LEX

TECHNICAL MANUAL

LEX Version 9

November 1985

LEXET

This manual was produced using LEX Version 9 and LEXET. LEXET allows the production of typeset output from LEX using a Laser printer.

For further details on LEXET please contact:

Ace Microsystems Ltd
Kew Bridge House
Kew Bridge Road
Brentford
Middlesex TW8 0EJ

Chapter 1. INTRODUCTION

This manual describes the operation of LEX from a systems point of view and is intended for users who are involved in customising the issued system for a particular application.

Originally called LEX-11 and operated on the Digital Equipment Corporation range of PDP-11 computers, LEX has now been extended for use with a wide variety of machines. All versions are compiled from the same set of sources so the only differences in operation are those caused by operating system or VDU constraints.

Before detailing the options with LEX it is worth reviewing its major features.

LEX is a menu driven system and whenever a menu is on the screen the required option can be selected by pressing the appropriate key. These menus may be customised to suit a particular need and may lead to other menus, documents, or programs.

1.1 MENU SWITCH OPTIONS

LEX has in built switch functions that may be included in the menu options. When a LEX menu is displayed they may be activated directly, bypassing the options shown on the screen. This is achieved by entering / followed by a character representing the option. Appropriate prompts then guide the user through the option.

The menu switch options are described in Appendix A.

1.2 CONTROL FUNCTIONS

Control functions are activated by pressing the CTRL key simultaneously with an alphabetic character key. Internally LEX uses a fixed set of 26 control codes defined by the full set of alphabetic characters A-Z. The meaning assigned internally by LEX to a particular control code is invariant, for example internal control code D always means 'delete character'.

In general a given CTRL and letter combination keystroke is associated with the internal code which uses the same alphabetic letter, for example CTRL and A is associated with internal code A which means 'insert a blank line'. A description of the full set of 26 control functions is given in Appendix B.

Certain operating systems regard some CTRL and letter combinations as special input characters which are intercepted by the operating system and not passed through to LEX at all. Also, on certain VDUs, use of the cursor arrow or other special function keys may send a CTRL and letter combination to the computer. In such cases some of the internal control codes used by LEX may not therefore be associated with the normal CTRL and letter combination. Caution must therefore be exercised when using Appendix B; the functions described therein are INTERNAL to LEX but may be activated on a particular system by a different CTRL and letter combinations or input escape sequences.

Control functions may be actioned when LEX recognises the visible equivalent of a control function in a keystore or system record. The visible equivalent of a control function is a single quote followed by the appropriate alphabetic character, for example CTRL and D has the visible equivalent 'D. Visible equivalents are extensively used in LEX programming.

Note that the visible equivalent of a control function ALWAYS has the meaning described in Appendix B. Thus 'D will always be interpreted by LEX as 'delete character' even if the CTRL and D combination input by the operator has been mapped onto some other internal function or even disabled by tailoring of the *VDUFORM system record.

1.3 ESCAPE SEQUENCES

The phrase 'escape sequence' is used to describe the series of options actioned by entering a sequence of characters in turn starting with ESC. For example ESC followed by * and E would be shown as ESC*E.

Appendix C lists the many sequences available. Whenever possible they are described in ascending alphanumeric order but where nonalphanumeric characters are used these are grouped together in logical order depending on function. For example all cut and paste sequences are grouped together.

Like control functions, escape sequences work in document edit mode and in the playground. Some also work at the various system prompts. The only escape sequences to work from a menu are ESC E and ESC Z which take LEX to the previous menu in the stack.

Again as with control functions they too may be used in LEX programming. A $ is used as the visible equivalent of ESC so that ESC *E for example, would be represented by $*E. In this way escape sequences can be actioned from keystores and records.

1.4 THE LEX DATABASE

LEX combines the functions of a word processor, database and calculator into one integrated system. The database features include accessing other LEX databases, list processing and mail merge. Database files consist of records containing the data, each with its own unique key.

1.5 CUSTOMISING AND TAILORING LEX

LEX may be customised to suit a particular requirement. In fact entire systems may be written in LEX. All system definitions are held as system records in the LEX database - the VMF. Therefore the menu options can be changed, keyboard layouts redesigned, printer details re-specified and the prompt and error messages altered.

By making use of these facilities, standard or repetitive tasks may be included as a menu option and reduced to a single key depression. LEX applications systems include personnel and recruitment record systems, conveyancing systems, training timetables, sales and stock control recording, insurance claims, car fleet maintenance and travel cost control.

LEX is not only capable of being customised to suit a particular application, but is equally adaptable for languages other than English. Where examples are given in this manual the system described represents the standard version as issued by Ace Microsystems Limited for use in English speaking countries. LEX, when issued by other suppliers, may already be tailored to suit particular requirements and in such cases queries should be directed to the supplier rather than to Ace Microsystems Limited.

1.6 THE MANUAL

This manual starts by giving background information on the structure of LEX, the files it uses and how to get it working.

That is followed by a section covering the configuration of a system, for a specific machine and interfacing it with VDUs and printers. That section also covers the prompts, messages and features which control the overall operation of LEX.

The next section describes how LEX can be customised to an end user's requirements and therefore provide him with his own individual system. It includes chapters on programming LEX and use of the database, list processing and mail merge.

The final section is a reference section with a complete list of LEX commands, options, error messages and other tables of information.

CHAPTER 2. LEX FILE STRUCTURE

A LEX system for any combination of machine and operating system consists of six fundamental parts, each normally in its own file with its own identifying extension name. These files are as follows and their functions are described in the remainder of this chapter.

The Start-up Module	- A file recognised by the operating system as an executable program, this file loads LEX.
The Run-time System	- The run-time system providing the links between LEX and the operating system.
The LEX Code	- This contains the program which performs the various LEX operations.
The Work File	- LEX uses a work file for storage of temporary information.
The Virtual Memory File	- The VMF is LEX's database which may be tailored by the user to suit individual requirements. It contains data records and control information including menus, screen formats, messages and record layouts.
Documents	- Letters, reports, memos, and other user created information are stored as documents. There can be any number of these. Each document consists of an ASCII file which conforms to the standard for the operating system being used.

2.1 THE START-UP MODULE

The start-up module is written specifically for each operating system that LEX runs on. It is responsible for loading the run-time system, and some of the actual LEX code. On completion it passes control to the loaded code and LEX itself starts to run. In some cases, the start-up module and the run-time system are held together in one file rather than as two separate files.

2.2 THE RUN-TIME SYSTEM

The run-time system (RTS) resides in memory and provides an interface between the LEX code and the operating system. During the course of its execution LEX makes requests on the operating system for particular actions such as reading information from a disk, obtaining input from the operator's terminal, sending output for printing, etc. Each operating system requires a particular structure for these requests and so the LEX code makes such requests via the RTS and not directly to the operating system. The RTS converts the requests into a form required by the operating system and also converts any information returned by the operating system into a form required by the LEX code.

For many implementations of LEX the LEX code is not held entirely in memory. For these implementations a region of memory known as the 'page pool' is obtained and discrete portions of the LEX code called 'pages' are brought into the page pool for execution as they are required. The RTS for these implementations contains the necessary routines to manage the page pool.

For some systems optional variants of the RTS are available. For example on the PDP-11 there are versions of the RTS which use different maths packages. One version uses a decimal maths package where numbers used by the LEX calculator are held as 32-bit signed integers with an additional byte used to hold the number of decimal places. An optional version uses the standard DEC 4-word floating point format. The latter provides a larger number range than the former but is slower in operation and, because the RTS is larger and thus reduces the amount of memory available for the page pool, requires more disc accesses to refresh the page pool which again causes a slowing down in operating speed.

On some multi-user systems there is also the possibility of there being two variants of the RTS; one where a single copy of the RTS is shared by all LEX users on the system, the other where each user has his own copy of the RTS loaded in his own job space.

2.3 THE LEX CODE

LEX is written in a decision table language, Filetab-D. The same source code is used to produce executable code for any given computer system for which LEX has been implemented. The compiler is instructed to generate executable machine code for a given processor chip via 'code generation tables' one of which has been prepared for all processors for which a LEX implementation has been produced.

By providing the interface of the RTS between LEX and the operating system, the LEX program source is independent of the operating system, which enables one common source version of LEX to be used for all machines.

This means that, as far as the user is concerned, LEX operates in an identical manner on all machines.

The output of the compiler is the LEX code file. By convention this file is given a name of the form:

```
LEXnnn.Dmm
```

where `nnn` identifies the version number of LEX and `mm` depends on the machine type on which this file is intended to be used, for example `.D11` for the PDP-11, `.D86` for the 8086 chip, etc.

2.4 THE WORK FILE

LEX uses a work file to hold information about the current document being processed, the current status of LEX and the contents of certain user-accessible buffers.

There are two types of work file; temporary and permanent.

Permanent work files are given a unique name by the user and allow an editing session to be interrupted and resumed at the point where it was left off. The name of the document being edited, cut and paste buffers, rulers, keystores and calculator memories are all retained.

If no permanent work file is specified LEX will create and use a temporary work file, whose name is unique and is based on the time, or on systems without a time facility a sequential number is allocated. On exiting from LEX a temporary work file is deleted.

Chapter 4 outlines how the type of work file may be selected and how a default file name may be defined. Note however that, on multi-user LEX systems, it is essential that each user has a unique work file. If more than one user is using the same work file DISASTROUS RESULTS WILL OCCUR.

Work files should be a minimum of 128 512-bytes blocks in size but should preferably be as large as possible so as to hold a copy of the entire document being edited. As a guide a work file should be twice as large as the largest document which will be edited with an additional 70 blocks added for work areas. The work file size is set in the **INST system record as described in Chapter 5.

During editing LEX uses an in-memory buffer which will hold a number of screenfuls of text from the document being edited. As editing proceeds and the user moves further through the document, text is removed from the start of the in-memory buffer and is written to the work file.

Provided that the work file is large enough, it is possible to scroll backwards and forwards through an entire document. If the size of work file is too small to hold the complete document being edited the work file will become full at some point during editing. At this point text from the start of the document is removed from the work file and written away to a temporary file. Scrolling back is then limited to the first line of text from the document which is still contained on the work file.

To return to the start of the document it is necessary to perform a 're-edit' operation, that is to end the current edit, thus causing the edited copy of the document to be written out as an ASCII file and re-commence editing from the first line of the document.

Note that, when a work file becomes full it is not possible to insert lines within the remaining text of the document being edited.

A work file may be initialised using menu switch option /2. This clears all buffers on the work file, clears calculator stores, clears abbreviations, clears the document name etc. It is also possible to switch to a different work file using menu switch option /Z.

2.5 THE VIRTUAL MEMORY FILE

The Virtual Memory File (VMF) is the LEX database. It contains any data records created by the user, such as names and addresses, and also provides the LEX prompts and control information. Such prompts and control information can be tailored by the user to suit his requirements, including modifying menus, error messages, forms, output formats, and programming instructions. Further, LEX is customised for individual terminals and printers by creating or amending records in the VMF.

Any number of VMF files may exist on a system and the user may switch from one to another as required. They are normally identified by the extension .VMF, although any valid filename may be used.

2.5.1 Structure of the VMF

The VMF is a particular form of a type of indexed file supported by the Filetab-D language known as a database file. The file contains an index based on alphanumeric keys which are held in ascending ASCII-Lexicographic order within the index. Each index entry points to a chain of data records which are stored elsewhere on the file. The chain of records is manipulated by LEX and presented to the user as a single logical record, for example a complete screenful of text in a form.

LEX contains a handler for the VMF. Physical access to the VMF by the handler uses a 512-byte unit of transfer known as a block. The VMF is thus divided into a number of 512-byte blocks of which the first on the file is numbered 0 (zero).

The VMF is logically divided into four sections:

Header block	Block 0
Top-level index block(s)	Start in block 1
Second-level index blocks	Start after top-level index block(s)
Data blocks	Start after second-level index blocks

With the exception of the header block, every block on the VMF has a block occupancy byte (BOB) as its first byte.

The header block contains the VMF structure mark number and information about the number of blocks used for the second-level index.

The top-level index (TLI) which starts in block 1 consists of an ordered set of key values, each key value being the highest key value which will be found in the second-level index block whose block number on the file corresponds to the position of the key in the TLI.

The first byte of each TLI block (BOB) has the value 255 decimal and the last byte is a continuation marker indicating whether or not this particular TLI block is the last one. The remaining 510 bytes contain the key values.

The number of TLI blocks depends on the number of second-level index blocks present. This is set when the VMF is initialised by the VMF reorganise function in LEX.

The key size on the VMF is ten bytes. Each TLI block can hold 51 keys. The number of blocks required for the TLI is thus:

$$\frac{\text{Number of second-level index blocks}}{51}$$

If the number of TLI blocks is 1 or 2 then, when LEX first opens the file, the whole of the TLI is read into memory and is held there until the VMF is closed. For a VMF with more than 2 TLI blocks a 2-block memory buffer is used and portions of the TLI need to be read into memory during operations on the file. LEX makes use of an additional level of index held in memory in such cases to speed up the index search. Note that the TLI is never altered on a VMF except when a new TLI is built when a VMF is reorganised.

The second-level index (2LI) starts in the block which follows the last TLI block. The first byte in each 2LI block (BOB) has the value 255 decimal. The second byte is used when the VMF is opened in multi-user mode and the 2LI block is about to be split into a home and overflow block. In these circumstances the VMF handler in LEX sets the second byte to a value which indicates that the complete 2LI block is locked against access by other users. The byte is reset to its normal value when the splitting operation is complete.

The next 4 bytes in the 2LI block consists of 2 16-bit integer values which are chain block pointers, the first points to the the next 2LI block in sequence the second points to the previous 2LI block in sequence.

The body of each 2LI block consists of 36 14-byte Index Entries which consist of: a 10-byte key, a 3-byte data record pointer and a single flag byte. The number of 2LI blocks required is:

$$\frac{\text{Number of Keys}}{36}$$

where Number of Keys is specified when the VMF is initialised by a LEX VMF re-organise operation.

The 3-byte pointer part of the Index Entry consists of a 16-bit byte-swapped integer specifying a block number and an 8-bit integer which specifies a position within a block. The pointer in the index entry points to the last record in the chain of data records associated with that index entry. One of the bits in the flag byte is used to mark a deleted index entry. The remaining seven bits in the flag byte are reserved for future use.

When a VMF is initialised at the start of a re-organise operation by LEX each 2LI block is initially filled with 'High Key' values, that is each index entry is set to 14 bytes all having the value 255 decimal. The re-organise will insert index entries into approximately half of the space available in each 2LI block.

Following the 2LI blocks on the VMF are data blocks which contain the chains of records associated with each index entry. Some blocks in this area of the file may also contain overflow 2LI index blocks. Overflow of keys is described later.

The first byte in each data block (BOB) indicates the next available position in the block where new data may be inserted. A completely empty data block has a BOB value of 1. The actual insert position in the block is given by the BOB value times 2 plus 1. The first data record in a block will therefore start at byte 3, with byte 0 being the BOB and bytes 2 and 3 never being used.

Within each data block each record consists of a 3-byte forward pointer to the next record in the chain. This pointer has a similar structure to the 3-byte pointer in the index entry. This is followed by a 1-byte record length which is followed by the actual contents of the data record. The theoretical maximum length of data in a data record is 255 bytes, however LEX imposes a smaller maximum than this to achieve more efficient packing and to improve speed of execution.

Each record in a data block is complete and does not span block boundaries. There will usually be a small number of unused bytes at the end of every data block.

In general the contents of data records are actual ASCII characters except that, when a record contains a number of consecutive blanks, these are converted into a single 'squashed' character in the record. Storage of 'squashed' characters is achieved using the 'low value ASCII' characters - with a value in the range 1 to 31 decimal - below the value used for the space character. Thus a 'squashed' character of value 5 will represent 5 consecutive blanks. Where a record contains more than 31 consecutive blanks multiple 'squashed' characters are used to represent them. Thus 73 consecutive blanks in a record will be stored as 3 'squashed' characters of value 31, 31 and 11.

When a data record is recalled by LEX any 'squashed' characters are unsquashed into the appropriate number of blanks. In fact the default character used for unsquashing is the underline character as this usually represents unused space in a form in LEX. With data records it is possible to specify the value of the character, strings of which are to be squashed on writing to the VMF or which character is to be used when 'squashed' characters are unsquashed on reading from the VMF. System records however MUST used underline as the squash character.

Note that LEX represents physical (RETURN) characters in a data record using NULL (value zero) characters in the record on the VMF, this is so that the standard ASCII character for (RETURN) (which is 13 decimal) may be used as a 'squash' character.

2.5.2 The Structure of the Keys

Each key comprises ten characters. If fewer than ten characters are entered, then LEX will pad the key with space characters.

The first character of the key is used to identify the record type; any character may be chosen except the square bracket characters [and]. For some records the second character of the key is used as an extension to the record type. LEX uses the following types of record internally so user defined records should avoid them.

TABLE 2.5 - Special Record Types

Record Type	Purpose	Short Title
**	System CONTROL record	** Record
)	VDU Record 1 (Main)	VDUFORM
))	2 (More sequences)	VDUFORM2
)=	3 (Screen attributes)	VDUFORM3
);	4 (Function set ups)	VDUFORM4
*)	VDU Character conversion table	VDUTABLE
*$	Printer record 1 (Main)	PRTFORM
*%	2 (Attributes)	PRTFORM2
*'	Printer petal conversion table	PETFORM
$(	Petal functions	PETFUN
*;	Proportional spacing	-
*.	Document extension details	PAPER
*,	Masking format	MASKFORM
*=	Spelling system	-
*-	Error messages	ERRORFORM
:	Document index record	-
*	General system record	System record

2.5.3 Multi User VMF

LEX may be directed to open the VMF in either single-user or multi-user mode. Multi-user mode is only available if the operating system under which LEX is running provides single-block locking. This means that the operating system will prevent the reading of a block on the file if that block has been locked (usually by being read) by another user. Provided that the operating system can support this basic requirement the VMF handler in LEX is able to provide true multi-access to a single VMF.

If a VMF is opened in multi-user mode the handler sets a lock bit in the index entry for a particular key when one of the users accessing the file performs the necessary 'Read with Lock' function within LEX.

A chain of records so locked may then only be updated by the user who set the lock, other users are only allowed to read the chain and are given a warning if they attempt to update it.

In order to invoke operating system block locking it is usually necessary to open the VMF in a special mode. If the file has already been opened by another user, some operating systems will not allow a file to be opened at all unless the current attempt to open the file specifies an opening mode which is identical to the mode used when the file was previously opened by the other user.

For example under the RSTS operating system on the PDP-11 a file may be opened MODE 0 (normal mode) or MODE 1 (update mode). MODE 1 invokes block locking by the operating system. If a file has already been opened in MODE 0 an attempt to open it in MODE 1 will fail and vice-versa. Also if a file is opened in MODE 0 by more than one user only the first user to so open the file will be allowed to perform write transfers to the file by RSTS.

LEX can be set up to open the VMF either in single or multi-user mode. Multi-user mode allows more than one user to have controlled access to the database and is useful when a number of users require read/write access to common data, for example customer names and addresses, stock records, etc. LEX will ensure that any updates to the file are correctly and unambiguously performed. It should be noted however that operating in multi-user mode of necessity significantly increases the number of disc transfers which the handler must make to and from the VMF which will cause slower operation of LEX.

If LEX is being used on a multi-terminal computer system and some VMFs are open in single user mode with other VMFs open in multi-user mode there can be problems when a given LEX user requires to access a secondary VMF, for example as a foreign file in list processing or for the purposes of copying records to or from the secondary VMF. This is because the secondary VMF may have been already opened by some other user in a mode which conflicts with the mode in which LEX requires to open it.

For foreign file access in the List Processor the user is able to control the mode in which the foreign file is to be opened. For VMF to VMF copying LEX will attempt to open the secondary VMF using whatever mode was used to open the main VMF. If the operating system refuses to open the file due to a mode conflict an error message is displayed.

If possible it is best if a multi-terminal system is set up so that the installation is all multi-user or all single user as far as LEX is concerned; that is all opens of VMFs by LEX will be either MODE 0 or MODE 1 and never MODE 0 by one user and MODE 1 by another.

Note that the VMF handler in LEX assumes that, if a file is opened in MODE 0 only one user will be attempting to write to the VMF. Some operating systems will not prevent write transfers from more than one user if the VMF is opened MODE 0. However it should be stressed that this practice is very likely to lead to corruption of the VMF.

2.5.4 Reading Records from the VMF

When LEX is required to read a record from the VMF it first searches the top-level index to determine which second-level index block will contain the required key, assuming that the key actually exists in the index. The appropriate second-level index block is read into memory if necessary. A read of the index block will be necessary unless the VMF is opened for single user access and the index block is currently in memory.

The contents of the index block are serially searched for a match with the required key. The search will terminate if a match for the key is found or if a key with a higher value than the required key is encountered. Note that, in the case of a matching index entry being found, the flag byte in the index entry is tested to ensure that the entry does not represent a deleted key.

If no equal or higher key is found but the last entry currently in the index block is less than the required key this means that key overflow has occured. The block chain pointer at the start of the index block is used to retrieve the overflow index block and this block is searched in a similar manner. Note that the overflow block may itself have an overflow block which may itself have an overflow block and so on. If no equal key is found in the main or overflow index block(s) then the required key does not exist.

Once the index entry for the required key has been located the 3-byte pointer part of the index entry is used to read the relevant data block which contains the last record in the chain of records associated with the given key. If there is more than one record in the chain the forward pointer at the start of the record in the data block will point to the first record in the chain.

The first record is retrieved and the forward pointer chain followed to read any subsequent records in the chain until the 'end of chain' record has been read. Unless all records in the chain occupy the same data block on disc this process will require further disc transfers. The chain of records is concatenated into one long logical record in memory and, if necessary, 'squashed' characters are expanded before the complete logical record is presented to the user.

2.5.5 Creating New Records on the VMF

LEX is required to create a new record on the VMF either when the create command is used or when the update command is used and no record with the given key is currently present on the VMF. In either case the VMF index is searched and the position in the index noted for the new key for the record about to be written to the file.

If there is no free space on the VMF LEX will not attempt to write to the file, instead issuing a (BELL) to the terminal.

If there is free space on the file the logical record to be written is first processed in memory to convert strings of 'squashable' characters into their 'squashed' representation. The transformed record is then divided up into the physical record units which will be written as a chain of records to the file. LEX imposes a maximum length of 124 bytes for each physical record written. LEX also ensures that certain consecutive bytes are always contained together in the same record, for example the visible equivalent of a control function, single quote followed by one of A-Z. If the division of the logical record into physical records would cause the quote to be at the end of one record and the following letter character to be at the start of the next, the length of the first record is reduced by one to make the quote become the first byte in the second record.

LEX keeps a note of which data block is currently accepting data. This block is retrieved and its BOB examined to see if the block has sufficient remaining free space to hold the first record in the chain about to be written. If not the file is searched from the next higher block number onwards to find a data block with a suitably low BOB value. The first record in the chain is inserted into the next free position in the data block, preceded by a forward pointer and a record length byte. For the first record in the chain the forward pointer will initially point to itself. The BOB value in the data block is updated to point beyond the record just inserted.

At this point the index is updated. The key value is inserted in the correct place in sequence in the index and the 3-byte pointer part of the index entry set to point to the record just inserted. When inserting the index entry any entries in the same index block with key values higher than the inserted value are moved to create space for the inserted entry and to preserve the sorted sequence in the index block. If the insert point is currently occupied by a deleted key the deleted key entry is overwritten by the new entry. Insertion of the new index entry may create an overflow index block as described below.

If the logical record being written consists of only one physical record this completes the creation process. Where the logical record requires a chain of physical records the remaining records in the chain are written to the file in a similar manner. If possible each succeeding record in the chain will occupy the same data block as the preceding record in the chain. When there is insufficient free space in a data block to accept the whole of the next record in the chain about to be written LEX searches further up the file for a data block with a sufficiently low BOB value.

During writing of the chain of records the forward pointers which are at the start of each record in the chain are adjusted, as is the pointer in the index entry. When the complete chain has been written, the pointer in the index entry will point to the last record in the chain, the forward pointer for the last record in the chain will point to the first record in the chain, the forward pointer for the first record in the chain will point to the second record in the chain and so on.

Normally each 2LI block has at least one unused high key value index entry at its end. If insertion of the new entry causes this last high key entry to be overwritten or 'shunted' off the end of the index block then the index block is full and LEX will create an overflow index block as follows:

LEX finds the next available completely empty data block on the VMF and alters the BOB value from 1 to 255 to mark it as being completely full. This data block will now be made into an overflow index block. Index entries from the upper half of the just-filled 2LI block are written into the lower half of this data block and the remainder of the block filled with high key values. The upper half of the just-filled 2LI block is filled with high key values. The chain block pointers in the 2LI block and the overflow index block are set up so that the forward pointer in the 2LI block points to the overflow block, the backward pointer in the overflow block points to the 2LI block and the forward pointer in the overflow block points where the forward pointer in the 2LI block used to point.

When index block splitting occurs the index is said to have gone into overflow. Note that index overflow can occur even if the index as a whole is far from full if a number of new records are created each with a similar key value.

2.5.6 Updating Records on the VMF

When an update of an existing record is required LEX first squashes the logical record in memory into its storage format. The logical record currently on file exists as a chain of physical records. The data content of each record in the chain currently on file is replaced by new data from the new logical record. If the new logical record is shorter than the existing logical record the length of the last record in the chain will need to made shorter If the new record is considerably shorter some of the existing records at the end of the chain will not be overwritten but will remain on file occupying space which will be recovered by a VMF reorganise.

If the new logical record is longer than the existing record one or more new physical records will be created at the end of the chain to hold the end of the data. These new records will be written into previously empty space in data blocks.

2.5.7 Record Deletion

When a logical record is deleted on the VMF the action of the handler is actually only to mark the index entry to indicate a deleted key. This is done by setting the delete bit in the flag byte in the index entry. The 3-byte pointer in the index entry is left unaltered and the chain of data records associated with the key is left on file. This makes it possible to restore a deleted record simply by resetting the deleted bit in the flag byte. Note however that the index entry for a deleted key may sometimes be re-used for a newly inserted key if the new key requires to occupy the position of the deleted key in the index. When a VMF re-organise is performed any deleted keys and their associated chains of data records are not transferred to the re-organised file.

2.5.8 Record Locking

In a Multi User VMF where a number of users may have read/write access to the VMF at the same time it is important to ensure that only one user can update a record at a time.

This is achieved by locking the record prior to updating it.

If a record is read in the ordinary way without locking, it is possible for a second user to read the record and perhaps update it without the first user realising that it has changed. Reading and locking the record guarantees that the second user will be given a 'read only access' warning when he subsequently reads it.

Once a record has been locked it remains locked until it is updated, another record is read or the user returns to a menu.

Because the Document Index in LEX consists of records on the VMF it is possible that the Document Index record for a given document may be locked by another user when the current user attempts to edit or view the document. If so the following error message (taken from the **TEXT system record) is displayed:

```
Document Index Locked
```

LEX allows the user to continue with the view or edit but the Document Index record will not be updated on completion. As a general rule a user should refrain from operating on documents whose Document Index record has been locked by another user as he is likely to be viewing or editing a document which another user may well be in the process of altering.

LEX implements locking on the VMF by setting a bit in the 3-byte pointer portion of the index entry for a given key. The first 2 bytes of this pointer comprise a byte-swapped 16-bit integer which is the (block number plus 1) of the data block on the VMF which holds the end of chain record associated with the given key. The sign bit of this integer is normally zero but when set indicates that a lock has been set for the given key.

As well as the above type of lock LEX may also set a lock on a complete second-level index block. This will occur when LEX is about to split the block and create an overflow block. Until this operation is complete other users must be prevented from accessing the index block and possibly modifying it. The second byte of the 2LI block, which normally has the value 0 or 255 decimal is set to the value 1 to indicate that the block is locked. Should other users attempt to read the block whilst the lock is set the handler will perform a sleep-retry loop until the lock byte has been reset to its normal value (when the split into overflow operation has been completed).

Because LEX sets locks by modifying bits and bytes on the VMF it follows that an abnormal termination of LEX, for example a system crash, may leave the VMF with some lock bits or index block lock bytes set in the locked state. Menu switch option /0 is provided to enable all such locks to be cleared. When running LEX to clear locks on a VMF the user should ensure that no other user has the VMF file open at the same time. This is best ensured by having the VMF open in single user mode whilst the lock clearing operation is performed.

2.5.9 Reorganising the VMF

From time to time it becomes necessary to reorganise the structure of the VMF file. This is achieved by menu switch option /3 and is described in Appendix A. These are two types of reorganisation. The type chosen depending on which of the following caused the need for a reorganisation.

1. As new records are added to the file so the data area may become full. A manual reorganisation is then necessary to extend the file data area for future use.
2. A large number of new records created in one area of the index may cause the index to go into overflow. Performance may suffer because of the extra disk reads that this entails even if there is still a lot of room in the index and data area. Normally an automatic reorganisation will retrieve the situation.
3. Even when no new records are added to the file it is still possible for the file to become full if a large amount of unused space accumulates in the data file. Such space is created when records are updated and extended since space occupied by the old, smaller record is not re-used. An automatic reorganisation will again normally suffice in these circumstances.

It is usual to find some combination of the above to cause the need for reorganisation. Generally, an automatic reorganisation is sufficient unless this does not provide enough free space for future usage. However the automatic reorganisation is built for speed rather than efficiency so an occasional manual reorganisation is useful to reformat the records more compactly.

The effects of a reorganisation are as follows. The top-level index is re-created based on the actual current distribution of key values on the file. Any deleted keys and their corresponding data records are not carried over onto the reorganised file and space occupied by unused portions of data records is recovered. The second-level index blocks are re-built only partially full which leaves room for a significant number of subsequent insertions without index overflow occurring. All the chains of records which form each logical record are written into a physically contiguous region of the file which minimises disk head movement required when the records are subsequently retrieved.

Because the index blocks are re-built only partially full and also because of the different packing of records into data blocks which occurs on reorganising there may be occasions when the reorganised file will need to be somewhat larger than the original file if it is to contain the same amount of free space.

2.6 DOCUMENTS

Documents contain letters, documents, and other work. There can be any number of them within the limits of disk storage available.

2.6.1 File Structure

Documents are ASCII files and so may be processed by other editors or operating system utilities.

LEX supports character attributes allowing up to five attributes to be associated with each character in a document. When a line of text in a document contains at least one character which has one or more associated attributes then the line of text is written to the ASCII file as two lines. The first line is the set of ASCII characters which form the line of text except that a special character is added to the end of the line in the ASCII file, immediately prior to the line terminating character(s) if these are present.

This special character is defined in the **MONT system record and usually has the value 255 decimal which means that it will not normally be treated as a printing character by operating system utilities. The presence of this special character indicates to LEX that the next line of text on the ASCII file should be interpreted as the visible representation of character attributes associated with characters in the line ending with the special character.

The attributes-defining line which follows consists of space characters and characters which are the visible representation of attributes. If the file is printed using an operating system utility each visible attribute character will appear directly under the character on the line above to which the attribute(s) apply. A table showing the characters used to represent attributes, together with some examples is given in Appendix I.

2.6.2 Document Naming Conventions

LEX documents exist as standard ASCII files in the file structure of the operating system being used. The name given to a document when it is originally created using LEX must therefore conform to the rules governing file names imposed by the operating system concerned. In addition there are some constraints applicable to file names over and above those imposed by the operating system of which the LEX user should be aware and it is these aspects which are covered here.

A document name can be up to 40 characters in length although not all operating systems will support names of this length.

a. Device/Account Details

An option in ****INST** allows LEX to be restricted to accessing documents in the users directory/account. Alternatively the device/account or directory can be allowed to appear whenever a document name is requested. e.g.

```
Document Name SY:(1,7)______________
```

In the latter mode LEX places the cursor immediately after the device/directory specification but the user may move the cursor back and change or delete these details. The ability for a user to access a particular device/account/directory may depend on permission settings in the operating system.

The characters used by LEX as the device/account delimiters may be patched to suit an operating system as discussed under configuring a LEX system in Chapter 4.

b. Body of the Name

The body of the document name follows the device/account details and is terminated by a . (dot). The maximum length of the name is dependent on the operating system but note that the key to a Document Index record which LEX constructs from the document name will use only the first six name characters specified by the user. To avoid possible duplication in Document Index keys it is therefore recommended that no more than six characters be used for the body of the document name.

Any printable characters may be used in the document name except those defined as device/account delimiters. Normally these include

```
/ ( ) [ ] < > $ ! % @ # :
```

Some operating systems treat upper and lower case case alphabetic characters in filenames synonymously, that is the names FRED, fred and Fred would be considered to be identical. Other operating systems allow mixed case and would consider FRED and fred to be different. LEX automatically translates input document filenames into upper case unless the case switch in the **INST system record is set to L (when translation to lower case is performed) or M (mixed case) when no translation takes place.

Mixed case is not recommended as the user must remember the exact case used when a document name was originally entered if the document is to be viewed or edited. Mixed case is used for example for Arabic users where the alphabet contains more than 26 characters, some of which are accommodated using ASCII codes normally used for lower case letters. In these circumstances no case conversion ever takes place.

Note that for the LEX Document Index key, which uses the first six characters of the document name in its construction, the name characters used are translated into upper case in the key as per *CASEFORM.

c. The Extension

The document filename extension follows the . (dot) at the end of the body of the name. The maximum number of characters in the extension is dependent on the operating system but it is strongly recommended that a maximum of three characters be used. This is because LEX uses the extension to define the class of documents to which a common set of rulers, keystores and printing details are applicable and expects the extension not to exceed three characters. Also only the first three characters of the extension are used in the construction of a key for the LEX Document Index and, by restricting the extension to three characters ambiguities may be avoided.

If the input document name does not contain a . (dot) then LEX appends the default extension defined in the **INST system record (see Chapter 5). It is possible to specify a filename with a blank extension by including the dot as the last input character in the filename with no following characters. On UNIX systems the dot is suppressed by LEX in these circumstances before the filename is presented to the operating system.

It is possible to direct LEX to apply the parameters normally associated with a given document filename extension to a document with a different filename extension by appending /*.aaa to the end of the input document name, where aaa is the extension whose associated parameters are to be used.

2.7 PLAYGROUND

Playground is not strictly a document. It refers to the operation of LEX in edit mode but with no document directly accessed. All input enters the workfile and it may be processed as normal including being scrolled off and on the screen. It is not subsequently stored as a file and is therefore lost on returning to a menu.

On entering the playground the document extension implied is .080, .132 etc. where the number is taken from the width of the VDU as defined in the current VDU record.

Chapter 3. INSTALLATION, SERIALISATION & START UP

The process of getting LEX running for the first time consists of copying the appropriate files from the issued distribution kit, serialising the copy onto the machine in question and finally going through the LEX start up procedure. This chapter describes the procedure in general terms and should be read in conjunction with the installation notes provided for a particular computer.

Once LEX is operating in this initial form Chapter 4 should be consulted to make the start up process a simple operation for end users. LEX is then in a state to be configured, customised, tailored or used.

LEX is licenced to run on one machine only. Therefore until the serialisation procedure has been performed it operates as a demonstration package. The limitations of LEX in its demonstration mode are outlined in this chapter along with the technique to make the package 'live'.

3.1 INSTALLING LEX

A typical LEX distribution kit contains the following files. Note that the extension will vary depending on the operating systems used.

```
READ  .ME
LEX9A .EXE
LEX9A1.D86
LEX9A .VMF

D     .EXE
LXSERL.D86
LXSERL.LIT
LXINST.D86
LXCONV.D86
LXCASC.D86
```

The first file is textual information which should be read before attempting to install a system. Release notes, the latest patching information and an installation guide may be issued in this way.

Next comes LEX itself. It consists of a VMF file plus from one to three other files depending on the computer to be used. The function of each of these files is described in Chapter 2. Note that their names all take the form

```
LEXnnn.aaa
```

where nnn is the version of LEX, in this case Version 9A.1, whilst the extension describes the purpose of the file.

These are the files that must be copied into the account/directory where LEX is to run.

On PDP-11 versions where the operating system supports floating point maths, both floating point and decimal versions of the run-time system (RTS) may be provided. The decimal version is normally used and it gives calculator precision to 10 digits. The floating point version is accurate to 16 digits but this must be traded against its slower speed. The decimal RTS file is named `LEXnnn.RTS` and the floating point version `LEXnnn.RTF`. LEX always uses the file `LEXnnn.RTS` as its runtime system so to install floating point maths rename that file as `LEXnnn.RTS`.

The final set of files in the distribution kit consists of a group of programs which perform specific functions when first setting up LEX. They are run as independent programs and are described in later sections in this chapter. They include such functions as the serialising program, a program to patch start up defaults within LEX and another which converts VMFs that were developed for earlier versions of LEX. The appropriate files must be copied and run where necessary but may then be destroyed.

3.2 START UP PROCEDURE

3.2.1 Selecting the work file

The command used to activate LEX depends on the operating system. Often it is as simple as:

```
LEX(RETURN)
```

At this point LEX looks for a valid work file using a built in search pattern. [redacted] as outlined in Kapitel 4 Chapter 4.

Temporary work files selected in LXINST

If temporary work files were selected with LXINST, LEX creates and opens `DEV:HHMMSS.WRK`, where [redacted] is the default work file device (see option 7 in LXINST). `HHMMSS` represents the time, or on systems without a time facility a sequential number is allocated. [redacted] option 8 in [redacted]

Permanent work files selected in LXINST

If permanent work files were selected with LXINST, LEX attempts to open `DEV:LEX9A.WRK`. If this file cannot be used for one of the following reasons:

a) LEX cannot find the filename
b) the file opened is not a valid work file
c) someone else is using the workfile

then LEX asks for the name of a valid work file with the prompt:

```
.wrk?
```

At the `.wrk?` prompt the user has three options:

a) specify a permanent work file
b) specify a temporary work file
c) chain to another program

These options are now considered in turn.

a) **Permanent work files**

   Permanent work files allow the user to continue from the previous session and to carry over information between sessions.

   If no extension is included in the file name entered in reply to the prompt then LEX applies the default extension of `.WRK` or other value as set by patching using LXINST - see Chapter 4. LXINST may also be used to set the default device.

   When LEX opens the work file it checks to see whether it is invalid or in use by examining the first byte of the file. If in use then a `**TEXT` warning message

   ```
   Already in use
   ```

   is displayed

   Note that LEX carries on and starts to use the work file. IT IS EXCEEDINGLY DANGEROUS FOR MORE THAN ONE USER TO BE IN A WORK FILE. On multi user systems if it is in use, exit immediately.

   To minimise such problems each user on multi user systems should be given a personal work file, for example `DAT:LEX:JIM.WRK`.

If LEX was terminated abnormally when the work file was last used then LEX would have been unable to reset the first byte. The work file appears to be still in use when next entering LEX and the above message is again issued. In such cases it is quite safe to continue as normal.

If the named work file is valid then LEX uses that file, restoring the user to where he left the work file. On the other hand if no file of that name exists then one is created.

b) **Temporary work files**

A temporary work file can be stipulated by answering (RETURN) to the `.wrk?` prompt. A temporary work file is then created as described in the section 'Temporary work files selected in LXINST'. The option is used when no specific work file is required as it is automatically deleted when LEX is exited.

LEX can be patched (Chapter 4) to suppress the work file prompt on entry and hence automatically create a temporary work file as though a (RETURN) was issued as above.

c) **Chain to external programs.**

This option does not enter LEX but chains to some external program. The option is, however, only available if supported on the operating system in use.

(ESC)#	enters the patch program (see Chapter 4)
(ESC)S filename	chains to the utility `filename`

3.2.2 Selecting the VDU at Run Time

Whether temporary or permanent files are specified at the `.wrk?` prompt, terminating with a (RETURN) leads to VMF selection below.

Alternatively, if it is terminated with (SPACE) then LEX prompts:

```
vdu?
```

Enter a valid key for a VDU record on the VMF and press (RETURN). This key overrides any key already set in the VMF or stored in the work file.

It is a useful technique when the user needs to run the same LEX system from different types of VDU.

3.2.3 Selecting the VMF

Having selected the work file, LEX now requires a VMF.

If the work file already exists then LEX continues from where it left off during the last session using the same VMF.

On the other hand if a new work file has been created, permanent or temporary, LEX looks for a default VMF. The name and extension of the default VMF can be patched into LEX using LXINST as described in Chapter 4. A primary and secondary device can be defined which LEX searches for the VMF.

Finally if a VMF cannot be found or the specified file does not have the correct access permission LEX prompts for a filename.

```
.vmf?
```

LEX will keep issuing this prompt until a valid VMF filename with the correct access is given, or CTRL c is entered to abort the operation.

The VMF will be opened as single or multi user depending on the value set by LXINST. Once the VMF has been opened this may be changed if the **INST value for single/multi user contradicts this.

As at the .wrk? prompt the file name may be terminated with a SPACE instead of RETURN. The effect is exactly as above in prompting for a vdu?.

The chain sequences ESC#RETURN and ESC$filenameRETURN allowed at the .wrk? prompt may also be used here.

3.2.4 LEX demonstration version

The steps described above complete the LEX start up procedure and LEX should display the main menu. The exception is the first time LEX is run on a machine or if it has not been serialised. In such cases a further stage is prompted by the message below.

```
LEX-11      9A1a (C) ACE 1985

Demonstration version. Only use to test.
Do you want to make LEX live
Yes to serialise. No for demo. Y/N
```

The serialisation process, automatically started by answering `Y` to the above message is described in the next section.

`N` enters the demonstration version of LEX. The demonstration version works exactly as a live version with the following restrictions:

a) On the foot of each menu appears the message:

`Demonstration version. Only use to test.`

b) Records on the VMF can be changed and updated but new records cannot be created.

c) Reorganisation of the VMF is impossible.

d) A random blank column is forced through all printed output.

e) Documents are limited to a maximum size of 99 lines.

3.3 SERIALISATION

Serialisation is a one time only operation which converts a LEX demonstration version into a fully operable live system. It takes just a few minutes and a telephone call, but must be carried out with care, ensuring all instructions are followed exactly.

Note that if major items of hardware are replaced or if the LEX files are refreshed from the release kit then serialisation may be required again.

For serialisation to be carried out the serialisation program and associated text file must be available. These are called `LXSERL.aaa` and `LXSERL.LIT` in the distribution kit.

On running the serialisation option an instruction message something like the following will appear.

```
Thank you for purchasing LEX.

The  software must now be serialised.  The serialisation  process
does not take long but it is  extremely important that you follow
the instructions exactly.

Ring  Ace Microsystems Ltd (01-847 4673) and ask for Installation
(9.30 to 5.30 Mon - Fri)

LEX file name is: <RETURN> to accept. LEX9A1.D86

This is LEX Version Number:-          LEX-86 9A1b (C)ACE 1985

Enter your Company Name (40 maximum): Ace Microsystems Ltd
```

The company name may be changed or a simple RETURN use to accept the current wording. The following extra information is then displayed.

```
Your serial no is:                    099.033.058.285  8601/0902

Your licence no is:                   *GB.001.001.297.718

Please make a careful note of the above details and repeat them to
the  Installation  Controller  exactly  as  they  appear  on  your
terminal

            !DO NOT PROCEED WITHOUT FURTHER NOTICE!
```

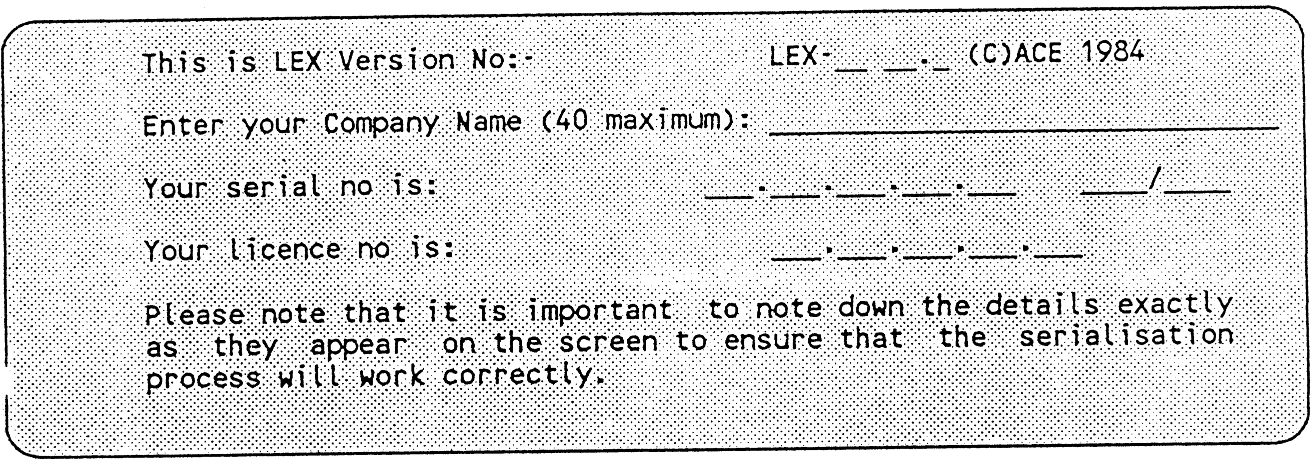
Having noted this information on the form provided (Figure 3.1) telephone the number shown, ask for the Installation Controller and follow the instructions given.

Errors will be dealt with by the Controller until, on successful completion, the following is seen.

```
Serialisation now complete
```

LEX is now serialised as a fully working version.

FIGURE 3.1 - Form for Noting Serialisation Details for LEX



```
This is LEX Version No:-                  LEX-__ __._ (C)ACE 1984

Enter your Company Name (40 maximum): ____________________________________

Your serial no is:                    ___.___.___.___.___    ____/____

Your licence no is:                      ___.___.___.___.___

Please note that it is important  to note down the details exactly
as  they  appear  on the screen to ensure that  the  serialisation
process will work correctly.
```

Chapter 4. SELECTING START UP OPTIONS

The first aspect to customising LEX to suit a computer configuration is to set those factors which cannot be altered when LEX is running, since they affect the actual start up of LEX.

Three aspects are considered in this chapter. The first is the use of an operating systems command file to start up LEX automatically. This is followed by a section on type ahead which may be used in command files but is described separately as it may be used elsewhere. Finally there are a number of patches which can be made to the LEX programs which set such factors as the search patterns to find the location of work files, VMFs, etc.

4.1 COMMAND FILE

On most computer configurations LEX may be entered automatically by setting up an appropriate command file in a format dependent on the operating system in use.

4.2 TYPEAHEAD

Depending on the operating system, typeahead can be used to preselect the work file, VMF, VDU and to type ahead into LEX. Visible equivalents for (RETURN) are used.

In this way statements can be included in a command file to call LEX, choose the work file, select the VMF and VDU type and possibly enter a specific document for editing. For example:

```
LEX /WPOF /FPOF /UVT220 //ETEST.LTR\\\//
```

This will work if E is a menu option, otherwise the following must be used:

```
///ETEST.LTR\\\//
```

Providing typeahead for work file, VMF and VDU type are used in this sequence they need only be preceded by one /. After these, or if these are required in a different order, two / characters precede any true typeahead string.

The typeahead string may contain any valid characters including visible equivalents. Thus in the example above, the \ character represents a (RETURN).

The string is terminated by two / characters.

To preselect a VDU, typeahead can also be input from the keyboard at the .wrk? and .vmf? prompts by pressing (SPACE). Similarly it may be used at the Key______________ prompt and switching between VMFs using the /F menu switch option.

The total length of the type ahead must not exceed 64 characters or the string will be truncated to this length. Any VMF or VDU specified overrides that which would normally be used, i.e. the record stored in the work file or specified in **INST.

If a workfile specified in typeahead does not exist then LEX creates one without prompting.

4.3 PATCHING LEX ENTRY OPTIONS

Some options concerning start up may be patched directly into the LEX files using the installation patch program.

LXINST is the patch program and must have been transferred from the distribution kit during installation if patching is to be accomplished.

Note that LEX has an inbuilt patch program accessible by menu switch option /5. This is provided mainly for correcting errors. It can be used for changing these options but only if the location for the variable is known. As these vary from release to release it is simpler to use LXINST.

4.3.1 Using the Patch Program

LXINST may be entered in three ways depending on your system:

a) On entry into LEX, at the .wrk? or .vmf? reply with (ESC)#(RETURN)

b) From within LEX, chain to the program with the /G menu switch option followed by the filename LXINST.D11, LXINST.D86, LXINST.D68 or LXINST.D32

c) From the operating system prompt, use the program supplied with the distribution kit and the command

```
D LXINST
```

See your installation notes for full details.

LXINST starts by asking for the name of the file to be patched.

The file depends on the version of LEX but usually takes the form LEXnnn.Dxx where nnn is the release number and xx the system for which it is derived. See your installation notes for more details.

LXINST shows the following display:

```
LEX9A1.D86                  LEX-86 9A.1b (C) ACE 1985

Current values are:
1- Multi/Single user (M/S)   S
2- Temp work files   (T/N)   N
3- 1st VMF device            C:
4- 2nd VMF device            E:
5- VMF Name                  LEX9A
6- VMF extension             .VMF
7- Work file device          C:
8- Work file extension       .WRK
9- Device/Acct Chars
A- serialise message         Do you want to make LEX live
B- serialise message - 2     Yes to serialise, No for Demo. Y/N
C- serialise program         D LXSERL

0 to update; * to abandon; Z to reset original values
Which field do you want to change:-
```

At the prompt Which field do you want to change the following replies may be entered:

n Enter the number of the field to be patched. This gives access to the appropriate field which may then be updated with the desired values. Note that this only changes the contents of the table and the file itself is not actually patched at this stage.

* Enter * to abandon the patch leaving the settings as shown on first entering the routine. No patching occurs.

U Updates the file to be patched with the values as shown in the table on display.

Z Resets the table to the values shown on first entering the routine.

The following may also be used:

DELETE or RUBOUT
BACKSPACE
RETURN
LINEFEED

4.3.2 Options to be patched

Most of these options describe the way in which the various LEX files are opened and where default files can be found. This is best explained by use of the example cited above which is for an MS-DOS system.

First LEX notes that the VMF is to be opened in single user mode.

A temporary work file is not required so LEX looks for `C:LEX9A.WRK`. If not found it prompts for the work file name. Next LEX looks for the VMF file. First it tries `C:LEX9A.VMF E:LEX9A.VMF` and if this fails, a prompt for the VMF filename is issued.

Once a valid VMF is identified LEX checks `**INST` for the multi/single user mode of the VMF. If it differs from the mode in which it opened LEX then re-opens it in the correct mode.

If at the above prompts file names are entered without extensions then the default extensions `.WRK` and `.VMF`, are appended to the user input file name.

Full descriptions of each patchable field are given below:

`1- Multi/Single user (M/S)`

If the VMF is to be opened as multi user this switch is set to `M` and if to single user to `S`.

Note that LEX opens in the mode described here but on entering a VMF checks `**INST` to see if the VMF's mode is the same. LEX then reopens in the correct mode if necessary.

`2- Temp work files (T/N)`

When set to `T` LEX overrides all other features and opens a new temporary work file with a unique name. Temporary work files are automatically deleted on exiting LEX and therefore cut and paste buffers and other useful stores cannot be carried over between editing sessions.

`3- 1st VMF device`

In the search pattern for the VMF LEX first tries this device and/or directory.

4- 2nd VMF device

Having failed to locate the VMF on the first device it then checks this one. Only after not finding it here does LEX prompt for a VMF filename.

5- VMF Name

In carrying out a search for the default VMF this is the filename used. It is also used for the default work file name.

6- VMF extension

In carrying out the search for a default VMF this field contains the filename extension. At the .vmf? prompt it is also used as the default extension when one is not included in the filename entered.

7- Work file device

Any work file created, temporary or permanent, will be located on the device specified here. The search for a work file is also conducted on this device.

8- Work file extension

If a filename is entered at the .wrk? prompt without an extension, then the extension specified here is appended.

9- Device/Acct Chars

Up to 15 standard delimiter characters are built into LEX and cannot be used in a file name. These characters may be changed by patching this location. The standard values will vary according to the system used but could be:

```
$[]/\(){}<>!%@#
```

: is also built into LEX but cannot be patched. Care should be taken not to enter any spaces.

A- serialise message
B- serialise message - 2

If, on entering LEX, the start up routine finds that LEX has not been serialised then the message on these two lines is displayed. From here the user may enter the serialisation process (or some other program) by entering Y or enter LEX for demonstration purposes with a N. The standard message may be re-defined here if required.

```
Do you want to serialise

Yes to serialise, No for Demo. Y/N
```

`C- serialise program`

On some systems a direct chain to the serialisation program may be achieved. In this case this field contains the name of the program (LXSERL) and, if necessary, the directory in which it may be found. On systems where the chain is not possible, this field contains brief instructions on how to execute the program.

Chapter 5. SYSTEM CONTROL RECORDS

System control records define the basic working system of LEX.

They are activated on entering a VMF and remain in force unless a new control record is specifically selected. The ability to change these records allows LEX to be customised to suit a particular application including translation to foreign languages.

The keys of system control records all start with two asterisks (**) as do the records containing the prompts and messages described in Chapter 6. The records considered in this chapter are **MONT **INST **PNST **MENU **START and their derivatives.

Note that for LEX to work a VMF must include **MONT **TEXT (Chapter 6) **INST and a VDU control record (Chapter 7). If any of these are missing an appropriate warning is displayed.

5.1 SYSTEM DEFINITION AND DATE - **MONT

```
[**MONT   091092093123124125060036062039034033064060061062058033040041038021
January,\February,\March,\April,\May,\June,\July,
August,\September,\October,\November,\December,
  \ \  \  \  \st\st\nd\rd\th\E\\Y\N\QFTOE\M\E
000000094037095124047038092255000\.\K\:
-JanFebMarAprMayJunJulAugSepOctNovDec\-50\AEIOU\\\\PLAC\HELP\
```

FIGURE 5.1 - EXAMPLE OF **MONT FOR ENGLISH

The **MONT record is the primary system control record and must exist for LEX to enter a VMF. It is mainly concerned with defining the control characters used by LEX but it also defines characters used to reply to specific prompts and sets the format for the date and time.

Once in LEX it is possible to activate another record with the same function as **MONT by using the menu switch option /1. Such other record must have the key **MONTa where a represents some other character or characters.

Figure 5.1 gives an example of **MONT designed for the English language. The following description of each sub-field is based on this example. Recommended changes for use with other languages are given in Table 5.1.

Each field in this record is separated by an actual RETURN at the end of a line or by a backslash \. The backslash is always used regardless of any subsequent redefinition of the visible equivalent of RETURN made in **MONT. Note that the relative position of each field is vital so even an empty or redundant field must still have its RETURN or \ separator.

```
091092093123124125060036062039034033064060061062058033040041038021
```

The first field in **MONT defines the decimal values of characters which are used within LEX for special functions. Other characters are used as visible equivalents in programming and these are also defined here. Note that this field is not a complete list of such characters. Others appear in later fields of **MONT.

If LEX detects the first three characters as 091 then the first 20 sub-fields default to the numbers shown in the example. (i.e. the whole field defaults except for the last two sub-fields shown as 038 and 021. [redacted]

If any non numeric character is found in the string then all characters from that point onwards default as shown.

[redacted]	Start of key
[redacted]	Visible equivalent of RETURN
[redacted]	End of key
[redacted]	Visible equivalent of CTRL H or ←
[redacted]	Visible equivalent of CTRL I or TAB
[redacted]	Visible equivalent of CTRL R or →
[redacted]	Visible equivalent of start of key
[redacted]	Visible equivalent of ESC
[redacted]	Visible equivalent of end of key
[redacted]	Visible equivalent of CTRL
[redacted]	Start and end of literal string. If this field is set to 000, then Australian programming mode is invoked where all characters are treated as literals until a start of programming character is found. An end of programming character then returns to all literals. These characters are set up in the extension of this field in **MONT.

033	!	Visible equivalent of question mark
064	@	Indicator to treat next character as a literal
060	<	When selecting criteria during list processing this is the less than operator.
061	=	When selecting criteria during list processing this is the equals operator.
062	>	When selecting criteria during list processing this is the greater than operator.
058	:	When selecting criteria during list processing this is the contains operator.
033	!	The pause indicator. This is set to 000 if a pause is only required on a question mark.
040	(	Select left bracket operator
041	)	Select right bracket operator

The remaining values do not default if the first is set to 091. They must be set to the desired value.

038	&	This character is used to signify that a line longer than the physical screen width is stored in a document. The character is removed when stored on disk and only appears in the final column on screen. Only one continuation character may be used on a line and the total length of a line may not exceed 225 characters.
021	^	This character, normally the CTRL U or ↑, emulates an ESC E when at a menu. The effect of this command at the top menu, usually **MENU, is to leave LEX. Setting the parity bit to this character, i.e. add 128 to it's value (giving 149 in this example) prevents this means of exiting LEX.

TABLE 5.1 - FOREIGN LANGUAGE SYSTEM CHARACTERS

German/Scandinavian		French		Swiss	
060	<	060	<	037	%
059	;	059	;	059	;
062	>	062	>	062	=
000		000		000	
000		000		000	
000		000		000	
040	(	040	(	040	(
036	$	036	$	036	$
058	:	058	:	058	:
039	'	039	'	039	'
034	"	034	"	034	"
033	!	033	!	033	!
064	@	038	&	038	&
060	<	060	<	060	<
061	=	061	=	061	=
062	>	062	>	062	>
058	:	058	:	058	:
000		000		000	
040	(	040	(	040	(
041	)	041	)	041	)
038	&	038	&	038	&
021	^	021	^	021	^

```
January,\February,\March,\April,\May,\June,\July,
August,\September,\October,\November,\December,
```

These fields define the format of the month when the full date sequence ESC Y is used. The range is from January to December.

```
  \  \  \  \ \st\st\nd\rd\th
```

Each of the next ten fields must be two characters long. They define the character suffix and prefix which may be applied to the day. Consider the full form of the date in English as:

```
23rd December, 1985
```

The ten fields provide the following information.

