICLST</td>
<td>LCS003</td>
<td>Lists *Dict working out start position</td>
</tr>
<tr>
<td>NAMLPF</td>
<td>LCS004</td>
<td>Rename pages on library page file</td>
</tr>
<tr>
<td>CHKLPF</td>
<td>LCS005</td>
<td>Check library file</td>
</tr>
<tr>
<td>GENTST</td>
<td>LCS006</td>
<td>Create any source file not only in RPL</td>
</tr>
<tr>
<td>DBUTIL</td>
<td>LCS007</td>
<td>Database utility program</td>
</tr>
<tr>
<td>SFINIT</td>
<td>LCS008</td>
<td>Sort file initialiser</td>
</tr>
<tr>
<td>PSFHDR</td>
<td>LCS009</td>
<td>Print sort file header</td>
</tr>
<tr>
<td>DBU</td>
<td>LCS010</td>
<td>Database utility program (alphanumeric)</td>
</tr>
<tr>
<td>ADDTAB</td>
<td>LCS011</td>
<td>Replaces spaces by tabs in ascii files</td>
</tr>
<tr>
<td>LCSCOP</td>
<td>LCS012</td>
<td>Copy a file</td>
</tr>
<tr>
<td>IDENT</td>
<td>LCS013</td>
<td>Rename a file</td>
</tr>
<tr>
<td>SETDAT</td>
<td>LCS014</td>
<td>Kill a file</td>
</tr>
<tr>
<td>SYSLOG</td>
<td>LCS015</td>
<td>Sort a file</td>
</tr>
<tr>
<td>TSP</td>
<td>LCS016</td>
<td>Prints control block of a file</td>
</tr>
<tr>
<td>DSKLIB</td>
<td>LCS017</td>
<td>Creates identity table on user diskette</td>
</tr>
<tr>
<td>LIBLST</td>
<td>LCS021</td>
<td>Creates and maintain diskette library</td>
</tr>
<tr>
<td>SYSLIB</td>
<td>LCS022</td>
<td>Lists diskette library usage</td>
</tr>
<tr>
<td>SYSPRRT</td>
<td>LCS023</td>
<td>System software utilities library additions</td>
</tr>
<tr>
<td>RPL</td>
<td>LCS024</td>
<td>Lists system program library</td>
</tr>
<tr>
<td>PATCH</td>
<td>LCS025</td>
<td>Maintain library control file</td>
</tr>
<tr>
<td>LTRPRT</td>
<td>LCS026</td>
<td>Run a program</td>
</tr>
<tr>
<td>LISTER</td>
<td>LCS027</td>
<td>Allows patches to files in ascii decimal</td>
</tr>
<tr>
<td></td>
<td>LCS028</td>
<td>Prints ascii files on TT: handling pages names</td>
</tr>
<tr>
<td></td>
<td>LCS029</td>
<td>Lists ascii files on TT: handling from feeds</td>
</tr>
<tr>
<td></td>
<td>LCS030</td>
<td>Interrogate the library control file</td>
</tr>
</tbody>
</table>
Where gaps appear in the 'Original Program Name' column functional programs were identified but time did not allow for these to be developed. These particular programs were omitted from the complete test sample documented in A3(ii).
REFERENCES


an"d user support", 

SCHACKEL, B., "The ergonomics of the man-computer interface", 

of computers across national boundaries", 

NEWMAN I.A., "Personalised interface to computer system", 

development", 
Loughborough University of Technology, Department of Computer Studies, Report No.73 (1978).


(1979).

Invited paper, Monotype Holding Ltd., Redhill, Surrey, U.K.,
(1979).


[12] HOWARD S., "What is an application package",

computer application",
Loughborough University of Technology, Department of Computer Studies, Report. No.82 (1979).

[14] EASON K.D., "Understanding the naive computer user",
Computer Journal, Loughborough University of Technology,

computer system",
Loughborough University of Technology, Department of Computer Studies, Internal Report, (1979).


[17] HEBDITCH D., "Design of dialogues for interactive commercial application",