Prefix if day is 01	Usually 2 spaces
Prefix if day is 21 or 31	Usually 2 spaces
Prefix if day is 02	Usually 2 spaces
Prefix if day is 03	Usually 2 spaces
Prefix for other day	Usually 2 spaces
Suffix if day is 01	Usually st
Suffix if day is 21 or 31	Usually st
Suffix if day is 02	Usually nd
Suffix if day is 03	Usually rd
Suffix for other day	Usually th

Note that strange results may occur if these fields are anything but two characters in size. If prefixes or suffixes are not required they should be blank. LEX removes leading spaces from the date and reduces consecutive spaces within the date to one space.

E

This field allows other styles of the date to be used. For example:

M	American	November, 2nd 1985
Y	Swedish	1985 November, 2nd
(other)	English	2nd November, 1985

For English this field is usually zero length.

-

Field not used and usually zero length.

Y\N

These two fields provide the valid responses whenever a query prompt is shown. The literal prompt usually associated with it is Answer (Y/N) and is found in **TEXT. In this example the response Y is used to indicate the YES response and N the NO. If they are missing Y and N are assumed respectively. Only the first character in each field is used.

The N is also used by the No query option when copying records from one VMF to another with the /4 menu switch option.

QFTOE

The Q is used at the Select question and when copying records from one VMF to another. It defines the character which indicates the query option i.e. it causes a prompt to be issued before each record is copied.

F and T are used when copying records between VMFs to indicate whether copying FROM or TO. They are associated with the **TEXT2 message

```
Copy From current file (F) or To current file (T) (F/T)
```

The O and E are used when printing a document with alternate page titles. They indicate the printing of ODD or EVEN pages respectively. They are associated with the **TEXTP message

Print odd or even numbered pages (O/E)

M

This field is used when re-justifying a paragraph. If set to S it assumes that a single space is required at the end of each sentence after the dot. Any other value and two spaces are automatically appended.

E

This field defines the rules to be applied to automatic hyphenation when re-justifying a paragraph. The following may be used.

E	English hyphenation rules
G	German hyphenation rules
N	Scandinavian hyphenation rules
(other)	Hyphenation disabled

000000094037095124047038092255255

This field is an extension of the first field and the notes given there are applicable.

000		Start of Australian programming mode. This character is used to turn on the interpretation of characters as visible equivalents. The value 096 (') is usually used.
000		End of Australian programming mode. From this character on all characters are treated as literals. The value 039 (') is usually used.
094	^	Visible equivalent of (CTRL)U or (↑).
037	%	Visible equivalent of (CTRL)J or (↓).
095	_	The standard paint character, usually underline. It should only be changed in exceptional circumstances as it is essential to most existing VMF routines.
124	|	The visible equivalent of (RETURN) for use in system records only. It allows a \ to appear in a system record. The \ is used in device definition under MS-DOS, so

		using | allows the \ to be included as text. It does not apply to the **MONT record itself. It is normally set to 092.
047	/	Used in the enhanced ESC F, ESC G and ESC -G sequences to represent a break between strings. i.e. aaa/bbb where the first occurrence of aaa OR bbb is found.
038	&	Used in the enhanced ESC F, ESC G and ESC -G sequences to represent a break between strings, i.e. aaa&bbb where the first occurrence of aaa AND bbb on the same screen is found.
092	\	Used in the enhanced ESC F, ESC G and ESC -G sequences to represent a break between strings, i.e. aaa\nn where the first occurrence of aaa in column nn is found.
255		This is the decimal ASCII value of the character used to flag a following attribute line in the input file. It should only be changed in exceptional circumstances. A visible terminator could cause problems by being used in the text.
000		Not used.

\

Field not used and usually zero length.

K

During list processing it is possible to delete the selected records by appending /K/K at the select line. The character which performs this kill option, K in this example, is defined in this field.

:

This character is used as the delimiter in the time. e.g.

20:45:09

`-JanFebMarAprMayJunJulAugSepOctNovDec`

This field sets the format for the short date obtained from the sequence ESC T. The first character defines the delimiter which separates the day from the month and the year. The next 36 characters represent the months, January to December, with three characters for each. In the above case a typical date would be:

```
31-Oct-85
```

`-50`

This field allows different years to be specified in the full form of the date ESC Y. It may be positive or negative and is added to the year.

It is normally set to zero length when the system date is being used as today's date but may contain a value when LEX is being used in countries not using the Julian calendar, e.g. Buddhist.

`AEIOU`

These characters are used as vowels for hyphenation purposes. Upper case only is required and the default is as shown. If using a 7-bit VDU and \ is to be defined as a vowel then its lower case equivalent | may be defined.

`\\\\`

Unused.

`PLAC`

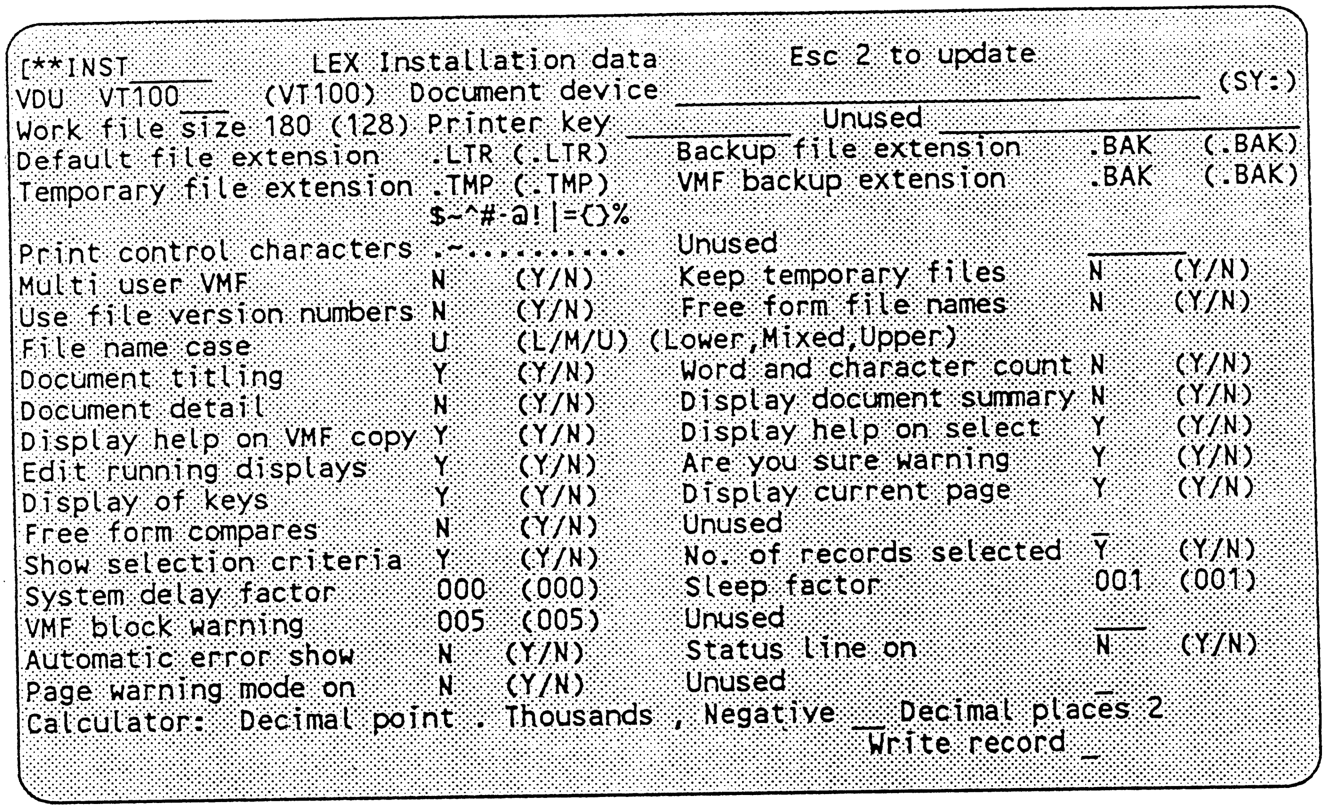
These characters are used in the status line to describe the position of the cursor when the option was called.

P	Page number
L	Line number on the current page
A	Actual line number in document
C	Column position

`HELP`

This field defines the string to be entered at the select prompt in the report generator to obtain the help screen. Note that the help screen may be displayed automatically if enabled in **INST.

5.2 INSTALLATION DETAILS - *INST



```
[**INST_____        LEX Installation data         Esc 2 to update
VDU  VT100___   (VT100)  Document device ________________________________ (SY:)
Work file size 180 (128) Printer key __________  Unused ______________________
Default file extension    .LTR (.LTR)   Backup file extension      .BAK   (.BAK)
Temporary file extension .TMP (.TMP)    VMF backup extension       .BAK   (.BAK)
                          $~^#-@!|={}%
Print control characters .~..........  Unused                     ______
Multi user VMF            N    (Y/N)    Keep temporary files       N      (Y/N)
Use file version numbers  N    (Y/N)    Free form file names       N      (Y/N)
File name case            U    (L/M/U) (Lower,Mixed,Upper)
Document titling          Y    (Y/N)    Word and character count  N      (Y/N)
Document detail           N    (Y/N)    Display document summary  N      (Y/N)
Display help on VMF copy  Y    (Y/N)    Display help on select    Y      (Y/N)
Edit running displays     Y    (Y/N)    Are you sure warning      Y      (Y/N)
Display of keys           Y    (Y/N)    Display current page      Y      (Y/N)
Free form compares        N    (Y/N)    Unused                     _
Show selection criteria   Y    (Y/N)    No. of records selected   Y      (Y/N)
System delay factor       000  (000)    Sleep factor               001    (001)
VMF block warning         005  (005)    Unused                     ___
Automatic error show      N   (Y/N)     Status line on             N      (Y/N)
Page warning mode on      N   (Y/N)     Unused                     _
Calculator:  Decimal point . Thousands , Negative __ Decimal places 2
                                                  Write record _
```

FIGURE 5.2 - EXAMPLE OF **INST

Basic installation details and the various LEX options are held in a system control record **INST. Unlike **MONT it is not divided into fields but is a string record which is best viewed by use of a form. *INSTFORM is provided for this purpose and is shown in Figure 5.2.

The various fields within the form and their meaning are considered in this section.

Note that **INST initialises LEX on entering the VMF. Subsequent changes to the record are not noticed by LEX until re-entering the VMF or using the menu switch option /1 to reset **INST. This is done by:

```
/1(RETURN)
```

Multiple **INST records may be held in a VMF but the default is **INST and it is this record that is used on first entering the VMF. Extra **INST records are particularly useful on multi user systems and take the form **INSTaaaa. For example **INSTSLOW might be set up for slow terminals reducing the amount of information sent to the screen to a minimum. Records like **INSTSLOW are set up by:

```
/1**INSTSLOW(RETURN)
```

or, at startup, by including in typeahead the string //1**INSTSLOW. See Chapter 4 for information on typeahead.

VDU ________ (VT100)

This is the key of a *VDUFORM record describing the VDU type that is to be assumed when first entering the VMF. The record type) of the VDU record must not be entered. If an existing work file holds a different VDU type key or if type ahead is used then this field is ignored. The default is shown in brackets.

Document Device ______________________________ (SY:)

This is the default device that documents are stored on. The default is shown in brackets. If the Free Form file name field is set to Y then these details are displayed whenever the document name prompt appears. If set to N then they are assumed but the user is not made aware of them.

Work File Size ___ (128)

This defines the minimum work file size required when this VMF is used. The larger the work file the further it is possible to scroll back through a document. To calculate the optimum work file size, take the size of the largest document anticipated, double it and add 70 blocks.

Printer key __________

The name of **PNST record to be used may be specified here. Do not enter the ** as they are assumed. The default is **PNST itself.

Unused ____________________.

Default file extension ___ (.LTR)

The extension applied to document names if one is not specified at the Document name prompt. If left blank then no extension is applied.

Backup file extension ___ (.BAK)

The extension applied to backups. If left blank no backups of documents are kept.

Temporary file extension ___ (.TMP)

The extension applied to the name of temporary work files. (See Chapter 3 for details on temporary work files). It is also used for temporary edit files. If left blank it defaults to .TMP.

`VMF backup extension ___ (.BAK)`

When reorganising a VMF it is possible to rename the new VMF and go straight into it. In such cases the old VMF is renamed with this extension.

```
                            $~^#@!|=()%
Print control characters ___________
```

A .C line in a document may be included to enable certain characters to represent special print time instructions such as paragraph numbering. This field sets the default values used unless a specific .C line is encountered.

The characters shown above this field are those suggested for normal usage in a document and are enabled by including them in the appropriate column of this field. A . (dot) disables a feature. Print time instructions are described in Appendix F.

`Unused _____`

`Multi user VMF          _    (Y/N)`

If set to Y this VMF is opened in multi user mode whereas if set to N it is opened in single user mode. If left blank the default mode set by LXINST is used (See Chapter 3).

`Keep temporary files    _    (Y/N)`

When set to Y temporary edit files are retained after abandoning with ESC Z.

`Use file version numbers _ (Y/N)`

This feature may only be used on operating systems that support version numbering (e.g. VAX/VMS or RSX). If set to Y then every version of a document is kept and allocated a new version number.

`Free form file names _ (Y/N)`

When set to Y the details shown in the Document device above are displayed whenever the document name prompt appears. The cursor may be moved back over these details to change them thereby allowing access to other accounts/devices. If an N is used here the document device details are fixed and transparent to the user.

```
File name case              _     (L/M/U)      (Lower,Mixes,Upper)
```

Some operating systems have lower case file names as a convention and others allow mixed lower and upper case names. Use of U or L causes LEX to translate document name characters into lower or upper case respectively. M leaves them as they are entered. The latter is not advisable unless essential (for example in Arabic versions) as the case of each file name must also be remembered. e.g FRED.DOC, FRED.doc, FrEd.DOc etc.

Note that the document index key is always translated to upper case or the range described by *CASEFORM.

```
Document titling            _     (Y/N)
```

A Y in this field enables the document indexing facility and causes the prompt:

```
Document title :________________________
```

to appear after the document name. Up to 40 characters including spaces are available for information to be stored on the document index. The prompt is taken from **TEXT.

```
Word and character count _ (Y/N)
```

If set to Y a word and character count of the document is kept. This feature only applies if the document titling field above has been set to Y to enable the document index. This feature will slow LEX down.

```
Document detail _ (Y/N)
```

When set to Y this field allows a further 40 characters of information to be stored on the document index. After the document title a further **TEXT prompt appears:

```
Document detail : ______________________
```

This feature only applies if the Document titling field is set to Y to enable the document index.

```
Display document summary _ (Y/N)
```

When set to Y it enables a document summary to follow the Document name prompt, e.g.

```
Created Edited vsn Access vsn Print vsn LstTim TotTim Lines  Words   Chars
03Nov85 03Nov85 3 03Nov85  3 03Nov85  1 101519 012025 000560 002477 023333
```

This feature only applies if the Document titling field is set to Y to enable the document index.

`Display help on VMF copy _     (Y/N)`

A Y in this field causes the help screen stored in *HELPV to be displayed automatically when reorganising a VMF or when copying records from one VMF to another.

`Display help on select _ (Y/N)`

When set to Y this causes the help screen associated with the output format in list processing or mailshot to be automatically displayed at the Select prompt. The default screen if not set in the output format record is *HELPS.

`Edit running displays _ (Y/N)`

When exiting or re-editing a document a running display appears at the bottom of the page with a line and page count as the document is written away. Setting this field to N disables the count.

`Are you sure warning _ (Y/N)`

If set to Y an ESC Z does not immediately abandon the document but first prompts the **TEXT message

`Abandon edit - Are you sure?`

This requires a Y response before the document is abandoned. Any other response causes editing of the document to continue.

`Display of keys _ (Y/N)`

During a list processing run a running display of the keys is shown as the search takes place. Setting this field to N disables this feature.

`Display current page _ (Y/N)`

Whilst printing, the current page being printed is displayed. Setting this field to N disables this feature.

`Free form compares        _     (Y/N)`

This field is used in the report generator. A Y results in fields with a numeric formatting character being set for a numeric rather than alphabetic comparison. See Chapter 13.

`Unused                        _`

```
Show selection criteria _      (Y/N)
```

If set to Y the selection criteria used is shown at the end of a list processing run. It is preceded by a **TEXTP message. For example:

```
Select criteria
NAME=JONES AND TOWN:GLASGOW
```

```
No. of records selected  _      (Y/N)
```

If set to Y the number of records selected is shown at the end of a list processing run. A **TEXT message precedes two numbers. The first is the number of records meeting the selection criteria and the second the number of records tested. For example:

```
No. of records selected   0005 / 0156
```

```
System delay factor       ___   (000)
```

This controls the delay factor and is added to that specified in *VDUFORM.

```
Sleep factor              ___  (001)
```

The sleep factor determines the length of time in seconds for which non-edit error messages such as Missing Menu are displayed before LEX returns to the previous prompt or menu.

```
VMF block warning         ___   (005)
```

If the number of free blocks in the VMF drops below this number LEX sends a (BELL) to the VDU and sets ERROR-NF each time an attempt is made to add new records to the VMF. A **TEXT message is also shown at the bottom of every menu in the form:

```
WARNING.  VMF is nearly full.  Reorganise now.
```

```
Unused                    ___
```

```
Automatic error show      _     (Y/N)
```

If set to Y LEX displays edit mode error messages automatically as if (ESC)H had been typed. If a status line is set up then the error may be displayed there, otherwise it is shown at the cursor position. Entering (RETURN) will remove the display and allow editing to continue.

```
Status line on         _     (Y/N)
```

If set to Y the status line is displayed continuously during editing, showing line, page and column numbers. This only applies to VDUs with a status line or scroll region and the appropriate escape sequences must be set in *VDUFORM3. Note that this feature will slow LEX down.

```
Page warning mode on   _     (Y/N)
```

If set to Y LEX will send a (BELL) and set ERROR-EP as a warning when reaching the bottom of a page. It is sent on every (RETURN) and (LINEFEED) below the point where LEX would put an automatic page break at print time.

```
Unused                 _
```

```
Calculator:
Decimal point _
```

This defines the character to be used as the decimal point in the calculator and may be . (dot) or , (comma). If left blank the default is . (dot).

```
Calculator:
Thousands     _
```

This defines the character to be used in the display of thousands in the calculator. For example , may be used in English to give 1,000. If the field is left blank then the number is left unchanged, i.e. 1000. If spaces are required use the print time visible equivalent of space.

```
Calculator:
Negative      __
```

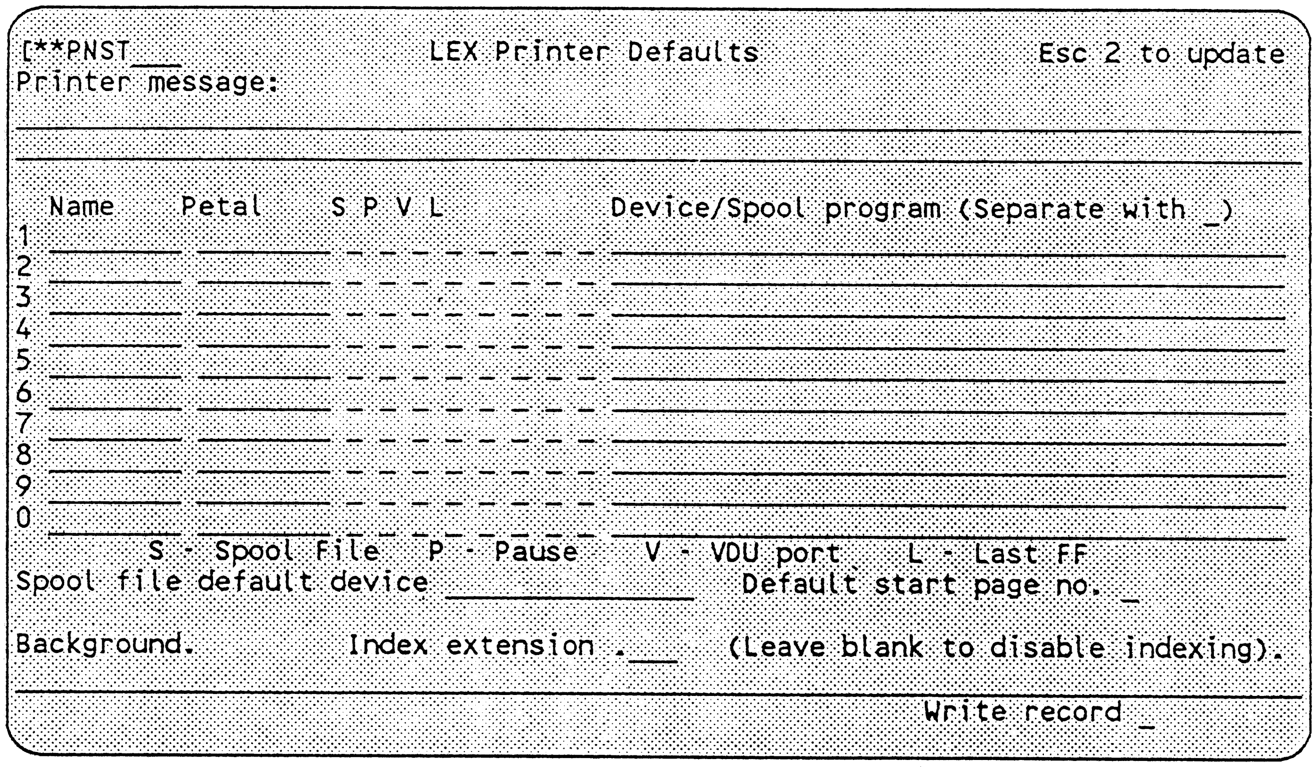
Putting brackets in this field, i.e. () causes negative numbers to be bracketed. If the field is empty then negative numbers are preceded by the minus, as in -45.

```
Calculator:
Decimal places _
```

This defines the default number of decimal places when the calculator is entered using (ESC)+. If the field is empty no decimal places are assumed.

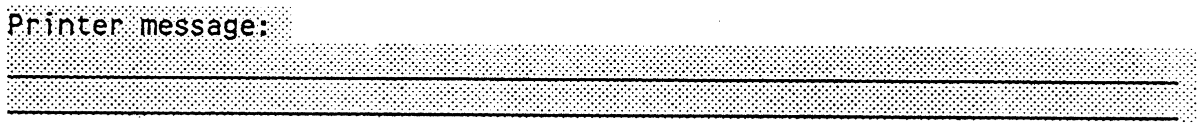
5.3 PRINTER INSTALLATION DETAILS - **PNST

Like the installation details, printer installation details are loaded into the system control record **PNST by using a form. The form is *PNSTFORM as shown in Figure 5.3.



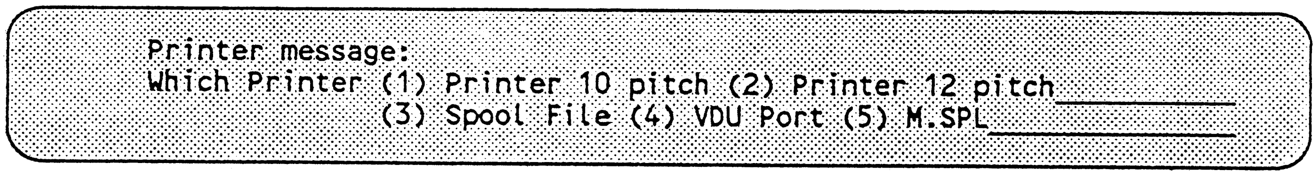
```
[**PNST___              LEX Printer Defaults                      Esc 2 to update
Printer message:
_______________________________________________________________________________
_______________________________________________________________________________
  Name     Petal      S P V L           Device/Spool program (Separate with _)
1 _______  _______  _ _ _ _ _ _ _ _ _  _______________________________________
2 _______  _______  _ _ _ _ _ _ _ _ _  _______________________________________
3 _______  _______  _ _ _ _ _ _ _ _ _  _______________________________________
4 _______  _______  _ _ _ _ _ _ _ _ _  _______________________________________
5 _______  _______  _ _ _ _ _ _ _ _ _  _______________________________________
6 _______  _______  _ _ _ _ _ _ _ _ _  _______________________________________
7 _______  _______  _ _ _ _ _ _ _ _ _  _______________________________________
8 _______  _______  _ _ _ _ _ _ _ _ _  _______________________________________
9 _______  _______  _ _ _ _ _ _ _ _ _  _______________________________________
0 _______  _______  _ _ _ _ _ _ _ _ _  _______________________________________
      S - Spool File   P - Pause     V - VDU port     L - Last FF
Spool file default device ______________     Default start page no. _

Background.          Index extension .___   (Leave blank to disable indexing).
_______________________________________________________________________________
                                                      Write record _
```

FIGURE 5.3 - EXAMPLE OF **PNST



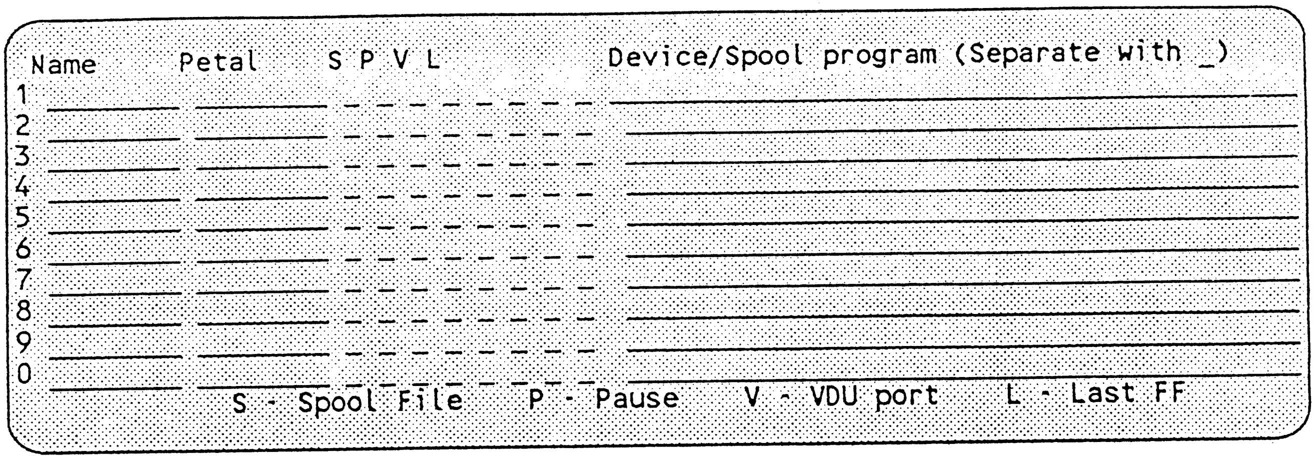
```
Printer message:
_______________________________________________________________
_______________________________________________________________
```

When the menu switch option to print a document is selected and the document name has been entered then this message is displayed. The message may contain up to 156 characters in free format describing up to 10 printers which have been defined in the next part of the record. For example:



```
Printer message:
Which Printer (1) Printer 10 pitch (2) Printer 12 pitch__________
                (3) Spool File (4) VDU Port (5) M.SPL____________
```

At print time the user replies with a single numeric digit corresponding to the printer required.



```
Name      Petal     S P V L          Device/Spool program (Separate with _)
1 _______ _______ _ _ _ _ _ _ _ _ _ _______________________________________
2 _______ _______ _ _ _ _ _ _ _ _ _ _______________________________________
3 _______ _______ _ _ _ _ _ _ _ _ _ _______________________________________
4 _______ _______ _ _ _ _ _ _ _ _ _ _______________________________________
5 _______ _______ _ _ _ _ _ _ _ _ _ _______________________________________
6 _______ _______ _ _ _ _ _ _ _ _ _ _______________________________________
7 _______ _______ _ _ _ _ _ _ _ _ _ _______________________________________
8 _______ _______ _ _ _ _ _ _ _ _ _ _______________________________________
9 _______ _______ _ _ _ _ _ _ _ _ _ _______________________________________
0 _______ _______ _ _ _ _ _ _ _ _ _ _______________________________________
            S - Spool File    P - Pause    V - VDU port    L - Last FF
```

Up to ten separate printers may be defined in this part of the form. Many of the fields may be omitted and defaults will be assumed.

`Device/Spool Program (Separate with _)`

The separator entered here is used to separate the name of the spool program (if entered) from the device. The separator must be a character which is not otherwise used in the Device/Spool program field.

`Name`

This contains the key of the record which defines the printer escape sequences as set up using *PRTFORM. (See Chapter 9) The *$ which appears at the start of the key of such records is supplied by LEX and must not be entered here. This field is optional. If left blank the defaults shown in brackets on the right hand side of *PRTFORM apply for functions such as form feed, half line feed, escape, etc.

`Petal`

This contains the key of the record which defines the petal sequences as set up using *PETFORM. (See Chapter 9). The *# which appears at the start of the key of such records is supplied by LEX and must not be entered here. This field is optional.

`S - Spool file prompt`

This field may be set to Y, N or _.

An entry of Y indicates that LEX is to prompt for a spool file name (using a literal defined in **TEXTP) to receive printed output. N indicates that no prompt is required and output should be directed to the device or file specified under Device/Spool Program. _ causes the default value of N to be used.

P _ Pause message

This field may be set to P, Y, N, or _.

P or _ indicates that the Pause after each page? prompt (defined in **TEXTP) should be displayed after the Which Printer message. The user reply to this prompt may be Y if printing is to halt before each new page or N if the printing is to be continuous.

If Y is entered then the message is not displayed and printing will halt before each page. If N is entered then the message is not displayed and printing is continuous.

V - VDU port

This field may be set to Y or _.

Y indicates that printing is to be performed through a printer port on the VDU. Any screen displays during printing are suppressed so that the printed output is not corrupted. Note that the Number from, Print from and End Print prompts are displayed when a .T line is encountered. Therefore the .T must appear at the start of the document otherwise the prompts will also appear on the printed output.

_ indicates that printing is not to be performed through a printer port.

L - form feed at end of document

This field may be set to Y, N or _.

Y or _ indicates that LEX is to perform a form feed at the end of each document printed. N indicates that there is no form feed at the end of a document.

Device/Spool Program

This field defines the destination of the printed output. It may be directly to a device (e.g. LP:) or to a file. An entry must be made in this field.

Optionally the entry may be followed immediately by a separator character and the name of a spool program to be chained, for example:

```
M.SPL\SPOOL.PRG /parameters
```

```
Spool file default device ______________
```

If an entry is made in this field then the entry must be a valid device. It is added as a prefix to any spool file names typed in by the user.

```
Default start page no. _
```

The value to be used for the default for the page number on the first page of output should be entered here. It will usually be 1.

```
Background.   Index extension .___  (Leave blank to disable indexing).
______________________________________________________________________
```

If the automatic index feature is required then the extension of the file to contain the index is entered here. The file is created at the same time as a document is printed and includes all entries marked in the document for inclusion in the index. (See Appendix G and the eighth character of the .C line). The file created will have a file name of the document being printed with the extension entered here.

If left blank then automatic indexing is disabled.

The background to be used for printing the index is also entered here. The text extracted from the file is placed over the first part of the background and the page number on which the text appears placed at the end of it. For example if the following entries are made:

```
Background.  Index extension NDX (Leave blank to disable indexing)
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  ____________
```

then the index will appear as:

```
9.6  Setting up and executing a program. . . .    4
     9.6.1 Programs in keystores. . . . . .        5
```

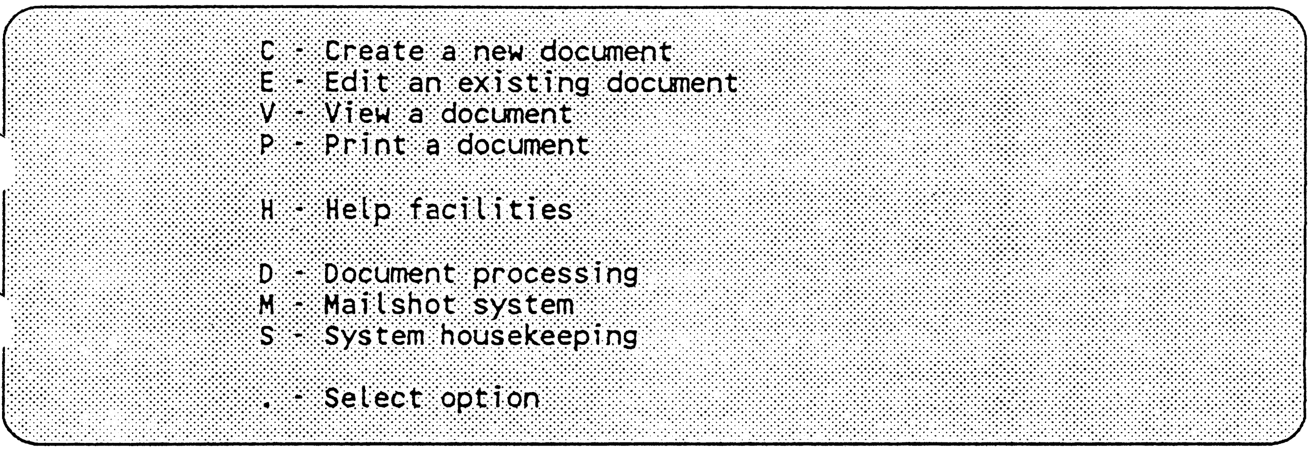
Any text which exceeds the length of the line is truncated.

5.4 MAIN MENU - **MENU

**MENU is usually the first menu called up on the system when LEX is initialised. Any menu may be specified as the first menu by setting the first menu field in the VDU specification record *VDUFORM. If the specified menu is not found on the VMF, then LEX looks for **MENU as the default.

It is suggested that this record should always be used as the first menu even if the layout is altered to suit a particular requirement. If it is not used as the first menu, it should still be present on the file so that LEX has something to work from should the specified first menu be missing.

A typical **MENU is the one shown below.



```
C - Create a new document
E - Edit an existing document
V - View a document
P - Print a document

H - Help facilities

D - Document processing
M - Mailshot system
S - System housekeeping

. - Select option
```

The use of menus is described in the User Manual and their creation and modification in Chapter 10 of this manual.

**WIDE is a special case of a top menu. It is the usual name given to the first menu called when using a VDU capable of 132 column wide mode. **MENU is still the default when all else fails.

5.5 LEX RE-ENTRY CONDITIONS - **START

Normally LEX enters the VMF at the main menu. If a system record with the name **START exists then it is activated first. LEX automatically enters playground and activates shutup mode with an implicit ESC SU. **START is then treated as a programming record. See Chapter 11 for details on LEX programming.

Note that it is necessary to include a $-SU in the program to unset shut up mode before returning to a menu. If this is not done the menu remains invisible. Alternatively from an invisible menu a /2 menu switch option re-sets the screen.

Chapter 6. PROMPTS, MESSAGES AND WARNINGS

All prompts, messages and warnings used by LEX may be customised to suit a particular application. Messages and errors which occur during the execution of menu switch options are generally held in the system records **TEXT, **TEXT2 and **TEXTP. Other errors and warnings caused when using escape sequences or control functions are held as records on the VMF. This Chapter considers these messages using the issue VMF as an example.

The system records **TEXT, **TEXT2 and **TEXTP contain the messages and consist of a set of fields separated by an implicit (RETURN) or by its visible equivalent. Usually the visible equivalent is \ but it may be changed in **MONT. The relative position of each field is vital so an empty or redundant field must still have its (RETURN) or \ separator.

Blanks within the fields are interpreted as such but a (RETURN) is automatically appended after the last non space character on a line. To force blanks after the last character use _ i.e. underline.

Two other characters have special meanings within these records and are independent of any settings in **MONT. They are:

! The exclamation mark is translated as a question mark ?

^ The up arrow symbol is translated to a space and causes a (BELL) to be sent to the VDU

FIGURE 6.1 - EXAMPLE OF **TEXT

```
[**text     *GB001001 - ACE Microsystems Ltd.
Document Name  : \Document Title : \Document Detail:_
Created Edited vsn Access vsn Print  vsn LstTim TotTim  Lines    Words   Chars
Old\New\Input\Output\View Mode\New Work
WARNING.  VMF nearly full, reorganise now. Blocks free^
No write access to VMF^\Work file too small\Not a work file\Already in use
File name error^\ already exists^\ doesn't exist^\You can't edit backup files
Missing menu\Invalid ruler in document format\Index locked\Key of current VDU_
Key \Answer HELP at selection prompt to display the help screen
Press any key for next screen^\LEX is copying\Deleted
- are you sure ?\(Y/N) \selected.  Please wait ..\Searching for -_
 end of document\string \Delay\Status \Error is \Repeat_
Hyphened \Key is \Foreign File is \Mask is \Merge_\Write out \advance to_
until \just \to end of \Find \Search for \Search backwards for_
Open (overwrite)- \Open (create)- \QS Key\QT command
New extension: \Attributes currently on \Exit\Abandon edit\Re-edit
Re-edit and return\\
```

6.1 GENERAL MESSAGES AND PROMPTS - **TEXT

**TEXT contains all the prompts and messages associated with escape sequences and the entry into LEX. Many of the menu switch options also take their prompts and messages from here.

The fields in the example of **TEXT at Figure 6.1 are described below.

`*GB001001 - Ace Microsystems Ltd.`

The first field holds the licence details and cannot be changed by the user.

`Document Name :`

This field is displayed when performing document handling activities such as creating, editing or printing. It is a prompt for a valid file name.

`Document Title :`

This field is a prompt for a document title used by the LEX internal indexing system. It appears after the Document Name prompt if enabled in **INST.

`Document Detail:`

This field is a prompt for further document information used by the LEX internal indexing system. It is only displayed if enabled in **INST and appears after replying to the Document Name prompt.

`Created Edited vsn Access vsn Print vsn LstTim TotTim Lines Words Chars`

After replying to the Document Name prompt, LEX looks for the file name and on finding it displays this heading with the information from the LEX internal document index.

`Old`

When using menu switch options for renaming documents, this field is used in conjunction with the Document Name to define the original version of the document.

`New`

When using menu switch options for renaming documents, this field is used in conjunction with the Document Name to define the name of the final version of the document.

`Input`

When using menu switch options for copying documents this field is used in conjunction with the Document Name to define the original version of the document.

`Output`

When using menu switch options for copying documents this field is used in conjunction with the Document Name to define the name of a new file to be created as a result of the option.

`View Mode`

When document handling using the /V menu switch option this heading is used above the Document Name prompt to show that view mode is being entered and that changes to the text will not update the document.

`New Work`

Menu switch option /Z is used to change to a new work file. This heading appears above the Document Name prompt to show that a new work filename is being opened.

`WARNING VMF nearly full, reorganise now Blocks free^`

This message indicates that the VMF file is nearly full and needs reorganising. It is issued when the number of blocks free falls below the level set in **INST and appears at the foot of every menu together with the number of blocks free.

`No write access to VMF^`

LEX displays this message when entering a VMF in single user mode if the VMF is already in use. The user is not prevented from entering the VMF or reading records but cannot write records on that VMF file.

`Work file too small`

This message appears if the size for the work file as specified in **INST is too small for LEX to use.

`Not a work file`

If LEX attempts to open a work file that is not in the correct format then LEX prompts with this error and awaits the entry of another filename. If the file is in the correct format then the file is entered while if there is no file of that name then LEX creates one. Note that LEX can be patched to use temporary work files, in which case this message is redundant.

`Already in use`

If, on start up an attempt is made to enter a work file that is currently in use (or was not closed properly) then this message appears temporarily before continuing to enter the work file. IT IS DANGEROUS FOR MORE THAN ONE USER TO BE IN THE SAME WORK FILE AS UNPREDICTABLE RESULTS WILL OCCUR FOR BOTH USERS AND WORK WILL BE LOST.

`File name error^`

This field shows the message that appears if an illegal file name is entered at a Document Name prompt. It is also generated if an invalid filename is specified during print merging.

`already exists^`

It is not possible to create a document that already exists. Therefore if an attempt to create a document with the same name as an existing document using the /C menu switch option or copying into an existing document then the filename and this message appears. LEX then waits for a new filename to be entered.

`doesn't exist^`

It is not possible to copy, edit or view a document that does not exist. In these cases the filename and this message appears. LEX then waits for a new filename to be entered.

`You can't edit backup files`

LEX will not allow editing of files with the backup file extension as defined in **INST. Therefore an attempt to use menu switch option /E into a file with such an extension results in this message. Note that the view option /V is legal.

`Missing menu`

When LEX cannot find the specified menu from the /M menu switch option or from the relevant VDUFORM entry then it displays this message. LEX attempts to recover by returning to the previous menu or failing that by returning to **MENU. If all else fails, it waits in a pseudo-menu state to allow the entry of a menu switch option.

`Invalid ruler in document format`

This appears whenever the document printing details for the file being edited contain an error in the rulers. The problem may be encountered with Ruler 1 when attempting to edit a wide document in narrow mode or vice versa. On detecting the error LEX uses printing details from the default document extension as defined in **INST.

`Index locked`

This message is used in a multi user VMF if another user has already accessed and locked the document index. The document index may therefore not be updated on completion of the work in that document.

`Key of current VDU_`

This prompt is displayed when the menu switch option /U is entered to change to a different VDU type. It is also used as a message in the LEX Status to introduce the key of the current VDU record. See menu switch option /W.

Note the _ which is used to force a space at the end of the literal. It is only necessary in this example where the field occurs at the end of a line and LEX inserts an implicit (RETURN) after the last character.

`Key`

When a menu switch option is selected where a mandatory system record key is required (such as /M and /S) this prompt appears. The prompt automatically applies the first * to the key name so this must not be entered.

`Answer HELP at selection prompt to display the help screen`

The contents of this field appear at the top of the screen whenever a list processing or mailshot selection is required. It is possible to display the HELP message automatically by enabling the option in **INST. In such circumstances the message is suppressed.

`Press any key for next screen^`

This message appears at the top of the screen if viewing list processing records when the screen is full. It is also used when LEX needs to pause to allow the user to assimilate the data on the screen before proceeding. Examples of its use are after the /T and /W menu switch options and on returning to LEX from an external program using /B.

`LEX is copying`

This field is displayed when LEX is copying a file using menu switch option /J. The file is copied block by block and a number is appended to this message indicating the number of blocks copied so far.

`Deleted`

When using the menu switch option /K to delete files, LEX indicates that it has performed the operation by displaying the file name and Deleted.

`- are you sure ?`

If the Are you sure warning is enabled in **INST, an (ESC)Z (ESC)CA issues this prompt as a precaution prior to abandoning an edit or deleting to the end of a document. It requires a Y response to continue. (See next **TEXT field)

`(Y/N)`

The prompt is used in conjunction with conditional testing query options, i.e. (ESC)CTQ. A reply of Y or N dictates the direction that a program takes. It also follows the Are you sure prompt above when enabled in **INST.

`selected. Please wait ..`

On exiting from a document this message is displayed as the file is being written to disk. Included on the screen is a **TEXT literal indicating the form of exit chosen, the name of the document and running counters of line and page number being written. The various exit options are listed below together with the sequence that calls them:

Exit	Update document; return to menu.	(ESC)E
Abandon edit	Return to menu without updating document.	(ESC)Z
Re-edit	Update document; re-enter at top of document.	(ESC)V
Re-edit and return	Update document; re-enter at calling point.	(ESC)QX

`Searching for -_`

When a search option, (ESC)F, (ESC)G or (ESC)-G cannot find the required string on the current screen, the screen is blanked and this message is displayed whilst the search continues through the document. It is followed by the text string that has been specified, the name of the document and running counters of line and page number as the search progresses.

Note the _ which is used to force a space at the end of the literal. It is only necessary in this example where the field occurs at the end of a line and LEX inserts an implicit (RETURN) after the last character.

`end of document`

During a search for the end of a document, (ESC)QE, the normal search message Searching for is followed by this literal.

`string`

This field is used as a prompt for a string input where a dedicated message is not available elsewhere in **TEXT. For example it is used with the merge until command (ESC)MU to prompt for the string to be searched for.

Delay

This field is used when displaying the delay factor in response to the command ESC SD.

Status -

When ESC L and ESC RL is used this field introduces the line containing the status details. These details include the filename, line and column of the cursor, line spacing etc. Full details are in Appendix G.

Error is

The last error detected may be viewed by the sequence ESC H. This field is used to prefix the error message displayed.

Repeat_

On issuing the ESC X of a repeat sequence this message prompts, in background mode, for the string containing the repeat details.

Note the _ which is used to force a space after the word Repeat. It is only necessary in this example where the field occurs at the end of a line and LEX inserts an implicit RETURN after the last character.

Hyphened

When escape sequence ESC QI is used to show how the current word would be hyphenated, this field prefixes the background display.

Key is

Normally a record is performed by [Key] but in doing so characters under the string are overwritten. The sequence ESC RK allows input of the key in background mode without effecting the foreground text. This message prompts for the entry of the key of the record to be performed.

Foreign file is

On issuing the command to set a foreign file, ESC SF, this prompt is displayed awaiting the string containing the foreign file definition.

`Mask is`

On issuing the sequence to set a mask, ESC SM, this message prompts, in background mode, for the key of the record containing the mask details.

`Merge`

When merge options are selected requiring the entry of a string, this field provides the first part of the prompt for that string.

`Write out`

When extract options are selected requiring the entry of a string, this field provides the first part of that string.

`advance to_`

This field follows the `Merge` or `Write out` literals to complete the prompt for the merge advance/extract advance sequences ESC MA and ESC WA. These sequences advance through the open document without transferring any text.

Note the _ which is used to force a space after the word `to`. It is only necessary in this example where the field occurs at the end of a line and LEX inserts an implicit RETURN after the last character.

`until`

This field follows the Merge or Write out literals to complete the prompt for the merge until/extract until sequences ESC MU, and ESC WU. These sequences transfer all text until the nominated string is reached.

`just`

This field follows the Merge or Write out literals to complete the prompt for the merge just/extract just sequences ESC MJ, and ESC WJ. These sequences transfer only those lines containing the nominated string.

`to end of`

This field follows the `Merge` or `Write out` literals to complete the prompt for the ESC MM and ESC WX sequences to merge to end of document.

`Find`

When the find command (ESC)F is entered this prompt appears in background mode awaiting the entry of a string to be found.

`Search for`

When the search command (ESC)G is entered this prompt appears in background mode awaiting the entry of a string to be found.

`Search backwards for_`

(ESC)-G initiates a search backwards through a document. On entering the command this field appears in background mode awaiting the entry of a string to be found.

Note the _ which is used to force a space after the word `for`. It is only necessary in this example where the field occurs at the end of a line and LEX inserts an implicit (RETURN) after the last character.

`Open (overwrite)`

This is the prompt for a document name resulting from the open document commands (ESC)OD and (ESC)OF. These commands open a document unconditionally, overwriting existing documents if necessary.

`Open (create)`

This is the prompt for a document name resulting from the open document command (ESC)OX. If the document exists the open will fail.

`QS Key`

This is the prompt requesting the name of a record containing the filename for an external program as required by the sequence (ESC)QS.

`QT Command`

This is the prompt requesting the filename of an external program as required by the sequence (ESC)QT. It is also used by the sequence (ESC)QW.

`New extension:`

This is the prompt for a document extension which follows the (ESC)CD sequence. The sequence applies the document details associated with the new extension to the current document.

`Attributes currently on`

This message is used with the (ESC)BU sequence to show which attributes are currently set.

`Exit`

When (ESC)E is entered in a document this message informs the user that LEX is leaving the document and returning to the menu. It appears with the other **TEXT message `Selected. WAIT...` and is shown whilst LEX is conducting the write away process. The document name and an indication of how writing away the file is progressing are also shown.

`Abandon edit`

When (ESC)Z is entered in a document this message is displayed before returning to the menu. It appears with the other **TEXT message `Selected. WAIT ..` and the name of the document.

`Re-edit`

When (ESC)V is entered in a document this message informs the user that LEX is writing away the document prior to re-entering it at the top of the document. It appears with the other **TEXT message `Selected. WAIT ..` The document name and an indication of how writing away the file is progressing are also shown.

`Re-edit and return`

When (ESC)QX is entered in a document this message informs the user that LEX is writing away the document prior to re-entering it at the point where the command was issued. It appears with the other **TEXT message `Selected. WAIT ...` The document name and an indication of how writing away the file is progressing are also shown.

`This screen is intentionally blank`

This message shown on entry into playground if enabled in **INST.

6.2 VMF MANAGEMENT MESSAGES - **TEXT2

```
[**TEXT2   .
VMF reorganise. Statistics of current VMF :- \Number of data blocks used
Number of data blocks free
Number of keys used
Number of keys on index
Number of keys in overflow
New VMF name\Automatic reorganise (Y/N)
New number of data blocks\New number of keys on index
Now creating file. Blocks written..
VMF copy records. Current file is \Copy VMF name
Copy From current file (F) or To current file (T) (F/T)
Keys \ New key \ copied\ not found\ already on file - overwrite\Not a VMF
Copy all records (Y/N) \ VMF too full
LEX patch program \Enter file name to patch \Can't find file
LEX status\VMF is\Work file is\Current menu position\Cut & paste buffers used
Foreign file parameters
VMF statistics
Total number of blocks
Calculator memories are in use
Reorganise finished. Rename and use new VMF (Y/N)
Clear locks for which VMF \Key Locked\Locks now being cleared.  Please wait..
Rename illegal across accounts/devices
Installation parameters\Multi user\Single user
```

FIGURE 6.2 - EXAMPLE OF **TEXT2

The messages and prompts held in **TEXT2 are all associated with six menu switch options, some appearing in more than one of the options. The menu switch options concerned are:

/0	Clear locks
/3	Reorganise the VMF
/4	Copy records from one VMF to another
/5	Patch
/T	VMF Statistics
/W	LEX Status

The fields in the example of **TEXT2 at Figure 6.2 are described below:

`VMF Reorganise.  Statistics of Current VMF :-`

Used to head the /3 option, this message is followed by the filename of the current VMF.

`Number of data blocks used`

Used in the /3 and /T options this message is followed by the number of data blocks actually used in the current VMF.

`Number of data blocks free`

Used in the /3 and /T options this message is followed by the number of unused data blocks in the current VMF.

`Number of keys used`

Used in the /3 and /T options this message is followed by the number of keys held on the current VMF.

`Number of keys on index`

Used in the /3 and /T options this message is followed by the maximum number of keys that the VMF index is set to accept.

`Number of keys in overflow`

Used in the /3 and /T options this message is followed by the number of records which are stored in overflow blocks.

`New VMF name`

Used in the /3 option this is the prompt for the filename of the VMF to be created.

`Automatic reorganise (Y/N)`

Under the /3 option it is possible to create a new reorganised VMF of the same size as the current VMF and containing all the records. An answer of `Y` to this prompt will achieve this whereas an `N` causes a manual reorganisation.

`New number of data blocks`

With a manual reorganisation under the /3 option this prompts for the number of data blocks to be generated in the new VMF. Entering (CR) assumes the same number of blocks as are present in the existing VMF.

`New Number of Keys on index`

With a manual reorganisation under the /3 option this prompts for the number of keys to be allowed on the index of the new VMF. Entering (RETURN) defaults to twice the number of keys currently being used subject to a minimum of 400. Note that any number entered may be rounded up to the relevant whole number of blocks to accommodate at least the number of keys required.

`Now creating file. Blocks written..`

This message appears under the /3 option whilst the new VMF file is being created. It is followed by a count of the blocks as they are allocated and initialised.

`VMF copy records. Current file is`

When menu switch option /4 is selected to copy records between VMFs this message is shown followed by the name of the current VMF.

`Copy VMF file name`

This is used under the /4 option to prompt for the new filename. LEX checks for this file, sending a BELL if it does not exist and the message Not a VMF described below if the file does exist but is not in the VMF format.

`Copy From current file (F) or To current file (T) (F/T)`

Under the /4 option it is possible to copy records from the current VMF into another or vice versa. The response to this prompt should be F to copy FROM the current VMF and T to copy TO it.

`Keys`

This is the prompt for the key of each record that is to be copied. It is used both by the copy records option /4 and by the reorganisation option /3 if manual reorganisation is selected.

`New key`

When LEX is copying records under the /4 option it pauses with the message `Already on file` (see below) if it finds that the record already exists. It also pauses if the query option /Q is added to the end of the record key entered. In either case it is then possible to change the name of the record as it is copied. Entering a / results in this prompt requiring the new key name for the record.

`copied`

This message announces that LEX has successfully copied the record under the /3 or /4 options.

`not found`

Under the record copy option /4, this message appears when LEX cannot find the key specified for copying.

`already on file - overwrite`

When LEX is copying records to an existing VMF under the /4 option it is possible that a record of the same name may exist. If LEX detects this it prompts with this message and pauses to allow the user to either copy the record, skip the copy, or copy the record and give it a new name.

`Not a VMF`

LEX will not permit records to be copied into a file which is not in the format of a VMF. Under the /4 option, if the filename points to a file which is not in the VMF format then this message results.

`Copy all records (Y/N)`

During a manual reorganisation under option /3, LEX first creates a VMF of the appropriate size and then allows the user to copy records into it. This may be done one by one or all records may be copied automatically if this prompt is answered by `Y`.

`VMF too full`

During a copy option /4, this message results if an attempt is made to copy into a VMF which is already full. The receiving VMF must be reorganised before it will accept further records.

`LEX Patch Program`

LEX has its own built in patch routine, /5. This is the message used to show LEX is entering that routine.

`Enter file name to patch`

On entering the patch routine using the menu switch option /5, you are prompted for the filename of the file to be patched with this message.

`Can't find file`

This message is displayed whenever LEX cannot find a file.

`LEX Status`

Menu switch option /W is headed by this message.

`VMF is`

Under the LEX Status and VMF Statistics menu switch options /T and /W this message precedes the filename of the current VMF.

`Work file is`

Under the LEX Status and VMF Statistics menu switch options /T and /W this message precedes the filename of the current workfile.

`Current menu position`

Under the LEX Status menu switch option /W the menu stack is displayed preceded by this message.

`Cut and Paste buffers used`

In the LEX Status menu switch option /W a list of cut and paste buffers in use is listed under this heading.

`Foreign file parameters`

If a foreign file is open then under the LEX Status menu switch option /W its name is preceded by this message.

`VMF Statistics`

This message is used as a heading for the VMF Statistics display under menu switch option /T.

`Total number of blocks`

Under the VMF Statistics menu switch option /T this message announces the total number of blocks in the current VMF.

`Calculator memories are in use`

Under the LEX Status menu switch option /W a list of calculator memories in use is displayed under this heading.

`Reorganise finished. Rename and use new VMF (Y/N)`

Under the reorganisation menu switch option, /3, it is possible to delete the current VMF, rename the newly created version with the name of the current VMF and finally enter it. A Y to this prompt achieves this. Note that another field in **TEXT2 is activated if an attempt is made to rename a VMF in a different account/device. In such instances LEX simply enters the new VMF without deleting the old one or renaming the new one.

`Clear locks for which VMF`

Under menu switch option /0 this prompts for the filename of the VMF whose locks are to be cleared. Note that this only applies to multi user systems.

`Key Locked`

This message is displayed on trying to overwrite a key that is locked whilst copying records between VMFs under menu switch option /4.

`Locks now being cleared Please wait ..`

Under menu switch option /0 this message appears whilst LEX is clearing the locks on the records of a foreign file.

`Rename illegal across accounts/devices`

Under the reorganisation menu switch option, /3, it is possible to delete the current VMF, rename the newly created version with the name of the current VMF and finally enter it. This is, however, only legal if both versions of the VMF are in the same account/device. If they are not then this message results and LEX simply enters the new version without changing the old VMF.

`Installation parameters`

Used in the display of LEX Status under menu switch option /W this message is followed by the keys of the installation records in use (**INST, **PNST and **MONT).

`Multi user`

A VMF can be opened as multi user or single user. This message is a heading to describe how the VMF is open in the display of LEX status and VMF Statistics under menu switch options /W and /T.

`Single user`

A VMF can be opened as multi user or single user. This message is a heading to describe how the VMF is open in the display of LEX status and VMF Statistics under menu switch options /W and /T.

6.3 PRINTING MESSAGES - **TEXTP

The messages and prompts held in **TEXTP are all associated with printing and list processing menu switch options.

FIGURE 6.3 - EXAMPLE OF **TEXTP

```
[**TEXTP
AND      &&OR      /&NOT      @@KEY     KSRECTYPE kSDATE    DSPAGE   dSSRECS  zS
TRECS    ZSTIME    tS\Can't find that format^\No output dictionary
 is not a valid output dictionary entry^
The FIELD NAMES listed below can be used to make your selection
Select criteria: \Select \Invalid selection^\LEX is searching\LEX is sorting^
Latest key \No. of records selected: \Totals are: \Number of copies_
Pause after each page ?\Line up paper, then press any key to continue^
Petals are \Page \Number from  \Print from \End print_
Print odd or even numbered pages (O/E)-\Device not available
Print file name \Invalid format
```

The fields in the example of **TEXTP at Figure 6.3 are described below:

```
AND      &&OR      /&NOT      @@KEY     KSRECTYPE kSDATE    DSPAGE   dSSRECS  zS
TRECS    ZSTIME    tS
```

This field defines special operands which can be used in the Select line during list processing or in the output formatting record. There are two types of operand in this specification:

a) Logical operands for use in the select line.

b) Standard dictionary entries. These are common to all dictionaries but are not shown in the select line help message.

Each special operand is a sub-field of 10 characters, the first 8 characters of each which defines-the variable name and is followed by a unique field-specifying character such as would be used in a dictionary. This character is reserved for this purpose and cannot be used elsewhere in a dictionary. The last character represents the field type which is used internally by LEX and must not be changed. The special selection operands are shown in the following Table 6.1.

TABLE 6.1 - SPECIAL LIST PROCESSING OPERANDS

Operand	Char	Field Type	Purpose	
AND	&	&	Logical operand for logical AND	
OR	/	&	Logical operand for logical OR	
NOT	@	@	Logical operand for logical NOT	
KEY	K	s	Dictionary entry	Key less the first character
RECTYPE	k	s	Dictionary entry	First character of key
DATE	D	s	Dictionary entry	Current date
PAGE	d	s	Dictionary entry	Page number
SRECS	z	s	Dictionary entry	Records selected so far
TRECS	Z	s	Dictionary entry	Records considered so far
TIME	t	s	Dictionary entry	Current time

`Can't find that format^`

This is used in the list processing menu switch options /L and /S if the record containing the specified output format cannot be found in the VMF.

`No output dictionary`

The record containing the output format in list processing menu switch options /L and /S refers to the dictionary to be used. If the dictionary record is not found on the VMF then this message is displayed.

`is not a valid output dictionary entry`

In list processing menu switch options /L and /S the record containing the output format refers to the dictionary it needs. If there any errors in the dictionary entries then this message results. Errors in the first field of **TEXTP (logical operands and standard dictionary entries) also result in this error

`The FIELD NAMES listed below can be used to make your selection`

When the select prompt is displayed using the list processing menu switch option /S, a help message may be displayed automatically if it has been enabled in **INST. If not enabled it may still be displayed by entering HELP at the select criteria line. In both cases the message in the field is used to head a breakdown of fields available from the dictionary for use in the list processing run.

`Select criteria:`

If enabled in **INST the selection criteria is displayed at the end of a list processing run. This message heads the details.

`Select`

In list processing menu switch options /L and /S this is the prompt to enter the selection criteria. It appears on the line above the field for entering the criteria.

`Invalid selection^`

In list processing if the selection criteria entered contains either a logic error or refers to a field which does not exist in the specified dictionary then this message is displayed. The user can then re-enter the selection criteria.

`LEX is searching`

This message is displayed whilst LEX is searching for records to be viewed in list processing. It may be followed by the Latest key literal described below.

`LEX is sorting^`

If LEX is to sort the data which has been selected by list processing then this message is displayed whilst the sort is taking place.

`Latest key`

During the list processing menu switch options /L and /S the previous messages may be followed by a running display of the key that LEX has reached. This message introduces the key. Note that the option must be enabled in **INST.

When list processing to a slave printer this field is suppressed.

`No. of records selected:`

If enabled in **INST a list processing run can show the number of records selected. This appears at the end of the list processing run as two numbers separated by a /. The first number is the number of records selected in this run and the second the total number of records processed.

`Totals are`

This field is displayed at the end of list processing if totalling has been selected for any of the fields. The fields selected for totalling and the totals of the fields appear on lines immediately following this message.

`Number of copies:`

During the multiple print menu switch option /Q this text prompts for the number of copies of the document required. A (RETURN) defaults to one copy.

`Pause after each page?`

When using printing to physical device options, **PNST allows LEX to be instructed to stop at the beginning of each printed page or to print continuously. Alternatively the decision can be made at print time in which case this message requests the required option

`Line up paper, then press any key to continue^`

During printing options to physical devices the previous prompt may be enabled to allow LEX to stop on every page break. On each page break this message then appears with appropriate instructions for the user.

When printing to a slave printer this message is replaced by two (BELL)s.

`Petals are`

The text held here appears at print time to indicate which type of petal keys are being used, it is followed by the key of the petal record. The message is suppressed when printing to a slave printer.

`Page`

When LEX is printing it gives a running indication of the page currently being printed preceded by this message.

When printing to a slave printer this output is suppressed.

`Number from`

This prompt appears when a title or footer (.T or .F) line is encountered in a document to be printed. The message prompts for the page number to be given to the next page in the document whether it is to be printed or not. The default, obtained by (RETURN), can be set in **PNST to 0 or 1.

`Print from`

This prompt follows the previous one and requires the page number of the first page to be printed. Default is the start page number. Note that if physical printing starts before the .T or .F line is detected, then this prompt is omitted.

`End print`

Following the previous two prompt messages, this one calls for the page number of the last page to be printed. Default is 9999.

`Print odd or even numbered pages (O/E)-`

When the title or footer line includes alternate printing then this additional prompt message is given to allow only odd or even numbered pages to be printed. Any character other than O or E allows all pages to be printed. The characters O and E can be set in **MONT.

`Device not available`

When LEX attempts to print on a physical device already in use then this message is displayed and the user is prompted to re-specify the device.

`Print file name`

Rather than print to a physical device **PNST allows printing to a spool file. This message prompts for the name of the spool file which may be any valid device/filename.

`Invalid format`

During list processing menu switch option /S, a formatting record is required. This message results from an error in the format of that record. e.g. a missing blank line separating the header from the data lines.

6.4 ERROR MESSAGES

Errors within LEX fall into three categories.

a) Those which occur when not in a document or playground. They are shown immediately and are known as Automatic Error Messages.

b) Those which occur when in a document or playground. These may be viewed by entering (ESC)H or automatically if show error mode is selected and are referred to as the Delayed Error Messages.

c) Errors in the Run Time System. Usually these are catastrophic errors.

The first two types of error messages may be changed as required.

6.4.1 Automatic error messages

When not in a document or the playground errors are reported by a (BELL) to the VDU and an error message displayed automatically. In certain cases only the bell is sounded and no message is displayed.

They usually occur under the various menu switch options and the text for these errors is taken from fields in **TEXT, **TEXT2 and **TEXTP. Each of these errors is discussed under the appropriate field in the descriptions earlier in this chapter.

6.4.2 Delayed error messages

Errors occurring in a document or in the playground are usually the result of escape sequences or control functions.

They are reported by a (BELL) to the VDU and a two character error condition is set. An appropriate error message can be displayed automatically if the show error messages option is enabled in **INST or by typing (ESC)H. The message is cancelled by hitting (RETURN).

These error messages are held as records, one record for each error. Each error message is stored with a key which has the form *-ERROR-aa, where aa is AA to ZZ. This is followed by a 40 character error message.

A list of all the standard error messages is given in Appendix E with a brief explanation of the cause of the error.

The error records may be changed or viewed and *ERRORFORM is provided for this purpose. An example of a page from *ERRORFORM is given in Figure 6.4.

FIGURE 6.4 - EXAMPLE OF ERROR RECORDS

```
Error Code      Message

[*-ERROR-AC_    ******* Unknown escape sequence *******  _
[*-ERROR-AL_    ****** Too much to cut and paste ******  _
[*-ERROR-AX_    ****** Cut too long for keystore ******  _
[*-ERROR-BB_    * ESC ] not followed by A,C,K,P,R or U*  _
[*-ERROR-BC_    *********  Key already used  **********  _
[*-ERROR-BF_    ****** Cut and paste buffer full ******  _
[*-ERROR-BK_    **********  Key not found  ************  _
[*-ERROR-BR_    ***** Invalid cut & paste block *******  _
[*-ERROR-BS_    ***** Save key no longer on file ******  _
[*-ERROR-CB_    **** Calculator, invalid use of ] *****  _
[*-ERROR-CC_    **** Invalid calculator character *****  _
[*-ERROR-CE_    ***** Invalid calculator syntax *******  _
[*-ERROR-CM_    ******* You haven't done a cut ********  _
[*-ERROR-CN_    ***** Invalid number in calculator ****  _
[*-ERROR-CO_    ***** Calculator number overflow ******  _
[*-ERROR-DL_    * Demonstration version file too big **  _
[*-ERROR-EB_    *********** End of document ***********  _
```

Records could, for example, be changed to give the user directions as to how to rectify the error. The error message `You are in INSERT Mode !!!` could become `You are in INSERT Mode. Use <ESC>-I to exit.` If there is not enough room in the error message then the user could be referred to one of the help menus.

6.4.3 Run time system errors

If a serious system fault occurs, LEX may terminate with a message followed by the prompt:

```
Please check your system before continuing

D Run-Time Monitor (ex to exit)>_
```

If this happens, the user may return to the operating system by entering EX(RETURN). The system should then be checked in line with the message displayed before continuing.

Unlike LEX errors, errors in the RTS cannot be customised by the user.

Some of the causes of run time errors are considered below.

There are two common types of disk errors that may be encountered; those caused by Bad Blocks on a disk and those caused when the disk is full. LEX supports the retry of I/O operations on 'Bad Block' and 'Disk is full' errors. 'Disk is full' is fairly common on small or heavily loaded systems. 'Bad Block' is common on floppy disk systems but may also appear occasionally with hard disks.

On finding a 'Bad Block' on a disk LEX sounds the bell twice then after a pause of ten seconds the operation is retried. If it happens repeatedly then it is a good idea to replace the particular disk involved. If the operation still does not proceed then the operation should be aborted (by CTRL c) and restarted with good media.

On finding that the 'Disk is Full' LEX again sounds the bell and pauses for four seconds. Some multi-user systems may support dynamic file extension and it may be possible to make sufficient space available by other means. If this is not possible then the operation should be interrupted (by CTRL c) and restarted with suitably empty media.

In the special case that the file being handled when an error occurs is an output file containing text (for example, when LEX is creating a new file, a spool file is being produced or when a new version of a file is being produced during an edit) then a special procedure is available for resuming with a continuation file. This facility is not available for other files, including VMF or input text files and is only available under RSTS, RT11, RSX and VMS.

The procedure is as follows :-

a) The retry loop is interrupted by DELETE or RUBOUT. On receiving this the output file is closed and truncated. This feature is only available where the system supports 'conditional terminal input'. (Note that the error may not be on the expected file).

b) LEX then prompts for the name of the new file to be used as the continuation file. The default extension is .CNT

c) If this works the user should exit from LEX and use whatever local system software is available to merge the files together.

If the same media is used for the continuation file then it is highly likely that the error that truncated the original file will also appear on the continuation and it may be necessary to construct several continuations.

Merging files can be tedious and it is often simpler to abort the whole operation and start again.

Chapter 7. CONVERSION TABLES

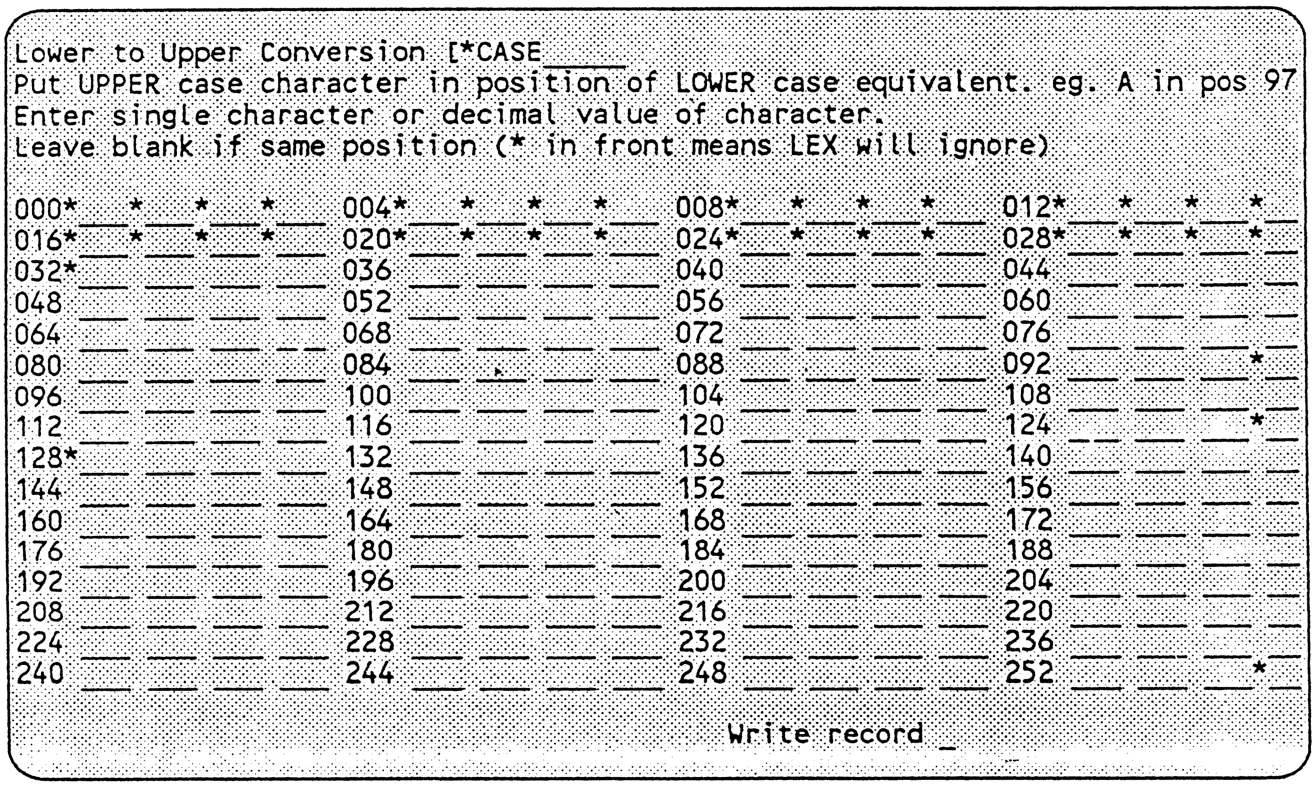
7.1 CASE CONVERSION - *CASEFORM

Case conversion is used by the CTRL K and ESC QQ commands and in key and document names. The latter may be converted to upper case or lower case or unchanged depending on a setting in **INST.

Generally when using the English language the standard ASCII case conversion will suffice, i.e. characters 96 - 126 convert to 64-94 and characters 224 - 254 convert to 192 - 222 and vice versa (e.g. abcd converts to ABCD). Note that this includes certain special characters. They are, with their case conversion:

```
^ { } [ ] \ | @ '
~ [ ] { } | \ ' @
```

In foreign language character sets using 8-bit characters the default may not be satisfactory and a system record *CASE can be used to control the conversion. The record is created using *CASEFORM which, when activated, appears as follows:



```
Lower to Upper Conversion [*CASE____
Put UPPER case character in position of LOWER case equivalent. eg. A in pos 97
Enter single character or decimal value of character.
Leave blank if same position (* in front means LEX will ignore)

000*___*___*___*___   004*___*___*___*___   008*___*___*___*___   012*___*___*___*___
016*___*___*___*___   020*___*___*___*___   024*___*___*___*___   028*___*___*___*___
032*___ ___ ___ ___   036 ___ ___ ___ ___   040 ___ ___ ___ ___   044 ___ ___ ___ ___
048 ___ ___ ___ ___   052 ___ ___ ___ ___   056 ___ ___ ___ ___   060 ___ ___ ___ ___
064 ___ ___ ___ ___   068 ___ ___ ___ ___   072 ___ ___ ___ ___   076 ___ ___ ___ ___
080 ___ ___ ___ ___   084 ___ ___ ___ ___   088 ___ ___ ___ ___   092 ___ ___ ___*___
096 ___ ___ ___ ___   100 ___ ___ ___ ___   104 ___ ___ ___ ___   108 ___ ___ ___ ___
112 ___ ___ ___ ___   116 ___ ___ ___ ___   120 ___ ___ ___ ___   124 ___ ___ ___*___
128*___ ___ ___ ___   132 ___ ___ ___ ___   136 ___ ___ ___ ___   140 ___ ___ ___ ___
144 ___ ___ ___ ___   148 ___ ___ ___ ___   152 ___ ___ ___ ___   156 ___ ___ ___ ___
160 ___ ___ ___ ___   164 ___ ___ ___ ___   168 ___ ___ ___ ___   172 ___ ___ ___ ___
176 ___ ___ ___ ___   180 ___ ___ ___ ___   184 ___ ___ ___ ___   188 ___ ___ ___ ___
192 ___ ___ ___ ___   196 ___ ___ ___ ___   200 ___ ___ ___ ___   204 ___ ___ ___ ___
208 ___ ___ ___ ___   212 ___ ___ ___ ___   216 ___ ___ ___ ___   220 ___ ___ ___ ___
224 ___ ___ ___ ___   228 ___ ___ ___ ___   232 ___ ___ ___ ___   236 ___ ___ ___ ___
240 ___ ___ ___ ___   244 ___ ___ ___ ___   248 ___ ___ ___ ___   252 ___ ___ ___*___

                                          Write record _
```

This form should be completed with the UPPER case character in the position of its LOWER case equivalent. e.g. A in position 97.

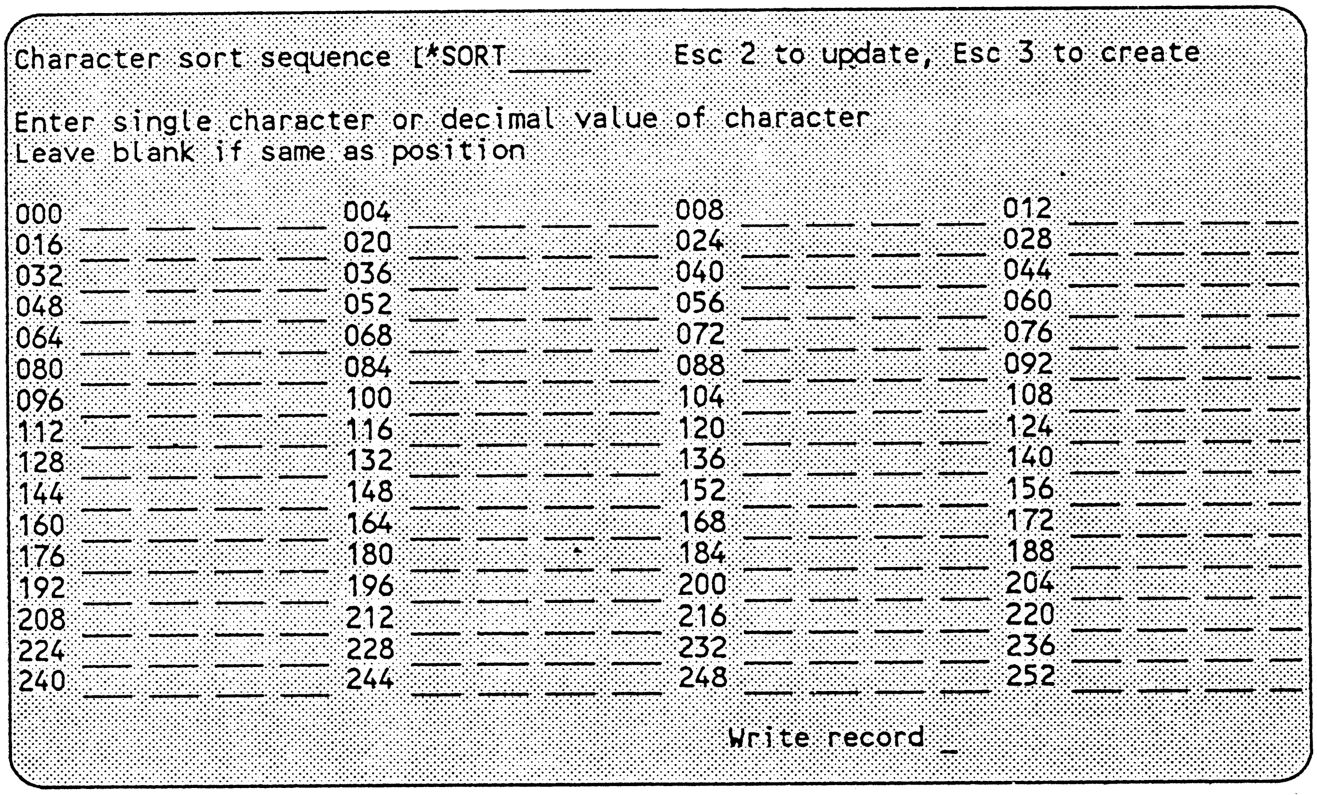
The first 33 characters are ignored as they are reserved by LEX as are 95, 127, 128, and 255.

Use this feature with care. DO NOT ENTER A CHARACTER TWICE AS THE REVERSE CONDITION WILL BE UNPREDICTABLE.

Remember too that it can be very difficult to remember a key and the case of every letter. i.e. ABCD, abcd, AbCd etc. So consider this when deciding on the conversion required.

7.2 SORTING - *SORTFORM

With certain foreign language character sets the ASCII sequence is not sufficient to achieve a strict alphabetic sequence, for example to put A-umlaut after the A. To overcome this LEX uses a system record *SORT. The record is created using *SORTFORM which, when activated appears as:



```
Character sort sequence [*SORT____        Esc 2 to update, Esc 3 to create

Enter single character or decimal value of character
Leave blank if same as position

000 __ __ __ __  004 __ __ __ __  008 __ __ __ __  012 __ __ __ __
016 __ __ __ __  020 __ __ __ __  024 __ __ __ __  028 __ __ __ __
032 __ __ __ __  036 __ __ __ __  040 __ __ __ __  044 __ __ __ __
048 __ __ __ __  052 __ __ __ __  056 __ __ __ __  060 __ __ __ __
064 __ __ __ __  068 __ __ __ __  072 __ __ __ __  076 __ __ __ __
080 __ __ __ __  084 __ __ __ __  088 __ __ __ __  092 __ __ __ __
096 __ __ __ __  100 __ __ __ __  104 __ __ __ __  108 __ __ __ __
112 __ __ __ __  116 __ __ __ __  120 __ __ __ __  124 __ __ __ __
128 __ __ __ __  132 __ __ __ __  136 __ __ __ __  140 __ __ __ __
144 __ __ __ __  148 __ __ __ __  152 __ __ __ __  156 __ __ __ __
160 __ __ __ __  164 __ __ __ __  168 __ __ __ __  172 __ __ __ __
176 __ __ __ __  180 __ __ __ __  184 __ __ __ __  188 __ __ __ __
192 __ __ __ __  196 __ __ __ __  200 __ __ __ __  204 __ __ __ __
208 __ __ __ __  212 __ __ __ __  216 __ __ __ __  220 __ __ __ __
224 __ __ __ __  228 __ __ __ __  232 __ __ __ __  236 __ __ __ __
240 __ __ __ __  244 __ __ __ __  248 __ __ __ __  252 __ __ __ __

                                         Write record _
```

If a *SORT record is created then each character in the sort key is translated using the table prior to the sort taking place. It applies to all the sorts; those in edit mode and during list processing.

If no differential between upper case and lower case is required then enter values 064 - 094 in fields marked 096 - 126.

To put A-umlaut after the A then all characters after the A must be shifted down by translation.

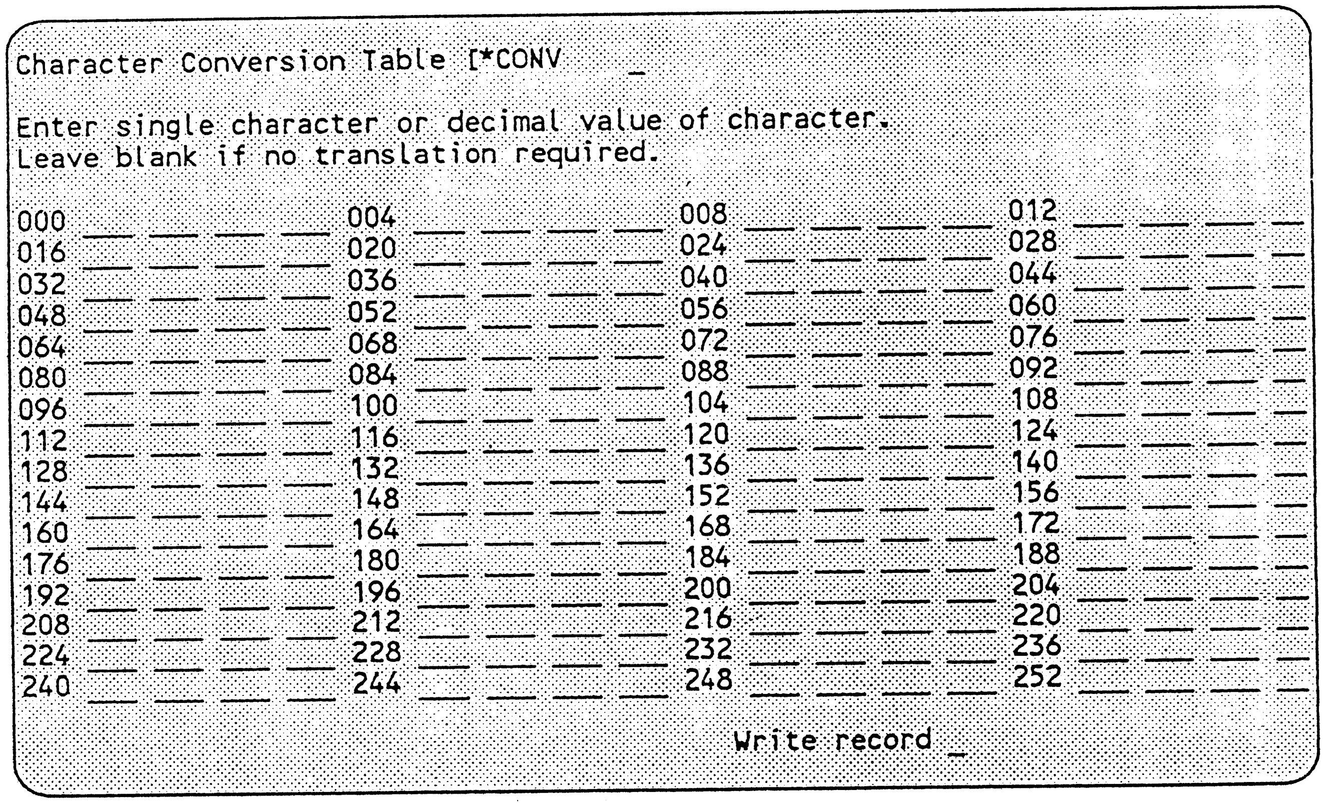
The space character (032) and paint, usually underline (095), have a special meaning and must not be translated.

Note that records *SORTaaa may be created. These have no effect on sorting, but are useful to retain sort sequences which may be required again.

7.3 7-BIT TO 8-BIT FILE CONVERSION - *CONVFORM

LEX supports 8-bit character codes i.e. a character set of 256 characters allowing for additional graphic and foreign characters. However not all VDUs and operating systems can take advantage of this and therefore LEX provides a means of converting documents between 8-bit and 7-bit formats.

Menu switch option /6 is provided for this. It makes use of *CONV to provide the data for character translation. *CONV records may be generated using *CONVFORM shown below.



```
Character Conversion Table [*CONV   _

Enter single character or decimal value of character.
Leave blank if no translation required.

000 __ __ __ __  004 __ __ __ __  008 __ __ __ __  012 __ __ __ __
016 __ __ __ __  020 __ __ __ __  024 __ __ __ __  028 __ __ __ __
032 __ __ __ __  036 __ __ __ __  040 __ __ __ __  044 __ __ __ __
048 __ __ __ __  052 __ __ __ __  056 __ __ __ __  060 __ __ __ __
064 __ __ __ __  068 __ __ __ __  072 __ __ __ __  076 __ __ __ __
080 __ __ __ __  084 __ __ __ __  088 __ __ __ __  092 __ __ __ __
096 __ __ __ __  100 __ __ __ __  104 __ __ __ __  108 __ __ __ __
112 __ __ __ __  116 __ __ __ __  120 __ __ __ __  124 __ __ __ __
128 __ __ __ __  132 __ __ __ __  136 __ __ __ __  140 __ __ __ __
144 __ __ __ __  148 __ __ __ __  152 __ __ __ __  156 __ __ __ __
160 __ __ __ __  164 __ __ __ __  168 __ __ __ __  172 __ __ __ __
176 __ __ __ __  180 __ __ __ __  184 __ __ __ __  188 __ __ __ __
192 __ __ __ __  196 __ __ __ __  200 __ __ __ __  204 __ __ __ __
208 __ __ __ __  212 __ __ __ __  216 __ __ __ __  220 __ __ __ __
224 __ __ __ __  228 __ __ __ __  232 __ __ __ __  236 __ __ __ __
240 __ __ __ __  244 __ __ __ __  248 __ __ __ __  252 __ __ __ __

                                       Write record _
```

The relevant field of the form is completed with the translated value either as a character or its decimal equivalent. The decimal equivalent is preferable as problems could be caused if the record was subsequently called on a different VDU.

Do not translate low ASCII values 0 to 27 as these are reserved for internal use. Likewise 127, 128, 254 and 255 are reserved.

Chapter 8. VDU AND KEYBOARD HANDLING

LEX can be used on any VDU which (a) supports direct cursor addressing and (b) reports every key depression directly to the host computer as soon as the key is pressed.

Different types of VDU behave in different fashions and respond to different sets of character sequences. Because of these differences, LEX refers all input received from a VDU to tables to determine how the character or characters that have been received are to be interpreted. A similar reference to these tables is made when LEX sends output to a terminal.

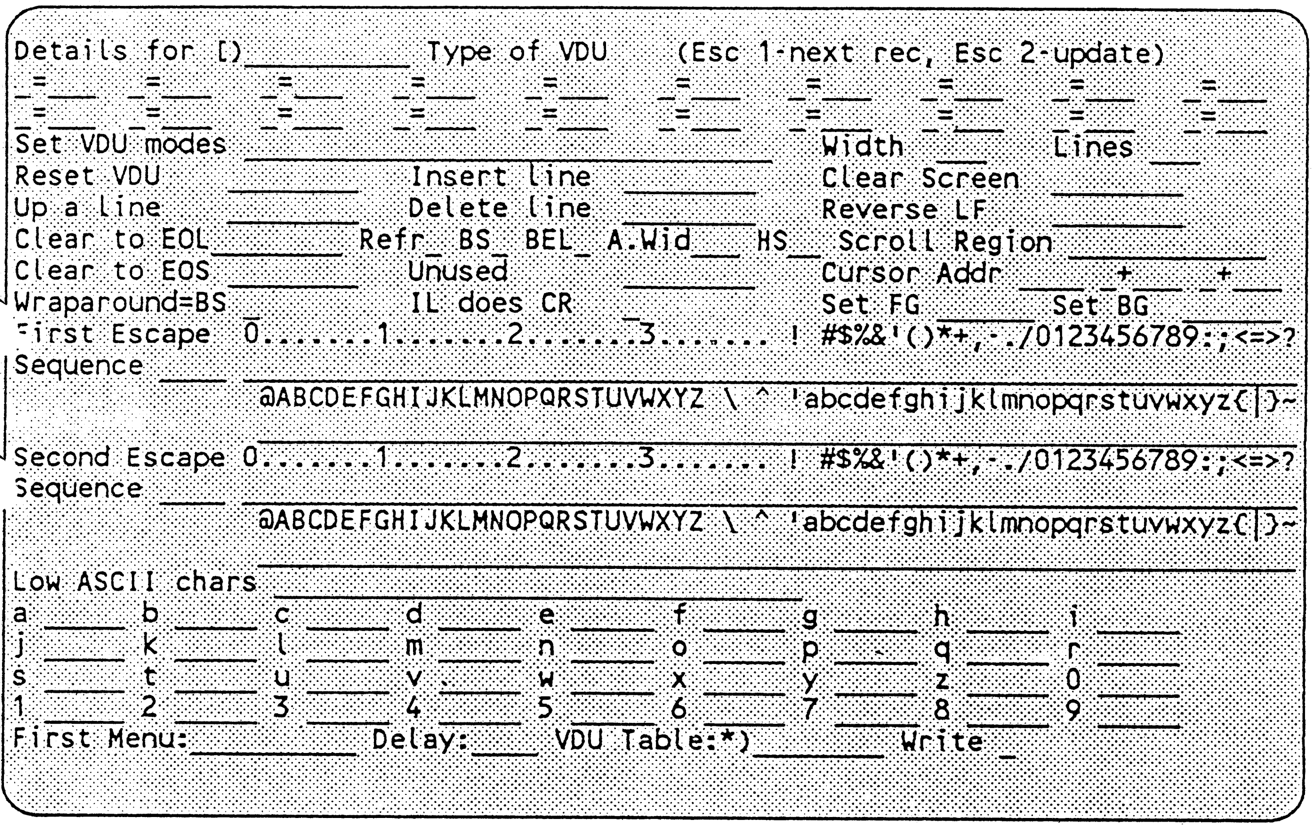
These tables are set up as system records on the VMF and are created and updated using the forms *VDUFORM, *VDUFORM2, *VDUFORM3, *VDUFORM4 and *VDUTABLE. The functions of the records defined using these forms are:

*VDUFORM	Defines VDU characteristics such as escape sequences for insert line, delete line, cursor addressing, screen size, etc. Also defines the LEX functions to be performed for received escape sequences (for example from function keys) and control codes.
*VDUFORM2	Defines additional LEX functions to be performed for received escape sequences. It also defines sequences to be sent to the screen for the (ESC)ssn function and to set menu attributes.
*VDUFORM3	Defines escape sequences for status line addressing and VDU attributes.
*VDUFORM4	Used to send long escape sequences to the VDU, e.g. to download function keys or to set up help facilities.
*VDUTABLE	Acts as a translate table for characters sent from LEX to the VDU.

At start up time LEX refers to the **INST system record to determine which terminal description records should be used. A different terminal description record can be selected at any menu by using menu option /U. Alternatively a different **INST record may be used by using menu option /1. A different VDU may also be selected using the typeahead facility.

8.1 DEFINING VDU CHARACTERISTICS - *VDUFORM

The form has the following format:



```
Details for [)__________ Type of VDU    (Esc 1-next rec, Esc 2-update)
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
Set VDU modes ________________________________    Width ___   Lines ___
Reset VDU      ________    Insert line  ________   Clear Screen ________
Up a line      ________    Delete line  ________   Reverse LF   ________
Clear to EOL________  Refr_ BS_ BEL_ A.Wid___ HS__ Scroll Region ________
Clear to EOS   ________    Unused       ________   Cursor Addr ___+___+___
Wraparound=BS _            IL does CR _            Set FG ______ Set BG ______
First Escape  0.......1.......2.......3....... ! #$%&'()*+,-./0123456789:;<=>?
Sequence ____ _____________________________________________________________
                @ABCDEFGHIJKLMNOPQRSTUVWXYZ \ ^ 'abcdefghijklmnopqrstuvwxyz{|}~
                _____________________________________________________________
Second Escape 0.......1.......2.......3....... ! #$%&'()*+,-./0123456789:;<=>?
Sequence ____ _____________________________________________________________
                @ABCDEFGHIJKLMNOPQRSTUVWXYZ \ ^ 'abcdefghijklmnopqrstuvwxyz{|}~
                _____________________________________________________________
Low ASCII chars ____________________________
a _____ b _____ c _____ d _____ e _____ f _____ g _____ h _____ i _____
j _____ k _____ l _____ m _____ n _____ o _____ p _____ q _____ r _____
s _____ t _____ u _____ v _____ w _____ x _____ y _____ z _____ 0 _____
1 _____ 2 _____ 3 _____ 4 _____ 5 _____ 6 _____ 7 _____ 8 _____ 9 _____
First Menu:__________ Delay:____ VDU Table:*)________ Write _
```

The VDU description records created and amended using this form have a key which commences). The) is automatically supplied in the form as the first character of the key.

The rest of the record key reflects the type of terminal being used. For example,)VT100 is used to describe the functions when using a VT100 in ANSI mode and)PCUK is used to describe an IBM PC terminal with a UK keyboard.

The rest of the record comprises 6 distinct sections.

8.1.1 Visible equivalents definition

The first section of the record appears immediately after the key and is used to define visible equivalents for characters which are used in sections two (see 8.1.2), five (see 8.1.5) and the lead-in to the Escape Sequence in section three (see 8.1.3.1) of the form. Characters for which visible equivalents must be defined are ones which cannot be displayed (e.g. the low value ASCII characters in the range 0 to 32 decimal) and characters which cannot be used in a form (e.g. [).

```
_=___ _=___ _=___ _=___ _=___ _=___ _=___ _=___ _=___ _=___
_=___ _=___ _=___ _=___ _=___ _=___ _=___ _=___ _=___ _=___
```

This area is divided into groups of four characters. The first character is the visible equivalent to be used, the rest of the field contains the decimal value of the character which will be represented by the visible equivalent:

```
a=001 $=027 {=091 /=013 _=___ _=___ _=___ _=___ _=___ _=___
_=___ _=___ _=___ _=___ _=___ _=___ _=___ _=___ _=___ _=___
```

Thus an entry of a=001 means that wherever the lower case character a appears in sections two (8.1.2), five (8.1.5) and the Escape sequence lead-in (8.1.3.1) of this form, it is to be interpreted as the decimal value of 1. Definitions made in this area are local to this screen and do not apply to *VDUFORM2, *VDUFORM3 and *VDUFORM4. Common definitions in this area are $=027 to define the dollar character as the visible equivalent of Escape and {=091 to define the open brace to be the visible equivalent of open square bracket.

8.1.2 VDU functions

The second section describes the sequences used by the terminal for the various functions needed by LEX:

```
Set VDU modes ________________________________ Width ___   Lines ___
Reset VDU     ________   Insert line ________   Clear Screen ________
Up a line     ________   Delete line ________   Reverse LF   ________
Clear to EOL________Refr_ BS_ BEL_ A.Wid__ HS__ Scroll Region __________
Clear to EOS________   Unused      ________   Cursor Addr ____+__ _+__
Wraparound=BS _          IL does CR  _          Set FG ______ Set BG ______
```

Delay Characters

Some terminals require a delay between some of the above operations, for example Clear Screen. This is achieved in LEX by sending delay characters to the terminal after an operation.

The sequence to define the delay character to be used is usually placed in the Set VDU modes field and is a two character string consisting of 127 (decimal) followed by 128 plus the delay character (i.e. the top bit of the character is set). This sets the delay character to the value sent minus 128 until another sequence is sent to reset it. The default delay character is a null.

A two character string consisting of 127 followed by a number less than 128 causes that number of delay characters to be sent to the screen. If more than 128 delay characters are to be sent to the screen then two or more sequences must be sent.

For example if ^=127, %=128 and :=100 had been defined as visible equivalents, then an entry of ^% in the Set VDU modes field would cause the delay character to be to null and ^: in the Clear Screen field would send 100 nulls to the terminal after a clear screen operation had been performed.

Delay characters are also useful to slow down screen output for use in help screens and rolling demonstrations.

The fields in this section are as follows.

`Set VDU modes`

This describes the sequence of characters to be sent to the terminal to set any special terminal characteristics. The sequence is sent when LEX is initiated or when a new terminal is selected with the /U menu option and is sent before any other characters are sent to the terminal. For example:

```
Set VDU modes $<$=${?3h____________________
```

The sequence above contains three escape sequences and causes:

a) `$<` the terminal to switch to ANSI mode

b) `$=` the numeric key pad to generate escape sequences rather than numbers

c) `${?3h` the width of the terminal to be set to 132 columns.

The Set VDU modes field allows for 32 characters. If this is not enough then longer sequences may be defined using `*VDUFORM4`.

The Set VDU modes field is also used for the special case of VDUs with VT52 type cursor addressing which have more than 96 columns. To access the right hand columns these VDUs need a second cursor address sequence in addition to the sequence defined in the Cursor Addr field. This is defined in the last 10 characters of the Set VDU modes field as follows:

4 characters of second cursor address lead-in sequence
3 characters of new X-bias
3 characters of new Y-bias

Width

This field contains the number of columns which may be displayed across the screen. Its maximum value is 240. If the terminal supports hardware horizontal scrolling this number will be greater than the actual width of the screen (see A.WID below). If horizontal scrolling is not used its maximum value is the same as the screen width, which in most cases will be 80.

Lines

The number of lines contained on a screen is entered in this field. In most cases this field will contain 24.

If the last line of the screen can be used as the status line then this field will be one less than the actual number of lines on the screen. The status line is defined using *VDUFORM3 (see 8.3).

Reset VDU

When LEX exits back to the operating system, the sequence specified in this field is sent to the terminal. A common usage for this sequence is to send the necessary characters to a terminal to switch the keypad back into numeric mode and to switch a terminal back into a mode expected by other programs on the system.

The Reset VDU field allows for 8 characters. If this is not enough then longer sequences may be defined using *VDUFORM4.

Insert line

Some terminals have a hardware insert line feature which inserts a line of spaces on the display at the line on which the cursor was positioned and moves the rest of the screen down one line. If the terminal supports this feature then the sequence to invoke it should be entered in this field. If the sequence causes the cursor to move to the start of the inserted line then a Y should be entered in the IL does CR field below.

If no entry is made in this field, LEX achieves the insertion of a blank line by re-displaying all the text on the screen from the current line downwards. This is less efficient than making use of hardware insert line, unless the screen refresh facility is available.

For ANSI terminals which support scrolling (i.e. VDU type in Cursor Addr field is A or a) this field should be left blank as the sequence defined in Scroll Region is used for insert a line.

Clear Screen

The sequence entered here is sent to the terminal when LEX requires to do a clear screen, for example when entering the playground or displaying a menu. The sequence should result in the terminal displaying a blank screen with the cursor finishing in the top left hand corner of the screen. This may involve issuing a sequence to position the cursor to "Home" before and/or after blanking the screen.

Up a Line

The sequence which when sent to the terminal causes the cursor to move up one line while remaining in the same column. This is used when the cursor is not at the top of the screen. The Reverse LF entry (see below) is used when the cursor is at top of screen.

Delete Line

Some terminals have a hardware delete line feature which deletes a line on the display at the line on which the cursor was positioned and moves the rest of the screen up one line. If the terminal supports this feature then the sequence to invoke it should be entered in this field.

If no entry is made in this field, LEX achieves the deletion of a line by redisplaying all the text on the screen from the following line downwards. This is less efficient than making use of hardware delete line. Again, the refresh mode, if available, may help this problem.

For ANSI terminals which support scrolling (i.e. VDU type in Cursor Addr field is A or a) this field should not be entered as the sequence defined in Scroll Region is used for delete a line.

Reverse LF

The sequence which when sent to the VDU causes reverse scroll to occur if the cursor is positioned on the top line of the screen. If the VDU has no specific reverse line feed command, but does have insert line, then the insert line sequence should be entered here.

`Clear to EOL`

This describes the sequence used to clear a line of the screen from the current cursor position to the end of the line.

`Refr`

If the VDU is memory mapped within the computer itself enter the applicable character here and LEX will take advantage of memory mapping. This is much faster than standard refreshing of screens. For the IBM PC enter an R in this field.

`BS`

Usually a terminal will perform a back space if a character of value 008 is sent. If the terminal requires a value other than 008 to cause it to backspace then the value should be entered here. If the field is left blank a value of 008 is assumed.

`BEL`

Usually a terminal will sound a bell if a character of value 007 is sent. If the terminal requires a value other than 007 to cause it to sound the bell then the value should be entered here. If the field is left blank a value of 007 is assumed.

`A.WID`

The actual width of the VDU, i.e. the number of characters which may be displayed across the screen. This field is used if the terminal supports hardware horizontal scrolling and enables LEX to centre menus and displays. If used it will be less than the value defined in the Width field above.

`HS`

Not yet used

`Scroll Region`

If the terminal supports vertical scrolling regions then the sequence to set the top and bottom margins should be entered here. The scrolling facility is used by the VMF re-organisation routine to scroll the keys copied while maintaining the fixed information in the upper part of the screen. It is also used for insert line and delete line if no entry has been made in these fields.

For ANSI terminals which support scrolling (i.e. VDU type is A or a) the sequence is `$<MM;NNr` where `$` and `<` are visible equivalents of Escape and `[` respectively. This sequence is assumed by default and need not be entered.

For non-ANSI terminals which support scrolling the sequence must be entered here. MM and NN should be entered to represent the line numbers of the top and bottom margins and LEX will substitute the values as appropriate.

Clear to EOS

This describes the sequence which clears the screen from the current cursor position to the end of the screen. This sequence clears both foreground and background displays. If the VDU does not have a clear to end of screen function then leave this field blank and LEX will use multiple clear line functions.

Unused

This field was used in earlier versions of LEX, so a sequence may be found here in some older terminal records. Any sequence in this field is currently ignored. This field may be used again in future versions of LEX.

Cursor Addr ____ _+___ _+___

This field is split into five sections.

The first of these is the lead-in sequence used in cursor positioning.

The second, single character area is used to say whether the X (row) or Y (column) co-ordinates are sent to the terminal first. One of the characters X or Y must be inserted here as appropriate. The characters x or y may also be entered here and in these cases the X or Y co-ordinate will be subtracted from the VDU width, this is used for some Arabic/Hebrew terminals.

The third area is the decimal value of the offset which must be applied to the first co-ordinate to be sent to the terminal. This value is always added to the co-ordinate. LEX always considers the Home position as X coordinate zero and Y co-ordinate zero. As an example, it would be necessary to add one to each coordinate if the terminal regarded "Home" as being 1,1.

The fourth area is a single character used to describe the type of terminal logic being used in cursor positioning. Some terminals follow a simple format, whereas others follow ANSI standards and others have discontinuities in some areas of the screen. The types that are currently supported, together with the relevant character to be used in the form are:

VDU type		Cursor address sequence	Comments
A	ANSI	CCCCXXX;YYYH	Leading zeroes removed, ANSI standard scrolling region assumed
a	old ANSI	CCCCXXX;YYYH	Leading zeroes retained, ANSI standard scrolling region assumed
B	ANSI	CCCCXXX;YYYH	Leading zeroes removed, ANSI standard scrolling region not assumed
b	old ANSI	CCCCXXX;YYYH	Leading zeroes retained, ANSI standard scrolling region not assumed
V	VT52	CCCCxy	For VDUs with more than 96 columns see Set VDU modes field
H	Hazletine	CCCCxy	If x>126 then 96 is subtracted
E	Perkin-Elmer	CCxCCy	
P	Hewlett Packard	CCCCXXXrYYYC	For HP terminals operating in VT100 emulation mode, this is similar to ANSI except the terminators ; and H are replaced by r and C. Leading zeroes are removed
Z	Split	CCCCZXXZYY	Z is the first character of the X and Y bias

where	CCCC	is the lead-in sequence
	XXX	is the X co-ordinate bias
	YYY	is the Y co-ordinate bias
	x	is the X co-ordinate bias as a byte
	y	is the Y co-ordinate bias as a byte

the order of X and Y depends on the second entry in the cursor address field.

The next area of this field is the offset in decimal to be applied to the first co-ordinate in the sequence.

The final area of this field is the offset in decimal to be applied to the second coordinate in the sequence.

`Wraparound=BS`

This defines what action is to be taken if a terminal wraps onto a new line when a character is entered into the last column on the screen. Three situations are catered for:

_	If the field is not entered, then it is assumed that the terminal leaves the cursor in the final column after the character has been entered.
x	The terminal wraps onto a new line, but entering a (BS) (backspace as specified in the BS field) wraps the cursor back onto the previous line.
l	The terminal wraps onto a new line and scrolls when on the last line of the screen (Lynwood style terminals). LEX will perform a reverse (LF) to restore the correct position of the cursor. If using an IBM type of terminal which does not have a reverse (LF), then LEX will not display the last character of the screen under scroll conditions. The screen may be properly refreshed by using the (ESC)QR sequence.

Any character other than the lower case l shown above assumes the default x.

`IL does CR`

On some terminals, the action of inserting a line causes the cursor to be positioned on the left hand edge of the inserted line effectively performing a carriage return as well as the insertion of the blank line. For such terminals, this field should be set to Y. Any other character at this point will assume that the terminal does not perform the (CR) as well as the line insertion.

`Set FG`

This is the display mode that LEX normally uses for all its character transmission while editing. It is also used as the mode for the protected parts of forms. This will normally be set as no attributes for most terminals.

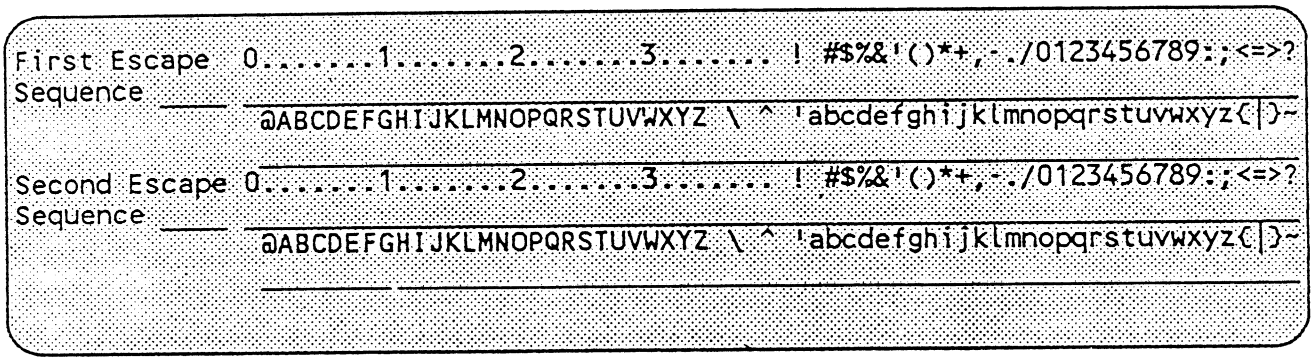
`Set BG`

This display mode is used as the unprotected areas of forms where data is entered by the user.

A common setting for this is double intensity or reverse video. Some terminals do not have the facility for displaying different forms of characters or this facility may not be desired, in which case both this field and the previous one should be left blank.

8.1.3 Escape sequences

The third section of this form describes incoming escape sequences and appears as:



```
First Escape  0.......1.......2.......3....... ! #$%&'()*+,-./0123456789:;<=>?
Sequence ____ ________________________________________________________________
               @ABCDEFGHIJKLMNOPQRSTUVWXYZ \ ^ 'abcdefghijklmnopqrstuvwxyz{|}~
              ________________________________________________________________
Second Escape 0.......1.......2.......3....... ! #$%&'()*+,-./0123456789:;<=>?
Sequence ____ ________________________________________________________________
               @ABCDEFGHIJKLMNOPQRSTUVWXYZ \ ^ 'abcdefghijklmnopqrstuvwxyz{|}~
               _______ ________________________________________________________
```

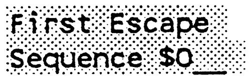
This section comprises two similar parts, each of which are split into two areas.

8.1.3.1 Lead-in sequence definition

The first area, which is 4 characters long, describes the lead in to the escape sequence as received from the terminal. Visible equivalents may be used. These sequences are normally sent by the arrow keys, the numeric keypad when working in a alternate keypad mode or by the function keys.

For example the arrow keys on an ANSI terminal generate: ESC O A, ESC O B, ESC O C, ESC O D. The ESC O is referred to as the lead in to the escape sequence.

The lead-in sequence may contain visible equivalents to represent special characters. For example for ANSI terminals, if $ had been defined as the visible equivalent of escape then the area would contain:



```
First Escape
Sequence $O__
```

If the lead-in sequence is greater than the four characters allowed in this area, the definition of the lead in characters may be continued into the second portion of each of these sections. A maximum of 10 characters may be defined as an escape sequence lead-in. The first underline character determines where the lead-in sequence terminates.

In the following example the escape sequence lead-in is ESC O Q ESC O:

```
First Escape      0.......1.
Sequence $0Q$     0_________     etc
                  @ABCDEFGHI
                  __________
```

VDU 29C

There are two special cases involved in this field.

i) Some terminals send sequences of the form lead in followed by a variable character followed by a terminating character. The ADM3A is an example of this in that its function keys send CTRL A followed by a variable character followed by CR. This is set up by entering a as the decimal value 001 and \ as decimal value 013 in visible equivalents at the start of this form. This field would then be set up as a \.

ii) The function keys on some terminals send a sequence of the form lead in followed by one or two numeric characters followed by a terminator. For such terminals the lead-in sequence is defined and followed immediately by a character which has been given a value of 254 in the visible equivalents section of the form. This 254 entry causes LEX to look for a sequence of the form:

    lead in number number terminator

Any non-numeric character is regarded by LEX as the terminator.

For example the function keys on the VT200 terminals generate:

    ESC [n n ~

where nn represents a number in the range 1-34 and ~ is the terminator. The visible equivalents would be set up as:

$027 [=091 [illegible]=254

and the lead-in as:

```
First Escape  0.......1.
Sequence ${*_  __________  etc
               @ABCDEFGHI

               __________
```

The number is used as an offset to index into the second part of this section.

8.1.3.2 Linking received sequences to LEX functions

The second portion of each of these sections describes which LEX function is to be performed for the received escape sequence. An upper or lower case character or a number is defined under the relevant variable character (or its decimal equivalent) received in the escape sequence.

If an upper case character is found, this implies that LEX should interpret the received sequence as a LEX internal control function.

LEX internal control functions are described in Appendix B

For example:

```
First Escape  0.......1.......2.......3....... ! #$%&'()*+,-./0123456789:;<=>?
Sequence $O__ _______________________________________________________________
               @ABCDEFGHIJKLMNOPQRSTUVWXYZ \ ^ 'abcdefghijklmnopqrstuvwxyz{|}~
               _UJRH_______________________________abc_defghijklm_____
```

The ⇧ key on a VT100 generates ESC O A, the ⇩ generates ESC O B. The lead-in to this sequence, ESC O, is entered as $O in the first part of this section. The LEX function code U is entered under the A of the second portion of the field and J is entered under the B. This indicates that the function cursor up is to be performed when the sequence ESC O A is received and cursor down when the sequence ESC O B is received.

If a lower case letter or a number is inserted, LEX refers to the table specified in the function mapping part of the screen (see 8.1.5) to determine how the incoming sequence should be interpreted. For example on a VT100 key 0 on the numeric pad generates ESC O p. If a lower case d is entered under the p of the second portion of this field then the LEX operation defined for d in the fifth section will be performed when key 0 on the numeric pad is pressed.

For sequences of the form:

`Lead-in number ; number terminator`

LEX uses the number as an offset to index into the second part of the section. For example the F13 key on a VT200 generates:

`ESC [ 2 5 ~`

to map this to a line feed (LEX function J) an entry would be made as follows:

```
First Escape  0.......1.......2.......3....... ! #$%&'()*+,-./0123456789:;<=>?
Sequence $(*_ _________________________________J________________________________
               @ABCDEFGHIJKLMNOPQRSTUVWXYZ \ ^ 'abcdefghijklmnopqrstuvwxyz{|}~
               ________________________________________________________________
```

1.4 Control characters

The fourth section of this screen is:

```
Low ASCII chars ________________________________
```

This describes which LEX function is to be performed when a control character is received from the terminal.

When the CTRL key is held down and another key is depressed, a low value ASCII character is transmitted from the terminal to the host computer. For example CTRL A generates the character with the decimal value of 001, CTRL B generates 002 up to CTRL Z which generates the character whose decimal value is 026. 027 is the ESC character.

LEX uses this area of the form as a translate table with the first character corresponding to null and the second character corresponds to CTRL A. Entries made in this field define the LEX function to be carried out. This function may be an upper case character representing the LEX internal control function (as listed in Appendix B) or it may be a lower case character or a number which is mapped to a sequence of characters in the function mapping part of the screen (see 8.1.5).

For example:

```
Low ASCII chars _ABCDEFGHIJKLMN_P_R_TUVWXYb_____
```

This defines that LEX function A (insert a line) is to performed when $\boxed{\text{CTRL}}$A is typed at the terminal, and the function defined by b is to be performed when $\boxed{\text{CTRL}}$Z is typed.

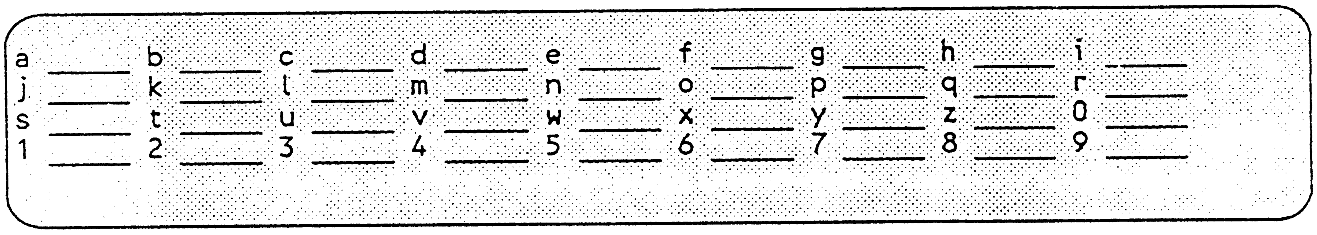
Some control characters are intercepted by the operating system and are never passed to LEX, for example $\boxed{\text{CTRL}}$S, $\boxed{\text{CTRL}}$Q. It is therefore not possible to map these characters to a LEX function. They should be mapped to an underline.

Note that if $\boxed{\text{CTRL}}$C is typed at a menu or in response to any question from LEX (e.g. Pause at head of form?) then LEX will respond in one of two ways depending on the mapping of the C position:

> if the C position is mapped to a C then LEX will perform $\boxed{\text{ESC}}$E
>
> if the C position is not mapped to a C then the $\boxed{\text{CTRL}}$C will be ignored.

8.1.5 Function on mappings

This fifth section is:



```
a _____ b _____ c _____ d _____ e _____ f _____ g _____ h _____ i _____
j _____ k _____ l _____ m _____ n _____ o _____ p _____ q _____ r _____
s _____ t _____ u _____ v _____ w _____ x _____ y _____ z _____ 0 _____
1 _____ 2 _____ 3 _____ 4 _____ 5 _____ 6 _____ 7 _____ 8 _____ 9 _____
```

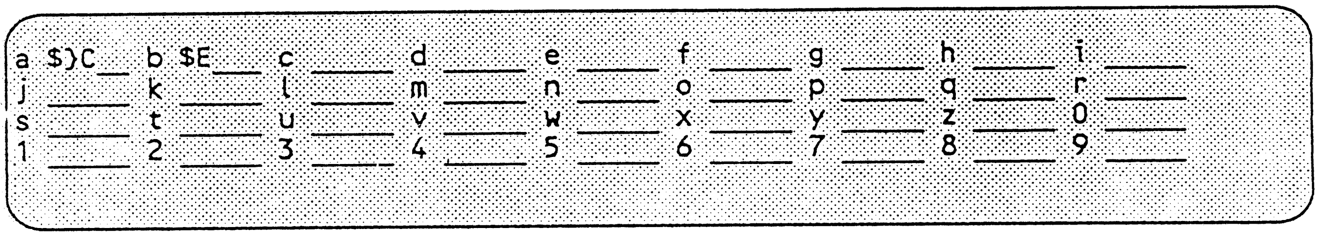
and is split into 36 sections of 5 character fields.

The information stored in each portion of this section is a symbolic representation of the characters that are to be used by LEX when the incoming sequence has been decoded. Each character in this area is transmitted as it is stored unless it has been previously defined in the first section of this screen as a visible equivalent.

References to these fields are made from the control characters section of the form (8.1.4) or the section linking received escape sequences to LEX functions (8.1.3.2).

An example of the use of this could be to translate the sequence generated by the key marked $\boxed{\text{PF3}}$ on a VT100 into the sequence required to create a system record on the VMF. This would involve the translation of the incoming sequence ESC O R into the internal sequence $\boxed{\text{ESC}}$]C to create a new record on the VMF. This could be achieved by defining $=027 and }=093 ($=(ESC) and }=]) in the visible equivalents section of the screen. The lead-in Escape sequence would be $O and under the R position of that sequence an a would be entered. $}C would be entered following the a in the function mappings as shown below.

If it was also required to cause (CTRL)Z to end an edit this would be done by entering b in the Low Value ASCII section of this form (see example above) and then defining $E for b



```
a $}C__ b $E___ c _____ d _____ e _____ f _____ g _____ h _____ i _____
j _____ k _____ l _____ m _____ n _____ o _____ p _____ q _____ r _____
s _____ t _____ u _____ v _____ w _____ x _____ y _____ z _____ 0 _____
1 _____ 2 _____ 3 _____ 4 _____ 5 _____ 6 _____ 7 _____ 8 _____ 9 _____
```

Each sequence is terminated by an underline, this allows sequences longer than 5 characters to be defined by continuing the sequence into the next field. This means that the following field may not be used, as it contains part of an escape sequence. Also if just five characters are used in a field then the first character of the next field must be an underline, effectively disabling the use of that field.

8.1.6 Control information

The final portion of this screen is

```
First Menu:________ Delay:____ VDU Table:*)_______ Write _
```

This comprises three fields. The final entry is the point at which the cursor must be positioned when creating or updating the record.

The first field is

```
First Menu:
```

This refers to the key of the first menu to be called up whenever this terminal type is invoked - either when LEX starts up or when menu option /U is actioned. If the specified name is not found on the file or this field is left blank, LEX automatically looks for **MENU as the first menu to be invoked.

The second field is

`Delay:`

This is necessary on some heavily loaded systems. It defines an internal delay invoked between reading characters of an incoming escape sequence. It is used to avoid the possibility of part of the sequence being lost with the result that spurious characters may be left in the body of a document being created or edited.

The delay is a wait loop and various values should be tried before settling on what appears to be the optimum for any given installation.

This value may be changed dynamically by using the ESC SD sequence. Typical values stored in this field are between 50 and 100. Values of less than 50 have little or no effect and values much greater than 200 significantly slow the response of the terminal.

Setting this value is only required where systems have inefficient keyboard I/O. It is not effective on most UNIX systems.

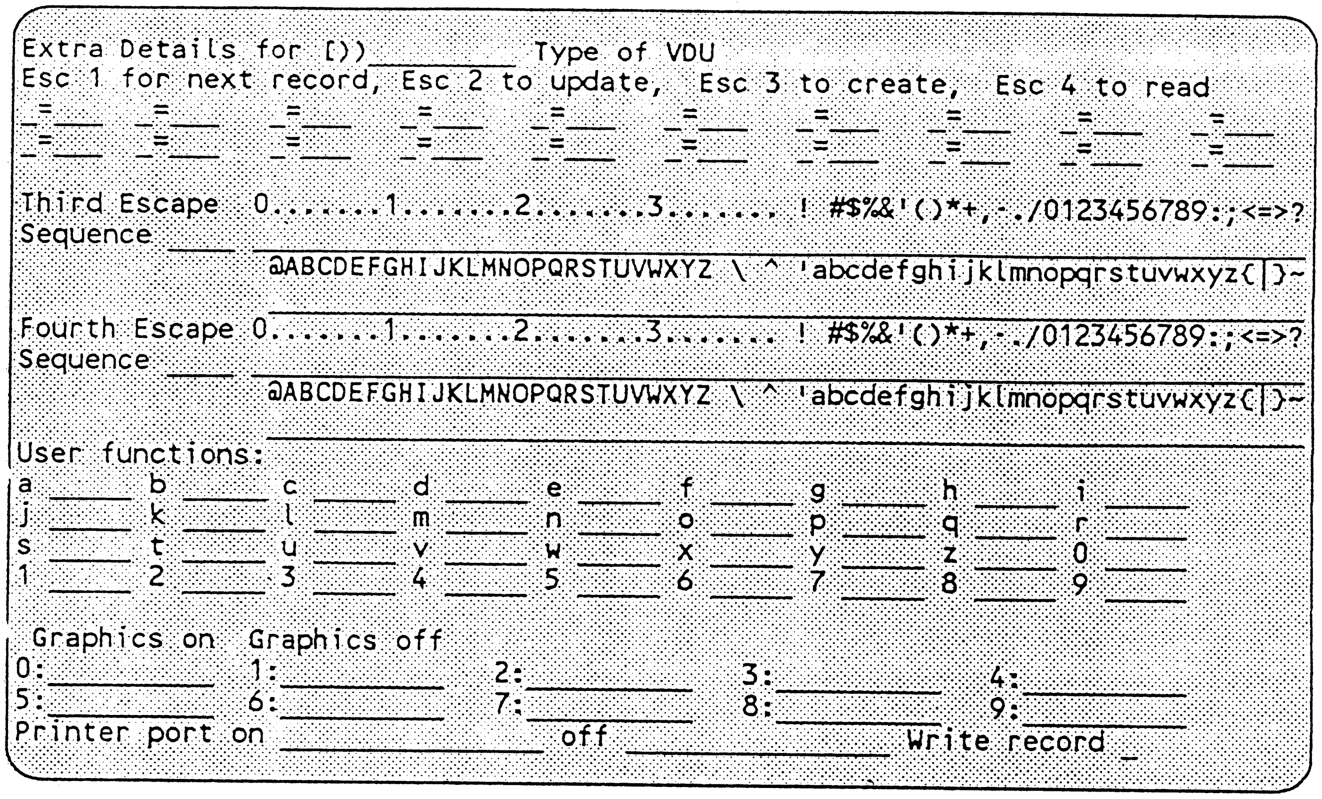
The third field is

`VDU Table:*)`

This refers to the relevant translation table used for displaying characters on the screen. This field contains the key of the relevant record. The contents of this record are described in the section on *VDUTABLE. No translation is carried out if this field is left blank.

8.2 FURTHER VDU CHARACTERISTICS - *VDUFORM2

This form is similar to *VDUFORM and allows extension records for the terminal description records described in 8.1 to be created. The form appears as:



```
Extra Details for [))________  Type of VDU
Esc 1 for next record, Esc 2 to update,  Esc 3 to create,  Esc 4 to read
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___

Third Escape  0.......1.......2.......3....... ! #$%&'()*+,-./0123456789:;<=>?
Sequence ____ _____________________________________________________________
                @ABCDEFGHIJKLMNOPQRSTUVWXYZ \ ^ 'abcdefghijklmnopqrstuvwxyz{|}~
                _______________________________________________________________
Fourth Escape 0.......1.......2.......3....... ! #$%&'()*+,-./0123456789:;<=>?
Sequence ____ _____________________________________________________________
                @ABCDEFGHIJKLMNOPQRSTUVWXYZ \ ^ 'abcdefghijklmnopqrstuvwxyz{|}~
                _______________________________________________________________
User functions:
a ____ b ____ c ____ d ____ e ____ f ____ g ____ h ____ i ____
j ____ k ____ l ____ m ____ n ____ o ____ p ____ q ____ r ____
s ____ t ____ u ____ v ____ w ____ x ____ y ____ z ____ 0 ____
1 ____ 2 ____ 3 ____ 4 ____ 5 ____ 6 ____ 7 ____ 8 ____ 9 ____

 Graphics on  Graphics off
0:_________  1:_________    2:_________    3:_________    4:_________
5:_________  6:_________    7:_________    8:_________    9:_________
Printer port on ____________ off ____________ Write record _
```

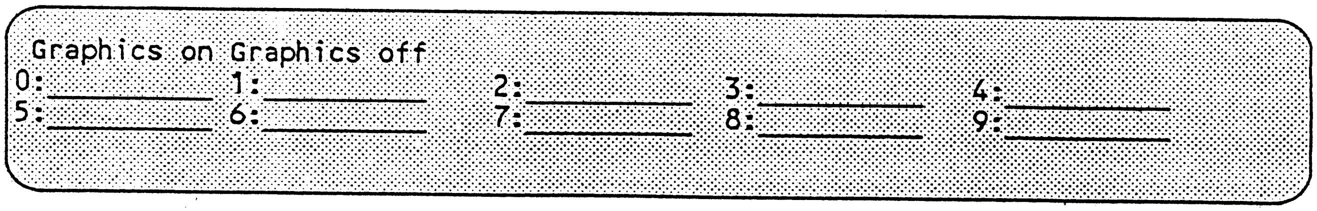
The key that is used for this record must commence with)) and then should be identical to the key of the associated record created using *VDUFORM. Thus if a record has been created with the key)VDUTEST using *VDUFORM then *VDUFORM2 should be used to create the extension record))VDUTEST.

The first portion of this screen after the key corresponds to the visible equivalents definition in *VDUFORM (8.1.). The definitions made in this section are local to this screen only and need not correspond with those defined for *VDUFORM.

The second portion of this form corresponds with the Escape sequences described for *VDUFORM (8.1.3). The third area corresponds to the function mappings (8.1.5).

The same lead in Escape sequences can appear in records created using *VDUFORM and *VDUFORM2.

The third portion of the form is



```
 Graphics on Graphics off
0:_________  1:_________    2:_________    3:_________    4:_________
5:_________  6:_________    7:_________    8:_________    9:_________
```

This is used to describe the sequences of characters which will be sent to the terminal when the escape sequence ESCssn is invoked (where n is a number in the range 0 to 9). Escape sequences entered here are also used for defining menu attributes (see Chapter 10).

Escape sequences 0 and 1 are usually used to switch into and out of the graphics or line drawing character set if one is supported.

Any characters defined at the head of the form as visible equivalents are translated before being sent to the terminal, all other characters are sent as defined. In other words, if $ is to be used as ESC then this must be set up as $=027 in the first section of the form. The sequences sent to the terminal are terminated by the first blank character, as shown by an underline, met in the sequence. Thus it is possible to extend the sequence beyond the ten characters allowed.

For example to select the line drawing character set on an ANSI terminal when the sequence ESCss0 is typed, the first entry would be set up as 0:$(0_______.

The escape sequences entered here may also be used for defining menu attributes by adding /S followed by three characters to the end of all menu lines for which attributes are required. The three characters must be in the range 0 to 9 and correspond to the escape sequences entered in this part of the form i.e.

```
[**menu     ./Smno
C           C - Create/Smno
E           E - Edit  /Smno
V           V - View  /Smno
P           P - Print /Smno
M*MENU      M - Menu  /Smno

              . - Select/Smno
```

For full details see Chapter 10.

The final portion of the form is:

```
Printer port on ______________ off ______________ Write record _
```

If printing is to be performed through the printer port, then the Escape sequences to switch the printer port on and off should be entered here. Selection of the printer port for printed output is determined in the **PNST system record by entering a Y in the VDU port field.

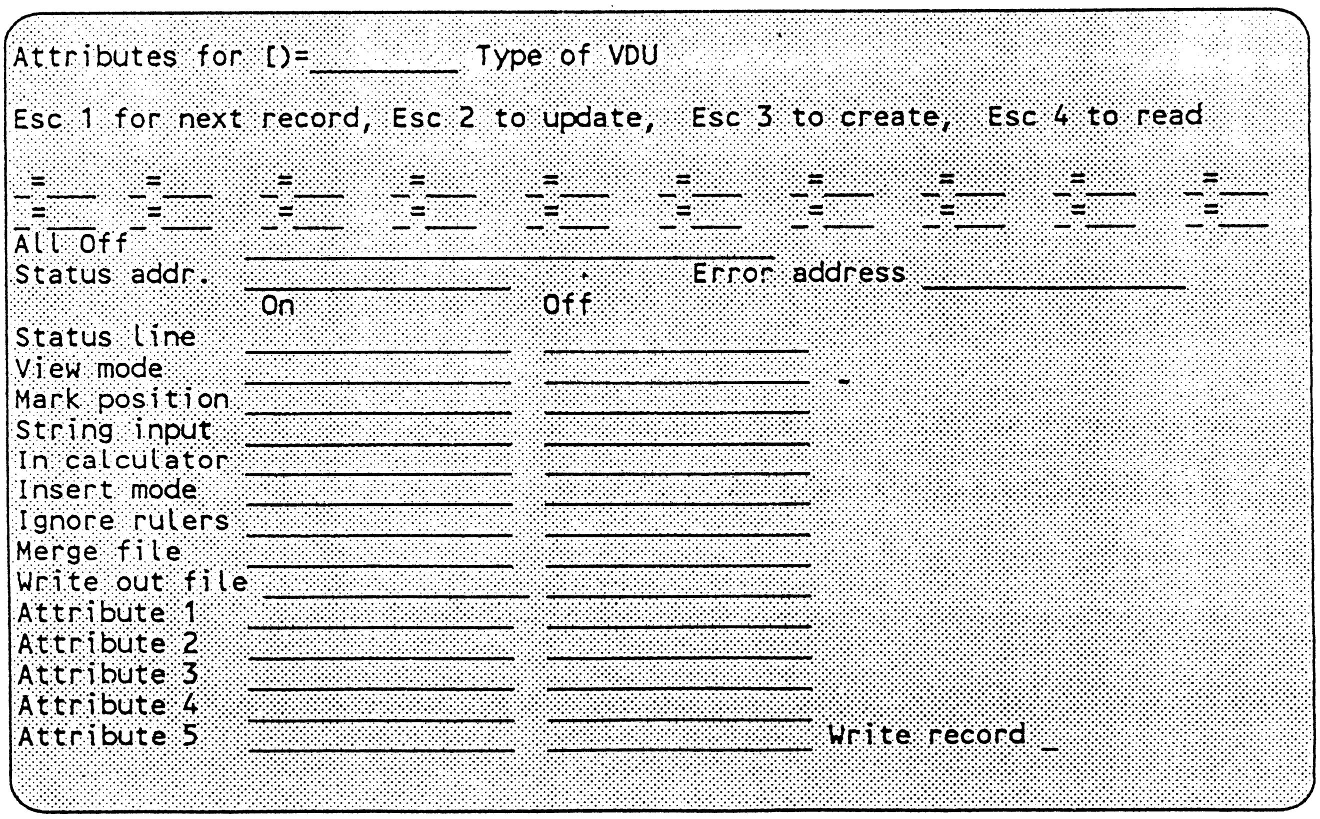
The sequences entered in these two fields may alternatively be entered in the Printer Set and Printer Reset fields of records defined using *PRTFORM.

8.3 VDU ATTRIBUTE DEFINITION - *VDUFORM3

This is an extension form for *VDUFORM and *VDUFORM2 and is used to create records which define the location of the status line (if required) and attributes.

The key that is used for this record must commence with)= and then should be identical to the second to tenth characters of the key of the associated record created using *VDUFORM. Thus if a record has been created with the key)VDUTEST using *VDUFORM then *VDUFORM3 should be used to create the extension record)=VDUTEST.

The form has the following format:



```
Attributes for [)=________  Type of VDU

Esc 1 for next record, Esc 2 to update,  Esc 3 to create,  Esc 4 to read

_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
All Off    ____________________________________
Status addr.   ________________       Error address ________________
               On                 Off
Status line    ________________   ________________
View mode      ________________   ________________
Mark position  ________________   ________________
String input   ________________   ________________
In calculator  ________________   ________________
Insert mode    ________________   ________________
Ignore rulers  ________________   ________________
Merge file     ________________   ________________
Write out file ________________   ________________
Attribute 1    ________________   ________________
Attribute 2    ________________   ________________
Attribute 3    ________________   ________________
Attribute 4    ________________   ________________
Attribute 5    ________________   ________________ Write record _
```

The form is divided into five sections described below.

8.3.1 Visible equivalents definition

The first part of the form allows local visible equivalents to be defined, as in *VDUFORM.

8.3.2 Switch off Attributes

`All Off ______________________`

The Escape sequence to turn off all attributes must be entered here. For attributes which do not have an escape sequence to switch an individual attribute off (for example the VT100 LEDs) then LEX will send this sequence and then reset the appropriate attributes.

It is important that this field is always entered even for those attributes which do have an 'off' Escape sequence.

8.3.3 Status and Error displays

`Status addr. ______________ Error address ______________`

LEX can optionally display a 32 character long status line showing the page number, line number, actual line number and character position of the cursor within the line. The status line can be a hardware status line provided by the VDU or a software generated at the foot of the screen.

This function is only applicable to ANSI standard VDUs.

`Status addr. ______________`

If a status line is required the Escape sequence to position the cursor on the required line should be entered in the Status Addr field. For example:

Put `$24;1H` in the `Status Address field`

For a software generated status line the VDU must support scrolling regions. The Escape sequence to define the scrolling region must be set up in the Set VDU Modes field in the associated *VDUFORM record and should include the whole screen except for the bottom line. Also the Lines field should be set to one less than than the screen size and the Reset field should include the sequence to unset scrolling regions. For example:

Put	$[1;23r	in the Set VDU field to set the scroll region
Put	$[r	in the Reset field to unset the scroll region
Put	23	in the Lines field

Additionally any forms which use the entire screen must be compressed by one line.

The status line is updated whenever carriage return, line feed, up arrow, down arrow, foot of screen or top of screen are performed.

Display of the status line is controlled by the system record **INST and also by the escape sequences (ESC)SL, (ESC)-SL and (ESC)CS.

Use of the status line degrades performance, particularly when scrolling through a document.

Error address

When LEX detects an error it beeps, subsequent actions depend on whether LEX is in show error mode and/or the error address field has been entered. Show error mode is controlled by the system record **INST and the Escape sequences (ESC)SE and (ESC)-SE.

When LEX is operating in show error mode the Error address field is used to position the cursor before displaying an error message when an error is detected.

If the Error address field is left blank then error messages are displayed on the line at which the cursor was positioned when the error occurred.

The Error address line is displayed on the same line as the status line. For example:

Put $24;39H in the Error Addr field

The error message requires 40 characters and the status message requires 32 characters. This leaves characters free for user defined displays.

8.3.4 Edit and view mode attributes

This section defines attributes which are used in edit and view mode as indicators on the screen and the keyboard. These attributes are displayed only on the screen or on the keyboard and are not stored as part of the document.

```
                 On                Off
Status line    ________________  ________________
View mode      ________________  ________________
Mark position  ________________  ________________
String input   ________________  ________________
In calculator  ________________  ________________
Insert mode    ________________  ________________
Ignore rulers  ________________  ________________
Merge file     ________________  ________________
Write out file ________________  ________________
```

For each of the above fields two Escape sequences may be entered one to turn a particular VDU attribute on and the other to turn it off. Note that if the same attribute is used for two different indicators then the attribute could be switched off when not appropriate. In this case leave the Off sequence blank and LEX will use the All Off sequence to turn the attribute off and then switch any relevant attributes back on again.

If there is no sequence to turn the attribute off leave the second field blank and LEX will use the All Off sequence and then reset the appropriate attributes.

`Status line`	If the status line is to be displayed with special attributes then the sequences to turn these on and off should be entered in the two Status Line fields.
`View mode`	The attribute selected is applied to the whole document.
`Mark posn`	Applies to the position marked at the start of a cut operation.
`String Input`	Applies to operations which require a string to be input, i.e. Find ESC F, ESC G, ESC -G, Repeat ESC X, Set Abbreviation ESC Sn and the status display ESC L.

`In calculator`	Applies to all entries made in calculator mode.
`Insert mode`	Used when insert mode is invoked by (ESC)I, the off sequence is sent when an exit from insert mode is performed, e.g. by (ESC)-I, (ESC)J, (ESC)E (ESC)SC(Z).
`Ignore rulers`	Used when ignore ruler mode is invoked by (ESC)SG, the off sequence is sent on exiting from this mode, e.g. by (ESC)-SG, (ESC)Mn, (ESC)Un, (ESC)E or (ESC)Z.

`Merge file and Write Out file`

Used when an input merge document is opened (ESC)MO and an output merge document is opened (ESC)WO respectively. The off sequence is sent when the document is closed by (ESC)MC, (ESC)WC, (ESC)E or (ESC)Z.

Entries for the first five fields i.e. Status line, View mode, Mark position, String input and In calculator will usually be Escape sequences which set VDU attributes such as inverse video or bold.

Entries for Insert mode, Ignore rulers, Merge file open and Extract file open will usually be Escape sequences to turn on the LEDs. Alternatively sequences to mark a position on the status line could be set up here.

The example below is set up for an ANSI VDU and selects the following attributes:

Status line	Bold
View mode	Reverse video
Mark posn	Bold
String Input	Underline
In calculator	Reverse video
Insert mode	LED 1
Ignore rulers	LED 2
Merge file	LED 3
Extract file	LED 4

```
                  On                   Off
Status line      $(1m______________   ______________
View mode        $(?5h_____________   $(?5l_________
Mark position    $(1m______________   ______________
String input     $(4m______________   ______________
In calculator    $(7m______________   ______________
Insert mode      $(1q______________   ______________
Ignore rulers    $(2q______________   ______________
Merge file       $(3q______________   ______________
Write out file   $(4q______________   ______________
```

in the example $ is the visible equivalent of escape and (is the visible equivalent of [.

8.3.5 VDU/printing attributes

This section is used to define attributes which are displayed on the screen when a document is being created or edited. The attributes are stored within the document and will be re-displayed when the document is next accessed. Attributes may also be printed if the appropriate *PRTFORM records are created. The attributes may take advantage of any features supported by the VDU such as embolden, underline, reverse video, alternate character sets or mathematical symbols.

```
                  On                   Off
Attribute 1      ________________    ________________
Attribute 2      ________________    ________________
Attribute 3      ________________    ________________
Attribute 4      ________________    ________________
Attribute 5      ________________    ________________   Write record _
```

For each of the above fields two Escape sequences may be entered, one to turn a particular VDU attribute on and the other to turn it off. Note that if the same attribute is used for two different indicators then the attribute could be switched off when not appropriate.

If there is no sequence to turn the attribute off leave the second field blank and LEX will use the All Off sequence and then reset the appropriate attributes.

The use of attributes is controlled by Escape sequences

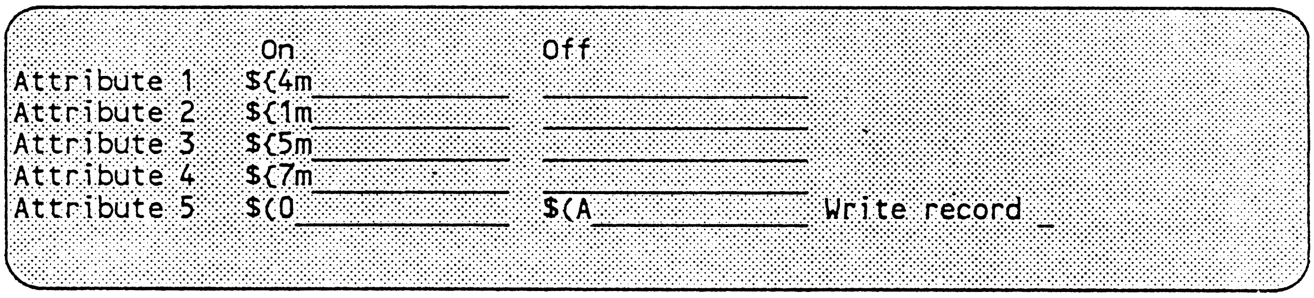
(ESC)An	Turn on attribute n
(ESC)Bn	Turn off attribute n
(ESC)Cn	Toggle attribute n on/off

ESC AX	Select combination of attributes (x may be A to O)
ESC BW	Apply current attributes to a word
ESC BL	Apply current attributes to a line
ESC BU	Show current attributes in pictorial form
ESC BT	Show attributes of the current line

Full descriptions of these escape sequences are given in Appendix C.

The example below is set up for an ANSI VDU and selects the following attributes:

Attribute 1	Underline
Attribute 2	Bold
Attribute 3	Blink
Attribute 4	Reverse video
Attribute 5	Select line drawing



```
             On                  Off
Attribute 1  ${4m___________     ______________
Attribute 2  ${1m___________     ______________
Attribute 3  ${5m___________     ______________
Attribute 4  ${7m___________     ______________
Attribute 5  $(0____________     $(A___________  Write record _
```

in the example $ is the visible equivalent of escape and { is the visible equivalent of [.

It may be that an attribute which is required for printing cannot be represented on the screen, for example super-script and sub-script. In this case any unused attribute should be selected to represent that the attribute has been selected. The escape sequences to cause the printer to move up half a line and down half a line should be entered in the appropriate attributes of the records created using *PRTFORM.

Attributes may be used additively, for example if attribute 1 has been selected and attribute 2 is also selected then in the example above text will be both underlined and bold.

Note the following points when using attributes:

1) Attributes are ignored in abbreviations or any string input such as file names, find, repeat loops, etc.

2) Find operations (ESC F, ESC G, ESC -G) ignore attributes when searching.

3) Partial memory file reads and writes must not be used if the VMF record contains attributes.

4) The internal buffer sort discards attributes before sorting.
5) The suppress linefeed character (usually tilde) should not be used on lines containing attributes.
6) In documents created using menu switch option /L (list processing with mail merge) attributes will be printed, unless justification is required in which case they will be discarded.
7) LEX files with attributes are not standard text files as LEX puts a special terminator at the end of the text and then follows it by a representation of the attributes. This visual representation of the attributes is shown below and in Appendix I:

Thus a string of text has the following form in a document:

```
string<terminator><CR><LF>attribute line<CR><LF>
```

Trailing spaces are dropped from the string and attribute lines. This means that attribute lines may be shorter than prime lines.

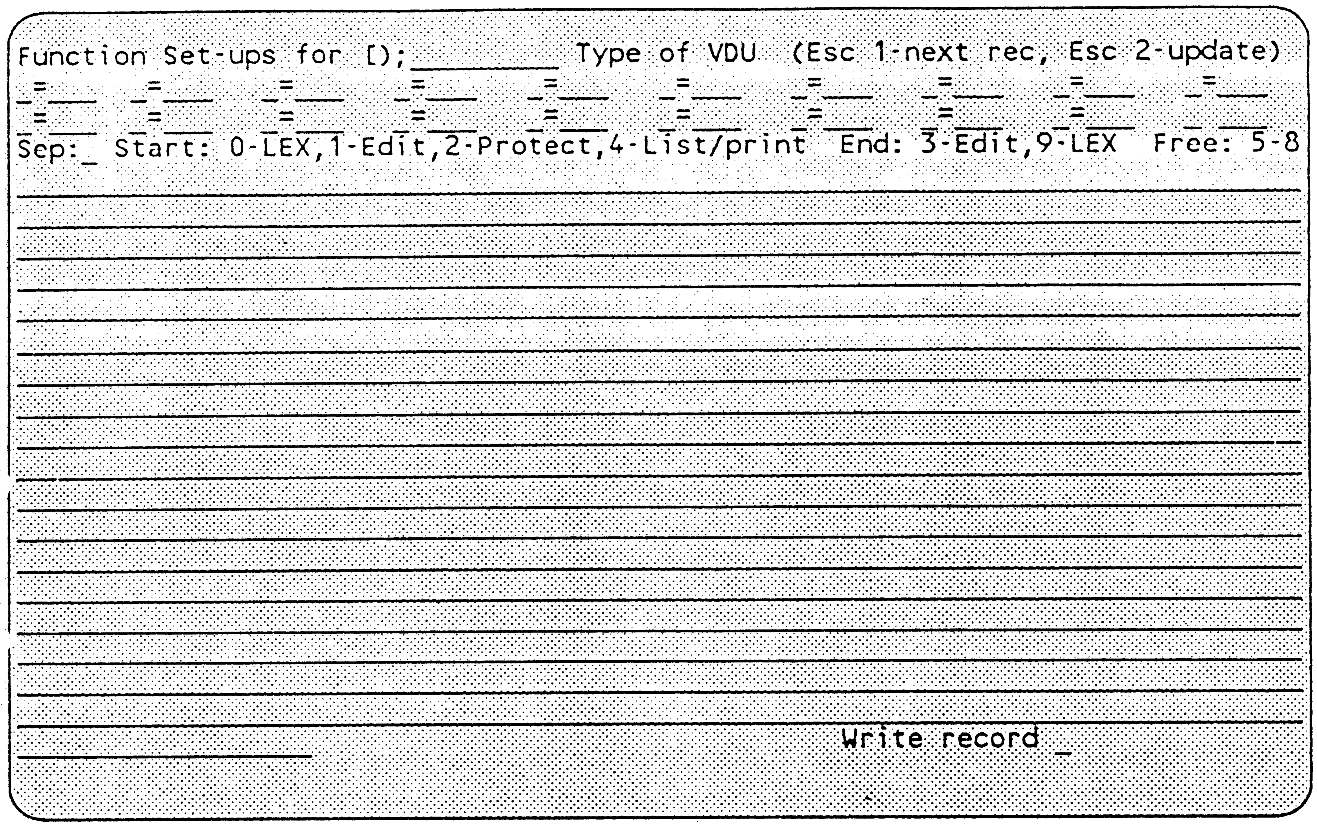
The default setting for the terminator is Octal 377 (Decimal 255). However this can be changed to a different character (e.g. a tilde ~) in the **MONT system record (see section 5.1). This is useful if the file is to be accessed by a different editor or if the document is to be printed on a printer which does not support attributes.

8.4 LONG VDU ESCAPE SEQUENCES - *VDUFORM4

This is used to allow long escape sequences to be sent to the VDU at various strategic points within LEX. The most common use of this will be to load function keys with Escape sequences and/or help facilities.

The key that is used for this record must commence with); and the key should be identical to the key of the associated record created using *VDUFORM. Thus if a record has been created with the key)VDUTEST using *VDUFORM then *VDUFORM4 should be used to create the extension record);VDUTEST.

The form has the following format:



```
Function Set-ups for ();________ Type of VDU  (Esc 1-next rec, Esc 2-update)
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
Sep:_ Start: 0-LEX,1-Edit,2-Protect,4-List/print  End: 3-Edit,9-LEX  Free: 5-8
                                                  Write record _
```

The form is divided into three sections described below.

8.4.1 Visible equivalents definition

The first part of the form allows local visible equivalents to be defined, as in *VDUFORM.

8.4.2 Separator character

`Sep:_`

The third part of the form defines up to ten Escape sequences each separated by a user defined character. The character to be used as the separator must be entered here.

8.4.3 Escape sequences

Up to ten Escape sequences may be entered in this section of the form, each separated by the separator character defined above. The Escape sequences are sent to the VDU at strategic points within LEX and by the user issued command $\boxed{\text{ESC}}$STn where n is 0 to 9.

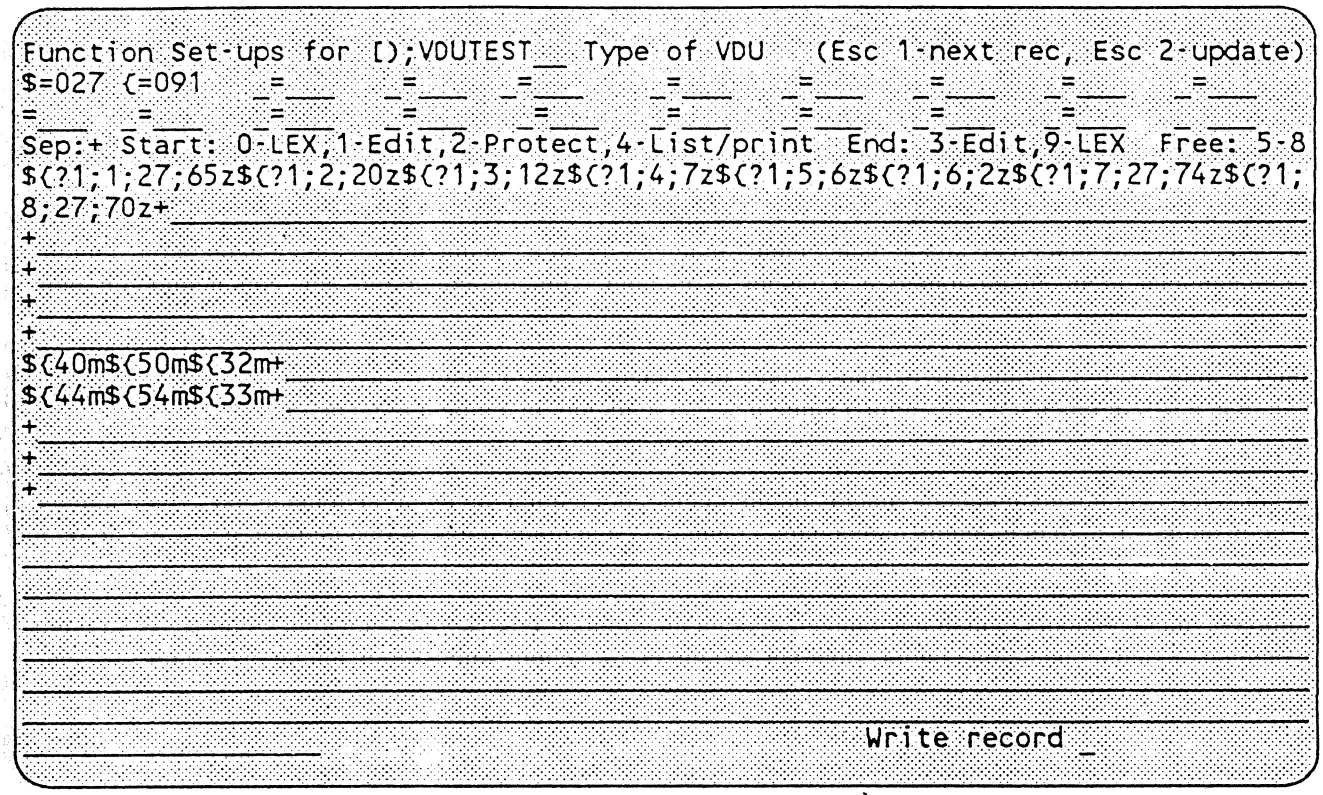
```
Sep:_  Start: 0-LEX,1-Edit,2-Protect,4-List/print  End: 3-Edit,9-LEX Free: 5-8
```

Escape sequence	Points at which Escape sequence is issued
0	At LEX start up or after /U. At start up the sequence is sent after any sequences entered in the Set VDU Modes field in *VDUFORM
1	Just prior to editing or creating a document
2	On entering protect mode when an ESC SP is issued
3	After exiting from editing or creating a document
4	Prior to printing
5-8	Free, available for use by the user
9	On exiting from LEX, it is sent after any sequences entered in the Reset VDU field in *VDUFORM.

Trailing underlines after a separator are ignored so that new sequences can start on a new line.

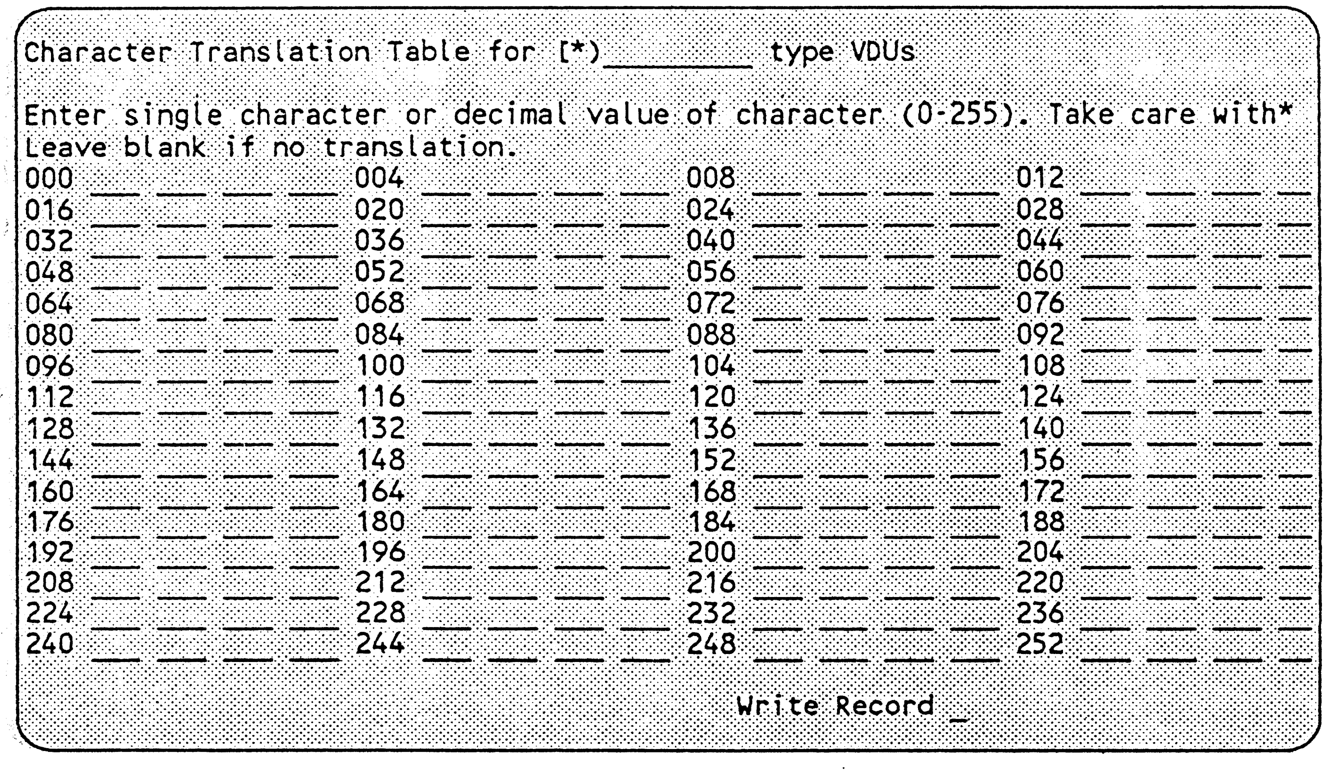
The example below shows a form in which three escape sequences have been entered:

Sequence 0	Downloads LEX commands into the function keys
Sequence 5	Sets colour as green on black
Sequence 6	Sets colour as yellow on blue



```
Function Set-ups for [);VDUTEST__ Type of VDU   (Esc 1-next rec, Esc 2-update)
$=027 {=091   _=___   _=___   _=___    _=___   _=___   _=___   _=___   _=___
=___  _=___  _=___   _=___   _=___   _=___   _=___   _=___   _=___
Sep:+ Start: 0-LEX,1-Edit,2-Protect,4-List/print  End: 3-Edit,9-LEX  Free: 5-8
${?1;1;27;65z${?1;2;20z${?1;3;12z${?1;4;7z${?1;5;6z${?1;6;2z${?1;7;27;74z${?1;
8;27;70z+_____________________________________________________________________
+_____________________________________________________________________________
+_____________________________________________________________________________
+_____________________________________________________________________________
+_____________________________________________________________________________
${40m${50m${32m+______________________________________________________________
${44m${54m${33m+______________________________________________________________
+_____________________________________________________________________________
+_____________________________________________________________________________
+_____________________________________________________________________________
______________________________________________________________________________
______________________________________________________________________________
______________________________________________________________________________
______________________________________________________________________________
______________________________________________________________________________
______________________________________________________________________________
_________________                                     Write record _
```

8.5 VDU CHARACTER TRANSLATION TABLE - *VDUTABLE



```
Character Translation Table for [*)________ type VDUs

Enter single character or decimal value of character (0-255). Take care with*
Leave blank if no translation.
000 ___ ___ ___ ___ 004 ___ ___ ___ ___ 008 ___ ___ ___ ___ 012 ___ ___ ___ ___
016 ___ ___ ___ ___ 020 ___ ___ ___ ___ 024 ___ ___ ___ ___ 028 ___ ___ ___ ___
032 ___ ___ ___ ___ 036 ___ ___ ___ ___ 040 ___ ___ ___ ___ 044 ___ ___ ___ ___
048 ___ ___ ___ ___ 052 ___ ___ ___ ___ 056 ___ ___ ___ ___ 060 ___ ___ ___ ___
064 ___ ___ ___ ___ 068 ___ ___ ___ ___ 072 ___ ___ ___ ___ 076 ___ ___ ___ ___
080 ___ ___ ___ ___ 084 ___ ___ ___ ___ 088 ___ ___ ___ ___ 092 ___ ___ ___ ___
096 ___ ___ ___ ___ 100 ___ ___ ___ ___ 104 ___ ___ ___ ___ 108 ___ ___ ___ ___
112 ___ ___ ___ ___ 116 ___ ___ ___ ___ 120 ___ ___ ___ ___ 124 ___ ___ ___ ___
128 ___ ___ ___ ___ 132 ___ ___ ___ ___ 136 ___ ___ ___ ___ 140 ___ ___ ___ ___
144 ___ ___ ___ ___ 148 ___ ___ ___ ___ 152 ___ ___ ___ ___ 156 ___ ___ ___ ___
160 ___ ___ ___ ___ 164 ___ ___ ___ ___ 168 ___ ___ ___ ___ 172 ___ ___ ___ ___
176 ___ ___ ___ ___ 180 ___ ___ ___ ___ 184 ___ ___ ___ ___ 188 ___ ___ ___ ___
192 ___ ___ ___ ___ 196 ___ ___ ___ ___ 200 ___ ___ ___ ___ 204 ___ ___ ___ ___
208 ___ ___ ___ ___ 212 ___ ___ ___ ___ 216 ___ ___ ___ ___ 220 ___ ___ ___ ___
224 ___ ___ ___ ___ 228 ___ ___ ___ ___ 232 ___ ___ ___ ___ 236 ___ ___ ___ ___
240 ___ ___ ___ ___ 244 ___ ___ ___ ___ 248 ___ ___ ___ ___ 252 ___ ___ ___ ___

                                           Write Record _
```

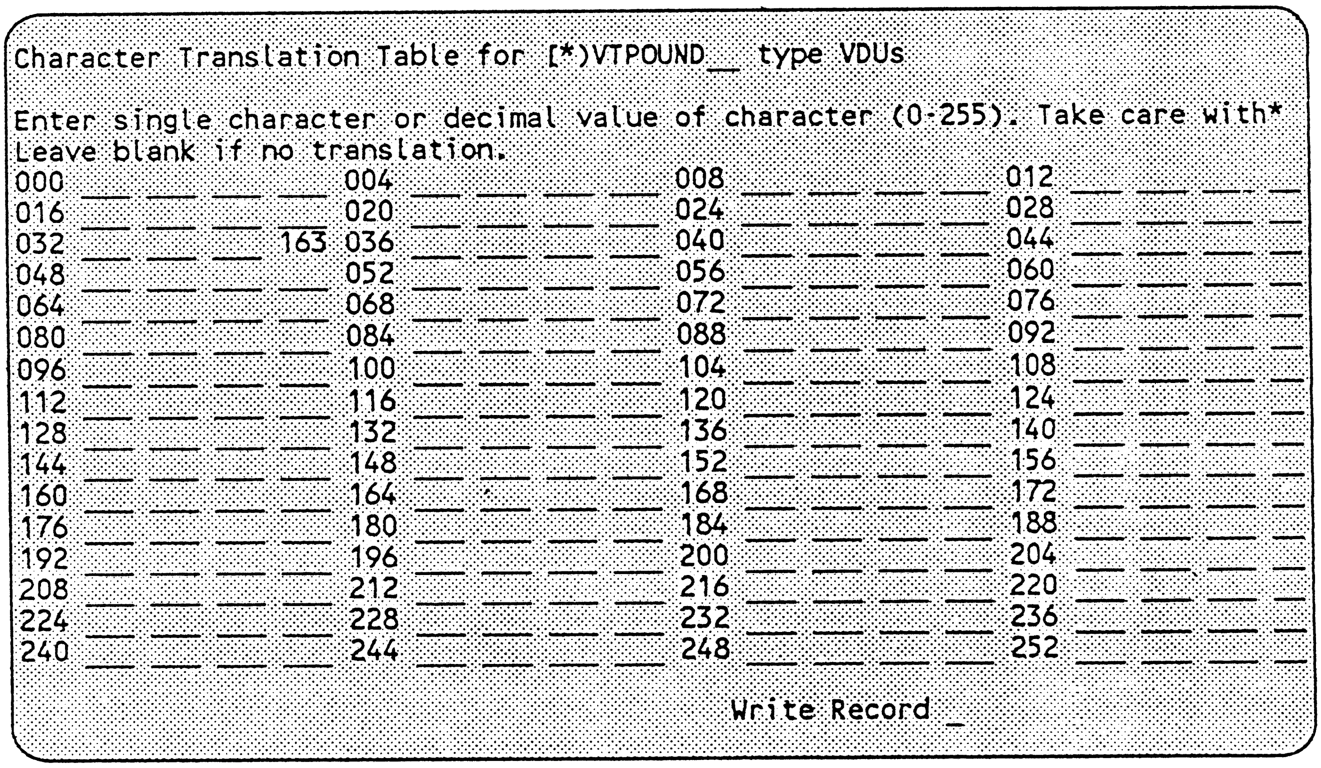
Records created using this form are used to translate characters sent from LEX to the terminal.

The keys of records created using this form commence with *). The record is used to translate characters if the key has been entered in the VDU Table field in the terminal description record (see 8.1.6).

The table is required in installations which have both 7 and 8 bit environments, e.g. a mixture of VT100 and VT200 terminals.

The numbers in the table represent the decimal value of characters as stored internally by LEX. The character to be displayed is entered in the unprotected part of the form corresponding to the internal value of the character. Characters may be entered as decimal values or as actual characters. Only characters to be translated need to be entered.

As an example the VT200 shift and 3 key generates decimal 35 and displays the hash sign. To cause this to display a pound sign which has a decimal value of 163 the table would be set up as follows:



```
Character Translation Table for [*)VTPOUND__ type VDUs

Enter single character or decimal value of character (0-255). Take care with*
Leave blank if no translation.
000 ___ ___ ___ ___ 004 ___ ___ ___ ___ 008 ___ ___ ___ ___ 012 ___ ___ ___ ___
016 ___ ___ ___ ___ 020 ___ ___ ___ ___ 024 ___ ___ ___ ___ 028 ___ ___ ___ ___
032 ___ ___ ___ 163 036 ___ ___ ___ ___ 040 ___ ___ ___ ___ 044 ___ ___ ___ ___
048 ___ ___ ___ ___ 052 ___ ___ ___ ___ 056 ___ ___ ___ ___ 060 ___ ___ ___ ___
064 ___ ___ ___ ___ 068 ___ ___ ___ ___ 072 ___ ___ ___ ___ 076 ___ ___ ___ ___
080 ___ ___ ___ ___ 084 ___ ___ ___ ___ 088 ___ ___ ___ ___ 092 ___ ___ ___ ___
096 ___ ___ ___ ___ 100 ___ ___ ___ ___ 104 ___ ___ ___ ___ 108 ___ ___ ___ ___
112 ___ ___ ___ ___ 116 ___ ___ ___ ___ 120 ___ ___ ___ ___ 124 ___ ___ ___ ___
128 ___ ___ ___ ___ 132 ___ ___ ___ ___ 136 ___ ___ ___ ___ 140 ___ ___ ___ ___
144 ___ ___ ___ ___ 148 ___ ___ ___ ___ 152 ___ ___ ___ ___ 156 ___ ___ ___ ___
160 ___ ___ ___ ___ 164 ___ ___ ___ ___ 168 ___ ___ ___ ___ 172 ___ ___ ___ ___
176 ___ ___ ___ ___ 180 ___ ___ ___ ___ 184 ___ ___ ___ ___ 188 ___ ___ ___ ___
192 ___ ___ ___ ___ 196 ___ ___ ___ ___ 200 ___ ___ ___ ___ 204 ___ ___ ___ ___
208 ___ ___ ___ ___ 212 ___ ___ ___ ___ 216 ___ ___ ___ ___ 220 ___ ___ ___ ___
224 ___ ___ ___ ___ 228 ___ ___ ___ ___ 232 ___ ___ ___ ___ 236 ___ ___ ___ ___
240 ___ ___ ___ ___ 244 ___ ___ ___ ___ 248 ___ ___ ___ ___ 252 ___ ___ ___ ___

                                             Write Record _
```

The table is used to translate characters sent from LEX to the display. The inverse table, which translates characters sent from the keyboard to LEX, is generated automatically from the table.

Chapter 9. PRINTER HANDLING

LEX can be used with most types of printing terminal even though printers differ widely in the facilities they offer and the commands to which they respond. Because of these differences all communication with printers from LEX is carried out through tables which may easily be changed by the user.

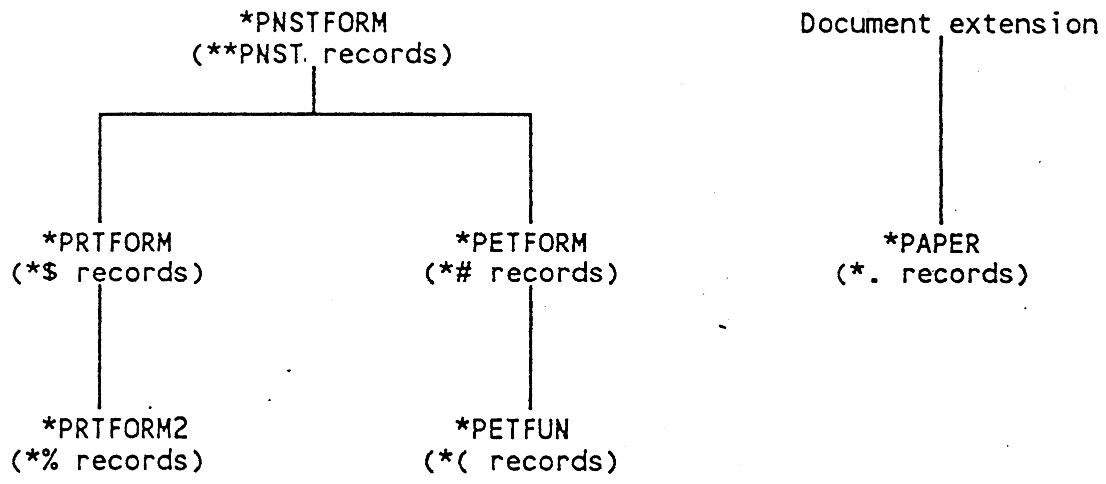
These tables are set up as system records on the VMF and are created and updated using the forms *PNSTFORM, *PRTFORM, *PRTFORM2, *PETFORM, *PETFUN and *PAPER. Setting up some of these records is necessary when configuring a system, whilst others are more a feature of particular applications. For completion they are all summarised below.

*PNSTFORM	This form is used to define the **PNST system record which describes the installation details for up to 10 printers. It contains the Which Printer prompt which appears when the print option is selected, and also defines characteristics, device types and names of records containing further details for each printer (see Chapter 5).
*PRTFORM	Defines escape sequences which may be sent to a printer for: initial set-up; switching on special attributes (e.g. underlining, line drawing character sets); setting up special form feeds (e.g. for double sheet feeders); printing special characters.
*PRTFORM2	Allows combinations of attributes to be defined for the ESC AX and ESC BX sequences.
*PETFORM	Acts as a translate table for characters held within LEX and the characters to be sent to the printer. The petal form records are mainly used for daisy wheel printers.
*PETFUN	A character being sent to the printer can be translated to a string of characters using petal function records.
*PAPER	Defines document extension details which include default rulers and keystores, as well as printing details such as left margin, page sizes and widow/orphan factors. These details are discussed in Chapter 13 of the User Manual.

Descriptions of these tables are given in this chapter or as indicated above.

The printer control commands which are used to control pagination, titles, footers, footnotes, line spacing, print control characters and pause points during printing are described in Appendix F.

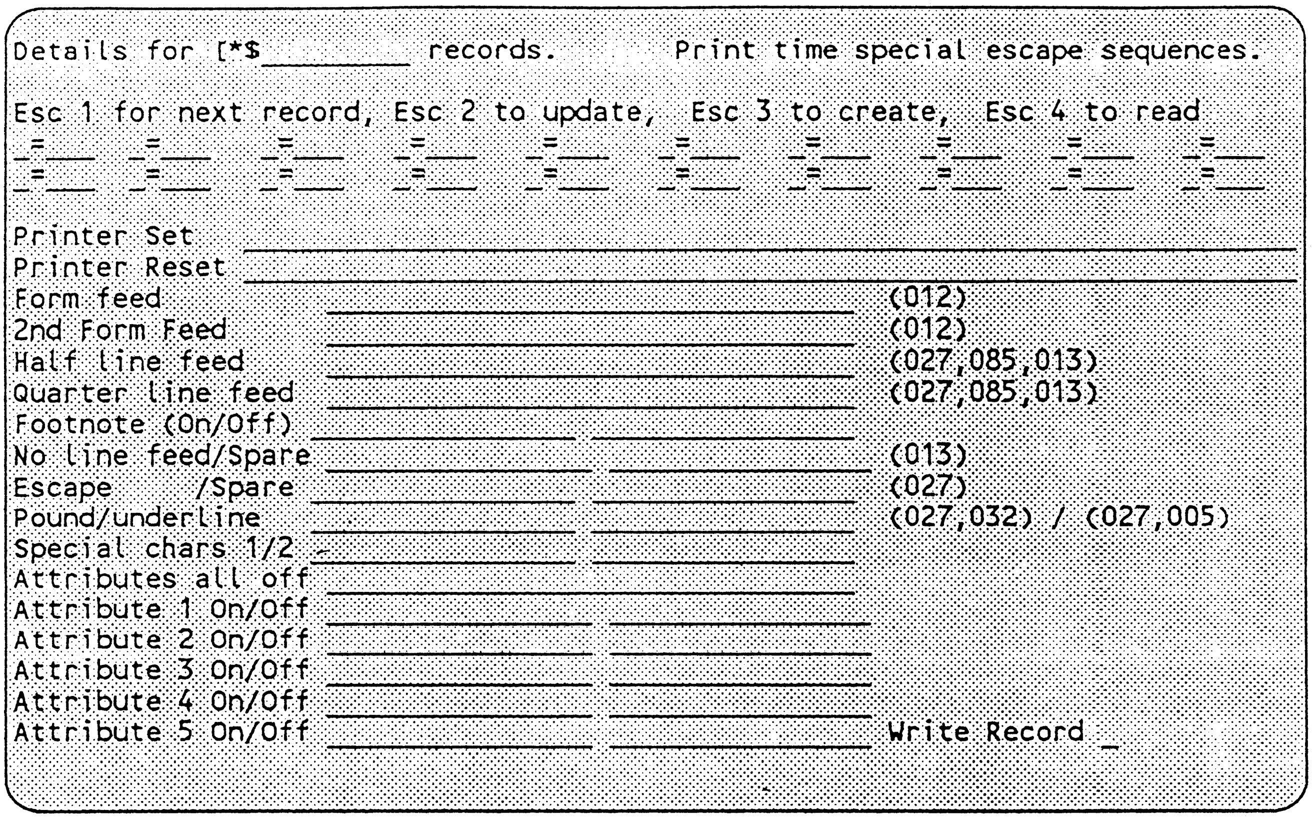
At start up time LEX refers to the **PNST system record to determine which printer characteristics records should be used. The relationship between the forms is summarised below:



9.1 ESCAPE SEQUENCES AND ATTRIBUTES - *PRTFORM

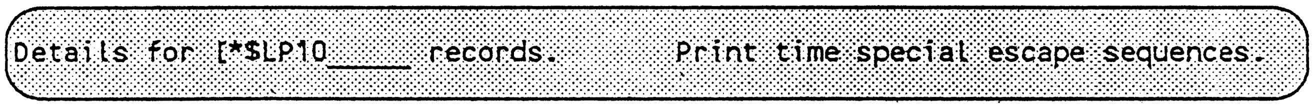
This form is used to create records which cause escape sequences and special characters to be sent to the printer. Records are created with a key which commences *$.

The form has the following format:



```
Details for [*$________ records.          Print time special escape sequences.

Esc 1 for next record, Esc 2 to update,  Esc 3 to create,  Esc 4 to read
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___

Printer Set    _____________________________________________________________
Printer Reset  _____________________________________________________________
Form feed           _____________________________  (012)
2nd Form Feed       _____________________________  (012)
Half line feed      _____________________________  (027,085,013)
Quarter line feed   _____________________________  (027,085,013)
Footnote (On/Off)  _______________  ______________
No line feed/Spare ______________  ______________  (013)
Escape    /Spare   _______________  ______________  (027)
Pound/underline    _______________  ______________  (027,032) / (027,005)
Special chars 1/2 _______________  ______________
Attributes all off _____________________________
Attribute 1 On/Off ______________  ______________
Attribute 2 On/Off ______________  ______________
Attribute 3 On/Off ______________  ______________
Attribute 4 On/Off ______________  ______________
Attribute 5 On/Off ______________  ______________  Write Record _
```

Up to 8 characters may be entered as the key of the record, e.g.



```
Details for [*$LP10____ records.          Print time special escape sequences.
```

This key is also entered in the Name field of the **PNST installation record. If a key is not specified in the Name field of the **PNST record then the defaults shown in brackets on the right hand side of the form are assumed. Values are shown in decimal; e.g. 012 is form feed.

The form is divided into two sections described below.

9.1.1 Visible equivalents definition

The first part of the form allows local visible equivalents to be defined, in the same way as *VDUFORM (see Chapter 8). For example the definition below sets up visible equivalents as:

$	escape
{	[
&	carriage return
*	null, with top bit set
!	form feed

```
$=027 {=091  &=013  *=128  !=012  _=___  _=___  _=___  _=___  _=___
_=___ _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
```

9.1.2 Escape Sequences and Special Characters

The second part of the form defines escape sequences and special characters to be sent to the printer.

LEX sends the escape sequence exactly as entered and does not append any line terminators. Therefore if a carriage return and/or a line feed is required after an escape sequence then it should be included in the sequence defined in this form.

If no sequence is entered in a field a default value is not assumed. The defaults shown in brackets at the right hand side of the form are only assumed if no *$ record is specified in the Name field in the **PNST record.

Printer Set and Printer Reset

The printer set sequence is sent to the printer as each document is about to be printed. Printer reset is sent at the end of each document. Each sequence may contain 64 characters.

For example to switch to 12 pitch certain printers require the following sequence to be sent:

`ESC [ 2 w`

to reset to 10 pitch the sequence is:

`ESC [ 0 w`

assuming `$=027` and `{=091` had been entered in the visible equivalents section to define `$` as ESC and `{` as `[`, then the entries in the fields to cause a document to be printed at 12 pitch and to reset to 10 pitch at the end would be:

```
Printer Set   ${2w___________________________________________________
Printer Reset ${0w___________________________________________________
```

Form Feed

This field and the Second Form Feed field contain the character or characters which cause the printer to do a form feed. Usually this will be the FF character (decimal value 012).

LEX does not do a form feed before printing a document, therefore, before printing commences a sheet must already be in position for printing.

If a double hopper sheet feeder is used then this field and the 2nd form feed field control the hopper from which the next sheet of paper is selected.

LEX uses the second form feed (see below) after each page of a document except after the last page when the form feed is used. This ensures that a sheet from the main hopper is left in position for the first sheet of the next document.

A document in this context includes:

- any document printed using the print option
- each copy printed using the multiple print
- a list processing report
- a mailshot letter

For example if `!=012` had been entered in the visible equivalents section then the usual entry for this field would be:

```
Form feed            !_________________________________   (012)
```

Second Form Feed

This is used in conjunction with the Form Feed sequence above. The sequence defined here is used after each page of a document except after the last page when the Form Feed is used.

Half Line Feed

The .S printer control line (see Appendix F) controls the line spacing to be used in a document. When this defines that half line spacing is required then the sequence entered in this field is sent to the printer.

For some printers the sequence ESC U causes the printer to perform a half line feed. This would be entered as:

```
Half line feed      $U&____________________________      (155,085,013)
```

where $ and & are visible equivalents of escape and carriage return respectively.

Quarter Line Feed

The .S printer control line (see Appendix F) controls the line spacing to be used in a document. When this defines that quarter line spacing is required then the sequence entered in this field is sent to the printer.

Footnote (On/Off)

The 12th operand in the .C control line or in the **INST installation record defines the character which is used to introduce a footnote.

Two escape sequences are entered in the footnote field, the first is sent before the footnote number is printed and the second is sent after it has been printed.

Typically the first sequence will be to move the paper half a line and the second sequence to move it back to its original position, for example if ESC K causes a half line feed down and ESC L causes a half line feed up then the field would be entered as:

```
Footnote (On/Off)  $L______________  $K______________
```

No line feed

The 2nd operand in the .C control line or in the **INST installation record defines a character to be used to indicate that no line feed is to be performed at the end of a line. Usually the character ~ is used for this. This field contains the characters which cause the printer to return to the start of the current line without performing a line feed. This will invariably be carriage return.

The ability to suppress the line feed after a line has been printed means that the next line is printed on top of the line just printed. This is useful for emboldening, underlining and strike through on printers which do not support these features as part of the hardware. For printers which do support these features the attributes section of this record should be used for underlining, emboldening, etc.

For example:

```
            This is useful for emboldening and underlining~
                               emboldening     ___________
```

will print as:

This is useful for **emboldening** and <u>underlining</u>

```
Pound
```

The fourth operand in the .C control line or in the **INST installation record defines a character to be used to print a pound sign. This is only needed for those printers which require a special escape sequence to print a pound sign. The sequence entered in this field is sent to the printer instead of the designated character.

For example if the printer requires ESC space to be sent to print a pound sign then, assuming $ had been set up as the visible equivalent of escape, this field would be set as:

```
Pound/underline     $ _____________ . _______________   (155,032) / (155,005)
```

If .c ...#.... is defined in the document then all occurrences of # will appear as a pound sign on the printer.

```
Underline
```

The fifth operand in the .C control line or in the **INST installation record defines a character to be used to print a double underline. This is only needed for those printers which require a special escape sequence to print a double underline. The escape sequence entered in this field is sent to the printer for each occurence of this character.

```
Special character 1 and 2
```

The tenth and eleventh operands in the .C line or in the **INST installation record identify two characters which are replaced at print time by the escape sequences entered in these two fields.

For example to set 12 pitch printing, some printers require the sequence ESC [2 w. To set the pitch to 10 pitch the sequence ESC [0 w is used. The special character fields would be set up as follows:

```
Special chars 1/2     $[2w____________   $[0w____________
```

then if .c ....<> is defined in a document any text enclosed introduced by < and terminated by > would be printed in 12 pitch.

Attributes

This section is used to define escape sequences which control printer attributes such as underline, embolden, select alternate character sets, etc. If the printer does not support escape sequences then this section does not apply and alternative means of underlining, emboldening, etc must be used as described above.

These attributes are set into a document during edit or creation of a document using the VDU attributes defined with *VDUFORM3 (see Chapter 8) and the attribute escape sequences (see ESC An, ESC Bn, ESC Ax, ESC Bu, ESC BL in Appendix C). The five attributes which may be defined in *VDUFORM3 correspond to the five printer attributes defined here.

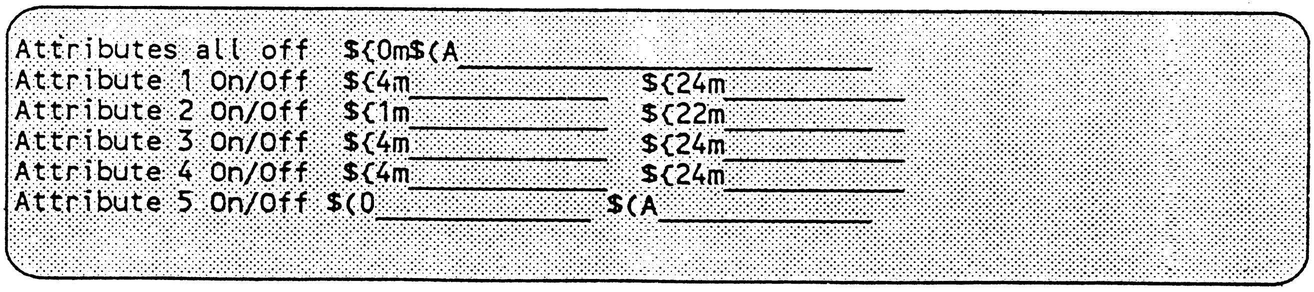
The first field must be set to the the escape sequence which switches off all attributes.

For each of the five attribute fields two escape sequences may be entered, one to turn an attribute on and the other to turn it off.

For example in Section 8.3.5 the VDU attributes were set up as:

Attribute 1	Underline
Attribute 2	Bold
Attribute 3	Blink
Attribute 4	Reverse Video
Attribute 5	Select line drawing set

Not all of these can be reproduced on the printer and so attributes 3 and 4 are printed underlined, therefore the corresponding printer attributes for an ANSI standard printer would be set up as:



```
Attributes all off  $(0m$(A___________________________
Attribute 1 On/Off  $(4m________________  $(24m________________
Attribute 2 On/Off  $(1m________________  $(22m________________
Attribute 3 On/Off  $(4m________________  $(24m________________
Attribute 4 On/Off  $(4m________________  $(24m________________
Attribute 5 On/Off $(0________________ $(A________________
```

For printers which do not have escape sequences to switch on underlining and emboldening, 1111 should be entered in Attribute 1 and 2222 in Attribute 2.

```
Attributes all off      ____________________________
Attribute 1 On/Off  1111__________    ______________
Attribute 2 On/Off  2222__________    ______________
Attribute 3 On/Off  ______________    ______________
Attribute 4 On/Off  ______________    ______________
Attribute 5 On/Off  ______________    ______________
```

Provided there is also an entry in the No Line Feed field (see above) LEX will then perform the underlining and/or emboldening by overprinting.

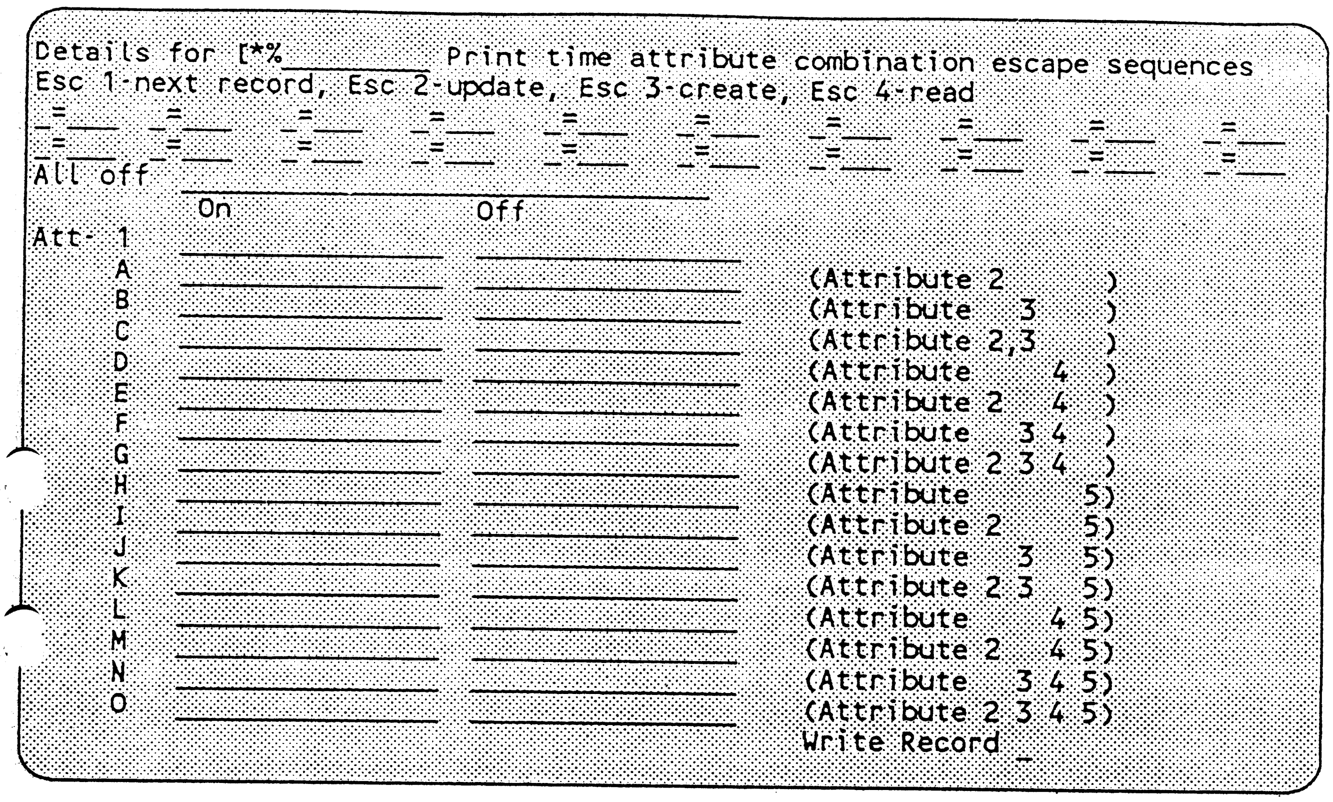
9.2 COMBINATIONS OF ATTRIBUTES - *PRTFORM2

If specific combinations of attributes are required in a document then *PRTFORM2 must be completed, this overides any attributes defined using *PRTFORM.

The key of the record created using this form commences *% and the

remainder of the key must be identical to the associated record created using *PRTFORM. For example if *$LP10 has been created using *PRTFORM then the associated record created with *PRTFORM2 will have a key *%LP10.

The form has the following format:



```
Details for [*%________   Print time attribute combination escape sequences
Esc 1-next record, Esc 2-update, Esc 3-create, Esc 4-read
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
All off ________________________________________
        On                 Off
Att- 1  _________________  _________________
     A  _________________  _________________   (Attribute 2      )
     B  _________________  _________________   (Attribute   3    )
     C  _________________  _________________   (Attribute 2,3    )
     D  _________________  _________________   (Attribute     4  )
     E  _________________  _________________   (Attribute 2   4  )
     F  _________________  _________________   (Attribute   3 4  )
     G  _________________  _________________   (Attribute 2 3 4  )
     H  _________________  _________________   (Attribute       5)
     I  _________________  _________________   (Attribute 2     5)
     J  _________________  _________________   (Attribute   3   5)
     K  _________________  _________________   (Attribute 2 3   5)
     L  _________________  _________________   (Attribute     4 5)
     M  _________________  _________________   (Attribute 2   4 5)
     N  _________________  _________________   (Attribute   3 4 5)
     O  _________________  _________________   (Attribute 2 3 4 5)
                                               Write Record _
```

The form is divided into two sections described below.

9.2.1 Visible equivalents definition

The first part of the form allows local visible equivalents to be defined, in the same way as *PRTFORM.

9.2.2 Attributes definition

The escape sequence which switches off all attributes should be entered in the All Off field.

For the remaining fields two escape sequences are entered, one to switch a combination of attributes on and the other to switch them off.

9.3 PETAL SEQUENCES - *PETFORM

Petal sequence records are used by LEX at print time as a translate table to translate the internal value of a character to the character which is to be printed. These records are particularly useful for non-standard daisy wheels. Keys of petal sequence records commence with **$#**.

Up to 8 further characters may be entered as the key of the record, e.g.

```
Daisy wheel petal sequence for [*#DAISY1___ type daisy wheels
```

This key is also entered in the Petal field of the **PNST installation record.

Petal functions (see section 9.4) cause a character to be replaced by a sequence of characters at print time.

The petal key field in the **PNST installation record defines the key of the petal record to be used when a document is printed.

The form used to amend and set up these petal records has the following format:

```
Daisy wheel petal sequence for [*#________ type daisy wheels
Esc 1 for next record, Esc 2 to update, Esc 3 to create, Esc 4 to read
Enter single character or decimal value of character
Alpha character in last position is a petal function

000 ___ ___ ___ ___ 004 ___ ___ ___ ___ 008 ___ ___ ___ ___ 012 ___ ___ ___ ___
016 ___ ___ ___ ___ 020 ___ ___ ___ ___ 024 ___ ___ ___ ___ 028 ___ ___ ___ ___
032 ___ ___ ___ ___ 036 ___ ___ ___ ___ 040 ___ ___ ___ ___ 044 ___ ___ ___ ___
048 ___ ___ ___ ___ 052 ___ ___ ___ ___ 056 ___ ___ ___ ___ 060 ___ ___ ___ ___
064 ___ ___ ___ ___ 068 ___ ___ ___ ___ 072 ___ ___ ___ ___ 076 ___ ___ ___ ___
080 ___ ___ ___ ___ 084 ___ ___ ___ ___ 088 ___ ___ ___ ___ 092 ___ ___ ___ ___
096 ___ ___ ___ ___ 100 ___ ___ ___ ___ 104 ___ ___ ___ ___ 108 ___ ___ ___ ___
112 ___ ___ ___ ___ 116 ___ ___ ___ ___ 120 ___ ___ ___ ___ 124 ___ ___ ___ ___
128 ___ ___ ___ ___ 132 ___ ___ ___ ___ 136 ___ ___ ___ ___ 140 ___ ___ ___ ___
144 ___ ___ ___ ___ 148 ___ ___ ___ ___ 152 ___ ___ ___ ___ 156 ___ ___ ___ ___
160 ___ ___ ___ ___ 164 ___ ___ ___ ___ 168 ___ ___ ___ ___ 172 ___ ___ ___ ___
176 ___ ___ ___ ___ 180 ___ ___ ___ ___ 184 ___ ___ ___ ___ 188 ___ ___ ___ ___
192 ___ ___ ___ ___ 196 ___ ___ ___ ___ 200 ___ ___ ___ ___ 204 ___ ___ ___ ___
208 ___ ___ ___ ___ 212 ___ ___ ___ ___ 216 ___ ___ ___ ___ 220 ___ ___ ___ ___
224 ___ ___ ___ ___ 228 ___ ___ ___ ___ 232 ___ ___ ___ ___ 236 ___ ___ ___ ___
240 ___ ___ ___ ___ 244 ___ ___ ___ ___ 248 ___ ___ ___ ___ 252 ___ ___ ___ ___

Function table *(________               Write Record _
```

The record is used by LEX as a translation table at print time. Each field in the table represents a character with the numbers in the table representing the decimal values of the characters, for example A has a decimal value of 65. Entries in the table can be made in one of three ways:

1) As a single character in the first position of the field representing the character to be printed.

```
064 +___ ___ ___ ___ 068 ___ ___ ___ ___ 072 ___ ___ ___ ___ 076 ___ ___ ___ ___
```

This defines that @ which has a decimal value of 64 should be printed as +.

2) As a three digit number representing the decimal value of the character to be printed.

```
064 043 ___ ___ ___ 068 ___ ___ ___ ___ 072 ___ ___ ___ ___ 076 ___ ___ ___ ___
```

This defines that @ which has a decimal value of 64 should be printed as + which has a decimal value of 043.

3) As a single character in the last position of the field indicating that the character is to be translated using the associated petal function table.

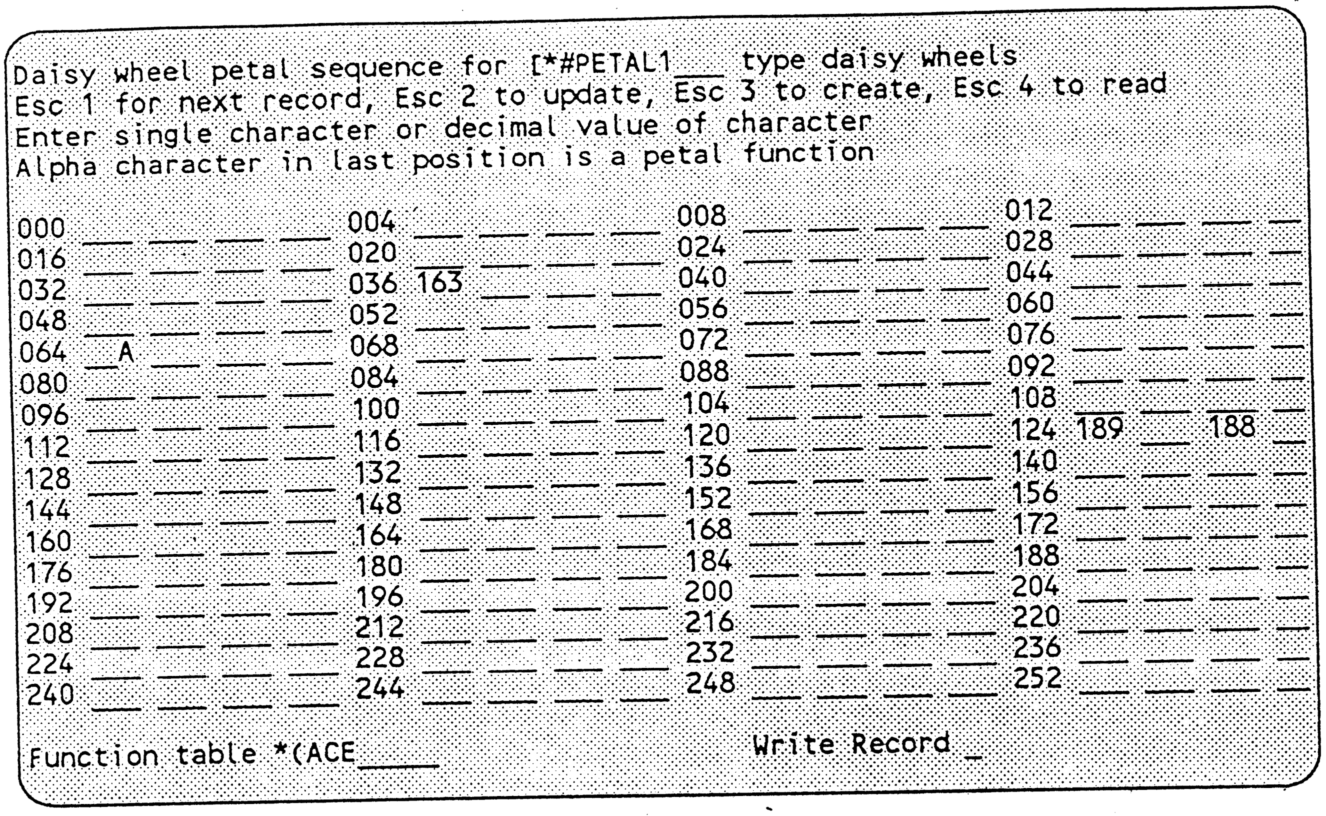
```
064 __A ___ ___ ___ 068 ___ ___ ___ ___ 072 ___ ___ ___ ___ 076 ___ ___ ___ ___
```

At print time @ will be replaced by the characters defined for entry A in the petal function form.

Only fields which need to be translated to a different value need to be entered. Thus in the above examples values for A, B, C, and so on were not entered as these characters are to be printed as the characters themselves.

An example for a petal record with the key *PETAL is given below; the record applies to systems which use an 8 bit character set for which pound, half and quarter signs are to be printed in place of the dollar, vertical bar and tilde respectively; the decimal values of each of these characters are listed below.

Character	Decimal value of character
Dollar sign	036
Vertical bar	124
Tilde	126
Pound sign	163
Quarter sign	188
Half sign	189



```
Daisy wheel petal sequence for [*#PETAL1___ type daisy wheels
Esc 1 for next record, Esc 2 to update, Esc 3 to create, Esc 4 to read
Enter single character or decimal value of character
Alpha character in last position is a petal function

000 ___ ___ ___ ___ 004 ___ ___ ___ ___ 008 ___ ___ ___ ___ 012 ___ ___ ___ ___
016 ___ ___ ___ ___ 020 ___ ___ ___ ___ 024 ___ ___ ___ ___ 028 ___ ___ ___ ___
032 ___ ___ ___ ___ 036 163 ___ ___ ___ 040 ___ ___ ___ ___ 044 ___ ___ ___ ___
048 ___ ___ ___ ___ 052 ___ ___ ___ ___ 056 ___ ___ ___ ___ 060 ___ ___ ___ ___
064 _A_ ___ ___ ___ 068 ___ ___ ___ ___ 072 ___ ___ ___ ___ 076 ___ ___ ___ ___
080 ___ ___ ___ ___ 084 ___ ___ ___ ___ 088 ___ ___ ___ ___ 092 ___ ___ ___ ___
096 ___ ___ ___ ___ 100 ___ ___ ___ ___ 104 ___ ___ ___ ___ 108 ___ ___ ___ ___
112 ___ ___ ___ ___ 116 ___ ___ ___ ___ 120 ___ ___ ___ ___ 124 189 ___ 188 ___
128 ___ ___ ___ ___ 132 ___ ___ ___ ___ 136 ___ ___ ___ ___ 140 ___ ___ ___ ___
144 ___ ___ ___ ___ 148 ___ ___ ___ ___ 152 ___ ___ ___ ___ 156 ___ ___ ___ ___
160 ___ ___ ___ ___ 164 ___ ___ ___ ___ 168 ___ ___ ___ ___ 172 ___ ___ ___ ___
176 ___ ___ ___ ___ 180 ___ ___ ___ ___ 184 ___ ___ ___ ___ 188 ___ ___ ___ ___
192 ___ ___ ___ ___ 196 ___ ___ ___ ___ 200 ___ ___ ___ ___ 204 ___ ___ ___ ___
208 ___ ___ ___ ___ 212 ___ ___ ___ ___ 216 ___ ___ ___ ___ 220 ___ ___ ___ ___
224 ___ ___ ___ ___ 228 ___ ___ ___ ___ 232 ___ ___ ___ ___ 236 ___ ___ ___ ___
240 ___ ___ ___ ___ 244 ___ ___ ___ ___ 248 ___ ___ ___ ___ 252 ___ ___ ___ ___

Function table *(ACE_____                    Write Record _
```

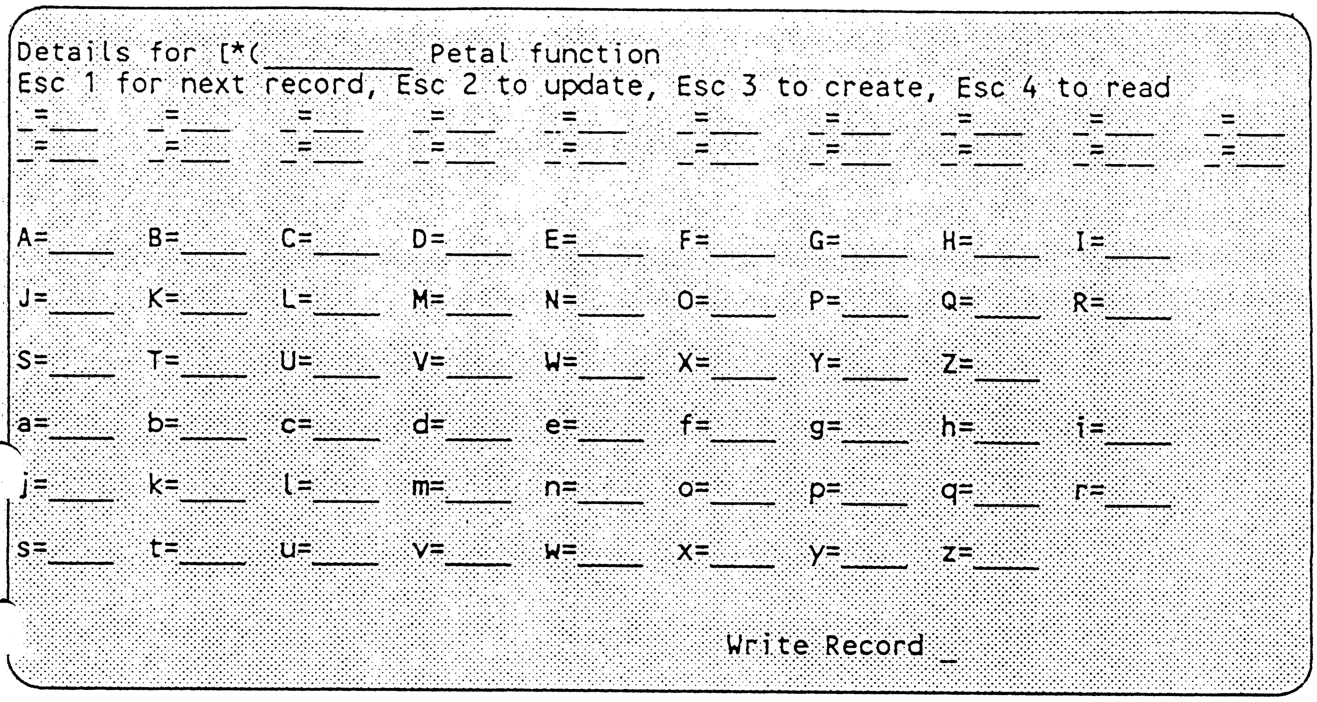
In the example a petal function table *(ACE has been defined and @ (decimal value 064) will be replaced by petal function A at print time, see below.

9.4 PETAL FUNCTIONS - *PETFUN

Records defined using the petal function table are used to translate a single character to a series of characters at print time. The series of characters could be an escape sequence.

LEX takes the name of the petal function table to use from the field at the end of the *PETFORM record.

The form has the following format:

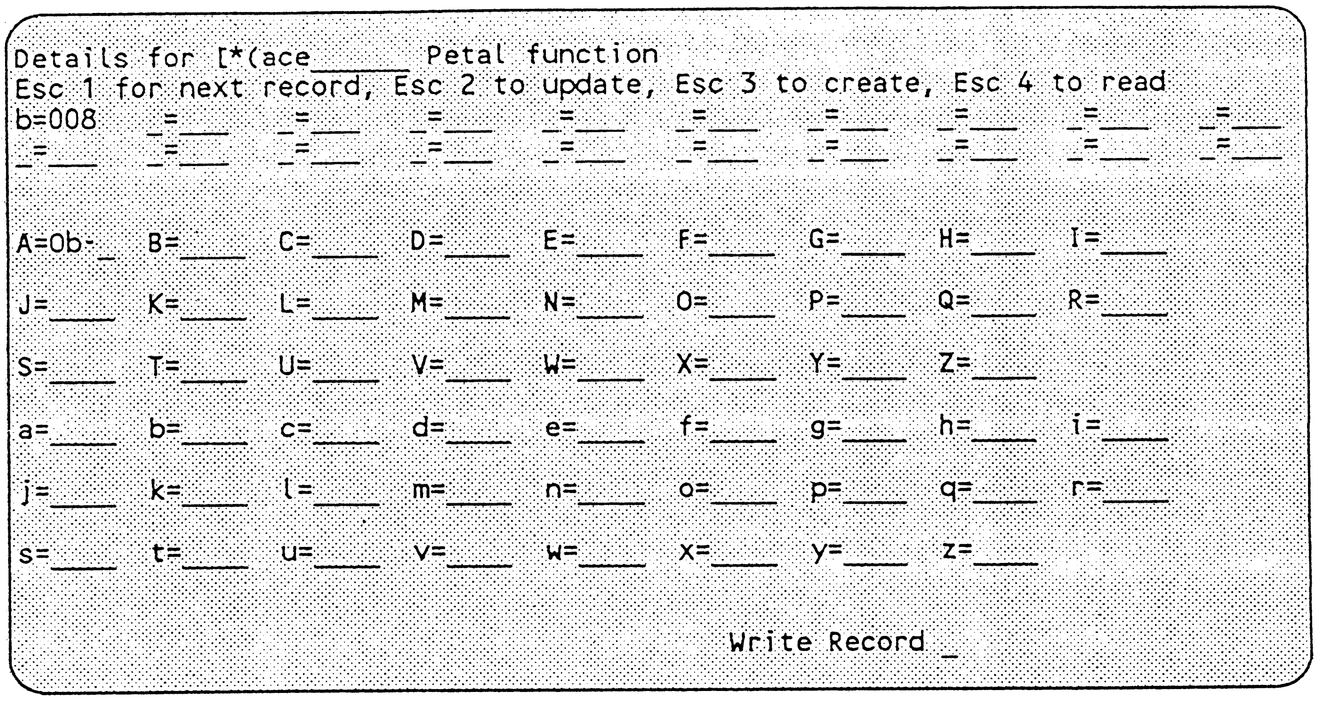


```
Details for [*(________ Petal function
Esc 1 for next record, Esc 2 to update, Esc 3 to create, Esc 4 to read
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
_=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___

A=____ B=____ C=____ D=____ E=____ F=____ G=____ H=____ I=____

J=____ K=____ L=____ M=____ N=____ O=____ P=____ Q=____ R=____

S=____ T=____ U=____ V=____ W=____ X=____ Y=____ Z=____

a=____ b=____ c=____ d=____ e=____ f=____ g=____ h=____ i=____

j=____ k=____ l=____ m=____ n=____ o=____ p=____ q=____ r=____

s=____ t=____ u=____ v=____ w=____ x=____ y=____ z=____

                                   Write Record _
```

The top part of the form is used to define local visible equivalents used in the lower part of the form. Characters for which visible equivalents must be defined are ones which cannot be displayed (e.g. the low value ASCII characters in the range 0 to 32 decimal) and characters which cannot be used in a form (such as [). The first part of *PRTFORM is similar.

The second part of the form defines the functions to be performed. Each function must be terminated by an underline. If more than four characters are used then the next field is used as a continuation. If a four character sequence is defined then the next field must not be used, as the underline at the start of it is used as a terminator.

As an example the Greek character Theta can be printed by sending the sequence O(BACKSPACE)- to the printer. This is achieved by assigning @ to petal function A as in the petal sequence example in section 9.3 and then completing the petal function record as below.



```
Details for [*(ace_____  Petal function
Esc 1 for next record, Esc 2 to update, Esc 3 to create, Esc 4 to read
b=008    _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___
_=___    _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___  _=___

A=0b-_   B=____ C=____ D=____ E=____ F=____ G=____ H=____ I=____

J=____   K=____ L=____ M=____ N=____ O=____ P=____ Q=____ R=____

S=____   T=____ U=____ V=____ W=____ X=____ Y=____ Z=____

a=____   b=____ c=____ d=____ e=____ f=____ g=____ h=____ i=____

j=____   k=____ l=____ m=____ n=____ o=____ p=____ q=____ r=____

s=____   t=____ u=____ v=____ w=____ x=____ y=____ z=____

                                   Write Record _
```

9.5 DOCUMENT PRINTING DETAILS - *PAPER

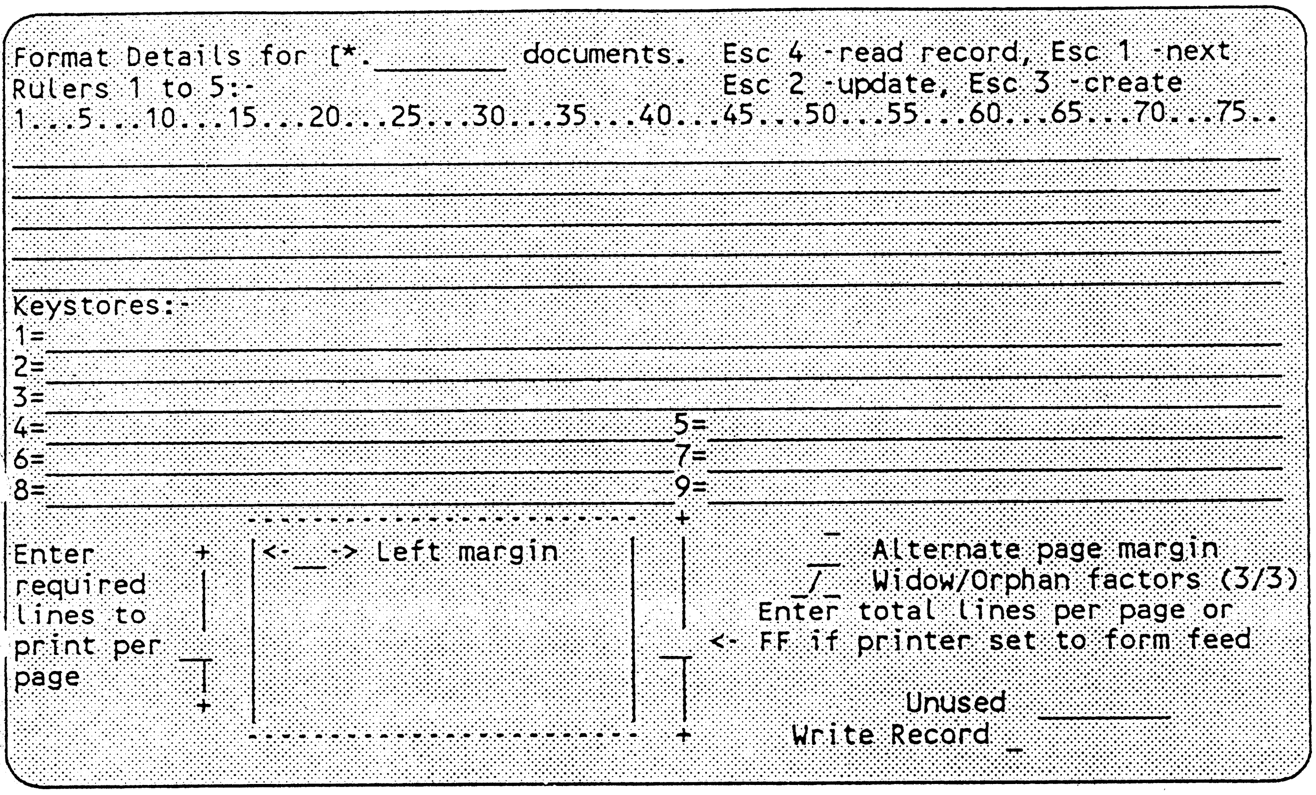
A LEX document name consists of two parts, a file name followed by an extension. The extension may be up to three characters long and is separated from the filename by a dot.

The document printing details record contains information which LEX uses to control the format of a document as it is printed, the record also contains the default rulers and abbreviations.

A different document printing details record may be set up for each extension. The key of the record commences *. followed by the extension. For example the document printing details record for all documents with an extension of .DOC is *.DOC. See the LEX User Manual for an introduction to document printing details.

The system record *PAPER is used as the form which is used when the document printing details for an extension are set up viewed or amended. In 132 wide mode the system record used is *PAPER132 or *PAPER13 if the VDU is unable to display 24 lines when in 132 mode.

The form has the following format:



```
Format Details for [*._______ documents.  Esc 4 -read record, Esc 1 -next
Rulers 1 to 5:-                           Esc 2 -update, Esc 3 -create
1...5...10...15...20...25...30...35...40...45...50...55...60...65...70...75..
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

Keystores:-
1=___________________________________________________________________________
2=___________________________________________________________________________
3=___________________________________________________________________________
4=_____________________________________5=____________________________________
6=_____________________________________7=____________________________________
8=_____________________________________9=____________________________________
                 - - - - - - - - - - - - - - - - - +
Enter      +   |<-__-> Left margin    |  |      _  Alternate page margin
required   |   |                      |  |     _/_ Widow/Orphan factors (3/3)
lines to   |   |                      |  |   Enter total lines per page or
print per _|_  |                      | _|_ <- FF if printer set to form feed
page       |   |                      |  |
           +   |                      |  |           Unused _______
                 - - - - - - - - - - - - - - - - - +   Write Record _
```

The form is divided into three sections. The first two sections are used during document create and edit operations and contain rulers and abbreviations.

If no document printing details record exists for an extension then LEX uses a record with the key *.xxx where xxx is the default document extension defined in the **INST installation record.

It is possible to force a document to accept a different document printing details record by appending /*.xxx to the document name when it is entered at the Document Name prompt, where *.xxx is the key of the record to use. This can be used when editing or creating a document to apply different rulers and abbreviations or when printing a document to apply different printing details such as the left margin offset.

[redacted] If a *.080 record is not found LEX obtains the rulers and abbreviations from a record with a key of *.xxx, where xxx is the default extension defined in the **INST installation record. If this record does not exist then no default rulers and abbreviations are available in the playground.

When operating in 132 wide mode in the playground LEX uses the document printing details record with a key of *:132.

9.5.1 Rulers

Five default rulers are defined at the start of the form. These can be called into a document at any point using the (ESC)Mn escape sequence where n is in the range 1 to 5 and represents the number of the ruler required.

If the terminal supports a 132 wide mode display then the form *PAPER132 (or *PAPER13 if the VDU is not capable of displaying 24 lines in 132 mode) allows document printing details records to be created which contain five rulers each 132 characters long.

9.5.2 Abbreviations (keystores)

Nine abbreviations may be entered. These can be used for frequently used text strings or sequences of keystrokes.

9.5.3 Printing Details

Left margin - the position the print head is to move to before printing commences on each line.

Enter required lines to print per page - this should be set to the number of lines to print on each page. LEX uses this field during pagination to insert .PP lines at the correct points in the document. If a document which contains no pagination commands (.P or .PP) is printed then this field is used to determine where to break each page.

Enter total lines per page - if the printer does not have a form feed facility this field should be set to the total number of lines available on the sheet. For standard 11 inch sheets this field should be set to 66. If the printer supports form feed then specify FF. If FF is entered then the characters sent to the printer to cause a form feed are taken from the Form Feed and 2nd Form Feed fields defined using *PRTFORM. If these fields are not entered a default of 012 is used.

Alternate page margin - this field is only used when the option to print alternate pages is selected at print time by entering a reply of O or E to the print alternate pages prompt. The field should be set to the position of the left margin to be used on odd numbered pages. The Left Margin field is used for the position of the left margin on even numbered pages. If the field is omitted then the value from the Left Margin field is used.

Widow/Orphan Factors (3/3) - if a page break occurs just after the start of a paragraph then one or two lines are left at the foot of the page, these lines are described as widows. If a page break occurs just before the end of a paragraph then one or two lines are carried over to the next page, these lines are described as orphans.

The Widow/Orphan fields allow LEX to over-rule the standard page breaks by forcing a page break on blank lines which occur W lines before the normal page break or O lines after it, where W and O are the entries for the Widow and Orphan factors respectively.

For example, assume Lines to Print per Page field is set to 55, Widow Factor is 3 and Orphan Factor is 5. Then LEX will page break on any blank line after line 52 (55 minus the widow factor of 3) and before 55. It will break on 55 unless there is a blank line between lines 55 and 60 (55 plus the orphan factor of 5) in which case it will break there.

The default Widow/Orphan factor is 3/3.

Some of the fields in this part of the record are not used at present and values should not be entered in them.

Chapter 10. MENUS

Menus control the logical structure of a LEX system and provide access to all the user features. Their use is described in the User Manual. This chapter considers creating and altering them to produce a customised system.

Menus are stored in the VMF as system records and can be accessed in exactly the same way as any other system record. An example of the stored format of a menu and a description of fields is given below in section 10.1.

On entering LEX the first menu displayed is specified in the *VDUFORM record for the VDU in use. It is normally called **MENU but a different menu may be nominated by altering the name in *VDUFORM. If the nominated main menu is not found on entering LEX it tries **MENU and if that too is not found a blank screen is shown. Such a blank screen is in a pseudo menu state where menu switch options such as /X can be used.

A menu may be activated by selecting an option in a menu or by use of the menu switch option /M. If a specified menu is not found on the VMF, LEX defaults to **MENU after showing the **TEXT message Missing Menu.

There is a limit of six nested menus, attempts to call menus beyond this limit can have unpredictable results. However if **MENU is called as a new menu from any other menu in the system, then the menu stack is reset. The name of the current menu and its position in the stack can be displayed with the /W menu switch option.

10.1 FORMAT OF MENUS

A typical menu is shown in Figure 10.1. Within the constraints of the format described here any menu can be tailored to suit the user's requirements. Options can be moved between menus, others deleted and new ones added.

FIGURE 10.1 - A Typical Menu Record

```
[*DOCMENU  EXTRA FACILITIES
X*DIRDSK   V  -  View Data Disk Directory

A          C  -  Copy a document
K          K  -  Kill a document
R          R  -  Rename a document

E*GLOBAL   G  -  Global replace
E*PAGINATE P  -  Pagination

X*PAPER    D  -  Document printing details

                 SPECIAL FEATURES
                 ----------------
M*HELPMENU H  -  Help facilities
FB:NEWLEX  X  -  Special Feature Disk
X*PASSWORD Y  -  System Housekeeping (Password Protected)

           .  -  Select option
```

The top line of the record immediately after the key is reserved for a menu heading. On activating the menu this heading is centred on the screen and is always followed by a blank line.

Each subsequent line defines an option that may be taken from that menu. The first 12 columns describe the internal LEX operations that take place when the line is selected and are not shown to the user. The first column must contain a valid menu switch option character. Columns two to eleven may be be blank or hold the key of a system record to be used in conjunction with that option. Menu switch options are described in Appendix A together with any system records they may require.

Column twelve is the character that is used to activate the option on that line. Any characters to the right of column 12 are simply text displayed with the menu. They usually describe an option but may be used for messages, etc.

The final line in the example above is a comment line to prompt the user.

Some of the lines in the above example are described below:

```
X*DIRDSK   V  -  View Data Disk Directory
```

Selecting V from the menu activates menu switch option /X (enter playground) and then automatically activates the record *DIRDSK. See Chapter 11 for details on how to program such a record.

```
A          C - Copy a document
```

Selecting C from the menu activates menu switch option /A to copy documents. No system record has been given and document name prompts will be as for the menu switch option.

```
E*GLOBAL   G - Global replace
```

Option G from the menu activates menu switch option /E to edit a document. As with manual use of the menu switch option the document name prompts will be issued but on entering the document a line is inserted at the top of the document and system record *GLOBAL activated. See Chapter 11 for how to program such a record.

```
M*HELPMENU H - Help facilities
```

Selecting option H activates menu switch option /M and goes to menu *HELPMENU.

```
FB:NEWLEX  X - Special Feature Disk
```

Selecting option X activates menu switch option /F which is used to change to another VMF. In this case the characters after the F do not refer to a record key but the name of the new VMF, in this case B:NEWLEX. As account/device details may be included in the string the ten characters available may not be enough to define the new VMF. This can be overcome by going into playground and then activating menu switch option by programming. For example the menu line could be:

```
X*CHANGE   X - Special Feature Disk
```

and the system record:

```
[*CHANGE   $E/FDP1:(40,100)NEWLEX.VMF\
```

See Chapter 11 for more details on programming.

10.2 MENU HIGHLIGHTS

In the menu as described above the heading line and characters which activate the options are displayed in background mode whereas the remainder is in foreground mode. Typically this would mean the former in double intensity or inverse video.

It is possible to use the options in *VDUFORM2 to change the modes of the display thereby allowing any attributes available on the VDU to be utilised. On the appropriate monitors this could even include double height or the use of colour.

The appropriate escape sequences for the VDU are set at the bottom of *VDUFORM2, where ten sequences can be defined. See Chapter 8 for details.

These highlights are selected by adding to a line in the menu an instruction of the form:

```
/Smno
```

For example

```
C          C  -  Create a new document/S463
```

On the heading line of the menu the escape sequence m is sent prior to the first literal in the heading and the escape sequence n after the last. o is ignored but should be set to a dummy character.

On subsequent lines the sequence m is sent prior to the first character and n after it. Sequence o is sent at the end of the literal.

To send no sequence a . (dot) should be placed in the appropriate m, n, o location.

The following shows a menu record using such attributes:

```
[**WP       WORD PROCESSING MENU/S233
C           C  -  Create a new document/S463
E           E  -  Edit an existing document/S463
Q           P  -  Print a document/S463

            OTHER OPTIONS/S2.3
M*DOCMENU   O  -  Further facilities/S463
FNEWLEX     X  -  Special option/S463

X*PASSWORD  .  -  Select option      /S452
```

If it is assumed that, for the VDU in use, the sequences are set as follows:

Sequence 2	-	select inverse video
Sequence 3	-	switch off all attributes
Sequence 4	-	switch on underline and bold
Sequence 5	-	switch on inverse video and bold
Sequence 6	-	switch off attributes and select underline

then, in the above example the title would be displayed in reverse video as would the heading part way down. The menu select characters would be underlined in bold and the menu descriptions underlined. The `Select option` would be inverse video and bold.

Note that if `/s` is omitted then the default described earlier applies. The `Set FG` sequence in `*VDUFORM` is used as the `m` sequence and `Set BG` sequence is used as the `n`.

The highlight selected could be double height characters if the VDU supports this. In this case two escape sequences need to be defined, one to display the upper part of the line in double height characters and the other to display the lower half in double height. For example if graphic sequence 7 and 8 are set to display upper and lower parts of a line respectively as double height characters, then the menu line should be set up as follows.

```
M*HELPMENU  H  Help facilities/S773
M*HELPMENU  H  Help facilities/S883
```

The title line of a menu is a special case as it cannot be entered twice in a menu. To overcome this an alternative system is provided. If the first character of the escape sequence in `*VDUFORM2` is set to 255 decimal, LEX repeats the heading on the line below. Take the example

```
Main menu/S23.
```

On some ANSI compatible VDUs such as the VT100 the following set in graphics sequence 2 results in double height characters.

```
{$#:}$#;$M
```

where `{`=255, `$`=027, `}`=010

These sequences have the following effect - on the VT100 ONLY:

`{`	causes the title to be displayed on the next line as well as this one
`$#:`	displays the first line as double height top half
`}`	causes a line feed
`$#;`	displays second line as double height bottom half
`$M`	moves the cursor back up a line

Note that doublē width is not recommended as it upsets automatic centring.

Chapter 11. VISIBLE EQUIVALENTS & PROGRAMMING

LEX programming is the storing of strings of characters representing literals and LEX functions. The programs may be stored temporarily or permanently in a keystore or permanently as a record on the VMF. Any LEX functions may be stored including RETURN, go to top of screen, display date, etc. Literals may be interspersed with these functions.

Programming is an advanced feature which when coupled with the database file structure of the VMF extends LEX from being a word processor to an information handling system capable of being adapted for most data processing applications.

Programming is used for:

- storing frequently executed keystrokes. Calculations or standard phrases can be made readily available.
- including commands in forms; for example to move the cursor to the top of the screen before displaying a form.
- creating standard documents. Paragraphs which are already stored on the VMF can be copied into the document.
- writing systems. LEX has been used to write invoicing, diary and ledger systems.

LEX programs can be invoked in a number of ways:

- directly from a keystore
- from a menu by selecting an option which has a record stored on the VMF associated with it
- by executing a VMF record either from within a document or from the playground

11.1 VISIBLE EQUIVALENTS

A stored LEX program contains what are known as visible equivalents. These are characters which may be displayed on the screen to represent a LEX function. The characters used as visible equivalents are defined in **MONT. The usual convention for the English language visible equivalents are shown in Table 11.1.

As LEX executes a program it assumes that any of these characters are to be treated as part of a LEX function. For example if the string:

```
'THELLO
```

is encountered then [illegible] is interpreted as [illegible] and the cursor moves to the top of the screen after which the literal string HELLO appears.

Some examples of programming using visible equivalents are given later in this chapter.

TABLE 11.1 - Standard Visible Equivalents

Visible Equivalent	LEX Function	Use
[illegible]	Escape	This is followed by one or more characters to define an escape sequence. (ESC)J is represented by $J
[illegible]	Control	When followed by a single letter it indicates a LEX control function. (CTRL)A is represented by 'A
[illegible]	(RETURN)	(RETURN) may also be achieved by [illegible] so an alternative visible representation is [illegible]
[illegible]	(TAB)	(TAB) is achieved by (CTRL)I so an alternative visible representation is 'I
[illegible]	Start of key	Visible equivalent of [
[illegible]	End of key	Visible equivalent of]
[illegible]	(→)	(→) is achieved by (CTRL)R so an alternative visible representation is 'R
[illegible]	(←)	(←) is achieved by (CTRL)H so an alternative visible representation is 'H
[illegible]	Start of literal string	Visible equivalent characters can be treated as ordinary literals if placed between a pair of these characters.

[illegible]	Next character literal	A single visible equivalent character can be treated as a literal if preceded by this character.
[illegible]	Pause indicator	This causes execution of a program to halt until a (RETURN) is typed.

11.2 SETTING UP A LEX PROGRAM

Programming in LEX is a screen based function. A LEX program emulates a sequence of commands that could be entered in by hand.

Programs may be held in keystores or as a record on the VMF. Each of these may be loaded or viewed in different ways as described in this section.

11.2.1 Programs in keystores

Keystores can be used to store short programs - up to 78 characters in length. They may be set up in four ways:

1) Self Programming. Simple programs can be created by manually running through the commands required and then using [redacted] [redacted] with the last 78 keystrokes entered at the keyboard. LEX functions are automatically converted to their visible equivalents which may reduce the number of keystrokes shown.

   [redacted]

   Programs generated in this manner may need slight modification before they run properly, particularly if such characters as [are involved. It is therefore always advisable to view and check the keystore before executing it.

2) Document Extension Details. The User Manual describes the document extension details where fields may be set to create permanent keystores associated with an extension.

3) Cut And Paste. Keystores may be set using cut and paste options. The start of the string is marked with ESC(and the end cut using ESC)na where n is the keystore number. The a results in a modification of the text from which the string is being cut. It can be S, B or L to represent squeeze, blank or leave respectively.

4) ESCSn This is a view and set keystore command where n is the number of the keystore. The sequence causes the contents of the keystore to be shown in background mode as in the following example of keystore 5.

```
5='T$CBContents of keystore 5
```

Overtyping the first character deletes the entire contents ready for a new string, whereas the remaining characters may be altered piecemeal. A RETURN finally loads the keystore and removes the display. Any changes made are not stored for the next session.

11.2.2 Programs in VMF records

Programs may be stored in records on a VMF. They can be set up either in a document or from the playground.

The record should commence with the key and programming starts in the 12th column. For example take a program to cut a screenful of information from a document and put it in a record called *CUT.

```
[*CUT      $('L$)S
```

It is important that the programming does not start before the 12th column, otherwise the first part of the program could be interpreted as part of the key, e.g. *CUT $

The record is created by placing the cursor immediately after the S and entering the create record sequence ESC]C. Note that if the program takes more than one line then there is an implied RETURN at the end of each line. To ignore the implied RETURN the sequence ESC>C can be used.

A similar method is used to read a record. Start with [and the key and then type ESC]R. Note the difference between reading a record and performing it; the latter executes visible equivalents as LEX commands, the former displays them as literals.

11.3 EXECUTING A LEX PROGRAM

LEX programs can be invoked in a number of ways:

- directly from a keystore
- by executing the VMF record containing the program from within a document or from the playground
- from a menu by selecting an option which has a record stored on the VMF associated with it.

When errors occur within a program it may continue to run, thereby giving strange results. However if an error occurs when opening a file no further processing occurs in that record and LEX returns to edit mode. The escape sequences that have this effect are (ESC)MO (ESC)OF (ESC)OX (ESC)SF (ESC)WO.

11.3.1 Programs in keystores

A program in a keystore may be performed when in normal edit mode either in a document or playground. It is executed by typing (ESC)n where n is the number of the keystore.

11.3.2 Programs in VMF records

Programs stored in VMF records may be performed when in edit mode in a document or in playground. A program is executed by entering a [, the record key and a] e.g.

[*CUT]

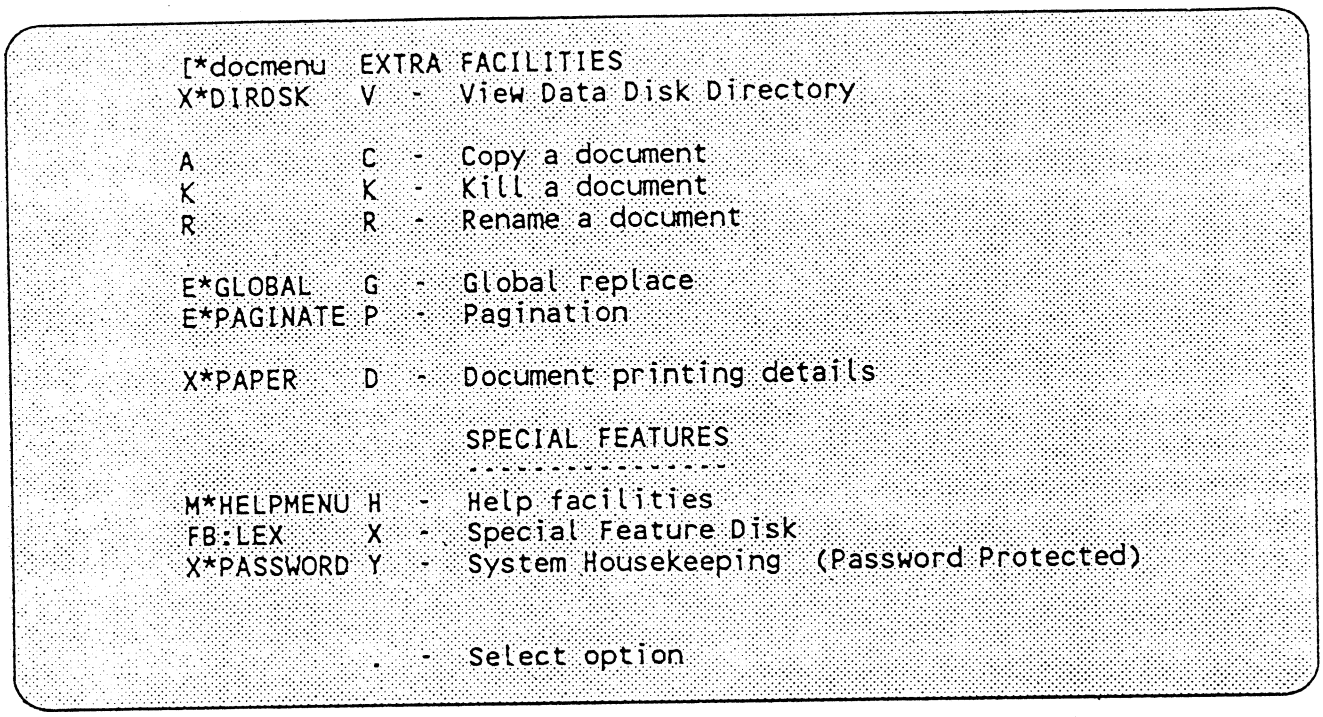
As entering the key may overtype text in a document there is an alternative method which allows the key to be entered in background mode. In this method the escape sequence (ESC)RK is entered which results in a Key is prompt in background mode. The key is entered followed by (RETURN) which removes the display from the screen and activates the keystore. e.g.

(ESC)RK

Key is *CUT (RETURN)

11.3.3 Executing records from a menu

Programs stored in records may also be executed from a menu when a document is opened or when the playground has been entered. The key is included in the menu instructions after the menu switch option character:



```
[*docmenu  EXTRA FACILITIES
X*DIRDSK   V  -  View Data Disk Directory

A          C  -  Copy a document
K          K  -  Kill a document
R          R  -  Rename a document

E*GLOBAL   G  -  Global replace
E*PAGINATE P  -  Pagination

X*PAPER    D  -  Document printing details

                 SPECIAL FEATURES
                 ----------------
M*HELPMENU H  -  Help facilities
FB:LEX     X  -  Special Feature Disk
X*PASSWORD Y  -  System Housekeeping  (Password Protected)


           .  -  Select option
```

When option V is selected, the playground (X) is entered and the record with key *DIRDSK executed. When option G is selected, a document is entered to edit (E) via the usual document name prompts. On entering the document the record *GLOBAL is executed. Similar programs can be run with other menu switch options including A, C, and I. See Chapter 10 for further information.

11.4 SOME PROGRAMMING FEATURES

11.4.1 Pause points

When LEX encounters a ? in a program it pauses until a (RETURN) is typed. This allows variable text or figures to be inserted in standard paragraphs or calculations. It can also be used for data prompts in more complex programs. For example:

```
Enter Customer Name $(?($)L
```

The above example marks the start of a string to be cut, pauses until the string has been entered and (RETURN) pressed, moves the cursor left over the last character and cuts the string into the cut and paste buffer.

Prompts inside system records and keystores are not shown. If a record contains, say, `$GABC\` the prompt `Search-ABC` is suppressed when the record is executed.

11.4.2 (ESC)E

A menu switch option can be activated in a program by exiting to the last menu and then entering the required switch. For example to get to the system menu a program could include:

```
$E/MSYSTEM\
```

The * in the record name `*SYSTEM` is supplied by LEX. Further the visible equivalent \ performs the (RETURN) at the end of the menu record key.

Note that if `$E` is used within a LEX program or keystore then only 40 characters after the `$E` will be interpreted by LEX. Any additional characters are ignored.

11.4.3 **Panic - (CTRL)C**

Whilst a program is running it is possible to stop it at any time by entering (CTRL)C. This can be useful if a repeat command goes into an infinite loop.

11.4.4 **Comments in a program**

Comments can be included in a program by use of two double quote signs. This is useful when documenting a program. Anything within the two double quotes will be ignored when a program is performed.

```
'T$CF""Go to top and clear screen""<*TEST>
```

If the double double quotes are at the end of a line then no (RETURN) is generated.

11.4.5 **Conditionals**

Conditionals are the branch commands in LEX. They contain a string to be activated on a true condition and another for the false case, the string to be used depending on the result of the comparison. LEX conditionals can be used to test:

a pair of strings
for a numeric field
for the existence of a document
for the existence of a record on the current or another VMF
for positive, negative or zero calculator store contents

Finally one conditional depends on a yes/no reply to a prompt at run time.

Conditionals in LEX are the only commands which cannot be issued from the keyboard but must be invoked from a keystore or record. They are escape sequences commencing with ESC CT and are fully described in the appropriate section of Appendix C.

Examples of the use of conditionals are included in the examples later in this chapter.

11.4.6 Shutup and shush modes

As LEX is a screen based system the progress of a routine can be seen as the program unfolds. Whilst this is most useful for debugging it can be rather disconcerting during long runs. Further all the output to the screen can slow LEX.

To overcome this there are two useful modes - shush and shut up.

Shush mode is achieved by ESC SH and results in no output to the screen whilst in the document or playground. Remember that the changes to the text have taken place, they are simply not visible. Shush mode is exited by a ESC-SH or returning to a menu. Once out of shush mode the hidden text on the current screen may be viewed by the refresh screen command ESC QR.

Shutup mode is a super quiet shush mode and is achieved by ESC SU. Even menus and other LEX prompts such as for keys are also invisible. To return to normal mode requires a ESC-SU if in edit mode or, at a menu by the /2 menu switch option.

11.4.7 Programming levels

All cases described so far in this chapter are examples of single level LEX programming. That is, any visible equivalents stored in records or keystores are executed in sequence as the program is recalled.

In more complex LEX programs it is useful to have nested programming levels. For instance during the execution of a program it may sometimes be necessary to set up a routine in a keystore. This requires the use of a second program level to prevent the commands in the routine from being executed as the program is recalled.

Programs with more than one level use the special visible equivalent characters @ and ". Strings enclosed between " characters are treated as literals whilst the single character after a @ is considered as a literal.

The visible equivalents were introduced in Section 11.1 as special characters used as a symbolic representation of LEX functions. The characters are:

```
$ ' \ | < > { } ?
```

As LEX executes a program it assumes that if it encounters any of the above characters that they are to be treated as a visible equivalent. If the string:

```
'THello
```

is encountered in a LEX program then the 'T is interpreted as the visible equivalent of (CTRL)T and the cursor moves to the top of the screen and the word Hello is displayed at the home position.

This can can cause problems if it is required to output a single quote character from a program, for instance. If an abbreviation is to be set up containing the string:

```
You mustn't hit the Break key
```

and it is set up as follows:

```
(ESC)S1=You mustn't hit the Break key
```

when (ESC)1 is typed the words

```
You mustn
```

will appear at the cursor position and the words

```
hit the Break key
```

will be displayed at the top of the screen. This is because the 't is interpreted as a visible equivalent and positions the cursor at the top of the screen.

The special characters @ and " are used to switch off the interpretation of visible equivalents.

The @ character is used in LEX programs to indicate that the next character is to be treated as an ordinary character and not as a visible equivalent, therefore if the abbreviation is set up as follows:

```
(ESC)S1=You mustn@'t hit the Break key
```

when (ESC)1 is typed the words will appear correctly at the cursor position as:

```
You mustn't hit the Break key
```

The @ is also used if an abbreviation is set up from a system record. For example abbreviation 1 would be set as follows to read the next record on the VMF:

```
(ESC)S1='T$N$>R
```

But if this is to be set up from a system record *NEXT, then it would be incorrect to code it as:

```
[*NEXT       $S1='T$N$>R\
```

because the 'TN>R would be interpreted as the record is invoked and not set into abbreviation 1 as required.

The correct form of *NEXT is:

```
[*NEXT       $S1@'T@$N@$@>R\
```

where each visible equivalent which is not to be interpreted as the record *NEXT is run, is preceded by an @ sign.

Double quotes " may also be used to switch off visible equivalent mode. When a LEX program encounters " it assumes all subsequent characters, until the next ", are not to be interpreted as visible equivalents. This gives a simpler way of writing the *NEXT record above as:

```
[*NEXT       $S1"'T$N$>R"\
```

The \ remains outside the " as it is to be interpreted as the carriage return which terminates the set abbreviation command when *NEXT is executed.

11.5 EXAMPLES OF PROGRAMS

11.5.1 Phrases and text blocks

```
Thomas J Mann\Marketing Director\Arrow Ltd\BRIGHTON
```

Examples such as this are often set in keystores using the document extension details. This one gives a name and address block. The \ is the visible equivalent of (RETURN).

It could be enhanced to give the address block anywhere on the screen starting at the cursor's location by by using the visible equivalent of [redacted]

11.5.2 Simple word processing program

[redacted]

In this routine the $J justifies the current paragraph and then $QF moves the cursor to the start of the next paragraph. This could be used within a repeat loop to work through an entire document re-justifying each paragraph if, for example, the initial margin settings had been changed.

11.5.3 Typical menu driven program

When LEX opens a document for editing it positions the cursor at the start of the document, displays the first screenful of text and allows editing to commence from this point. Sometimes it is more convenient to open the document and position the cursor at the end of the document. For example if LEX is being used to write a manual then new text will invariably be added at the end of the existing text.

This can be achieved by writing a simple program to go to the end of the document.

```
[*END      $QE
```

and adding a line to a menu:

```
E*END      Z - Edit at end of document
```

The program is executed when the document has been opened for editing.

11.5.4 Use of the pause

```
[*MEMO
MEMORANDUM

To:    ?                                Date:  $Y

From:  ?                                Ref:   ?
```

In the example a memorandum heading is stored in a record with a key *MEMO. When it is invoked the heading is displayed and pauses after the word To: until a name terminated by

(RETURN) is typed. The [redacted] is used to display today's date after which it pauses again at the prompts `From:` and `Ref:`. In each case the text entered is terminated by a (RETURN).

11.5.5 Calculations in ### rulers

The following would be used under a three column calculator ruler:

```
....................#####.##.....#####.##.....#####.##............
```

[redacted]

It calculates VAT at 15% and the total:

`|`	Tab to the next field - it should be a calculator field thereby entering the calculator.
`S1`	Store the contents of the first field in store S1.
`|`	Tab to the next field - another calculator field.
`R1`	Recall the contents of store 1.
`*.15`	Multiply the number by .15 (15%)
`=`	Display the answer.
`|`	Tab to the next field - the third calculator field.
`H`	Calculate the horizontal total from all calculator fields on this line.

11.5.6 Calculations on the spot

This example requests a price and a quantity then calculates and displays the total value:

[redacted]

When the program is invoked the result appears in the form:

```
Price =      15.50       Quantity =  30.00        Value =      465.00
```

Explanation of the program:

`$U7`	Ruler 7 consisting of tab stops every 8 characters is used
`Price =`	The prompt is displayed
`||`	Skip two tab stops

`$+`	Enter the calculator
`?`	Pause and request a number
`s1`	Save the number entered in accumulator 1
`|Quantity =|`	Tab, display the prompt and tab again
`$+?`	Enter the calculator, pause and request a number
`s2`	Save the number entered in accumulator 2
`|Value =||`	Tab, display the prompt then tab twice
`$+`	Enter the calculator
`R1*R2=`	Compute and display the value of accumulator 1 multiplied by accumulator 2
`\`	Carriage return to exit from the calculator

11.5.7 Documents with standard paragraphs

This example shows how a licensing agreement could be produced using LEX. For each agreement produced the same clauses are used; only the name of the product being licensed changes.

```
[*LIC     Software  to be licensed  $(___________$SP'T?'H$)9L$UP'X

                    LICENCE TO USE $9 SOFTWARE


 WHEREAS:

 A. Ace Microsystems own the copyright in the Software known as $9.

 NOW THIS AGREEMENT WITNESSES AS FOLLOWS:

 <*CLAUSE1>
 <*CLAUSE2>
```

This program would probably be activated from a menu option to create a document and automatically invoke system record `*LIC`. On entering the document

```
Software to be licensed ?__________
```

is displayed with the cursor positioned on a question mark at the first underline. The name of the software to be licensed is then entered followed by RETURN. The program then sets up the licence in the document:

```
                    LICENCE TO USE LEX SOFTWARE

WHEREAS:

A. Ace Microsystems own the copyright in the Software known as LEX.

NOW THIS AGREEMENT WITNESSES AS FOLLOWS:

1. This licence shall be valid for a period of 15 years from the date
   hereof.

   etc
```

Explanation of the program:

$(	The reply to the prompt Software to be licenced will be cut into keystore 9, this command marks the start of the cut.
__________	The underlines mark the area of the screen where the reply is entered.
$SP	Set protect mode, so that the cursor is restricted to the underlines.
'T	Move to the first of the underlines
?	Pause and display a ?. The name of the software is now entered.
'H	Move the cursor left over the last character of the name.
$)9L	Cut the name into keystore 9 and leave it on the screen.
$UP	Unset protect mode
'X	Delete the line containing the prompt and the reply.
$9	Whenever $9 appears in the record *LIC or in any of the paragraphs called in by it, then it is replaced by the string cut into keystore 9.
<*CLAUSE1> <*CLAUSE2>	The paragraphs *CLAUSE1, *CLAUSE2 are enclosed in < and > which are the visible equivalents of [and] They are therefore activated as system records in turn.

11.5.8 Use of conditionals

The following program provides a simple password system. The record *PASSWORD would be activated in playground from a menu.

```
[*PASSWORD
Enter Password _________$SP'T$SU?
'T$UP$CTSSWORDFISH "$E/MSPECIAL\/2"
"$-SU\\Incorrect Password - Press <RETURN> to continue ?$E"\
```

Explanation of the program:

`Enter Password _________`	The prompt is displayed on the screen.
`$SP'T`	Set protect so that the cursor is restricted to underline characters. Then go to the first underline.
`$SU`	Enter shut up mode. No entries are shown on the screen, neither are menus or other prompts displayed.
`?`	Pause for the entry of a password. The password will not be displayed. The program continues on receiving a RETURN.
`'T$UP`	The cursor is taken to the first character of the entered password and the screen then unprotected.
`$CTSSWORDFISH`	The $CTS sequence is a conditional which compares the string under the cursor with the password SWORDFISH [redacted]
`$E/MSPECIAL\/2`	This string is activated if the password is correct. It performs an ESC E to return to the last menu, menu switch option /M to change menu and then enters the new menu name. The leading * is supplied by LEX so LEX goes to menu *SPECIAL. The \ is a visible equivalent for RETURN to indicate the end of the key. At this stage LEX is still in shut up mode and the menu not visible. Menu switch option /2 resets the mode leaving LEX in the password protected menu *SPECIAL.

```
$-SU\\Incorrect Password - Press <RETURN> to continue ?$E
```

This string is activated if the password is incorrect. It first takes LEX out of shut up mode and then displays an appropriate message. It then waits for a RETURN before performing an ESC E to the calling menu.

11.5.9 Use of repeat loops

LEX contains a repeat facility which allows a string of text, a calculation or a section of a program to be repeated. The repeat loops may be performed a fixed number of times (unconditionally) or until a condition is met (conditionally).

Note that misuse of the repeat loops can cause LEX to enter infinite loops or even to crash because of stack overflow problems.

The format of the repeat command is described in Appendix C as:

```
ESC Xnn"String"\
```

Example:

To draw a row of 80 dots across the screen, a program could be set up as follows:

```
[*DOTS      $X80.\
```

Example:

To replace the next 5 occurrences of 'black' by 'brown' in a text, an abbreviation would be set up as:

```
$X5"$Gblack\brown"\
```

Example:

If the number of times the loop is to be repeated is set to 0, then the loop repeats indefinitely or until the end of the document is reached. To replace all occurrences of 'black' by 'brown' in a document an abbreviation would be set up as:

```
$X0"$Gblack\brown"\
```

Example:

The system record *VSRN, which is invoked when the view system record names option on the system menu is selected, contains a command to set up abbreviation 1:

```
$S1"'T$X75@$N_\'T"\
```

*VSRN displays a form with room for the names of 75 system records. The user is asked to type ESC 1 to get the next screenful of records. This performs the abbreviation set up above:

`$X75@$N_\`	repeat 75 times the command $N (get next key). Each of the protected fields into which the system record names are read is 11 characters long, as each key is 10 characters the underline after the $N is to force the cursor to move to the next field. Because the string is being set into an abbreviation from a system record the entire string is enclosed in double quotes, the $ within the repeat loop is further protected by an @ sign.

Example:

This program, stored in a record with the key *GLO, does a global replace changing one string for another.

```
[*GLO
'T'AReplace What $(?'H$)8S
'U'VWith What $(?'H$)9S
'U'VHow many times $(?'H$)7S
'U'X$X$7@$G'V$8\$CU$I$9$-I'F@\
```

Explanation of the program.

`'T`	Go to top of screen
`'AReplace What`	Insert a line and add the literal Replace What
`$(?`	Mark the start point for cutting and pause.
`'H`	On receiving a RETURN move one place left over the last character entered.
`$)8S`	Cut the string into keystore 8 and squeeze it off the screen. There is an implicit return at this point.
`'U'V`	Up a line to the home position and delete to end of line. This removes the Replace What
`With What$(?$)9S`	This is similar to the above except that the with what string is cut into keystore 9. There is an implicit return at this point.

`'U'VHow many times $(?'H$)7S`

Again this is similar to the above except that the number of times that the repeat is to be performed in cut into keystore 7.

There is an implicit return at this point.

`'U'X` — Up a line to the home position and delete the line which was created earlier to hold the prompts.

`$X` — Repeat. Everything from this point to the next (RETURN) or \ will go into the repeat string. In this case it is the last character in the record. (The earlier \ is between double quotes and is treated as a literal at this stage.)

`$7` — $7 is activated adding to the string the number of times to repeat.

`"$G'V$8\$CU$I$9$-I'F"`

As this string is between doubie quotes it is treated as a literal string to be entered into the repeat string.

Note that the repeat string now contains the following:

`nn$G'V$8\$CU$I$9$-I'F`

where nn is the number of times that the rest of the string is repeated.

`$G'V$8\$CU$I$9$-I'F`

When activated this string works as follows. (Remember this will be done nn times)

`$G` — Get. Everything from this point to the next (RETURN) or \ will go into the get string.

'V	Clears contents of get string.
$8	String to be replaced.
\	Activate get.

Cursor will move to first occurrence of the string.

`$CU` — Delete the found string.

`$I` — Go into insert mode.

`$9`	Insert the replacing string.
`$-I`	Leave insert mode.
`'F`	Move forward a word.

11.5.10 Conditional Repeat Loops

The format of the conditional repeat commands are described in Appendix C as:

(ESC)`XWString1\String2`	for a repeat while loop
(ESC)`XUString1\String2`	for a repeat until loop

For the repeat while command, String2 is performed while String1 is true. For the repeat until command, String2 is performed until String1 is true.

In both cases String1 is compared with the text on the screen from the cursor position onwards, if it matches the text then the test is true.

When one of the commands is included in a system record any visible equivalents in String1 and String2 must be enclosed in double quotes (") or must be preceded by @. This also applies to the \ used as the delimiter between String1 and String2.

Example:

The system record *PAGINATE, which is invoked when the paginate a document option is selected from the Document Processing Menu is selected, is set up as follows:

```
[*PAGINATE $SH$S1"$P"\$S2"'X"\$S3"$-X'U$L?'M'M"\$-SH'T.PP=0001$X35==\
$XW".PP\'X"\
```

The second line of this record contains a repeat while loop, which says that as long as the cursor is positioned on a line commencing .PP then delete the current line. This ensures that when a document is re-paginated that there is only one .PP at the start of the document.

The repeat while loop could also be written as:

```
$XW.PP@\@'X\
```

or

```
$XW.PP"\'X"\
```

Example:

The system record *LISSYS, which is is invoked when the Print System Records option in the System Menu is selected, contains a repeat while loop:

```
[*LISSYS   $ODSYSREC.DOC\
L......................................................R.................L....
.T
Listing of System Records on $T                        Page PPPP|
-------------------------------------                  ---------|
.C.........
SG$S1"'H'H'H'H'H'H'H'H'H'H"
*>'H'H$N$1$XW*@\"'R'R'R'R'R'R'R'R'R'R$>R))\\\<$N$1"\'H'V$E/P
```

`$XW*@\`

*LISSYS is used to print system records only, that is those records with a key commencing with *. The routine repeats while the cursor is positioned on asterisk, i.e. it stops when the first record which is not a system record is read.

The \ terminating the conditional test part of the command is preceded by an @, otherwise it would be interpreted as the \ which terminates the entire command.

`"'R'R'R'R'R'R'R'R'R'R$>R))\\\<$N$1"\`

The string to be repeated moves the cursor right 10 positions to position it at the end of the system record name which was displayed by the $N command. The record with this key is read and)) is added to the end of it representing the cursor position when the record was written to the VMF. Three carriage returns are performed. A [is displayed, the next key is found and executing abbreviation 1 positions the cursor on the first character of the key ready for the repeat while loop to be repeated.

The entire string to be repeated is enclosed in double quotes to protect the visible equivalents.

Chapter 12. ARRANGING A LEX DATABASE

LEX allows data to be stored in the VMF as records in a similar manner to the system records and menus described in Chapters 10 and 11. Each data record may be up to 2,400 characters in length and the VMF may hold up to 100,000 such records.

A record is identified by a unique 10 character key, the first character of which defines the type of the record. For example in the issue VMF the key . (dot) is used to indicate records containing name and address information.

LEX uses some record types for its own purposes as described in Chapters 4 to 9. It is therefore advisable to avoid record types

*) $:

These system record types are listed in Table 2.1.

The example data base system as supplied on the issue VMF has been set up for the purpose of maintaining an indexed customer file. It could easily be modified to suit any application which involves the storing and retrieval of information and need not necessarily be restricted to names and addresses. For example variations could be created for:

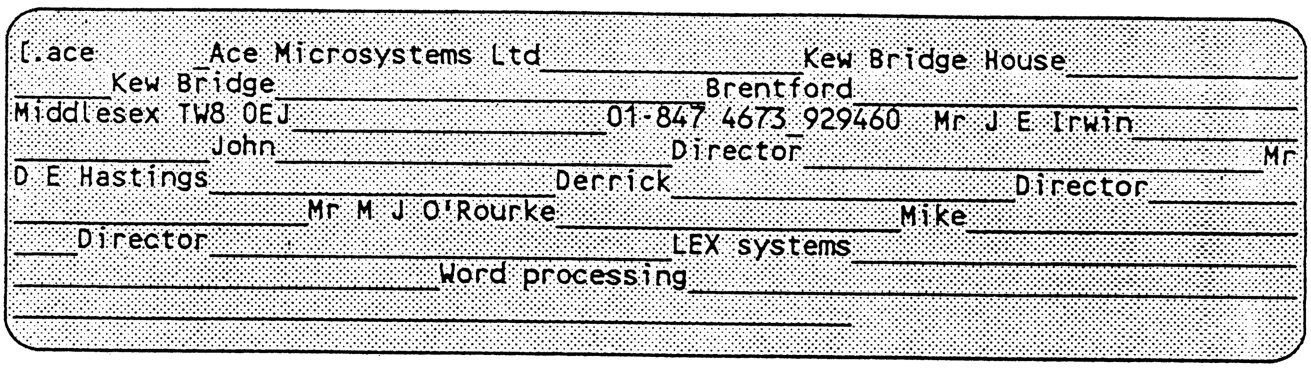
stock files
ledgers
bibliographies
personnel information
medical records
examination questions

Such techniques are explored in this chapter.

Once a data base has been set up, the LEX report generator described in Chapter 13 may be used to access the data for mailshot and list processing purposes. Using these options the name and address system contained in the issue VMF can produce standard letters and answer general enquiries.

12.1 FORMS

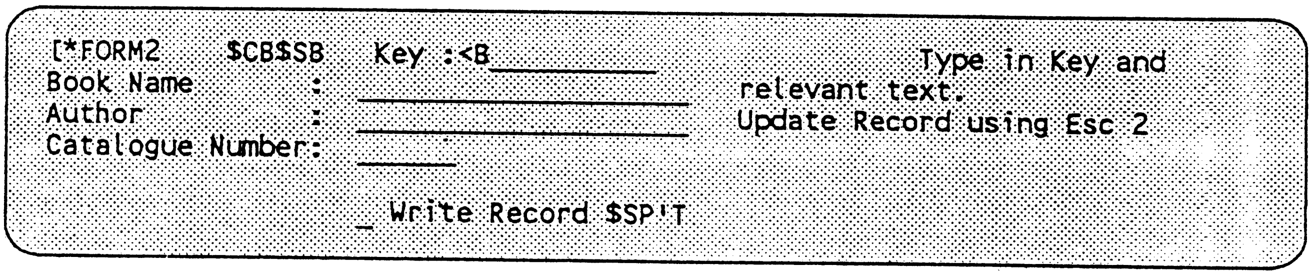
Data records read from the VMF appear as a string of characters, for example:



```
[.ace      _Ace Microsystems Ltd_________________Kew Bridge House_________
_______Kew Bridge__________________________Brentford________________________
Middlesex TW8 OEJ_________________01-847 4673_929460  Mr J E Irwin_________
___________John________________________Director_____________________________Mr
D E Hastings__________________Derrick_____________________Director__________
_____________________Mr M J O'Rourke___________________Mike_________________
____Director_____________________________LEX systems________________________
_________________________Word processing____________________________________
_______________________________________________________
```

Note the underline or paint character identifying blank characters (See (ESC)sq in Appendix C for more details).

Accessing fields within such a string is difficult, so forms are used in LEX as an aid to data input. They are stored on the VMF in the same way as a system record. An example used to catalogue library books follows:



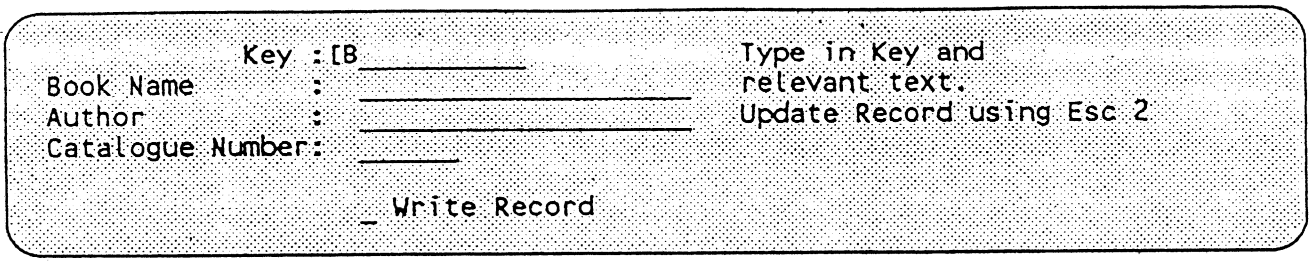
```
 [*FORM2    $CB$SB   Key :<B__________                Type in Key and
 Book Name          :  ____________________    relevant text.
 Author             :  ____________________    Update Record using Esc 2
 Catalogue Number:     _____

                        _ Write Record $SP'T
```

Three areas can be identified in such a record.

Prompts and descriptions
Data input areas
Embedded LEX commands

The area used for prompts and descriptions is protected, that is it cannot be overwritten by the user. The area used for data input is marked by underlines and is unprotected. The cursor is restricted to the unprotected part of the screen when an (ESC)sp (set protect) command is issued. This command can be found at the end of the form as an embedded command.

When activated, the record above appears on the screen as shown below with the underline part in background mode which could be bold, inverse video, etc. The cursor can only access the unprotected underlined area.



```
                Key :[B_________            Type in Key and
Book Name          : ________________       relevant text.
Author             : ________________       Update Record using Esc 2
Catalogue Number:    _____

                     _ Write Record
```

An explanation of this record follows:

CBSB	Embedded LEX commands to clear the screen and set background mode.
Key :	A prompt which appears as shown.
<	The visible equivalent for a [to define the start of a key.
B	This character is used to define the record type. It can be any character other than those used for another purpose.
B________	The next 9 underlines are reserved for the remainder of the key. The tenth underline is used for positioning the cursor to allow the read record command to be used.

```
                                     Type in Key and
Book Name          : ________________  relevant text.
Author             : ________________  Update Record using Esc 2
Catalogue Number:  _____
```

These lines show the prompt messages that will appear on the screen when the form is activated and the unprotected area in which the data may be entered. Note that each underline represents one character in the data record and therefore defines the field size.

_Write Record	When a command to update or create a system record is given, data is written away from the cursor position to the previous [. This field is used for positioning the cursor when the record is written to ensure that all data in the record is written away.
$SP	Sets protect mode and restricts the cursor to the underlined parts of the screen.
'T	Positions the cursor on the first underline in the form.

12.1.1 Using forms

Forms are activated in the same way as any other system record. They can be either set into a menu as described in Chapter 10 or by typing [key], for example:

[*FORM2]

When the form has been displayed on the screen, data can be entered into the unprotected area. Note that all entries in the unprotected area are in background mode.

To read a record over a form enter its key and the command ESC]R. Data is then inserted character by character from the record into the unprotected area of the screen.

To create or update records type in the key and relevant data. Then move the cursor to the last unprotected character and type ESC]C to create a record and ESC]U to update it.

It should be noted that to save space in the VMF LEX squashes underline characters to a single byte. See ESCSQ in Appendix C and Chapter 2 for further information.

12.1.2 Editing restrictions in protected fields

The standard LEX editing functions work as normal when entering data in a form, with some restrictions. These are noted here.

CTRLB CTRLF CTRLG CTRLL CTRLT can be used to move around the form as well as TAB and RETURN and the arrow keys.

CTRLV deletes to end of field, CTRLX cannot be used.

CTRLN or CTRLE inserts a space into existing text but will not push text down onto a new line or across into another field. The text disappears at the end of the line or field.

Similarly CTRLP and CTRLD delete characters and pull the paint character into the right hand end of the field. Text is not pulled from the next field.

ESCI cannot be used to insert text and causes ERROR-PM to be set and a BELL sent to the VDU.

The use of cut and paste in forms mode is limited to cut and leave and paste and overlay. However, cut and paste should be undertaken with care, as problems may arise if text cut from a protected area is pasted over another protected area of a different size/length.

(CTRL)Y copies the character from the line above irrespective of whether the character above is in a protected area or not.

12.1.3 The use of rulers in forms

Rulers may be included in a form. If there is more than one ruler then the ruler in force is the last one above the cursor.

```
[*INPUT  $CB$SB   <P__________ $SH$SG
...........F.......................................M.....
$-SH
Product
Description ____________________________________________
            ____________________________________________
            ____________________________________________ $SH
...........####.##.........##............................
$-SH
Cost Price  ______    Stock __
Retail Price______
                                 _ Write Record $SP$UB$-SG'T
```

In this example as the product description is typed it is word wrapped onto the next line when the text meets the right hand margin set by the first ruler.

Also, using the tab command under the calculator ruler causes figures input to be decimally aligned or right justified.

The following are also of note in this example.

$SH	Puts LEX into shush mode and therefore does no display the rulers when the form is in use
$-SH	Exits from shush mode
$SG	Forces LEX to ignore the automatic assertion of rulers.
$-SG	Restores LEX to normal mode where rulers are automatically asserted

12.1.4 Masking

In the previous section an entire record was read onto the unprotected area of a form in the order that it is stored. LEX's masking facility allows users to read and update specific parts of a record.

The uses of masking are:

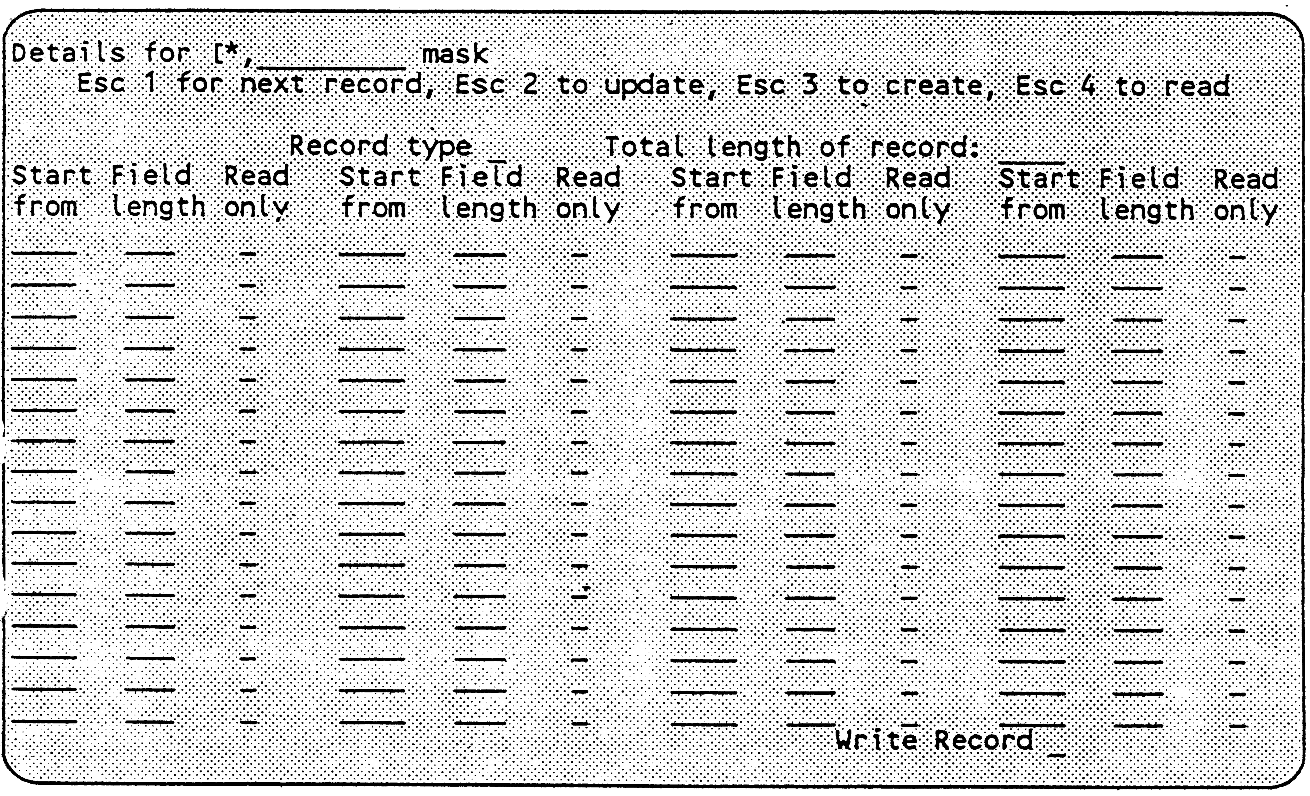
a) For records which occupy more than one screen (up to 5,000 characters).

b) In cases where only part of a record is to be displayed - perhaps because a field is not to be made available to all users.

c) Where fields are to be displayed in a different order from the order in which they are stored.

d) When a field is to be displayed but protected from being updated.

A system record called *MASKFORM is supplied with the VMF file to allow masking details to be set up. It may be called from an option is the system housekeeping menu or by typing [*MASKFORM] in the playground. It takes the following form.



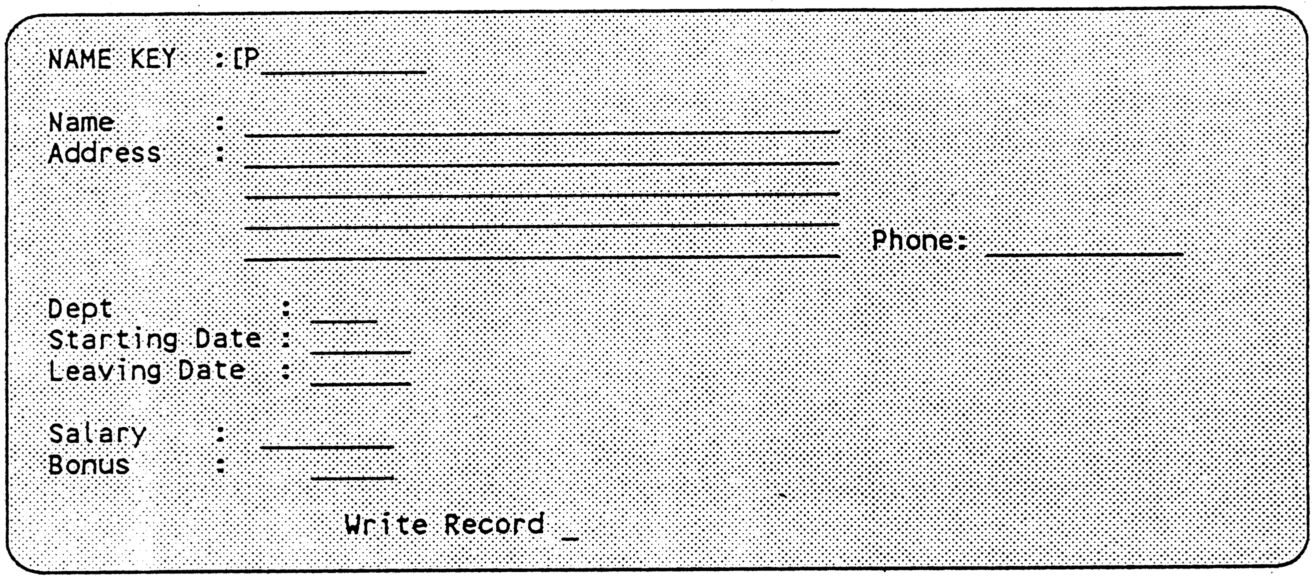
```
Details for [*,________ mask
    Esc 1 for next record, Esc 2 to update, Esc 3 to create, Esc 4 to read

                 Record type _       Total length of record: ____
Start Field  Read   Start Field  Read   Start Field  Read    Start Field  Read
from  length only   from  length only   from  length only    from  length only

____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
____  ___     _     ____  ___     _     ____  ___     _      ____  ___     _
                                              Write Record _
```

Masks are stored in records with a record type of *,

The record type, in the mask format line, specifies the record type to which this mask applies. The length is the total length of the record and may be greater than the size of the screen.

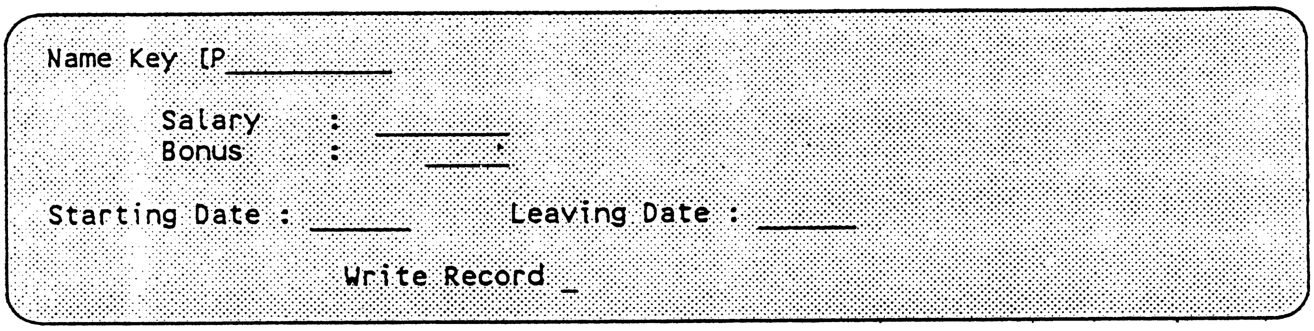
Up to 64 position and length fields may then be entered. The order in which fields are entered is the order in which they will appear on the screen although this need not be the order in which they are stored in a record. Each field is defined by the start position of the field in the record. The first character in the record after the key is position 1. The length defines the number of characters in the field and each field may be marked with an R if it is to be read only.

Consider the following form used to access data in the VMF.



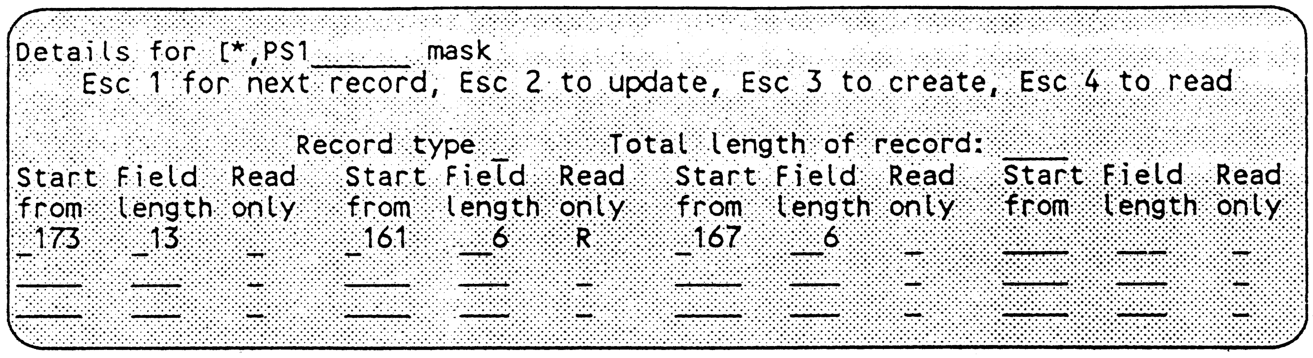
```
NAME KEY   :[P__________

Name       : ____________________________________
Address    : ____________________________________
             ____________________________________
             ____________________________________
             ____________________________________  Phone: ____________

Dept          : ___
Starting Date : ______
Leaving Date  : ______

Salary     :  ________
Bonus      :     _____

                     Write Record _
```

To demonstrate the use of masking assume that another screen is to take the form below where only some fields are shown and these in a different order from that above. Further assume that the starting date is to be a read only field when this form is used.



```
Name Key [P__________

       Salary   :  _______
       Bonus    :     _____

Starting Date : ______       Leaving Date : ______

                     Write Record _
```

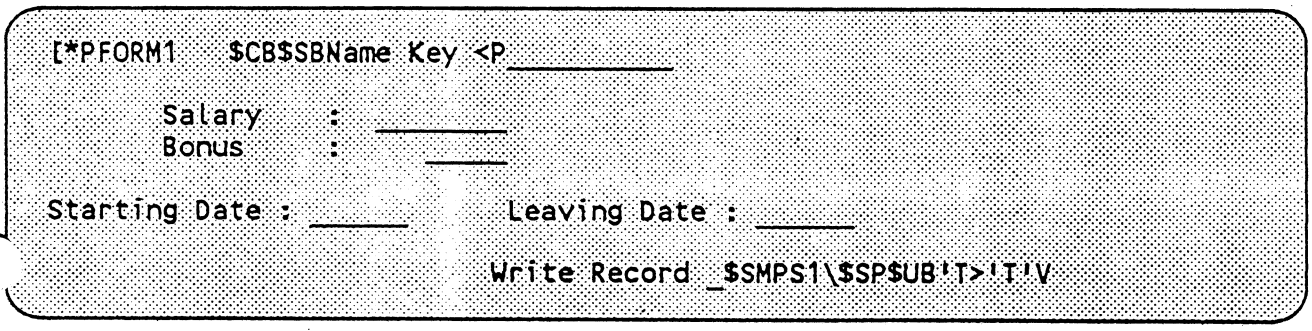
If the mask is to be known as PS1 then *MASKFORM should be used to create a mask record.



```
Details for [*,PS1______ mask
   Esc 1 for next record, Esc 2 to update, Esc 3 to create, Esc 4 to read

                   Record type _      Total length of record: ____
Start Field  Read   Start Field  Read   Start Field  Read   Start Field  Read
from  length only   from  length only   from  length only   from  length only
_173  _13    _      _161  __6    R      _167  __6    _      ____  ___    _
____  ___    _      ____  ___    _      ____  ___    _      ____  ___    _
____  ___    _      ____  ___    _      ____  ___    _      ____  ___    _
```

This mask allows update access to those records of type P from position 173 in the record for a length of 13 (salary and bonus inclusive). Read access only is allowed to the field of length 6 starting at position 161 (Starting Date) and update access is allowed to the field of length 6 starting at position 167 (Leaving Date). The fields are displayed in this sequence.

A mask is enabled by the escape sequence (ESC)SMmaskname(RETURN). This may be performed at the keyboard, e.g. (ESC)SMPS1(RETURN) or in a system record. Below is a system record to produce the screen format described above.



```
 [*PFORM1   $CB$SBName Key <P_________

        Salary     :  _______
        Bonus      :     ____

 Starting Date : _____         Leaving Date : _____

                               Write Record _$SMPS1\$SP$UB'T>'T'V
```

Note that masks are designed to work in protected forms. If used in normal edit mode trailing spaces are automatically removed on writing records which may cause data to be out of phase.

A particular (ESC)SM command remains in force until either another (ESC)SM command is executed, an (ESC)UM is executed to unset masking or (ESC)E is sent.

12.1.5 Accessing records in other VMFs

The (ESC)SF command (set Foreign File) allows a specified record type in a different VMF file to be accessed. The VMF file can be opened for reading or writing using the following command:

```
(ESC)SFabcd@filename@
```

where abcd have the following significance.

a	The record type to be associated with the foreign file.
b	The file type. Set to V to signify a VMF.
c	If set to R then read only access is given. Any other character provides read/write access.
d	This character sets the mode in which the foreign file is to be opened. M assumes multi user, N assumes single user and any other character uses the system standard as set in **INST of the main VMF, not the foreign file.

See (ESC)SF in Appendix C and Chapter 2 for further details.

The following example shows foreign file access sequence:

```
(ESC)SF,VRML.VMF(RETURN)
```

opens the VMF file L.VMF as a foreign file for Read only access in multi user mode. Any reads or writes of records of type , are performed on L.VMF and not on the main VMF. The foreign file currently open is closed by the command (ESC)UF or menu switch option /2 after which all references to record type , would be made on the main VMF.

Note that foreign file access should be used for reading data only and it should not be used to execute a record. However all read/write escape sequences may be applied to foreign files including partial memory reads. (See Field Referencing below).

12.1.6 Field referencing (partial record reads)

It is possible to recall part of a record stored on the VMF rather than the whole record. This is achieved by specifying the key followed by a displacement defining the start position in the record, and the number of characters to read. The general form is:

```
[aKEY+nnn/mmm
```

where a is the record type, KEY the record key, nnn the start position in the record and mmm its length.

i.e. if the full record is:

```
[,KEY _0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

then

```
[,KEY+11/10(ESC)]R
```

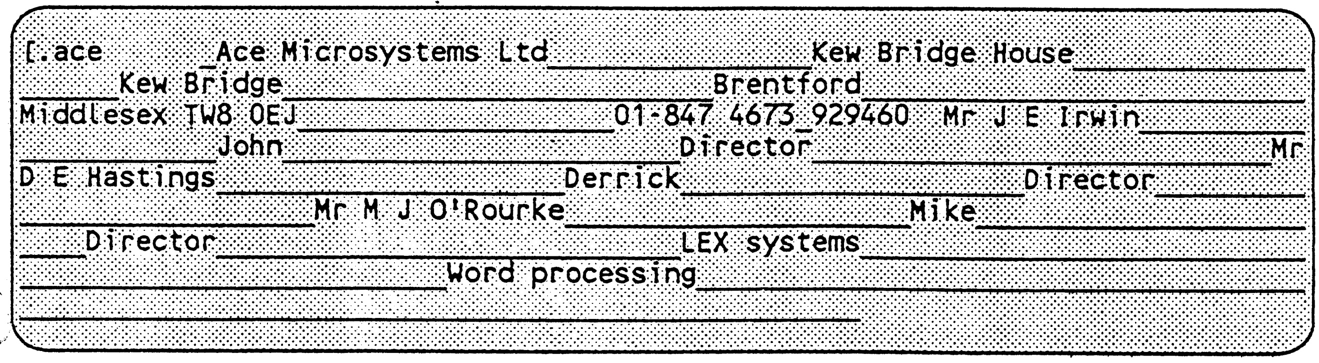
will display:

```
[,KEY+11/10 ABCDEFGHIJ
```

This technique may be used to read or write parts of a record in exactly the same way as a normal read except that the key and field reference occupies 18 characters in length and masks must NOT be in force.

As with a standard record read, a record may be actioned by terminating the key with]. The key is removed from the screen and the text from the data record displayed. In this case there is a fundamental difference between partial memory reads and the normal reads in that only text is shown and no visible equivalent conversion is performed. This is because the feature is designed for accessing data including names and addresses which may include O'Reilly, etc.

Consider the data record:



```
[.ace      _Ace Microsystems Ltd________________Kew Bridge House______________
______Kew Bridge__________________________Brentford___________________________
Middlesex TW8 0EJ___________________01-847 4673_929460  Mr J E Irwin__________
____________John________________________Director____________________________Mr
D E Hastings_____________________Derrick_____________________Director_________
__________________Mr M J O'Rourke_____________________Mike____________________
____Director____________________________LEX systems___________________________
__________________________Word processing_____________________________________
___________________________________________________
```

If [.ACE+1/36] is entered then 36 characters starting at character 1 of the record with key of .ACE are displayed:

```
Ace Microsystems Ltd________________
```

or if $SQ [.ACE+1/36] is entered (squash character is set to spaces) then the underlines are replaced by spaces and the line is displayed as:

```
Ace Microsystems Ltd
```

This technique can be used to create a letter to a known contact. Consider the record *LETTR2 which could be used to send a letter to a contact. It is a variation on the record *LETTR which can be found in the issue VMF.

```
[*lettr2   $SQ 'T$CF                    Customer Key __________ 'T$SP'T$(?'H$)0L$UP
'T'V1) $SH<.$0+201/36$-SH>\2) $SH<.$0+301/36$-SH>\3) $SH<.$0+401/36$-SH>\
Which Contact - 1, 2 or 3 __ $SP'T$(?'H$)9L$UP'T$CF    $+P0$9+1= 'B$('G{$)9B
F.......................................................................M.
$Y\\\$SH<.$0+$901/36$-SH>
$SH<.$0+1/36$-SH>
$RK.$0+37/36\
$RK.$0+73/36\
$RK.$0+109/36\
$RK.$0+145/36\\\\
Dear $SH<.$0+$937/28$-SH>\\
```

When this is invoked by entering [*LETTR2] a prompt is displayed:

```
Customer Key __________
```

If ACE is used as the key response then the user is allowed to choose from one of three contacts in the named company.

```
        1) Mr J E Irwin
        2) Mr D E Hastings
        3) Mr M J O'Rourke

        Which Contact - 1, 2 or 3 __
```

Having selected a contact the prompt messages are removed from the document and the appropriate name and address entered in the document. If 2 was chosen:

```
F.......................................................................M..
27th October 1985



Mr D E Hastings
Ace Microsystems Ltd
Kew Bridge House
Kew Bridge
Brentford
Middlesex TW8 0EJ



Dear Derrick
```

There follows a full explanation of the *LETTR2 record:

`$SQ`	Set the squash character to spaces so that any underlines that were squashed when the record was written away are unsquashed as spaces. The space is the blank character left between $SQ and 'T.
`'T$CF`	Go to the home position and clear the screen.
`Customer Key ________`	Prompt for the key to be input
`'T$SP'T`	Set protect mode so the cursor is limited to the underlines and position on the first underline.
`$(`	After the key has been entered it is going to be cut into keystore 0, this marks the start of the cut.
`?`	Pause until the key has been entered and terminated by carriage return.
`'H$)OL`	Move cursor left over the last character of the key and cut it into keystore 0.
`$UP`	Unset protect mode.
`'T'V`	Delete the top line which contained the prompt.
`1) $SH`	The characters 1) are printed and then LEX goes into shush mode.
`<.$0+201/36$-SH>\`	This is a partial memory read. < is the visible equivalent of [. This is followed by the record type . and the contents of keystore 0. Keystore 0 was set up above with the key of the name/address record required. The 201/36 looks for the 36 characters from position 201 i.e. the name of the first contact. All these entries have been in shush mode so do not appear on the screen but this mode is now returned to normal by $-SH before activating the partial memory read with]. The name appears followed by a RETURN signified by the \.

`2) $SH<.$0+301/36$-SH>\3) $SH<.$0+401/36$-SH>\`

In a similar manner to the last sequence described, the second and third contact are extracted from the data record and displayed on the screen.

`Which Contact - 1, 2 or 3 _`

Prompt for the contact to whom the letter is to be addressed.

`$SP'T` Set protect and go to the top of the unprotected screen - the only underline.

`$(?'H$)9L` The underlined position is marked for a cut and paste operation before pausing for entry of a number followed by (RETURN). On receiving the (RETURN) the number entered is cut into keystore 9.

`$UP'T$CF` Unset protect, go to the top of the screen and clear the screen leaving it blank.

`    $+P0$9+1=` The `$+` puts LEX into calculator mode and the four spaces prior to it ensures enough space for the calculation. `P0` sets no zero places. The contents of keystore 9, a number, is then entered into the calculator and one added to it. The final blank exits the calculator.

`'B$('G($)9B` The screen now has a number displayed with the cursor beyond it. This sequence goes back to that number, cuts it into keystore 9 overriding its previous contents and then blanks it out.

`F....................................................M.....`

Future entries will obey the new ruler. Otherwise the screen is blank.

`$Y\\\` Today's date is entered followed by three (RETURN)s.

`$SH<.$0+$901/36$-SH>`

In shush mode a key is entered consisting of keystore 0 as entered above. The start location of the contact required is established by the number in keystore 9 and 01. That is if contact 1 was wanted the location is 201, for contact 2 it is 301 and for the third contact 401. The length of this name field is 36 characters.

```
$SH<.$0+1/36$-SH>
```

The $0 is replaced by the string entered in response to the customer key prompt, and 36 characters starting at character 1 of the record are displayed, i.e. the first line of the address.

```
$RK.$0+37/36\
```

This accesses the second line of the address. It makes use of the $RK sequence which activates a record without displaying it, in the same way as shush mode has been used above. Note that the \ is the terminator to perform the action.

```
$RK.$0+73/36\
$RK.$0+109/36\
$RK.$0+109/36\
$RK.$0+145/36\\\
```

These commands access the remainder of the address, the last line being followed by three RETURNs.

```
Dear $SH<.$0+$937/28$-SH>\\
```

The final part of the program writes Dear and pulls the contacts colloquial name from the record at location 237, 337 or 437 in a similar manner to the full name shown above. It is followed by two RETURNs. The user now types the text of the letter.

Chapter 13. REPORT GENERATING, LIST PROCESSING AND MAILSHOT

The report generating facility accesses selected records in a LEX database and outputs parts of them in a user defined manner. The facility consists of two options, list processing and mailshot, both of which are considered in this chapter.

Before either system can be used a VMF database of suitable records must be created as described in Chapter 12. This chapter assumes that an appropriate database exists.

List processing is the term used to generate reports and statistics from the data records. It is initiated by the /S menu switch option and allows output to the screen or printer. Headings are provided at the top of the output and these are repeated on top of subsequent pages or screens. Under the headings nominated fields are displayed and repeated for each record selected.

Mailshot usually refers to the production of personalised letters each to be headed with a different address and salutation taken from the data base. In this option the output is effectively the opposite way round to list processing. The heading part of each letter is taken from the data records and merged with a standard document containing the body of the text. This is then repeated for each record selected. Menu switch option /L initiates this facility.

Both report generating options are similar in that the data records referenced may be selected to satisfy given conditions. Further, two special system records are required to define the output. One, a dictionary, defines the fields held in the data records whilst the other describes the output format.

In summary, report generation consists of three stages:

a) Using the menu switch option to select list processing or mailshot.

b) Defining the format of the output.

c) Selection of which records are going to be used in the output.

Before considering each of these stages in turn it is necessary to consider the dictionary where the fields within a data record are defined.

13.1 THE DICTIONARY

A dictionary associates field names and format characters with fields in a data record. It is mandatory in all report generating processes.

Dictionaries are stored in the VMF as system records. It is useful to include *DICT as part of the key for ease of identification.

The structure of the dictionary record consists of fields separated by spaces. For convenience they are usually structured in columns as in the example below.

```
[*DICT
N NAME       36     A ADDRESS  144     P PHONE      12     T TELEX      8
1 CONTACT1   36     2 CLQNAME1  28     3 TITLE.1    36     4 CONTACT2  36
5 CLQNAME2   28     6 TITLE.2   36     7 CONTACT3   36     8 CLQNAME3  28
9 TITLE.3    36     I INSERT1   66     J INSERT2    66     L INSERT3   66
a ADD1 ADDRESS+0/36                    b ADD2 ADDRESS+37/36
c ADD3 ADDRESS+73/36                   e ADD4 ADDRESS+109/36
R RECORD     NAME+0/698
```

The fields in the dictionary are defined in the same order as they are stored in the record. This is particularly important to remember when records are stored under masking where the order of input may differ from the structure of the stored record.

Although not shown in the example it is possible to put comments in a dictionary. Anything to the right of a ! on a line is ignored.

Associated with each field are three items:

- Formatting character
- Field name
- Field length

13.1.1 Formatting character

The first item is a single character known as the formatting character. It is used in the output format system records to define field positions for printing as described later in this chapter.

Up to 77 dictionary fields may be defined by using any formatting character except those reserved for special purposes or features. The following **MONT defined characters are illegal:

!	Indicates ignore rest of line
&	indicates continuation line
'	Indicates literal
<	Field in format
>	Field in format
[	System record
]	System record
_	Paint character

(SPACE) may not be used and certain alphabetic characters are also reserved. These are defined in **TEXTP and are associated with special features. They are listed in Table 13.1 below.

Therefore the formatting character may be:

```
#$%()*+-./0123456789:;=?"ABCEFGHIJLMNOPQRSTUVWXY\^
abcefghijlmnopqrstuvwxy{|}
```

Note however that if a field is to be used for selection purposes the formatting character must be alpha-numeric, i.e.

```
0123456789ABCEFGHIJLMNOPQRSTUVWXYabcefghijlmnopqrstuvwxy
```

All other allowable formatting characters are for use in output formats only.

TABLE 13.1 - SPECIAL FEATURE DICTIONARY ENTRIES

Formatting character	Dictionary name	Length	Use
K	KEY	9	Record key, excluding record type.
k	RECTYPE	1	Record type - first character of key.
D	DATE	22	Today's date as formatted in **MONT.
d	PAGE	4	Current page number in a list processing report.
T			Reserved for future use.
t	TIME	8	Time in the form HH:MM:SS.
Z	TRECS	5	Total number of records of the type specified.
z	SRECS	5	Number of records selected that conform with the selection criteria.

13.1.2 Field name

The field name is used to refer to a field during the record selection phase of producing reports. It can also be used in the output formats.

Any combination of alpha or numeric characters may be used including a . (dot) but the length is limited to 8 characters.

As with formatting characters some names are reserved for special features and must not be defined by the user. They are defined in **TEXTP and listed in Table 13.1 above.

13.1.3 Field length

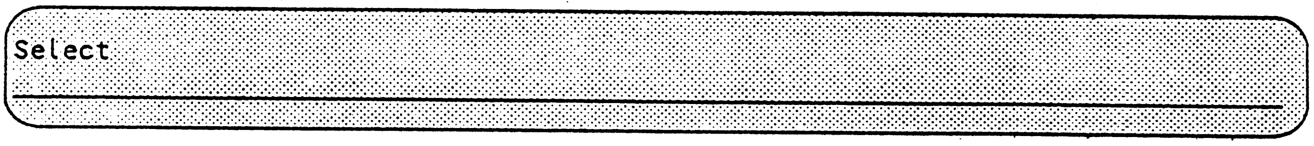
The final item of information is the field length.

Entries in the dictionary may also be made using a displacement and length from an existing field. In the examples shown above the field ADD2 refers to a data field commencing 37 characters from the start of the ADDRESS field and has a length of 36 characters. If the address consists of four lines of 36, totalling 144 characters, ADD2 defines the second line of the address.

In the example the field RECORD refers to the entire data record. Note that negative displacements may also be used. This is a useful method of getting at the very first character in a record or part of the key.

13.2 THE SELECT FUNCTION

Both the /L and /S menu switch options lead to a select prompt where criteria can be entered to select data records.



```
Select
```

A field name is compared against a given string by entering, for example:

```
NAME=JONES
```

where NAME is a field name from the dictionary in use and JONES is a string. This example selects all records where the field NAME is equal to JONES. The operator = may be one or more of:

=	equal to
>	greater than
<	less than
:	contains
-	between

Choices may be combined using the logical operators AND, NOT and OR. The following example selects records for plumbers with London or Glasgow in the address.

```
ADDRESS:London,Glasgow AND JOB=Plumbing
```

A count of the number of records processed can be displayed at the end of a mailshot or list processing output by setting the relevant fields in **INST. The display consists of two numbers. The first is the number of records meeting the selection criteria and the second is the number of records that were tested. The display takes the form:

```
Number of records selected: 0044 / 0154
```

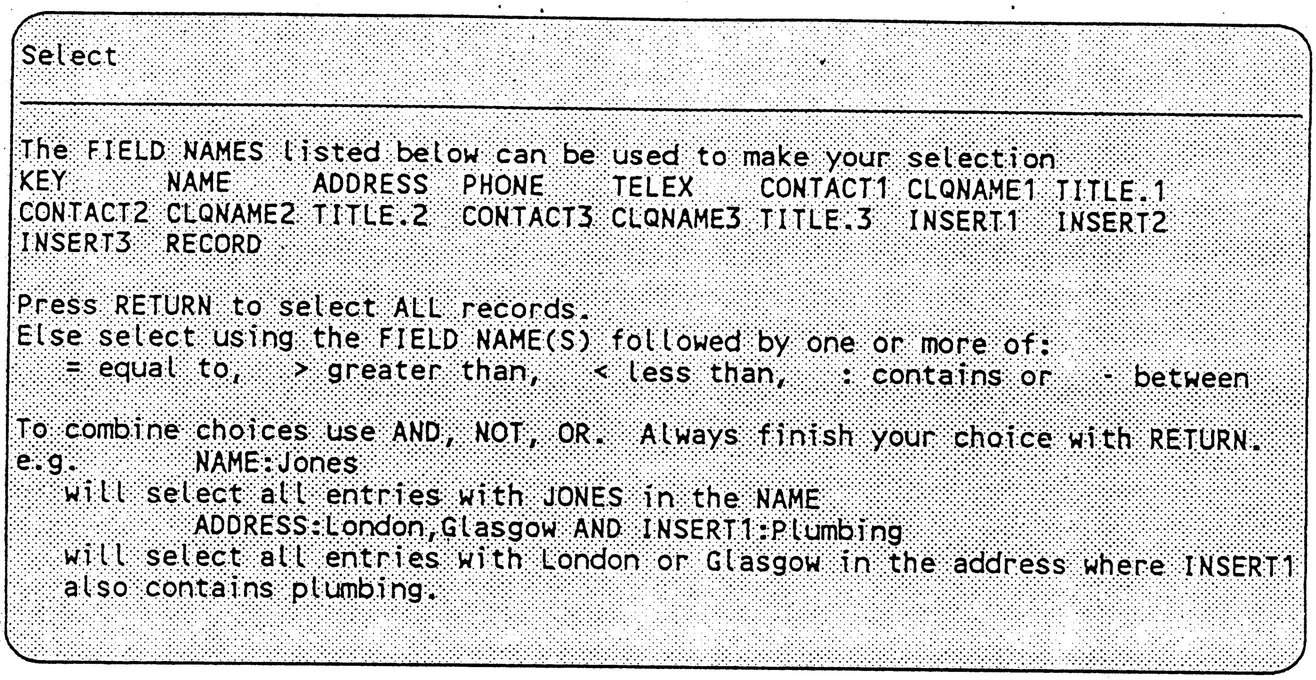
Similarly the selection criteria can be displayed at the end of a mailshot or list processing output by another setting in **INST. In this case the display follows the number of records selected in the form:

```
Selection Criteria
NAME=JONES AND ADDRESS:GLASGOW
```

Further features may be associated with the select function and they are now discussed.

13.2.1 The Help Message

A reply of HELP to the select prompt displays a list of all fields available for selection as shown in the dictionary in force. The list is followed by the message contained in a system record. The key of the help message record may be defined in the first line of the output format but it defaults to *HELPS. In the issue VMF the resulting message is:



```
Select
________________________________________________________________________

The FIELD NAMES listed below can be used to make your selection
KEY      NAME     ADDRESS  PHONE    TELEX    CONTACT1 CLQNAME1 TITLE.1
CONTACT2 CLQNAME2 TITLE.2  CONTACT3 CLQNAME3 TITLE.3  INSERT1  INSERT2
INSERT3  RECORD

Press RETURN to select ALL records.
Else select using the FIELD NAME(S) followed by one or more of:
   = equal to,   > greater than,   < less than,   : contains or   - between

To combine choices use AND, NOT, OR. Always finish your choice with RETURN.
e.g.      NAME:Jones
   will select all entries with JONES in the NAME
          ADDRESS:London,Glasgow AND INSERT1:Plumbing
   will select all entries with London or Glasgow in the address where INSERT1
   also contains plumbing.
```

Note that the message may be displayed automatically with the select line if the feature is enabled in `**INST`.

13.2.2 Searching a range of keys

If in response to the Select prompt

```
KEY>G
```

is specified as the first element of the select option, then LEX does not test keys of a lesser value; it starts searching from the specified point. Therefore, all records on file whose keys commence with a character with a lower value than `G` will not be included in the search.

Similarly, `KEY<G` as the first element of the selection criteria causes LEX to ignore any record whose key commences with a character greater than or equal to `G`.

Note that only the first element of the selection functions works in this manner so care should be taken to ensure the best method of searching files:

```
ADDRESS:LONDON AND KEY<G
```

results in all records being searched, whereas

```
KEY<G AND ADDRESS:LONDON
```

restricts the search to records before the first key commencing with `G`. This is therefore a faster option.

The logic of some selection criteria may override this. For example the OR in the following case causes the whole set of records to be searched.

```
KEY>H OR NAME:ACE
```

To restrict a search to a range of keys a - can be used.

```
KEY=ACE-MAN
```

This causes only those records with keys from ACE to MAN inclusive to be accessed.

13.2.3 Use of double quotes in selection

Searches for fields containing the operands such as = : < and other special characters may be achieved by putting the string between double quotes. Quotes can also be used when the search parameter is more than one word, e.g.

```
NAME="Ace Microsystems Ltd"
```

13.2.4 Use of wildcards in selection

Strings used in response to the Select question may contain wildcard characters. A wildcard character is indicated by a ? in the string, for example:

```
CODE=A?E?
```

this will cause records to be selected in which the field CODE is A(any character)E(any character).

Note that wildcards are only applicable to selections made using the equals criteria and the fields within the record and not when selecting on key names.

13.2.5 Case independent searching

When list processing and using the contains criteria (usually :) the specified string is searched for without checking the case of the string. Thus:

```
ADDRESS:London
```

selects all records where the ADDRESS field contains any of the strings, LONDON, London or london.

With this option the number of embedded spaces is not significant.

13.2.6 Numeric/alphanumeric comparisons

Comparisons in the report generator may be performed on a numeric or a character string basis. In a numeric compare the value of 12 is greater than 2. In a character string compare the string 2 is greater than the string 12.

Numeric comparison is achieved by setting the free form indicator in **INST. When this indicator is set, select comparison for fields with numeric formatting characters are performed on a numeric basis. Fields with alphabetic formatting characters are still considered on a string basis.

Free form compare can also be nominated in the output format records (see 13.3.1)

As an example consider the following records containing fields and formatting characters as shown.

Field name	DEPT/INDIV	HOTEL	TRAVEL
Format character	A B	H	2
Record 1	100/01	60.00	50.00
Record 2	100/01	120.00	89.00
Record 3	100/02	100.00	150.00

Character string selection criteria using TRAVEL>50 results in no records being selected. On the other hand with numeric selection the following records are selected.

DEPT/INDIV	HOTEL	TRAVEL
100/01	120.00	89.00
100/02	100.00	150.00

Note that as the formatting character of HOTEL is alphabetic (H), then select criteria HOTEL>50 will never output any records whatever the free form indicator. Use of an alphabetic formatting character in this instance therefore inappropriate.

13.2.7 Automatic deletion of records

If /K is typed at the SELECT option then those records selected are deleted. As a safeguard two /Ks must be entered or, alternatively /K/Q. For example

```
NAME:ACE/K/K              NAME:ACE/K/Q
```

Note that the letter to enable this kill option, K in this instance, is definable in **MONT. If omitted from **MONT then this facility is disabled.

13.2.8 Using system records for selection

In response to the Select prompt a system record containing the selection criteria may be activated in the normal manner. e.g.

```
[*SEL]
```

The record *SEL should then contain the selection criteria, say

```
[*SEL     NAME:ACE AND ADDRESS:LONDON
```

The selection details are not displayed on the select line but still appear at the end of the output if the selection criteria display is enabled in **INST.

This technique is useful when the select line would be longer than 80 characters or when standard and complex selections are often repeated.

13.3 OUTPUT FORMATS

An output format determines the way in which LEX views or prints the record details on file.

It is a good idea when creating output formats to include *OUT in the key to ease identification.

The output formats are split into three sections, the first of which defines:

- the dictionary to be used
- the record type of the records to appear in the output
- whether output should be printed or viewed
- page or screen limits
- fields to be totalled and other ancillary options

The second part only applies to list processing and not mailshot. It shows the headings that are required on the first and subsequent pages of output.

The third part defines which of the fields are to be output for each selected record and their format.

For list processing or mailshot the output format record must conform to the layout detailed below.

13.3.1 Top line

The first line of the output format is described below using an example:

```
[*OUTS1     *DICT P6066 .05F KEY=? /U:12:#
```

If there is not enough room on the line for all details then the continuation character, usually & may be appended in column 80 thereby making the next line a continuation.

*DICT	The first entry on the top line is the name of the dictionary that is to act as the reference map to the data record.
(SPACE)	The dictionary name must be followed by one or more space characters.
P	The next character is separated from the dictionary name by one or more spaces and defines whether output is to be printed or viewed. If V is found in this position then output is displayed on the screen. If P is found then the Which printer prompt allows the user to choose the output device.
60	Immediately after the P or V the page or screen size is defined The first two figures are the number of lines to print per page or screen.
66	The second two figures give the page size. They are ignored when viewing the output as the number of lines on the VDU is taken from *VDUFORM. Use FF in place of 66 if a form feed is required.
(SPACE)	The page size definition must be followed by a space.
.	The character in the next position is the record type. Only records with this record type will be selected for output. For example if as here a . (dot) is specified then only records with a . in the first position of the key are output.
05	The two numeric characters immediately after the record type are used to set the left hand margin of the printed report. A left hand margin setting of 00 is assumed if this field is omitted.

F	If F is specified in this position then LEX performs numeric comparisons at select time. (For example 2 is less than 10 in a numeric compare but greater in an alphabetic compare). The numeric compare is performed only on fields which have a numeric formatting character in the associated dictionary. If this field is blank then the free format field in **INST determines whether compares are alphabetic or numeric.
(SPACE)	One or more space characters must follow these codes.
KEY=	Part or all the selection criteria may be entered on this line. See 13.2 for details on select criteria.
?	The ? causes LEX to display the select prompt described in 13.2. When used in conjunction with selection criteria such as the KEY= example, the prompt becomes: `Select KEY=` Selection works as described in 13.2 with, in this example, KEY= supplied by LEX as the first part of the logic. If the ? is omitted then no select prompt is displayed. All records of the specified record type will be selected unless some criteria is included in this top line.
/U:12:#	Optional operands may be defined on the header line, these all start with a / and are described in 13.5. The options include:

/A	To sort selected output records into ascending order
/D	To sort selected output records into descending order
/C	To assert print time special characters
/H	To assign a help message at select time
/P	To set a paint character
/S	To set line spacing
/T	To define fields to be totalled
/F	To access foreign files
/U	To update files during list processing

13.3.2 Heading lines

Heading lines apply to list processing only.

If the list processing menu switch option /S is used then all the lines directly under the top line until a blank line is met are taken as the report header. These lines appear once at the top of every page or screen. If the blank line is missing in list processing then a run time error occurs with the **TEXTP message Bad Format.

The information included in heading lines may be text, field names (in angle brackets) or formatting characters (in single quotes). Field names and formatting characters are taken from the dictionary in current use or from the special reserved names listed in Table 13.1.

Use of formatting characters fixes the position of the display whereas field names in angle brackets ignore trailing spaces in the output field. The following example shows these different techniques. Formatting character N is associated with field name NAME.

```
[*OUTX3    *DICT V2123 .     ?
List of Customers - Page'PPPP'
<DATE> <RECTYPE><KEY>
<NAME>One
'NNNNNNNNNNNNNNNNNNNN'Two
```

would display:

```
List of Customers - Page    1
21st May, 1984 .ACE
Ace MicrosystemsOne
 Ace Microsystems     Two
```

13.3.3 Detail lines

The lines following the blank line (or top lines in mailshot) specify the layout of the output. This may consist of fields from the record and literals. If any character is included in column 80 on a detail line the following line is assumed to be a continuation line thereby allowing wide lines to be output.

Literals

Literals may be included in the detail lines by enclosing the string in single quotes. Characters not in quotes are assumed to be formatting characters as described below. (Note that in the heading lines it is the formatting characters that are in quotes and not the text.)

Use of field names

The position of a field may be defined using the dictionary field name.

Consider a dictionary entry with two fields, NAME and CITY. These are both 20 characters in length.

The dictionary name is enclosed in angle brackets.

```
<NAME><CITY>
```

will output

```
Mr E ArkwrightAberdeen
```

Note that trailing spaces are omitted and the next field is moved to the left.

Use of Formatting Characters

Alternatively the position of a field may be defined using the formatting character.

The formatting characters fix the position of the output by ensuring trailing spaces in a field are printed. If, in the previous example, the formatting characters N and C are associated with NAME and CITY respectively, then

```
[*OUTX2     *DICT P6066 .        ?
NNNNNNNNNNNNNNNNNNNN CCCCCCCCCCCCCCCCCCCC
```

will output

```
Mr E Arkwright       Aberdeen
```

If the number of formatting characters used is less then the length of the field then the field will be truncated. If the number of formatting characters used is greater than the length of the field then characters in the following field will be displayed.

The special field names and formatting characters may also be used in the detail lines.

13.4 THE MAILSHOT DOCUMENT

Mailshot requires a document to be used in conjunction with the output format. List processing requires the output format only.

A mailshot document is a standard ASCII document created in the same way as any other document. With a mailshot document however, it is possible to include variables in the text. This is done by enclosing in angle brackets the dictionary name of the required field. For example a document may be set up as follows.

```
F.................................................................M............
Thank you for your request for information on <INSERT1>. I have
enclosed an introductory brochure which I hope will be of
interest.

If you and your colleagues at <NAME> would like to see a
demonstration of <INSERT1> please do not hesitate to contact
me.

Yours sincerely



E ARKWRIGHT
```

At run time the contents of the fields defined are entered into the document in insert mode obeying the margins in force.

Note that the output format record is inserted above the text in this document which, in this case would contain the address and salutation. . If however a system is to be set up which uses a mailshot document with its inserts but does not require any field details to be inserted at the top, then a dummy output format record may be used. Such a record simply consists of the top line of the record.

13.5 OUTPUT FORMAT EXTENSION OPTIONS

The formatting capabilities may be extended by using the options described in this section. These options are specified at the end of the header line of the output format and introduced by a /x:.

```
[*OUTQ1    *DICT V2123 .  ?  /x:aaaaa
```

where x defines the option and aaaaa additional information necessary for that option.

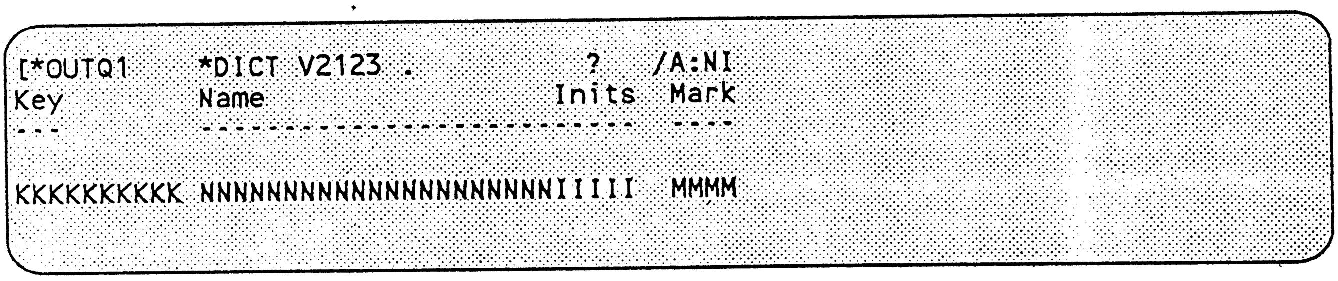
Any number of options may be used but each must be preceded by its own /.

13.5.1 Sorting the output

The sorting facility in the LEX report generator allows the selected records to be sorted into an appropriate order before displaying. Without a sort they appear in lexicological order of keys.

Two sorts are available. /A implies an ascending order whilst /D implies descending order. In either case the order is a strict alphabetical sequence (i.e. no numeric or date capability which means that 345 comes before 4.)

The sort can be on any group of fields defined in the associated dictionary by appending the field formatting character to the /A:



```
[*OUTQ1    *DICT V2123 .        ?  /A:NI
Key        Name             Inits  Mark
---        -----------------------  ----

KKKKKKKKKK NNNNNNNNNNNNNNNNNNNNIIIII  MMMM
```

In the above example the output is sorted into surname/initials order. Replacing the sort instruction with /A:M would print the students in order of merit.

Note that sorting may take up to three times as long as a conventional list process run depending on the number of records selected and the length of the fields to be sorted.

13.5.2 Help message

At Select time a help message is displayed automatically if the facility is set in **INST. Alternatively it may be obtained by entering HELP at the select prompt. In either case the default message shown is contained in the system record *HELPS.

Another system record can be nominated to supply the help message by putting `/H:key` in the top line of the output format. In this way messages appropriate to a customised process may be nominated. In the following example the help message is contained in the record `*HELPX`.

```
[*OUTQ1 *DICT V2123 .  ? /H:*HELPX
```

13.5.3 Paint character

The report generator assumes a paint character of SPACE. To use another character `/P:p` is included in the top line of the format where `p` is the new paint character to be applied.

13.5.4 Line spacing

When output from the report generator is printed it uses single spacing. This option allows other spacing to be defined in exactly the same way as .S is used in a document. For example two and a half line spacing is used in the following example.

```
[*OUTQ1 *DICT V2123 .  ? /S:2.5
```

13.5.5 Column totalling

This option applies to the list processor only.

The totalling facility in the LEX list processor allows designated fields to be totalled by column. The formatting characters of the fields to be totalled are defined following /T:

```
[*OUTS3     *DICT2 P6066 ,05F     ?      /T:123
DEPT/INDIV        HOTEL                 TRAVEL          MISC

LLL/MM          111111111           222222222       333333333
```

At the end of the report a form feed is thrown and the final totals printed as below:

```
Totals are...
1 HOTEL         220.00
2 TRAVEL        289.00
3 MISC               0
```

The literal `Totals are...` is as specified in `**TEXTP`. Each total is on a separate line preceded by the single letter formatting character and the field name as defined in the appropriate dictionary.

It has already been mentioned that a form feed is thrown before the totals are printed. To suppress the form feed so that the totals follow on directly from the detail a ; may be used in place of the : in the top line, i.e. in the above example:

```
[*OUTS3    *DICT2 P6066 ,05     ?     /T;123
```

13.5.6 Accessing data records in foreign files

It is possible to direct the LEX report generator to search through records held in a different VMF from the current one. The other VMF is referred to as a foreign file and access is achieved by a /F. It is used in a similar manner to the escape sequence (ESC)SF and takes the general form:

```
/F:abcd@filename@
```

where:

a	represents the record type for the group of records on which the search is to be performed.
b	represents the file type, usually V to indicate a VMF file.
c	is set to R for read only access (this is the default if not specifically entered)
d	to define multi user (M) or single user (N) mode. All other values open the foreign file in the same mode as the main VMF.

Consider the following example.

```
[*OUTA3    *DICT   V2123    ?     /F:=VRMSTOCK.FIL
```

Records of type = are to be selected from the VMF file `STOCK.FIL` which is to be opened as a multi user file. Records cannot be updated as read only access is defined.

The file remains open after the list processing run is complete. A specific file close (ESC)UF must be executed if the foreign file is no longer required.

Note that the filename may include account/device details.

This facility may be used to store records for customer names and addresses or a library of standard paragraphs on a central file where they can be accessible to users from other VMF files.

13.5.7 Updating records during list processing

A record that has been selected during a list processing run can be flagged with a character string or the date. This is achieved by making use of of an option of the form:

```
/U:nnnn:@string@:
```

Each record selected has characters from the nnnnth position overwritten by the characters contained in string. If the final : is not found then only one character is updated - the first character in the string.

For example, the record

```
[*OUTQ1 *DICT V2123 .  ?  /U:200:Q
```

causes the 200th character of all records output in a list processing run to be overwritten with the letter Q.

Note that /U:200: at the end of a line will not apply space as the updating character but the implicit RETURN. As a consequence the next line is lost. This problem can be overcome by using /U:200: :

The date, in the form YYMMDD, can be added to selected records in the six characters from the nnnnth position. To achieve this the first character in the string is set to #. e.g. format becomes:

```
[*OUTL1     *DICT P6066 .  ?  /U:633:#
```

13.5.8 Print time instructions

During the printing of an ordinary document instructions can be performed at print time by associating special characters with these functions. Such visible equivalents are defined in **INST or .C lines in the text of ordinary documents.

The same effect can be achieved in list processing by use of the /C option as shown in the example.

```
[*OUTQ1     *DICT V2123 .          ?     /C:.~.."!|...
```

The characters used as standard and their position after the /C: is exactly as in .C lines. See Appendix F for a full description but for reference; a summary is given here.

$	Escape
~	Suppress line feed on carriage return.
^	Control
#	Pound
_	Special underline
@	Space
!	Paragraph numbering
|	Indexing
=	Hard hyphen
{	Special Character 1
}	Special Character 2
%	Footnotes

Chapter 14. DOCUMENT INDEX

Chapter 19 of the User Manual describes the document indexing feature of LEX. This consists of a list of documents which have been created using LEX. For each document the index contains information such as the document name, title, date created, document size, etc.

Document indexing is an optional feature of LEX and is enabled in the **INST installation record.

This chapter describes how LEX stores the document index information and the system records which allow the document index to be selectively viewed or printed.

14.1 STORING DOCUMENT INDEX INFORMATION

If document indexing is enabled then each time a document is accessed or created LEX amends or creates a document index entry for that document on the VMF. The document index entries are stored on the VMF as records with a key in the following format:

```
:NNNNNNEEE
```

:	is the record type
NNNNNN	is the first 6 characters of the document name
EEE	is the document extension

The data stored in each record has the following format:

Chars	Length	Contents
0-9	10	The key :NNNNNNEEE
10	1	Unused
11-50	40	Full document name including any device or account specification
51-90	40	Document title
91-130	40	Document detail
131-137	7	Date of creation of document in form DDMMMYY
138-144	7	Date on which document was last accessed for any purpose (edit, copy, view or print), date is in DDMMMYY form
145-146	2	Number of times the document has been accessed
147-153	7	Date on which document was last accessed for editing, date is in DDMMMYY form

Chars	Length	Contents
154-155	2	Number of times the document has been edited
156-162	7	Date on which document was last accessed for printing, date is in DDMMMYY form
163-164	2	Number of times document has been printed
165-170	6	Time document was last accessed in HHMMSS form
171-176	6	Total time spent accessing document in HHMMSS form
177-182	6	Number of lines in document excluding non-printing lines, e.g. rulers and commands such as .P, .C etc
183-190	8	Number of words in document excluding rulers and commands
191-198	8	Number of characters in document excluding rulers and commands
199-204	6	Document creation date in form YYMMDD
205-210	6	Document last access date in form YYMMDD
211-216	6	Document last edit date in form YYMMDD
217-222	6	Document last print date in form YYMMDD

Document index entries are written if the document titling entry in the **INST installation record has been set to Y.

No entries will be made in the line, word and character count fields unless the word and character count field in **INST has been set to Y.

No document detail will be prompted for unless the document detail field in **INST has been set to Y.

14.2 DOCUMENT INDEX DICTIONARY

The dictionary used to produce list processing reports from the document index data is *DTDICT:

```
[*DTDICT
N NAME      40   I TITLE    40   B DETAIL   40
C CREATED   7
A ACCESS    7    1 ACVER    2
E EDITED    7    0 EDVER    2
P PRINTED   7    7 PRVER    2
2 LASTIME   6    3 TOTTIME  6
4 LINES     6    5 WORDS    8    6 CHARS     8
c DATECR    6    a DATEAC   6    e DATEED    6  p DATEPR   6
X XTENSION KEY+6/3   R TYPE  KEY+0/1
```

14.3 DOCUMENT INDEX REPORTS

The standard reports which may be produced from the document index may be selected from the Document Processing Menu. The section of this menu below shows which records are used to produce the reports:

```
X*DTIR      Q - Quick document index
S*VDTI      I - View document index (full details)
S*PDTI      X - Print document index (full details)
```

Chapter 19 of the User Manual describes each of these options.

Further reports may be created as described in Chapter 13 of this manual. As an example the following report is produced in date sequence and shows document names, titles and last access dates.

```
                      DOCUMENTS SORTED BY ACCESS DATE

         Document      Title                            Last access date

         JEI   .LTR    Electricity Board letter         12-Dec-85

         T909  .MAN    Chapter 9, Technical Manual      15-Dec-85

         T914  .MAN    Chapter 14, Technical Manual     08-Jan-86

         INCOME.DOC    Financial figures for 1984/5     14-Jan-86
```

The record to produce this report is named *DTLIST and is added to the document processing menu as follows:

```
X*DTIR      Q - Quick document index
S*VDTI      I - View document index (full details)
S*PDTI      X - Print document index (full details)
S*DTLIST    O - Print document list in access date sequence

            . - Select option
```

The contents of *DTLIST are:

```
[*dtlist     *DTDICT V2323 :   ?  /H*HELPD    /A:a
                  DOCUMENTS SORTED BY ACCESS DATE
ocument      Title                                        Last access date


KKKKKK.KKK   IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII   AA-AAA-AA
```

Note that the sort sequence specified after the /A defines that sorting is to be carried out on the last access date stored in YYMMDD format. The last access date is displayed in DD-MMM-YY format.

[redacted]

SUMMARY

KEY

M	Mandatory System Record	F	Filename
O	Optional System Record	F1	Input Filename
X	Other Prompt	F2	Output Filename

Document Processing Options

Menu Option	Description	Parameters (see key)	Section
/C	Create a document	O F	*A.3*
/E	Edit a document	O F	*A.5*
/I	Edit a document; display first line only	O F	*A.9*
/V	View a document	O F	*A.21*
/K	Delete a document	F	*A.11*
/R	Rename a document	F1 F2	*A.17*

Print Options

Menu Option	Description	Parameters (see key)	Section
/P	Print a document	X	*A.15*
/Q	Print a document; multiple copies	X	*A.16*

Copy Files Options

Menu Option	Description	Parameters (see key)	Section
/A	Copy and edit a document	F1 F2	*A.1*
/J	File copy by block	F1 F2	*A.10*

VMF Maintenance Options

Menu Option	Description	Parameters (see key)	Section
/F	Switch to another VMF	F	*A.6*
/M	Switch to another menu	M	*A.13*
/U	Switch to another VDU type	M	*A.20*

Menu Option	Description	Parameters (see **key**)	Section
/Z	Switch to another work file	F	*A.24*
/1	Switch system control record	M	*A.26*
/2	Initialise work file		*A.27*
/T	VMF Statistics		*A.19*
/W	LEX Status		*A.22*
/X	Enter playground	O	*A.23*
/0	Unset VMF file locks	F	*A.25*
/3	Reorganise VMF	X	*A.28*
/4	Copy records between VMFs	X	*A.29*

Chain/Fork Options

Menu Option	Description	Parameters (see **key**)	Section
/G	Chain to external program	F	*A.7*
/B	Fork to external program and return	M F1	*A.2*
/D	Fork to external program and return	M F2	*A.4*
/H	Fork to external program and return	M F1 F2	*A.8*
/N	Fork to external program and return	M	*A.14*

List Processing Options

Menu Option	Description	Parameters (see **key**)	Section
/L	List processing with mail merge	M F1	*A.12*
/S	List processing with reports	M X	*A.18*

Miscellaneous Options

Menu Option	Description	Parameters (see **key**)	Section
/5	Patch routine	X	*A.30*
/6	Convert ASCII formats	M F1 F2	*A.31*

A.1 OPTION /A — COPY AND EDIT A DOCUMENT

Purpose:

This option copies an existing document to a new one and then opens the new document ready for editing.

Method:

It prompts for

```
Input
Document Name: ________________
```

followed by the following prompts and messages if enabled in **INST.

```
Created Edited vsn Access vsn Print  vsn LstTim TotTim Lines  Words   Chars
12Oct85 12Oct85 05 12Oct85 05 12Oct85 02 110119 012025 000560 0002477 002333

           Document Title:  ______________________________________
           Document Detail: ______________________________________
```

The document title and data are to help in identifying documents and are held in records on the VMF as an index of existing documents.

This is followed by a similar sequence of prompts for the "Output Document Name". Finally LEX enters the newly created document with the first screenful ready for editing.

Notes

When a document is copied in this way the data in the LEX index containing date created, time, number of characters, etc is preserved. The access date and version number are updated.

Note that only documents can be copied using this option. See menu switch option /J for copying VMF and other non ASCII files.

The Customised Menu Version

When customised into a menu it is possible to define a system record to be activated on opening the file. LEX inserts a blank line and executes the record. See Chapter 10.

A.2 OPTION /B FORK TO EXTERNAL PROGRAM & RETURN

Purpose:

On systems where fork options are not available, this option causes LEX to be dumped as a memory image onto disk and the relevant program loaded into memory for execution. Where a true fork is available this is used. On termination of the external program, LEX continues after `Press any key to continue.`

Method:

The option may only be used by programs written in Filetab-D or with programs conforming to a particular format. Its primary purpose is for programs such as SPELL and ADD when supplied as fully integrated programs for use with LEX. Programs not conforming to this format can only be executed using the chain menu switch option /G.

Data passed to the called program under this option is transferred through memory.

The mandatory system record required by /B contains in the first 32 characters the full filename of the program to be called. The remainder may contain other data. LEX prompts for `Output Document Name` the reply to which is also transferred. The `Input Document Name` is passed as spaces.

Notes

Note the three related menu switch options /D /H and /N.

A.3 OPTION /C CREATE A DOCUMENT

Purpose:

This option is used to create a new document file.

Method:

LEX first issues the prompt:

```
Document Name: ________________
```

followed by the following prompts and messages if enabled in `**INST.`

```
Created Edited vsn Access vsn Print  vsn LstTim TotTim Lines  Words   Chars
12Oct85

          Document Title:  ________________________________________
          Document Detail: ________________________________________
```

The document title and data are to help in identifying documents and are held in records on the VMF as an index of existing documents. As the document does not exist at this stage most of the data is blank.

LEX checks that the document name entered does not already exist. If it does exist an error message is displayed and LEX waits for another file name to be entered. If the file does not exist then LEX creates the file entry on the directory, opens the file and displays the first ruler from the document details associated with the document's extension. The cursor is positioned on the first free line with the remainder of the screen blank.

The Customised Menu Version

When customised into a menu it is possible to define a system record to be activated on opening the file. LEX inserts a blank line and executes the record. See Chapter 10.

A.4 OPTION /D FORK TO EXTERNAL PROGRAM & RETURN

Purpose:

This option causes LEX to be dumped as a memory image onto disk and the relevant program loaded into memory for execution. On termination of this program, LEX continues after a prompt `Press any key to continue.`

Method:

It is similar to menu switch options /B /H and /N the workings of which are described under /B.

The mandatory system record required by /D contains in the first 32 characters the full filename of the program to be called. The remainder may contain other data. LEX prompts for `Input Document Name` the reply to which is also transferred. The `Output Document Name` field is passed as spaces.

A.5 OPTION /E — EDIT A DOCUMENT

Purpose:

This option opens a file ready for editing provided it already exists on the system.

Method:

It prompts for:

```
Document Name: ______________________
```

followed by the following prompts and messages if enabled in **INST.

```
Created Edited vsn Access vsn Print  vsn LstTim TotTim Lines  Words   Chars
12Oct85 12Oct85 05 12Oct85 05 12Oct85 02 110119 012025 000560 0002477 002333

         Document Title:  _____________________________________________
         Document Detail: _____________________________________________
```

The document title and data are to help in identifying documents and are held in records on the VMF as an index of existing documents.

LEX checks that the requested document exists. If it does not an error message is displayed and LEX awaits a new filename. If the file is found LEX opens the document and displays the first screenful of text, leaving the cursor at the top left hand corner of the screen.

Special Features

Note the similar option /I which only displays the first line of text.

The Customised Menu Version

When customised into a menu a system record can be defined. In this case a blank line is inserted at the top of the document and the record activated. See Chapter 10.

A.6 OPTION /F — SWITCH TO ANOTHER VMF

Purpose:

This option is used to switch to a different VMF.

Method:

The prompt

```
VMF: ________________
```

appears and the new VMF is specified in the following 32 characters. Device and account details may be used in the specification and if the document extension is omitted .VMF is assumed. If no record is specified LEX will look for the default VMF file as patched into the system. On failing to find an appropriate file LEX prompts for another entry with

```
.vmf?
```

The relevant system control records are only reset when loading a VMF. When changes are made to **TEXT for example it is necessary to re-open the current VMF with the /F option for the changes to be effective. Changes to the system control records **MONT, **INST and **PNST can be applied in this way but may also be activated using the /1 menu switch option.

Typeahead can be used at the VMF prompts to set the *VDUFORM or use any other menu switch option. For example:

```
VMF: XYZ.VMF /UVT100 //ETEST.LTR \
```

Note that the termination of each string is a space and that ordinary characters should be entered in // as in //ABCDEF//.

The Customised Menu Version

The option may be customised in a menu as described in Chapter 10 but the file specification is then limited to 10 characters. Alternatively it may be invoked from a system record in which case the full 32 characters may be utilised. Programming is described in Chapter 11.

A.7 OPTION /G CHAIN TO EXTERNAL PROGRAM

Purpose:

This option causes LEX to chain to the program specified by the file specification following the menu option. This only applies to those operating systems which will support a chaining action.

Method:

If activated from the keyboard with the /G switch the prompt

`Key ________________`

appears allowing 32 characters for the file specification including any device/account details and the file extension.

If the file extension is D11 LEX assumes a Filetab D program whereas any other extension is taken to be a compiled program executable on the operating system in use. When no extension is given LEX attempts the following in order:

a) Adds extension D11 and if such a program exists runs it as a Filetab D program.

b) Adds the operating system default extension for a compiled program and if such a file is found runs it.

c) Exits to the operating system with the **TEXT message

`Can't find File`

On leaving LEX to chain to the external program, Core Common (where relevant) is set.

LEX does not automatically return on completion of the external program. It is however possible to chain back to the top menu of LEX or to the place where the option was called.

The Customised Menu Version

If activated from a customised menu as described in Chapter 10 then the title specification is limited to 10 characters by the menu record format. The command can, however, be programmed into a system record and when activated in this way the full 32 characters may be utilised. e.g. `$E/GDP1:(2,5)SPECIAL.EXE\.`

A.8 OPTION /H FORK TO EXTERNAL PROGRAM & RETURN

Purpose:

This option causes LEX to be dumped as a memory image onto disk and the relevent program loaded into memory for execution. On termination of this program, LEX is automatically reactivated at the point where the other program was called via a prompt `Press any key to continue.`

Method:

It is similar to menu switch options /B /D and /N the workings of which are described under /B.

The mandatory system record required by /H contains in the first 32 characters the full filename of the program to be called. The remainder may contain other data. Also LEX prompts for Input Document Name the reply to which is transferred. Finally the reply to the Output Document Name prompt is also passed.

A.9 OPTION /I — EDIT A DOCUMENT

Purpose:

Menu switch option /I is similar to option /E except that on opening a document for editing, only the first line of the document is displayed rather than the whole first screenful. This option is usually used when editing files using a slow line such as a dial-up modem.

Method:

Prompts and limitations are identical to those for the /E option. Whilst only the first line is shown the rest of the page is available and can be displayed by scrolling or refreshing the screen with $\boxed{\text{ESC}}$QR.

The Customised Menu Version

As with option /E a system record can be customised into a menu as described in Chapter 10. A blank line is inserted at the top of the document and the record activated.

A.10 OPTION /J — FILE COPY BLOCK BY BLOCK

Purpose:

Menu switch option /J initiates a block mode copy of one file to another.

Method:

The prompts issued are as for option /A but the input file is read and copied block by block rather than line by line. As a result this is a quicker method of copying files and can be used for non-ASCII files, such as VMFs. Use of this option to copy documents or other files not occupying a whole number of 512 byte blocks on systems where an exact byte count is held for a record size may lead to corruption occuring in the copied document.

Notes

The internal LEX index is updated appropriately.

The copy menu switch option /A is designed to leave the user at the top of the new document ready to edit it. The option /J on the other hand simply performs the copy and returns LEX to the menu.

A.11 OPTION /K DELETE A DOCUMENT

Purpose:

This option deletes a document or file.

Method:

When this option is taken, LEX prompts for:

```
Document Name: ________________
```

followed by the following prompts and messages if enabled in `**INST`.

```
Created Edited vsn Access vsn Print  vsn LstTim TotTim Lines  Words   Chars
12Oct85 12Oct85 05 12Oct85 05 12Oct85 02 110119 012025 000560 0002477 002333

            Document Title:  ________________________________
            Document Detail: ________________________________
```

The document title and data are to help in identifying documents and are held in records on the VMF as an index of existing documents.

A message appears to the effect that the document has been deleted and LEX waits at the `Document Name` prompt ready for another filename to be entered.

If the file is not found, LEX sends a BELL to the VDU and re-prompts for another document filename.

A.12 OPTION /L LIST PROCESSING WITH MAIL MERGE

Purpose:

This option enters the mail merge option of LEX. It produces multiple copies of an existing document, each version having information from selected list processing records merged into it.

Method:

A system record is mandatory to define the format of the information to be merged into the head of the document from the selected list processing records. Therefore if selected from a menu LEX prompts for the system record with:

```
KEY *________________
```

On receiving a legal system record LEX then prompts for:

```
Document Name: ________________
```

followed by the following prompts and messages if enabled in **INST.

```
Created Edited vsn Access vsn Print  vsn LstTim TotTim Lines  Words   Chars
12Oct85 12Oct85 05 12Oct85 05 12Oct85 02 110119 012025 000560 0002477 002333

        Document Title:  ______________________________
        Document Detail: ______________________________
```

The document title and data are to help in identifying documents and are held in records on the VMF as an index of existing documents.

The document name is the name of the document to be printed headed by the details contained in the system record. The document may also contain references to fields in the list processing records as described in Chapter 13.

The Customised Menu Version

If customised into a menu as described in Chapter 10 the system record key is picked up from the menu and the user is prompted for the Document Name only.

A.13 OPTION /M SWITCH TO ANOTHER MENU

Purpose:

This option is to initiate another menu.

Method:

A system record key defining the new menu record is mandatory and it must have the correct format for a menu (see Chapter 10) or the results are unpredictable.

Notes

Menus can be nested to a maximum depth of six levels but returning to the top menu, usually **MENU, resets the stack.

To exit from LEX it is necessary to move back through all levels of nesting unless an exit is made using the /B /D /G /H or /N menu switch options.

A.14 OPTION /N FORK TO EXTERNAL PROGRAM & RETURN

Purpose:

This option causes a true fork where the external program is loaded into memory for execution. On termination of this program, LEX continues after a prompt `Press any key to continue.` Note that it is only available on operating systems which support this feature.

Method:

It is similar to menu switch options /B /D and /H the workings of which are described under /B.

The mandatory system record required by /N contains in the first 32 characters the full filename of the program to be called.

A.15 OPTION /P PRINT A DOCUMENT

Purpose:

This option causes LEX to print a document, any LEX print time commands are interpreted and sent to the printer as the appropriate commands. A full description of the print time commands is given in Chapter 9.

Method:

A number of prompts are issued under the print option and they are described in the following steps.

Step 1

```
Document Name: ______________________________
```

followed by the following prompts and messages if enabled in **INST.

```
Created Edited vsn Access vsn Print  vsn LstTim TotTim Lines  Words   Chars
12Oct85 12Oct85 05 12Oct85 05 12Oct85 02 110119 012025 000560 0002477 002333

        Document Title: ______________________________
        Document Detail: _____________________________
```

The document title and data are to help in identifying documents and are held in records on the VMF as an index of existing documents.

Step 2

```
Which Printer (1) Slow (2) LP: (3) Spool to a file
```

The above message is taken from **PNST and prompts the user to define where the output is to be directed. Up to nine output channels are possible and associated with each there may be a system record to describe any special print time effects which may be required. See Chapter 5 for **PNST where these details are set.

If the channel selected is nominated in **PNST as a spool file, 3 in this case, then LEX issues an extra prompt.

```
Spool File Name _______________
```

The user may enter the name of a disk file, complete with account/device details if appropriate, or a valid output device name.

If the device specified either implicitly by the number prompt or directly at the Spool File Name, is already in use, LEX replies with `Device not available` and awaits input of another channel number.

Step 3

```
Pause at head of form  (Y/N)
```

This is an optional prompt enabled in **PNST, and requires a reply of Y or N . A Y causes LEX to pause before every page of printed output with the message

`Line up paper, then press any key to continue`

Step 4

Next LEX displays information on petal keys and petal functions, if appropriate, according to details set up in **PNST. No response is required.

Step 5

`Number from ____   Print from ____   End print ____`

These prompts are only issued if the document includes a page header or footer i.e. .T or .F line.

`Number from` refers to the page number associated with the first page of the document. The default is 0 or 1 as set in **PNST but may be overridden in the document itself by use of an * on the .P line.

`Print from` defines the first page to be printed thereby allowing pages at the begining of a document to be missed. At the `Print from` prompt the page number may be followed by a (LINEFEED) instead of a (RETURN). Such a response causes only the nominated page to be printed after which LEX returns to this prompt for another page number to be entered.

`End print` defines the last page number to be printed.

Step 6

`Print even or odd numbered pages (O/E)`

This prompt only appears if alternate headers and footers have been specified in a document. It allows LEX to print odd pages, even pages or both. A reply of O prints odd numbered pages, E prints even numbered pages whilst any other response prints all pages. This option allows a document to be printed on both sides of the paper.

A.16 OPTION /Q PRINT A DOCUMENT, MULTIPLE COPIES

Purpose:

The /Q option is similar to /P except that it allows the printing of multiple copies of a document to be printed.

Method:

After the step 3 prompt `Pause at head of form.` LEX asks

`Number of Copies ____`

The remaining prompts are as before.

Notes

Each copy is printed in full before the next copy is started. Any questions that normally follow the prompt for pausing at head of form are repeated before each copy. It is therefore possible to specify two copies and to print odd numbered pages during the first run and even numbered pages during the second.

A.17 OPTION /R — RENAME A DOCUMENT

Purpose:

This option allows a document to be renamed with with the associated index record being updated accordingly.

Method:

LEX prompts for

```
Old
Document Name: ____________________
```

followed by the following prompts and messages if enabled in **INST.

```
Created Edited vsn Access vsn Print  vsn LstTim TotTim Lines  Words   Chars
12Oct85 12Oct85 05 12Oct85 05 12Oct85 02 110119 012025 000560 0002477 002333

        Document Title:  ______________________________________
        Document Detail: ______________________________________
```

The document title and data are to help in identifying documents and are held in records on the VMF as an index of existing documents.

This is followed by a similar set of prompts for

```
New
Document Name: ____________________
```

with the attendant document title and document data if enabled in **INST. All normal defaults apply to the document names but the new document name must contain the same account and device specifications if the function is to be successful.

A.18 OPTION /S LIST PROCESSING WITH REPORTS

Purpose:

This initiates the LEX list processing option to generate reports from list processing records on the VMF.

Method:

The prompts made by LEX depend on the type of record being used and are discussed in Chapter 14. Briefly, LEX requires the specification of some selection criteria to define which records are to be output and then a mandatory system record defines the output format and the dictionary to use. If the output is to a print device then prompts for printing selections are displayed similar to those required by the /P option.

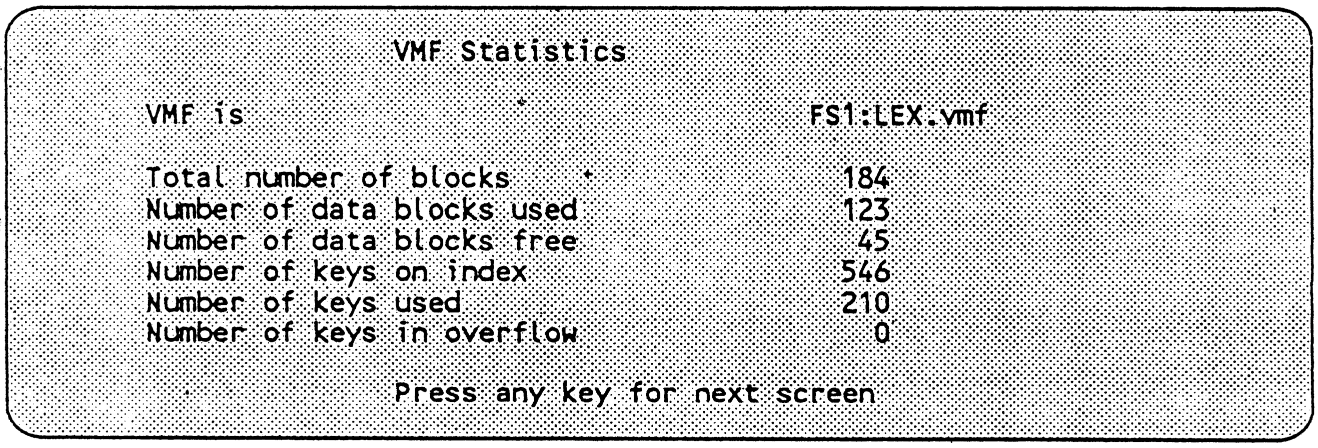
A.19 OPTION /T MEMORY FILE VMF STATISTICS

Purpose:

The selection of this option causes LEX to display the status of the current VMF.

Method:

On issuing the command LEX blanks the screen and displays the following:



```
                 VMF Statistics

VMF is                                   FS1:LEX.vmf

Total number of blocks                      184
Number of data blocks used                  123
Number of data blocks free                   45
Number of keys on index                     546
Number of keys used                         210
Number of keys in overflow                    0

              Press any key for next screen
```

All descriptive parts of the display are taken from **TEXT. The significance of the statistics is outlined is the section in Chapter 2 describing the structure of the VMF file.

The display remains until any key is depressed when LEX returns to the menu from which it was called.

A.20 OPTION /U SWITCH TO ANOTHER VDU TYPE

Purpose:

LEX operates on many different types of VDU all of which have their own commands and characteristics. A set of system records must be defined, as described in Chapter 8, to interface LEX with the appropriate VDU. This option allows the records for a different VDU to be activated.

The option can also be used to change a VDU's mode of working from, say 80 column wide to 132 columns, when the VDU has such a facility.

Method:

If the option is customised into a menu as described in Chapter 10, the record type `)` must be included in the system record name. eg. `)VT100`. If it is called using /U the record type is assumed and the user enters the rest of the key. eg `VT100`.

LEX resets the menu stack to start at the first menu as defined in the system record containing the VDU's details.

A.21 OPTION /V VIEW A DOCUMENT

Purpose:

This option is used to view a document file.

Method:

LEX first prompts

```
Document Name: ____________________
```

followed by the following prompts and messages if enabled in `**INST`.

```
Created Edited vsn Access vsn Print  vsn LstTim TotTim Lines  Words   Chars
12Oct85 12Oct85 05 12Oct85 05 12Oct85 02 110119 012025 000560 0002477 002333

         Document Title:  ____________________
         Document Detail: ____________________
```

The document title and document detail are to help in identifying documents and are held in records on the VMF as an index of existing documents.

Notes

A document may be viewed safely without the possibility of text being altered. Editing may be carried out but any changes are not stored. i.e. an (ESC)E to leave the document acts as an (ESC)Z.

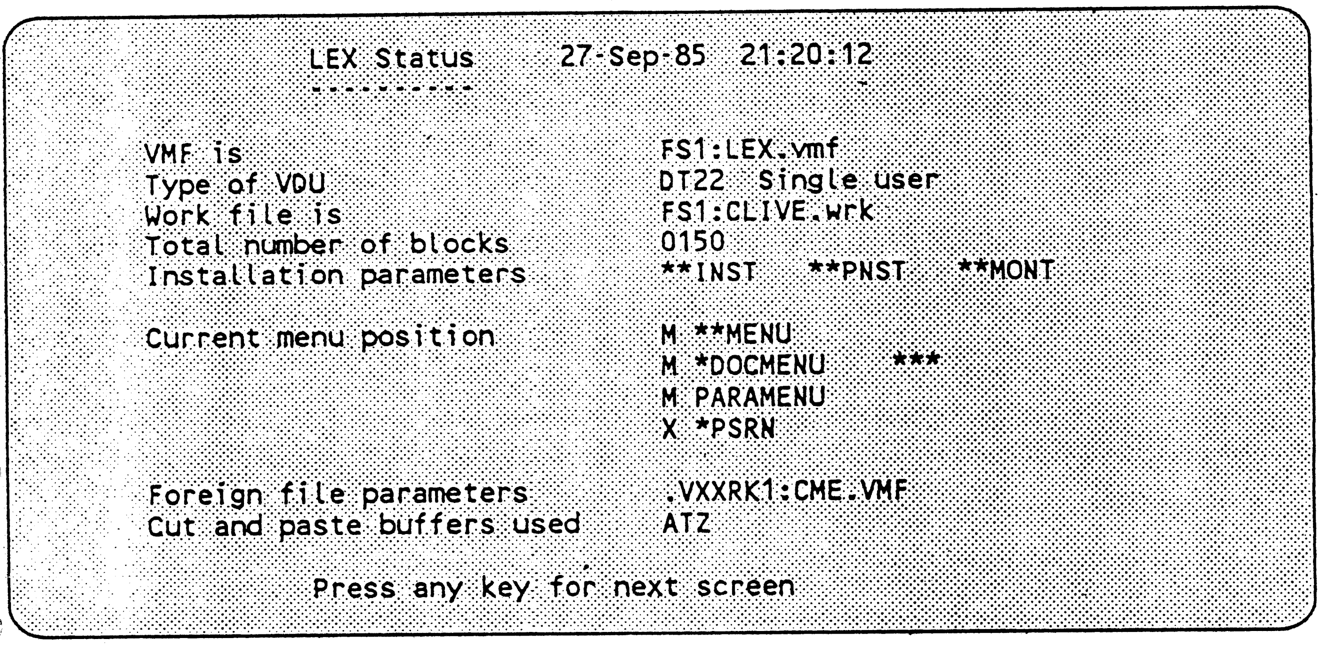
A.22 OPTION /W LEX STATUS

Purpose:

This menu switch option is used to show the various files in use and the state of the menu stack. It also shows which buffers have information in them.

Method:

The command results in a display similar to that shown below.



```
              LEX Status     27-Sep-85  21:20:12
              ----------

VMF is                               FS1:LEX.vmf
Type of VDU                          DT22  Single user
Work file is                         FS1:CLIVE.wrk
Total number of blocks               0150
Installation parameters              **INST   **PNST   **MONT

Current menu position                M **MENU
                                     M *DOCMENU    ***
                                     M PARAMENU
                                     X *PSRN

Foreign file parameters              .VXXRK1:CME.VMF
Cut and paste buffers used           ATZ

              Press any key for next screen
```

where the descriptions are taken from **TEXT2 and the content is as described below. The display remains until any key is depressed when LEX returns to the menu from which it was called.

`VMF is`

The name of the current VMF file, followed by an indication of whether the VMF is open for single or multi-user access.

`VDU type`

The name of the current VDUFORM key describing the VDU.

`Work file is`

The name of the current work file.

`Total number of blocks`

Total number of 512 byte data blocks used by the current VMF.

`Installation parameters`

The names of the system control records currently active which describe the installation.

`Current menu position`

The state of the menu stack where the *** marks the current position. Note that M refers to a menu record and X to a record activated from the playground. In this instance the user has returned down the stack using (ESC)E to the current position.

`Foreign file parameters`

When a foreign file is open full details are shown here in the same format as in the escape sequence to open the foreign file. See (ESC)SF for more details.

`Cut and paste buffers used`

A list of all cut and paste buffers which still contain information is shown here. A refers to the main buffer and the others can be T U V W X Y and Z.

`No write access to VMF`

Whilst not shown in the above example, the no write access message will also be shown if applicable to the main VMF or the foreign file.

A.23 OPTION /X ENTER PLAYGROUND

Purpose:

This option causes LEX to enter the playground. Default rulers, keystores and printing details are taken by assuming a document extension of .080, .132, etc. where these numbers are picked up from the width field in the appropriate *VDUFORM.

Method:

On issuing the command the screen is blanked with the cursor at the top left hand corner.

The Customised Menu Version

When customised from a menu an optional system record may be defined which will be activated automatically when the screen has been blanked. This operation is used for a large variety of tasks within the standard issue VMF and is most commonly used to enter forms mode. Chapter 10 gives more details of customising menus and Chapter 12 describes forms.

A.24 OPTION /Z SWITCH TO ANOTHER WORK FILE

Purpose:

The menu switch option /Z is used to close the current workfile and open another. It is the equivalent of leaving LEX and entering LEX again with the new workfile.

Method:

On issuing the command a prompt appears calling for the filename. LEX then performs the normal startup sequences as described in Chapter 2.

The name of a VDU can be input at the same prompt by separating the work filename from the VDU name by a space.

A.25 OPTION /0 UNSET VMF FILE LOCKS

Purpose:

In a multi-user system LEX ensures that only one user at a time has write access to individual records. This is achieved by "locking" records which are opened with a view to updating them. If a system crash occurs then the locks may be left set. This menu switch option clears all existing locks in a named VMF.

Method:

On issuing the prompt LEX prompts the **TEXT2 literal:

```
Clear Locks.  File name: ___________
```

Whilst clearing any locks a further **TEXT2 message is displayed

```
Locks now being cleared. Please wait ..
```

Finally control returned to the calling menu.

A.26 OPTION /1 SWITCH SYSTEM CONTROL RECORDS

Purpose:

LEX requires certain installation details when entering a VMF and it takes this information from the records **MONT **INST and **PNST. It may be required to switch to system control records with some other name and this is achieved with the /1 option. Note that **MONT type system control records must start with **M, **INST type with **I and **PNST type with **P.

Method:

LEX prompts for the name of a system control record. Enter its name and (RETURN).

A.27 OPTION /2 INITIALISE WORK FILE

Purpose:

This option clears the work file of all its buffers, resets various defaults, deletes the menu stack and sets the menu back to the VDU default. The buffers and modes that are initialised are as follows.

a) Unsets foreign files.
b) Unsets shutup mode
c) Removes all cut and paste buffers.
d) Clears the calculator stores.
e) Clears the keystores.
f) Sets the rulers 6-9 back to their defaults.
g) Clears the default document name.

A.28 OPTION /3 REORGANISE VMF

Purpose:

The need for reorganising the VMF is outlined in Chapter 2 and this option performs the operation.

Method:

When activated a number of prompts and messages occur as described in the following steps. Except where stated the wording of the prompts is taken from **TEXT2. See Chapter 6.

Step 1

```
VMF reorganise.  Statistics of current VMF :- FS1:LEX.vmf
```

This message confirms that the reorganisation option has been selected and displays the name of the current VMF. If account/device details were entered when the VMF was selected then these also appear. If not the system device can be assumed. Note that .vmf is in lower case in this example. This has no significance; it is how it was originally entered.

No response is required to this message.

Step 2

```
Number of data blocks used 123 Number of keys used 211
Number of data blocks free 44 Number of keys on index 546
```

The significance of the statistics shown is described in Chapter 2.

No response is required to this message.

Step 3

```
New VMF name ____________________________
```

A filename is required into which the VMF is to be organised. Account/device details may be included, the default being the system device. If an extension is not shown then .VMF is appended.

An (ESC)E at this prompt abandons the reorganise and returns LEX to the calling menu.

Step 4

```
Automatic reorganise (Y/N)
```

A N to this prompt instigates a manual reorganisation. The following steps then control the size of the VMF to be created and allow all or some of the records in the current VMF to be copied over.

A Y creates a new VMF the same size as the current one and copies all records into it. Steps 5 to 9 for manual reorganisation still appear on the screen but only as messages with the appropriate response automatically added.

An (ESC)E at this prompt abandons the reorganise and returns LEX to the calling menu.

Step 5

```
New number of data blocks (00168) _____
New number of keys on index (00422) _____
```

These prompts control the structure of the VMF to be created. The number in brackets shows the number of data blocks in the current VMF, and twice the current number of keys used. If, rather than a number, a RETURN is entered, then LEX defaults to the number of data blocks/keys shown in brackets.

An ESC E at this prompt returns LEX to the `New VMF name` prompt at Step 3.

No response is required to this message under automatic reorganisation; the figures in brackets are automatically entered.

Step 6

```
Now creating file. Blocks written..000123
```

Whilst LEX is creating a new VMF this message is displayed to show which block is being initialsied. If there is not enough disk space for the new VMF LEX hangs at this point until space becomes available or on a single user system the system may need to be re-started.

Step 7

```
Copy all records (Y/N)
```

A response of `Y` copies all records in the current VMF into the newly created VMF. LEX then bypasses the selection of individual records in steps 8 and 9. In an automatic reorganisation a `Y` is implicit.

`N` copies only system control records whose keys commence with ** or ") . Other records to be copied must be selected as shown in steps 8 and 9.

Step 8

```
All keys are 10 characters with  trailing spaces where applicable.
Type  the key then <CR>.  Wildcards may be used eg.  *??DICT  will
copy keys starting * and  ending  DICT  with any two characters in
between  (*12DICT, *DTDICT etc).
/+  after key copies all  records  with  keys  equal to or greater
    than that key.
/N  forces the key to be copied.
/Q  after a key queries each key (can be used with /+ ie /+/Q).
      Y will cause the record to be copied
      / allows you to rename the key as it is copied
      * returns to the Key prompt.
Any other response and the record is ignored.
```

A help message, an example of which is shown above, may be displayed automatically if the appropriate switch is set in **INST. The message is contained in the record *HELPV and may also be displayed by typing HELP in step 9.

No response is required to this message.

Step 9

```
Keys ______________
```

Typing HELP at this prompt displays the contents or the record *HELPV as described in step 8.

To copy a record its key must be entered followed by (RETURN). The character ? may be used as a wild card so that a group of similar records may be copied.

As each record is copied the message copied appears but if the record is already on the new VMF LEX queries the transfer with the message:

```
already on file ?
```

The following replies are possible.

Y	causes the record to be copied
/	allows a record to be copied with a different name. The / causes a prompt New key for the new record name.
(ESC)E	returns to the Key prompt.

Any other response and the record is ignored.

At the Key prompt a number of options may be appended to the key entered:

/+	copies all records with keys equal to or greater than the key entered.
//key2	copies all records in the range between the key entered and key2. It is inclusive; the keys entered are also copied.
/N	forces the key to be copied even if it is already on the new VMF.
/Q	queries each key in turn with the prompt already on file ? as previously described.

These options may be used in conjunction with each other. e.g. /+/N

The copy option is terminated by (ESC)E when LEX moves to step 10.

Step 10

`Reorganise finished. Rename and use new VMF (Y/N)`

A reply of N returns to the calling menu of the current VMF. The new VMF can then be entered by using the /F menu switch option.

If Y is entered the following actions occur:

a) LEX deletes the existing VMF backup if it exists.

b) LEX renames the current VMF with the VMF backup extension defined in **INST.

c) LEX renames the new VMF with the deleted VMF's name.

d) LEX enters the new VMFn with an implicit /F.

Note that renaming the VMF only occurs if both new and old VMFs are in the same account/device/library. If not LEX issues a message

`Rename illegal across accounts/devices`

leaves the old VMF in place and enters the new VMF.

A.29 OPTION /4 COPY VMF RECORDS

Purpose:

This option is used to copy records between the current VMF and another. Records can be copied either from or to the current VMF.

Method:

A series of prompts and messages are issued during the option as described in the following steps. The messages are contained in **TEXT2 and described in Chapter 6.

Step 1

```
VMF copy records. Current file is RK1:LEX.vmf
```

This message confirms that the record copy option has been selected and displays the name of the current VMF. If account/device details were entered when the VMF was selected then these also appear. If not the system device can be assumed. Note that .vmf is in lower case in this example. This has no significance; it is how it was originally entered.

No response is required to this message.

Step 2

```
Copy VMF name ________________________
```

A filename is required from or to which records are to be copied. Account/device details may be included, the default being the system device. If an extension is not shown then .VMF is appended.

An (ESC)E at this prompt abandons the copy and return LEX to the calling menu.

Step 3

```
Copy From current file (F) or To current file (T) (F/T)
```

T results in records being copied TO the current VMF and F copies them FROM the current VMF.

All keys are 10 characters with trailing spaces where applicable.

Type the key then (RETURN). Wildcards may be used e.g. *??DICT will copy keys starting * and ending DICT with any two characters in between (*12DICT, *DTDICT etc).

/+ after key copies all records with keys equal to or greater than

/+ after key copies all records with keys equal to or greater than that key.

/N forces the key to be copied.

Step 3

```
Copy From current file (F) or To current file (T) (F/T)
```

T results in records being copied TO the current VMF and F copies them FROM the current VMF.

Step 4

```
All keys are 10 characters with  trailing spaces where applicable.
Type  the key then <CR>.  Wildcards may be used eg.  *??DICT  will
copy keys starting * and  ending  DICT  with any two characters in
between  (*12DICT, *DTDICT etc).
/+  after key copies all  records  with  keys  equal to or greater
    than that key.
/N  forces the key to be copied.
/Q  after a key queries each key (can be used with /+ ie /+/Q).
      Y will cause the record to be copied
      / allows you to rename the key as it is copied
      * returns to the Key prompt.
Any other response and the record is ignored.
```

A help message, an example of which is shown above, may be displayed automatically if the appropriate switch is set in **INST. The message is contained in the record *HELPV and may also be displayed by typing HELP in step 5.

No response is required to this message.

Step 5

```
Keys ________________
```

Typing HELP at this prompt displays the contents or the record *HELPV as described in step 4.

To copy a record its key must be entered followed by (RETURN). The character ? may be used as a wild card so that a group of similar records may be copied.

As each record is copied the message copied appears but if the VMF file is nearly full a (BELL) will be sent to the VDU as a warning. If the VMF is full then the copy is barred and a warning message is displayed.

At the `Key` prompt a number of options may be appended to the key entered:

`/+`	copies all records with keys equal to or greater than the key entered.
`//key2`	copies all records in the range between the key entered and key2. It is inclusive; the keys entered are also copied.
`/N`	forces the key to be copied even if it is already on the new VMF.
`/Q`	queries each key in turn with the prompt `already on file ?` as previously described.

These options may be used in conjunction with each other. e.g. `/+/N`

Step 6

If the record is already on the receiving VMF, LEX queries the transfer with the message

```
Already on file?
```

The following replies may be made.

`Y`	causes the record to be copied
`/`	allows the record to be copied and given a different name. On entering the `/` it is followed by a prompt `New Key` for the new record name.
ESC E	returns to the Key prompt
RETURN	Any other response and the record is ignored.

Step 7

ESC E terminates the option returning LEX to the calling menu.

A.30 OPTION /5 PATCH ROUTINE

Purpose:

LEX has the ability to patch its own programs, including the LEX code file and the run time system.

Method:

The steps are shown below use prompts taken from **TEXT2.

Step 1

```
LEX Patch program version 9A

Enter name of file to patch ____________________
```

The filename of the file to be patched is required. Account/device details may be included, the default being the system device. If the filename does not exist then a (BELL) is sent to the VDU and LEX waits for another entry.

Step 2

```
*
```

This is the patch routine prompt and the following may be entered.

Location/	Opens WORD location and gives content in both octal and ASCII, the latter in brackets. Non displayable ASCII characters are shown as a "."
Location\	Opens BYTE location and gives content in both octal and ASCII, the latter in brackets. Non displayable ASCII characters are shown as a "."

Once a location is open the following may be entered.

nnnnnn	set word/byte to nnnnnn decimal
Caa	set word/byte to ASCII aa
(LINEFEED)	open next location
(RETURN)	return to * prompt ready to open new location.

Step 3

An E or (ESC)E at the * prompt terminates the patch returning LEX to the menu from which the option was called.

Notes

Besides the menu switch /5, the patch program LXINST.D11 can be used. This program is specifically to patch those options which are required on entering LEX and is convenient since it does not require a knowledge of the locations to be patched. It is entered at the .vmf? prompt by ESC# as described in Chapter 4.

As a safeguard take a back up of the file being patched using /5

A.31 OPTION /6 CONVERT ASCII FORMATS

Purpose:

This menu switch option converts documents from one ASCII format to another. It is primarily for foreign language users who want to make use of 8 bit codes for, say, accented characters.

Method:

It works like any copy option calling for an Input document name and Output document name and uses a record *CONV to define the translation characters. *CONV is set up by activating *CONVFORM and entering the details. See Chapter 7 for details of the conversion form.

Appendix B. CONTROL FUNCTIONS

SUMMARY

Moving Around the Screen

CTRL H	Left Arrow	*B.8*
CTRL R	Right Arrow	*B.18*
CTRL U	Up Arrow	*B.21*
CTRL J	Down Arrow	*B.10*
CTRL F	Forward Word	*B.6*
CTRL B	Back Word	*B.2*
CTRL G	End of Line	*B.7*
CTRL I	Tab	*B.9*
CTRL M	Return	*B.13*
CTRL Q	Top Document	*B.17*
CTRL S	End of Document	*B.19*
CTRL L	Forward Screen	*B.12*
CTRL T	Back Screen	*B.20*

Text Editing

CTRL E	Insert Character - Ignore rulers	*B.5*
CTRL N	Insert Character - Obey rulers	*B.14*
CTRL A	Insert Blank Line	*B.1*
CTRL D	Delete Character - Ignore rulers	*B.4*
CTRL P	Delete Character - Obey rulers	*B.16*
CTRL W	Delete Word	*B.23*
CTRL V	Delete to End of Line	*B.22*
CTRL X	Delete Line	*B.24*
CTRL K	Change Case	*B.11*
CTRL Y	Copy Character Above	*B.25*

File Handling

CTRL O	Exit	*B.15*
CTRL C	Panic	*B.3*
CTRL Z	Abandon Edit	*B.26*

B.1 CTRL A INSERT BLANK LINE

CTRL A inserts a blank line into existing text above the line on which the cursor is positioned. The rest of the text on the screen is moved down to make space.

If the screen has the facility of a hardware actioned insert line then this should be defined in the Insert a Line field in *VDUFORM. If no such facility exists, LEX will insert a blank line and refresh the entire screen from the current line onwards. As the action of refreshing the entire screen may involve the transmission of up to 2000 characters or more the hardware facility should be used wherever possible.

B.2 CTRL B BACK WORD

When CTRL B is entered, the cursor moves back to the start of the previous word. Use of this function is an efficient way of moving backwards through the text as the cursor will, if necessary, move beyond the start of a paragraph or even beyond the top of the screen. The only limitation is the top of the buffer, which if the work file is large enough, is the top of the document.

B.3 CTRL C PANIC

If CTRL C is passed through to LEX then processing of the current task will be abandoned, the actual effect depending on the nature of the task.

Menu. At a menu or in answer to a Yes/No type question CTRL C will act as an ESC E

Printing. LEX will stop printing and return to the menu. Data already in the printer buffer will still be output.

List Processing. As for printing

Editing. When in a repeat loop LEX will halt the process and return control to the user. In Find, Get or Merge mode LEX will stop at the current line. On all other occasions it is ignored.

Programming. If actioning a keystore or system record when CTRL C is received then LEX will stop and return to normal edit mode.

B.4 CTRL D DELETE CHARACTER - IGNORE RULER

This is used to delete the character currently marked by the cursor. The entire line following the cursor, including a continuation line, is moved one place left ignoring all tab and margin settings in the current ruler. Therefore using this function in multiple columns will distort the layout to the right of the cursor. (Note that CTRL P deletes, obeying the rulers and ignoring continuation lines.) The last character deleted with CTRL D can be reinstated by ESC-D.

B.5 CTRL E INSERT CHARACTER - IGNORE RULERS

The opposite of CTRL D is CTRL E in that it opens a space at the cursor position, shifting all characters to the right of the cursor one place further right. As with CTRL D all margin and tab settings in the ruler are ignored and tabulated work will be distorted. Characters pushed beyond the right hand edge of the screen cause a continuation line character to be appended with the overspill characters wrapped onto the next line. Note that CTRL N may be used to insert a space obeying the current ruler.

B.6 CTRL F FORWARD WORD

CTRL F moves the cursor forward through the text to the first character of the next word on the screen. If necessary moves beyond the end of the currently displayed screen by scrolling forward and amy be used until the end of the document is reached.

B.7 CTRL G END OF LINE

Entering CTRL G causes the cursor to be moved to the position after the last character of any text on the current line. The function has no effect when at a prompt for Key or Document Name. Other special cases are:

a) If the cursor is to the right of the last character then it moves back to get to this position.

b) If the cursor is already on the first blank after the last character then it moves down to the end of the next line.

c) If there is no text on the current line then the cursor moves to the extreme left hand side of the screen.

B.8 CTRL H **LEFT ARROW**

This function moves the cursor one position to the left without affecting the text. It is usually mapped on the keyboard as the ← key. Operation of CTRL H when in the first (lefthand) column of the screen causes the cursor to move to the extreme right hand edge of the previous line. Note that it is faster and more efficient to use CTRL B to move backwards word by word along a line of text as fewer characters are transmitted to and from the computer to achieve the desired movement.

B.9 CTRL I **TAB**

This moves the cursor to the next margin or tab setting as specified by the ruler in force. It is exactly the same as the TAB key and is the code usually generated by that key on a keyboard. Note that back tab is available by use of ESC CTRL I or ESC TAB.

The TAB command is also a formatting command when used under rulers containing the characters C, R, or #. When text is entered between C...M margins or after a R.... a TAB moves the text to the appropriate position, centering or right justifying. A TAB to a ...###.. causes LEX to enter calculator mode. Note that back tab, ESC TAB, will enter the calculator but not perform formatting functions.

In protect mode TAB moves the cursor to the next protected area on the screen whilst at a prompt for Key or Document Name it has no effect at all.

B.10 CTRL J **DOWN ARROW**

Use of this function moves the cursor one line down the screen in the same column without affecting the text. It is normally mapped as the ↓ key on the input keyboard and is also generated by the LINEFEED key. If the cursor is on the bottom line of the screen when this function is used, LEX causes the screen to scroll forward one line.

B.11 (CTRL)K CHANGE CASE

This function changes upper case characters to lower case and lower case characters to upper case from the cursor position to the end of the word. If the cursor is positioned on a space, then the next word on the line is changed. The cursor is left on the first space after the word just altered or on the first underline character encountered.

In standard applications, besides the expected change of case for the alphabetic characters, some other characters are altered. These are as follows:

```
^ ~ { } [ ] \ | @ "
~ ^ [ ] { } | \ " @
```

For foreign languages and other special applications the standard ASCII case conversion may be unsatisfactory. LEX caters for this by a table of conversions in *CASEFORM. Any character may be defined as the opposite case from any other character. Case conversion tables are considered in Chapter 7.

B.12 (CTRL)L FORWARD SCREEN

This is used to advance the cursor by one complete screen of text. The number of lines advanced is as specified in the Lines field of *VDUFORM. The cursor remains in the same column on the screen as it was when the function was actioned. If the cursor is positioned beyond the end of text in the document then an error bleep occurs and the function has no effect.

In protect mode the cursor goes to the last unprotected location on the screen whilst at a prompt for Key or Document Name it has the same effect as (RETURN).

B.13 (CTRL)M RETURN

(CTRL)M is equivalent to (RETURN). It has the effect of moving the cursor down one line and onto the left hand column of the screen. Any margin or tab settings specified in the ruler are disregarded; it always goes to the first column.

At prompts such as Key and Document Name, (RETURN) or (CTRL)M is the terminator causing LEX to continue processing. It is also the terminator at the ? prompt - the programmable pause.

B.14 CTRL N — INSERT CHARACTER - OBEY RULER

This function is similar to CTRL E in that it opens up a space at the position marked by the cursor and moves text to the right to create the space. The difference is that unlike CTRL E, text outside the current margins is not affected. Text pushed beyond the margins will be word wrapped down onto the next line within the same margins.

B.15 CTRL O — EXIT

CTRL O terminates the current activity and returns to the previous one. Its main use is on completion of document edits and is identical to ESC E.

Many operating systems use CTRL O for special purposes and intercept the character before it is received by LEX. In these cases the function may only be used in programming or if a function key is allocated to the command. The latter is achieved using *VDUFORM. (See Chapter 8)

B.16 CTRL P — DELETE CHARACTER - OBEY RULER

This is the inverse of CTRL N. The effect is to delete the character marked by the cursor and pull one place left all the remaining text to the right of the cursor and within the margins. If the ruler contains tab settings but not true margins then only a single word will be pulled to the left leaving all text after the first word in position. The last character deleted using CTRL D or CTRL P, can be reinstated using ESC-D.

.17 CTRL Q — TOP DOCUMENT

Whilst few systems will be able to use it directly, the internal meaning of CTRL Q is to go to the top of a document, i.e it is identical to ESC QM.

Most operating systems use CTRL Q for special purposes and intercept the character before it is received by LEX. In these cases the function may only be used in programming or if a function key is allocated to the command. The latter is achieved using *VDUFORM. (See Chapter 8)

B.18 CTRL R **RIGHT ARROW**

CTRL R is usually mapped to the → key. The cursor moves over text without affecting it and any margin or tab settings are ignored. If the cursor is in the extreme right-hand column of the screen, it will move to the extreme lefthand column on the next line down.

B.19 CTRL S **END OF DOCUMENT**

The complement to CTRL Q is CTRL S. In this case the very last screenful of a document is displayed with the cursor left on the last line of text. It is identical to ESC QE.

Most operating systems use CTRL S for special purposes and intercept the character before it is received by LEX. In these cases the function may only be used in programming or if a function key is allocated to the command. The latter is achieved using *VDUFORM. (See Chapter 8)

B.20 CTRL T **BACK SCREEN**

This function moves the cursor to the top left hand corner of the screen - the home position. If the cursor is already at the home position LEX scrolls backwards one screen of text so that the first line on the screen becomes the last. Trying to back screen beyond the start of a document causes ERROR-TB and a BELL sent to the VDU. The function has no effect at a prompt for Key or Document Name.

B.21 CTRL U **UP ARROW**

CTRL U moves the cursor up one line, staying in the same column. It is normally mapped on the keyboard as the ↑ key. If the cursor is on the top line of the screen, LEX scrolls backwards through the text.

When this function is used at either a menu, Select, Document Name or Printer prompt it has the same effect as ESC E unless **MONT has been modified to enable another character or function to be used (see Chapter 5).

B.22 CTRL V — DELETE TO END OF LINE

CTRL V deletes all characters from the current cursor position to the end of the line. The character marked by the cursor is included in the text deleted. The last string of text deleted using this function can be restored by ESC-V.

Where possible, this function should be set to make use of the VDU hardware method for deleting to the end of a line by defining it in the `Clear to EOL` field of `*VDUFORM` (see Chapter 8). If not available, LEX will send the requisite number of spaces to delete the text from the screen, a less efficient solution.

B.23 CTRL W — DELETE WORD

If the cursor is positioned anywhere within a word then this function deletes the entire word and shifts left the remainder of the text on that line within the current margins. The cursor is left at the beginning of the next word which is now in the position previously occupied by the first character of the deleted word. Text outside the current margins is not affected. If the cursor is positioned on a space, this function takes no action. The last word deleted by this function can be reinstated by using ESC-W.

Note that this function does not work in protect mode nor at a prompt for Key or Document Name.

B.24 CTRL X — DELETE LINE

CTRL X causes the entire line on which the cursor is positioned to be deleted with the remainder of the document shifted up one line. The cursor is left in the same position on the screen as before the deletion occurred. It has no effect protect mode nor at a prompt for Key or Document Name. The last line deleted by using this function can be reinstated by ESC-X.

Where possible this function should be set to make use of the VDU hardware method for deleting a line by defining it in the `Delete line` field of `*VDUFORM`. If this is not available then the entire screen from the cursor position to the end is refreshed, a less efficient solution.

B.25 CTRL Y — COPY CHARACTER ABOVE

CTRL Y causes a single character to be copied from the line above. This function ignores rulers and moves the cursor one place to the right after copying the character.

If CTRL Y is used when in the calculator it copies the entire number above into the current calculator field.

B.26 CTRL Z — ABANDON EDIT

This function is used by LEX to exit a document without applying the changes since the last edit i.e. it is identical to ESC Z. If enabled in **INST a prompt Are You Sure? appears requiring Y or y to proceed.

ANNEX TO APPENDIX B

Problems peculiar to specific operating systems

The following notes describe the CTRL characters which are intercepted by the operating system. In these cases it is not possible to type the CTRL characters directly at the keyboard. However the function can still be used by mapping it to a function key using *VDUFORM or *VDUFORM2. The function can also be used within a program or keystore.

VAX/VMS

CTRL Q	causes a stalled screen to continue.
CTRL S	causes the screen to stall until a CTRL Q is received.
CTRL X	is used to discard unused input.
CTRL Y	aborts the current program thereby leaving LEX.

RSX

CTRL Q	causes a stalled screen to continue.
CTRL S	causes the screen to stall until a CTRL Q is received.
CTRL X	is used to discard unused input.

RSTS

CTRL C	may be used for LEX internal functions but the ^C characters will be forced onto the screen. ESC QR will refresh the screen and remove the characters.
CTRL Q	causes a stalled screen to continue.
CTRL S	causes the screen to stall until a CTRL Q is received.

MS DOS/PC DOS

CTRL C	may be used for LEX internal functions but the ^C characters will be forced onto the screen. ESC QR will refresh the screen and remove the characters.
CTRL N	in certain circumstances when LEX is busy CTRL N may be intercepted by MS-DOS and causes the printer to echo every key depression.

CTRL P	in certain circumstances when LEX is busy CTRL P may be intercepted by MS-DOS and causes the contents of the screen to be sent to the printer.
CTRL Q	causes a stalled screen to continue.
CTRL S	causes the screen to stall and no further output to be accepted until a CTRL Q is received.

UNIX

CTRL C	may be used for LEX internal functions in the prescribed manner on UNIX System 3 and later. On earlier versions CTRL C has no effect at all.
CTRL Q	causes a stalled screen to continue.
CTRL S	causes the screen to stall until a CTRL Q is received.

Appendix C. ESCAPE SEQUENCES

SUMMARY

STATUS AND INFORMATION MESSAGES

ESC H	Display help message	*C.38*
ESC SE	Show error mode on	*C.88*
ESC -SE	Show error mode off	*C.121*
ESC L	Display edit status line	*C.42*
ESC RL	Insert edit status line in document	*C.77*
ESC SL	Show automatic edit status line	*C.92*
ESC -SL	Switch automatic edit status line off	*C.124*
ESC CS	Clear automatic edit status line	*C.32*
ESC SW	Set page warning mode	*C.100*
ESC -SW	Switch page warning mode off	*C.126*
ESC RD	Document name	*C.73*
ESC RN	Document key	*C.79*
ESC RI	Installation parameters	*C.75*
ESC RM	VMF name	*C.78*
ESC RO	Current menu switch option	*C.80*
ESC RV	VDU key	*C.82*
ESC RW	Work file name	*C.83*

MOVING AROUND THE SCREEN

ESC RETURN	Carriage return to ruler margin	*C.3*
ESC QM	Top document	*C.61*
ESC QE	End of document	*C.53*
ESC QL	Last character on screen	*C.60*
[redacted]		*C.54*
ESC QG	Back paragraph	*C.55*
ESC QK	Left edge of current line	*C.59*
ESC QU	End of last word	*C.66*
ESC QV	End of next word	*C.67*
ESC TAB	Back tab	*C.1*
ESC S(n	Mark position n on screen	*C.84*
ESC <n	Return to marked position n	*C.15*

CLEAR SCREEN

ESC CA	Clear to end of document	C.27
ESC CB	Clear background of current screen	C.28
ESC CF	Clear foreground of current screen	C.30
ESC CP	Clear protected area and paint characters	C.31

RULERS

ESC Mn	Display and assert ruler n	C.43
ESC Wn	Temporarily store ruler	C.109
ESC SG	Ignore automatic assertion of rulers	C.90
ESC -SG	Re-establish automatic assertion of rulers	C.122
ESC Un	Use ruler n - retain auto assertion	C.103
ESC UUn	Use ruler n - and ignore auto assertion	C.108
ESC MZ	Insert tab in ruler	C.45
ESC -MZ	Remove tab from ruler	C.120

TEXT EDITING

ESC #	Right align	C.5
ESC @	Centre	C.6
ESC &	Wide line	C.8
ESC LINEFEED	Split paragraph	C.4
[redacted]	[redacted]	C.40
ESC QJ	Partial justify	C.58
ESC QA	Copy line above	C.49
ESC QB	Copy line below	C.50
ESC QC	Swap characters	C.51

DELETE AND RECOVER DELETED TEXT

ESC -D	Restore last deleted character	C.116
ESC -W	Restore last deleted word	C.128
ESC CTRL X	Restore last deleted line	C.129
ESC -X	Restore last deleted line	C.129
ESC -V	Restore last deleted part line	C.127

EDITING MODES

(ESC)\	Vertical Mode - Enter	*C.7*
(ESC)-\	Vertical Mode - Exit	*C.115*
(ESC)I	Insert Mode - Enter	*C.39*
(ESC)-I	Insert Mode - Exit	*C.118*
(ESC)QD	Insert Mode - Toggle in and out	*C.52*
(ESC)QH	Manual hyphenation mode	*C.56*
(ESC)QI	Show hyphenation	*C.57*
(ESC)+	Enter calculator	*C.17*

CUT AND PASTE

(ESC)(	Mark for Cut & Paste	*C.9*
(ESC))B	Cut & Blank - To main buffer	*C.10*
(ESC))xB	Cut & Blank - To buffer x	*C.10*
(ESC))nB	Cut & Blank - To keystore n	*C.10*
(ESC))L	Cut & Leave - To main buffer	*C.10*
(ESC))xL	Cut & Leave - To buffer x	*C.10*
(ESC))nL	Cut & Leave - To keystore n	*C.10*
(ESC))S	Cut & Squeeze - To main buffer	*C.10*
(ESC))xS	Cut & Squeeze - To buffer x	*C.10*
(ESC))nS	Cut & Squeeze - To keystore n	*C.10*
(ESC).B	Cut & Blank Sentence - To main buffer	*C.11*
(ESC).xB	Cut & Blank Sentence - To buffer x	*C.11*
(ESC).L	Cut & Leave Sentence - To main buffer	*C.11*
(ESC).xL	Cut & Leave Sentence - To buffer x	*C.11*
(ESC).S	Cut & Squeeze Sentence - To main buffer	*C.11*
(ESC).xS	Cut & Squeeze Sentence - To buffer x	*C.11*
(ESC)*E	Paste (Elbow) - From main buffer	*C.12*
(ESC)*xE	Paste (Elbow) - From buffer x	*C.12*
(ESC)*I	Paste (Insert) - From main buffer	*C.12*
(ESC)*xI	Paste (Insert) - From buffer x	*C.12*
(ESC)*O	Paste (Overlay) - From main buffer	*C.12*
(ESC)*xO	Paste (Overlay) - From buffer x	*C.12*
(ESC)*D	Delete Cut & Paste buffer	*C.12*
(ESC)*xD	Delete Cut & Paste buffer n	*C.12*

ESC }E	Paste - Elbow in blanks	*C.13*
ESC }I	Paste - Insert blanks	*C.13*
ESC }O	Paste - Overlay blanks	*C.13*

SEARCH

ESC F	Find	*C.36*
ESC G	Get	*C.37*
ESC -G	Back search	*C.117*

DATE AND TIME

ESC K	Date - YYMMDD	*C.41*
ESC T	Date - Short form	*C.102*
ESC Y	Date - In full	*C.113*
ESC QN	Time	*C.62*

ATTRIBUTES

ESC An	Set attributes n	*C.19*
ESC Aa	Set combination of attributes	*C.19*
ESC Bn	Unset attribute n	*C.20*
ESC BL	Append attributes to end of line	*C.23*
ESC BW	Append attributes to next word	*C.26*
ESC BT	Show attributes of current line	*C.24*
ESC BU	Display attributes currently set	*C.25*

MERGE AND EXTRACTS

ESC MA	Merge - Advance	*C.44*
ESC MC	Merge - Close file	*C.44*
ESC ME	Merge - To end	*C.44*
ESC MI	Merge - Line	*C.44*
ESC MJ	Merge - Selected lines	*C.44*
ESC MM	Merge - Entire document	*C.44*
ESC MO	Merge - Open	*C.44*
ESC MS	Merge - Skip	*C.44*
ESC MU	Merge - Until	*C.44*
ESC MV	Merge - View	*C.44*
ESC RG	Merge file name	*C.74*
ESC WC	Extract - Close file	*C.111*
ESC WE	Extract - To end	*C.111*
ESC WJ	Extract - Selected lines	*C.111*

ESC WO	Extract - Open file	*C.111*
ESC WU	Extract - Until	*C.111*
ESC WX	Extract - To end of file	*C.111*
ESC OD	Open document from playground - Overwrite	*C.47*
ESC OF	Open document from playground - Overwrite	*C.47*
ESC OX	Open document from playground - Create	*C.47*

SORTING

ESC BA	Line sort - Ascending	*C.21*
ESC BD	Line sort - Descending	*C.21*
ESC BHA	Column sort - Horizontal - Ascending	*C.22*
ESC BHD	Column sort - Horizontal - Descending	*C.22*
ESC BVA	Column sort - Vertical - Ascending	*C.22*
ESC BVD	Column sort - Vertical - Descending	*C.22*

RECORD MANAGEMENT

ESC]A	Add to record - including implicit RETURNs	*C.16*
ESC >A	Add to record - excluding implicit RETURNs	*C.16*
ESC]C	Create record - including implicit RETURNs	*C.16*
ESC >C	Create record - excluding implicit RETURNs	*C.16*
ESC]K	Kill record - including implicit RETURNs	*C.16*
ESC]L	Lock record	*C.16*
ESC]P	Put record - including implicit RETURNs	*C.16*
ESC >P	Put record - excluding implicit RETURNs	*C.16*
ESC]R	Read record	*C.16*
ESC]S	Reinstate deleted record	*C.16*
ESC]U	Update record - including implicit RETURNs	*C.16*
ESC >U	Update record - excluding implicit RETURNs	*C.16*

ESC N	Next key	*C.46*
ESC -N	Previous key	*C.119*
ESC QY	Next deleted key	*C.70*
ESC RK	Perform system record	*C.76*

PROGRAMMING FEATURES

ESC SB	Set background	*C.86*
ESC UB	Unset background	*C.104*
ESC SD	Set delay factor	*C.87*
ESC SF	Set foreign file	*C.89*
ESC UF	Unset foreign file	*C.105*
ESC SH	Enter shush mode	*C.91*
ESC -SH	Unset shush mode	*C.123*
ESC SU	Enter shut-up mode	*C.98*
ESC -SU	Unset shut-up mode	*C.125*
ESC SM	Set mask	*C.93*
ESC UM	Unset mask	*C.106*
ESC SP	Set protect	*C.94*
ESC UP	Unset protect	*C.107*
ESC SQa	Set squash character to a	*C.95*

REPEAT SEQUENCE

ESC Xn	Repeat n times	*C.112*
ESC XO	Repeat to end of document	*C.112*
ESC XU	Repeat - until	*C.112*
ESC XW	Repeat - while	*C.112*
ESC XW	Repeat previous escape sequence	*C.18*

CONDITIONALS

ESC CTD	Conditional - on document	*C.33*
ESC CTF	Conditional - on foreign file key	*C.33*
ESC CTK	Conditional - on key	*C.33*
ESC CTN	Conditional - on numeric field	*C.33*
ESC CTQ	Conditional - on query	*C.33*
ESC CTS	Conditional - on string	*C.33*
ESC CT+	Conditional - on accumulator positive	*C.33*
ESC CT-	Conditional - on accumulator negative	*C.33*
ESC CTO	Conditional - on accumulator zero	*C.33*

CHAINS AND FORKS

ESC QS	Fork and return - direct	*C.64*
ESC QT	Fork and return - via prompt	*C.65*
ESC QW	Fork and return - via prompt	*C.68*

MISCELLANEOUS COMMANDS

ESC SPACE	Cancel escape	*C.2*
ESC Sn	Set keystore n	*C.85*
ESC n	Recall keystore	*C.14*
ESC Rn	Load keystore with last keystrokes	*C.71*
ESC RS	Recall last keystrokes	*C.81*
ESC RC	Clear keystroke stores	*C.72*
ESC CD	Change document details	*C.29*
ESC CU	Clear function string	*C.34*
ESC P	Paginate	*C.48*
ESC =	Repeat previous escape sequence	*C.18*
ESC QR	Refresh screen	*C.63*
ESC SSn	Send sequence n to VDU	*C.96*
ESC STn	Send sequence n to VDU	*C.97*
ESC SV	Send to VDU	*C.99*
ESC SX	Numeric value of keyboard keys	*C.101*
ESC V	Re-edit - to top of document	*C.109*
ESC QX	Re-edit - to current position	*C.69*
ESC E	Exit	*C.35*
ESC Z	Abandon edit	*C.114*

C.1 ESC TAB **BACK TAB**

Whereas CTRL I or TAB takes the cursor on to the next tab on the current ruler, ESC CTRL I or ESC TAB moves the cursor back to the last tab mark. If it encounters a # in the ruler then LEX enters calculator mode but unlike the forward TAB it will not perform any formatting on R (right justify) or C (centre) rulers.

C.2 ESC SPACE **CANCEL ESCAPE**

To cancel a sequence after the ESC has been pressed, type a SPACE (Space Bar). The BELL will not sound.

C.3 ESC RETURN **CARRIAGE RETURN TO RULER MARGIN**

ESC RETURN performs a carriage return to the appropriate left margin as defined by an F or J in the current ruler. If margins are not set then a standard RETURN is performed.

C.4 ESC LINEFEED **SPLIT PARAGRAPH**

This sequence is used to break the current line of text into two portions.

Text from the cursor position to the end of the paragraph and within the current margins is deleted and re-inserted as a new paragraph starting at the left hand margin on the line below. Text outside the margins is not affected although a space may be opened up to make way for the inserted blank line. The cursor is left at its original position.

C.5 ESC # **RIGHT ALIGN**

This is a convenient way to right-align text without an ..R.. in the ruler. The text effected is that between the cursor and the next tab mark in the ruler to the left of the cursor. This text is right aligned onto the next tab or margin setting.

C.6 ESC @ **CENTRE**

This is a convenient way to centre text without using a C...M ruler. The text on the current line between the cursor and the next tab mark to its left is centered between that tab mark and the next tab setting to the right of the cursor.

C.7 ESC\ ENTER VERTICAL MODE

This sequence causes LEX to enter vertical typing mode so that each character is placed immediately below its predecessor. It provides a convenient way of overwriting a column and can be used in conjunction with insert mode to elbow in a set of values.

There are three ways to return to normal typing mode.

ESC\	Toggles in and out of vertical mode.
ESC-\	Forces LEX into normal typing mode.
ESCE	To return to a menu.

C.8 ESC& WIDE LINE

If the continuation character as defined in **MONT (usually &), is placed in the extreme right-hand column of the screen then the following line of text is stored as a continuation line.

It is possible to type the continuation character on the extreme right hand edge but this is often difficult to locate quickly. ESC& positions the ampersand correctly and return the cursor to the position from which the sequence was executed.

As the continuation character is not stored in the text file, text created on an 80 column wide VDU will, when subsequently viewed on a 132 column wide VDU, appear as single lines of up to 132 columns. Using this technique, it is possible to generate text of up to 255 columns width. Note however that a line can only have one continuation. Therefore on an 80 column wide VDU the maximum width of text is 160 characters.

When using this extend line facility, it is not possible to have rulers set up to wrap through the line extension. It is therefore suggested that the ruler be set with tab settings but without hard margins.

C.9 ESC(MARK FOR CUT & PASTE

In any cut and paste operation it is necessary to mark the top left hand corner of an area of the screen which is to be cut.

The character at the position so marked is displayed using the mode defined in *VDUFORM3. Typically this may be double intensity or inverse video. This marking only stays on the screen as long as the marked position remains on the screen.

If the text is scrolled off the screen and subsequently brought back, the indication is lost although the marked spot is still recorded.

C.10 ESC)a ESC)xa ESC)na **CUTTING TEXT**

Text may be cut into various buffers. ESC)a cuts text into the main cut and paste buffer whilst ESC)xa stores text in auxiliary buffers defined by the character x. The auxiliary buffers are T, U, V, W, X, Y and Z. The third form of cut sequence, ESC)na, stores the text in the keystore nominated by the number n in the range 0 to 9. In all these options the character a describes the effect on the remaining text and can be B for blank, L for leave, or S for squeeze as described below.

As a keystore is limited to 78 characters, the ESC(, and ESC)na, must appear on the same line.

If both the MARK and CUT sequences occur in the extreme left hand column of the screen, then the complete lines between the two points are moved into the buffer. If, however, either or both the marked points are away from the left hand column, then a rectangular block of text is stored with the block defined at the TOP LEFT hand corner by the MARK and at the BOTTOM RIGHT hand corner by the CUT sequence. If the CUT is used without the MARK having been previously marked, or if the CUT is to the LEFT or ABOVE the MARK then ERROR-AN is signified by a BELL.

When the CUT is made it is not necessary for the MARK to be on the screen, the amount cut being limited only by the available space in the work file.

After cutting text from a screen, the marked position for the top left hand corner remains. Subsequent cuts may be made using this position.

Cut and paste can be used to transfer text between documents.

To delete the cut and paste buffers use ESC*xD to empty buffer a or menu switch option /2 to delete them all.

A maximum of 16,384 characters (32 blocks) are available for the storage of text in cut and paste buffers.

(ESC))B (ESC))xB (ESC))nB **CUT & BLANK**

This sequence stores the block of text defined by the position of the MARK and this CUT sequence into the nominated cut and paste buffer or keystore. Text in the block on the screen is overwritten with space characters.

(ESC))L (ESC))xL (ESC))nL **CUT & LEAVE**

Here the text is cut into the nominated buffer or keystore but the screen is left unaffected with all text still in place.

(ESC))S (ESC))xS (ESC))nS **CUT & SQUEEZE**

The cut and squeeze sequence cuts the text into the nominated buffer or keystore and then squeeze the text off the screen. The squeeze will depend on the type of cut being made. If both MARK and CUT sequences are in the left hand column of the screen, indicating a cut of entire lines, then these lines will be erased from the screen with the remaining text moved up into the space created. If a block of text is being cut and squeezed, then all text to the RIGHT of the block will be shifted left to compensate for the gap created. Text marked as continuation lines (see (ESC)&) is not pulled back into the squeezed text.

C.11 (ESC).a (ESC).xa **SENTENCE CUT**

Whereas other cut sequences move a rectangular block of text to the buffer, these sequences work on character strings, such as a sentence. Sentence cut and paste must take place under justifying margins, i.e. F/J.....M. ERROR-NJ and a (BELL) is sent to the VDU if outside these limits.

The first character to be cut is marked with the MARK, (ESC)(and the last with the appropriate (ESC).a or (ESC).xa sequence. All the buffers can be used so x can be T, U, V, W, X, Y or Z.

The length of the string is limited only by the amount of free cut and paste space in the work file.

ESC.B ESC.xB **CUT & BLANK SENTENCE**

The text is cut into the nominated cut and paste buffer and is replaced by blanks on the screen.

ESC.L ESC.xL **CUT & LEAVE SENTENCE**

The text is cut into the nominated cut and paste buffer and the screen left unchanged.

ESC.S ESC.xS **CUT & SQUEEZE SENTENCE**

The text is cut into the nominated cut and paste buffer and replaced by blanks on the screen. The paragraph is then automatically re-justified.

C.12 ESC*a ESC*xa **PASTING TEXT**

Once text has been cut into one of the cut and paste buffers it can be re-inserted into the text elsewhere. In the general form of these commands x defines the appropriate buffer and a controls the format of the text returning to the screen.

If a is the character D then the nominated cut and paste buffer is emptied. Other paste options do not destroy the buffers so the paste can be repeated in more than one place.

ESC*E ESC*xE **PASTE - ELBOW**

The entire text from the current cursor position is elbowed to the right, in a similar manner to CTRL E, thus making space for the block of text stored in the nominated cut and paste buffer. If any characters are moved beyond the end of a line they are lost; extension lines are not created.

If the text in the cut and paste buffer has been cut using a sentence cut sequence, ESC.a, then this paste option enters the text in insert mode and then re-justifies the paragraph.

ESC*I ESC*xI — PASTE - INSERT

In this sequence a blank line is inserted at the cursor position for every line in the block of text. The text is then pasted into the space created with the cursor position marking the top left hand corner of the block. Any characters pasted beyond the end of the line are ignored; continuation lines are not created.

ESC*O ESC*xO — PASTE - OVERLAY

With the cursor marking the top left hand corner of the block, this sequence over-writes any text on the screen with the block of text from the nominated buffer. No extension lines are created so text beyond the edge of the screen is lost.

If the text in the cut and paste buffer has been cut using a sentence cut sequence, ESC.a, then this paste option overlays the text on the current line character by character within the margins. In the same way as overtyping text, on reaching the right margin a blank line is inserted below for the remaining text which again is inserted between the margins.

ESC*D ESC*xD — DELETE CUT AND PASTE BUFFER

This sequence empties the nominated buffer x. Note that menu switch option /2 deletes all buffers.

C.13 ESC)a — PASTING BLANKS

A set of escape sequences allow blocks of blank characters to be inserted into areas already filled with text.

First the top left corner of an area of text must be marked using ESC(. Then a sequence of the above general form is entered with the cursor at the bottom right of the block. If the mark and paste are both entered with the cursor in the extreme left hand column of the screen then complete blank lines are opened up. The a can be E for elbow, I for insert or O for overlay as outlined below.

Whilst these sequences are similar in nature to cut and paste, they do not use or affect the cut and paste buffers.

ESC}E **PASTE - ELBOW IN BLANKS**

This sequence is used to insert blanks by elbowing text to the right of the screen. Text will be lost irretrievably if characters are elbowed off the right hand edge of the screen.

ESC}I **PASTE - INSERT BLANKS**

This sequence inserts whole lines of blanks from the marked position to the final position. No text is lost.

ESC}O **PASTE - OVERLAY BLANKS**

This sequence causes blanks to be written over the text previously on the screen. The overwritten text is destroyed and cannot be recovered.

C.14 ESCn **RECALL KEYSTORE**

The sequence ESCn recalls the keystore n, where n defines keystore 0 to 9.

The keystore must have been set up first by one of three methods.

a) Using the document extension details (default settings).
b) Using the cut and paste sequence ESC)na.
c) Using ESCsn to view and load a keystore.

Keystores are a useful means for quickly recalling standard phrases, sentences or abbreviations.

They can also be used to hold small programs since ESCn automatically interprets visible equivalents as the appropriate command. (See Chapter 11). Note that at the Key and Document Name prompts no visible equivalent translation takes place.

The command may be used in calculator mode where the contents of the keystore are treated as though typed in.

Once set up keystores are preserved in the work file between editing sessions.

C.15 ESC<n RETURN TO MARKED POSITION

The sequence ESC<n returns the cursor from anywhere on the screen to a previously marked position n. The positions are marked by the ESCs(n sequence where n is 0 to 9. The marked position for 0 is achieved automatically by the select command, ESC(which is used to mark the top in a Cut and Paste sequence. See ESCs(n for further details.

Note that in programming mode a < is interpreted as [. To overcome this the sequence must be entered as $@<n.

C.16 ESC]a ESC>a RECORD HANDLING

Chapter 2 describes the structure of the VMF file, its records and the keys to the records. The following sequences are used to access these records.

When a record handling sequence is invoked LEX searches back up the screen a [which indicates the start of a record key. If one is not found LEX sends ERROR-NX and a BELL. The 10 characters following the [are the key of the system record.

The character a defines the action to be taken, such as read, update, kill, etc, and are all described below. Use of the] or > effects implicit RETURNs.. In the ESC]a form RETURNs are left in the record while with ESC>a the RETURNs are replaced with spaces. The ESC>a sequences are used primarily to store standard paragraphs with text which is to be recalled under a potentially different ruler than that under which it was created. They may also be used to create system records where all RETURNs are explicitly defined by use of the relevant visible equivalent.

In multi user mode safeguards are built into LEX to ensure that a record is only updated by one user at a time. This is achieved by locking a record. On such systems the various write options described below will only work if preceded by the locking action of ESC]L.

The sequences are often used in conjunction with forms and protected screens. Details can be found in Chapter 12 on arranging a database.

ESC]A ESC>A APPEND TO RECORD

This sequence is used to append the current screen contents onto the end of an existing record.

If the specified record is not already on the file, LEX simply treats the request as if it were a create or update without indicating any error.

ESC]C ESC>C **CREATE RECORD**

This sequence is used to create a new record on the VMF. If the record is already on file, ERROR-BC is set and a BELL sounds as a warning and the file is not updated. To overwrite an existing record the sequence ESC]U or ESC>U must be used.

When creating a record from a protected form the updated area is cleared with the paint character, usually the underline.

ESC]K **KILL RECORD**

To remove a record from the system file, the key is followed by ESC]K. If the specified key is not found, ERROR-BK is set and a BELL is sent to the VDU.

In practice LEX simply flags the record and does not physically delete it until the next reorganisation of the VMF. It is possible to reinstate deleted records using ESC]S.

ESC]L **LOCK RECORD**

In multi user mode it is necessary to lock a record before it can be updated. ESC]L performs an ESC]R and locks the record. If the record is already locked ERROR-LK is set and a BELL sent to the VDU.

ESC]P ESC>P **PUT RECORD**

This sequence is used to update the VMF regardless of whether or not the specified record is already on the file.

It performs the same function as ESC]U or ESC>U except that when in protected forms the record remains visible and is not cleared from the screen after writing away.

ESC]R **READ RECORD**

If a record key is terminated with a] it is activated and visible equivalents interpreted when found. This sequence however allows a record to be examined since it displays the contents of the record with all visible equivalents shown as stored.

It is often used to call data onto a protected form on the screen. If an attempt is made to read a record larger than the protected form ERROR-TM is set and a (BELL) sent to the VDU.

(ESC)]S REINSTATE DELETED RECORD

When a record is deleted, LEX flags the record but does not physically delete it until the next reorganisation of the VMF. It is therefore possible to reinstate deleted records by entering (ESC)]S after the record key.

Note that (ESC)QY can be used to search through deleted keys.

(ESC)]U (ESC)>U UPDATE RECORD

This sequence is used to unconditionally update the VMF regardless of whether or not the specified record is already on the file.

It performs the same function as (ESC)]P or (ESC)>P except that when updating a record from a protected form the unprotected area is cleared with the paint character, usually the underline, as it is written away.

C.17 (ESC)+ ENTER CALCULATOR

There are two ways of entering the calculator, (TAB) into a calculator ruler ..####.##.. or use (ESC)+. LEX exits from calculator mode on receiving (TAB), (RETURN), (→), (←), (SPACE) or (ESC)E.

Before entering the calculator, it is important to ensure sufficient room to the LEFT of the cursor for the maximum number of digits that will be required. Should the calculator require more space then LEX sets ERROR-CO and sends a (BELL) to the VDU. If the overflow is the result of a calculation the field will be padded with ****.

Note that standard versions of LEX uses decimal maths and works to a maximum of 9 significant figures. For some operating systems with a floating point maths facility, then an optional version of the LEX RTS is available which allows the calculator up to 16 significant digits. This version is, however much larger than the standard. In either version LEX sets ERROR-CO and sends a (BELL) if an attempt is made to exceed the number of significant digits.

The commands CTRL U or ↑ and CTRL J or ↓ move the cursor up or down without leaving the calculator. Similarly, CTRL B and CTRL F enable the previous or next relevant numeric field to be entered with the cursor left in the correct position for entering values.

A number of options for the calculator can be set in **INST. These include the decimal marker, normally a . (dot) for the English language version. A thousands delineator, normally a , (comma) in the English version, can be set or disabled. There is also the option to automatically bracket negative numbers.

The functions that can be used whilst in the calculator mode are detailed in Appendix D. Note that memories are retained in the work file and if no longer required must be specifically deleted. The menu switch option /2 clears all the memories.

C.18 ESC= PREVIOUS ESCAPE SEQUENCE

This sequence performs a repeat of the previous escape sequence. It only works with escape sequences not control functions. Further it will not work in the calculator, or during manual hyphenation. Nor does it apply to itself!

After an error the previous escape sequence is lost and this command fails.

C.19 ESC An ESC Aa SET ATTRIBUTE

Any character in a LEX document can be flagged with up to 5 attributes. When LEX displays a character with attributes set then appropriate escape sequences from *VDUFORM3 are sent to the VDU. Likewise when these characters are printed the escape sequences defined in *PRTFORM3 are performed.

All text typed in is flagged with any of the attributes that are set until they are specifically changed or unset, or on leaving the document with ESC E.

ESC An sets the attribute n where n is in the range 1 to 5. If attributes other than n are set they remain set and if off they remain off. ESC Bn turns attribute "n" off and ESC A0 turns all attributes OFF.

ESC Aa is similar to ESC An except that a defines a combination of attributes. a may be an upper case letter in the range A to O as shown in Appendix I. All attributes are redefined by this sequence except for attribute 1. As lower case letters are translated to upper case when interpreting escape sequences, attribute 1 must be set separately by using ESC A1.

In LEX it is usual to define attribute 1 as underline and attribute 2 as embolden.

C.20 ESC Bn — UNSET ATTRIBUTE

Attributes are turned on by the command ESC An while ESC Bn turns them off. There can be up to 5 attributes, identified by n in the range 1 to 5. If n is zero then all attributes are turned off.

C.21 ESC Bx — LINE SORT

Full lines of text can be sorted into ascending or descending order depending whether x is A or D. The sort is based on the content of the text between the current cursor position and the end of the line. All subsequent lines are sorted until reaching a blank line, ruler, print control line or the end of the buffer. The lines themselves remain the same, only their order changes.

ESC BA — LINE SORT - ASCENDING

Full line sort into ascending order.

ESC BD — LINE SORT - DESCENDING

Full line sort into descending order.

C.22 ESC Bax COLUMN SORT

In this form of sort, columns and rows can be sorted into ascending or descending order. Each column must be the same number of characters long as defined by L tabs in the current ruler. The sort only considers columns on or to the right of the cursor position and rows on or below it until reaching a blank line, a ruler, a print control line or the end of the document. In the general form of the sequence the x is defined as ascending (A) or descending (D) order whilst the a orders the columns and rows horizontally (H) or vertically (V).

In the examples given below the cursor is assumed to be positioned on the a in:

```
L.........L.........L.........L..........L.........L......
0         a         l         b          2
v         5         f         7          x
8         t         r         s          9
```

ESC BHA COLUMN SORT - HORIZONTAL ASCENDING

```
L.........L.........L.........L.........L.........L......
0         2         5         7         9
v         a         b         f         l
8         r         s         t         x
```

ESC BHD COLUMN SORT - HORIZONTAL DESCENDING

```
L.........L.........L.........L.........L.........L......
0         x         t         s         r
v         l         f         b         a
8         9         7         5         2
```

ESC BVA COLUMN SORT - VERTICAL ASCENDING

```
L.........L.........L.........L.........L.........L......
0         2         9         f         s
v         5         a         l         t
8         7         b         r         x
```

ESC BVD COLUMN SORT - VERTICAL DESCENDING

```
L.........L.........L.........L.........L.........L......
0         x         r         b         7
v         t         l         a         5
8         s         f         9         2
```

C.23 (ESC)BL — SET ATTRIBUTES TO END OF LINE

As text is entered the attributes set are applied to each character as it is typed. This command sets the current attributes on existing text from the cursor position to the end of the line. Note that the spaces are attributed.

C.24 (ESC)BT — SHOW ATTRIBUTES

The command (ESC)BT displays the attributes set on each character of the current line. The attributes set on each character are shown on the line below by a visual representation. Appendix I shows the characters used to represent combinations of attributes. The display is in background mode and (RETURN) causes a return to normal edit mode removing the display.

C.25 (ESC)BU — DISPLAY ATTRIBUTES

This command shows which attributes are currently set, that is, the attributes that will be applied to every new character typed in. The display consists of a message from **TEXT and the numbers 1 to 5 if that attribute has been set. A @ indicates that an attribute is NOT set. In the following example attributes 2 and 4 are set.

```
Attribute is @2@4@
```

The indication is temporary and removed on a (RETURN).

C.26 (ESC)BW — APPEND ATTRIBUTES TO NEXT WORD

As text is entered the attributes currently set are applied to each character as it is typed. This command sets the current attributes on existing text from the cursor position to the end of the next word. Note that spaces in front of the word are attributed.

C.27 (ESC)CA — CLEAR TO END OF DOCUMENT

This sequence clears a document from the cursor position to the end of the file. Note the complementary function (ESC)CF which clears to the end of the page and differs in that text beyond the current screenful remains intact.

In protect mode it has no effect.

On some systems ESC CA blanks the entire screen. In such cases ESC QR will refresh text above the cursor position.

C.28 ESC CB CLEAR BACKGROUND OF CURRENT SCREEN

This sequence clears the background area of the screen from the cursor position; all text set as background is replaced with spaces. Text beyond the current screen is not affected. If the VDU does not have a foreground/background feature or this has not been enabled, then all characters from the cursor position are cleared.

In protect mode the background area is protected and the command has no effect.

On some systems ESC CB blanks the entire screen. In such cases ESC QR will refresh text above the cursor position.

C.29 ESC CD CHANGE DOCUMENT DETAILS

When a document is opened LEX uses the rulers, keystores and printing details as set up for the extension of the document name. Use of this command allows the details for a different extension to be applied in the current edit.

The command ESC CD initiates the **TEXT prompt New extension in background mode. Entering the required extension and RETURN changes the details applicable to the new document extension for the duration of the current edit.

Note that there is a second way of applying different document extension details to an edit session. At the Document Name prompt the required extension can be added to the document name after a slash. For example to apply the details for extension ABC to the file FILE.DOC, use

```
FILE.DOC/*.ABC
```

C.30 ESC CF CLEAR FOREGROUND OF CURRENT SCREEN

This is the complement of the ESC CB sequence. It clears the foreground area of the screen (rather than the background) from the cursor position to the end of the screen. Text beyond the screen is not affected.

ESC CTD — CONDITIONAL - DOCUMENT

This conditional checks for the existence of a document or a file.

The document or file name, terminated with a space or paint character, is entered as string1. LEX performs prog1 if the document or file exists or prog2 if it does not.

ESC CTF — CONDITIONAL - FOREIGN FILE

This conditional checks for the existence of a key on a foreign VMF file.

The key, up to 10 characters terminated with a space or paint character, is entered as string1. LEX performs prog1 if the key is found or prog2 if it is not. If the foreign file is not open for the record type then LEX assumes that the record does not exist and the fail option, prog2, is performed.

ESC CTK — CONDITIONAL - KEY

This conditional checks for the existence of a key in the current VMF.

The key, up to 10 characters terminated with a space or paint character, is entered as string1. LEX performs prog1 if the key is found or prog2 if it is not.

ESC CTN — CONDITIONAL - NUMERIC

This conditional checks that the character string under the cursor is numeric.

String1 is not used and Prog1 is performed if all characters from the cursor position to the next-space or paint character are numbers. If a letter or non numeric character is encountered then Prog2 is performed.

ESC CTQ — CONDITIONAL - QUERY

This conditional prompts the user to give a Y/N reply to a question.

LEX temporarily leaves shush mode to display the **TEXT prompt Answer Y/N-. The route depends on the user's answer to this question; prog1 is performed if Y or y is selected and prog2 if the answer is N or n. ESC E is treated as a No reply whereas ESC z or CTRL c forces LEX to abandoning and return to native edit mode. All other inputs are ignored. String1 is not used.

ESC CTS CONDITIONAL - STRING

This conditional allows string comparison.

String1 is a literal terminated by a space or paint character. Prog1 is performed if the cursor is currently on an identical string (not including the space), otherwise prog2 is performed.

ESC CT+ CONDITIONAL - ACCUMULATOR POSITIVE

This conditional checks if a value held in a calculator store is greater than zero.

String1 is a single character 0-9,A-Z representing a calculator memory. Prog1 is performed if the memory is greater than zero, otherwise prog2 is performed.

ESC CT- CONDITIONAL - ACCUMULATOR NEGATIVE

This conditional checks if a value held in a calculator store is negative.

String1 is a single character 0-9,A-Z representing a calculator memory. Prog1 is performed if the memory is less than zero, otherwise prog2 is performed.

ESC CT0 CONDITIONAL - ACCUMULATOR ZERO

This conditional checks if a value held in a calculator store is zero.

String1 is a single character 0-9,A-Z representing a calculator memory. Prog1 is performed if the memory is equal to zero, otherwise prog2 is performed.

C.34 ESC CU CLEAR FUNCTION STRING

Options such as ESC F (find), ESC G (get) or ESC X (repeat) require entry of a string of text as part of the sequence. For example in ESC F the string to be found must be defined. This is achieved by switching to background mode and issuing a **TEXT literal to prompt for the required string. Any previous string is shown following the prompt. Changing the first character in the string deletes it so that a new string may be entered. Alternatively if the first character is bypassed the string may be changed piecemeal.

ESC CU deletes the character string so that only the **TEXT prompt appears on the next occasion that a sequence of the ESC F type is entered.

C.35 ESC E EXIT

This is the general exit sequence. It is used to terminate the current activity and return to the previous one. If the current activity is editing a document, then that document is written away and LEX returns to the menu from which the edit was requested.

If LEX is displaying a menu, then it goes back to the previous one. At the top menu LEX either returns control to the monitor or has no effect depending on a **MONT setting.

C.36 ESC F FIND

The find sequence is a special case of the get sequence, ESC G, specifically designed for slow terminals. The **TEXT prompt is Find - and the found string is shown on the top line of the screen with the remainder blank. It is possible to scroll the text either way from this point even though the text is not immediately visible. A screen refresh ESC QR will bring the text back into view.

C.37 ESC G GET

This sequence initiates a text search facility. After ESC G is entered, a **TEXT prompt Search for - is shown and the system pauses to await the entry of the text string to be sought. This string is displayed in the background format and is terminated by use of RETURN or LINEFEED. LEX then searches forward through the text file, ignoring attributes, until the specified string is found or the end of the document is reached.

If the string is terminated by (RETURN), then LEX seeks any occurrence of the string regardless of the case of the characters involved and ignoring any blank characters in the text. If (LINEFEED) is used as the terminator, then LEX looks for an exact match including spaces and the case of characters. Note also that (ESC)G_(RETURN) will find a space while (ESC)G_(LINEFEED) will find an underline.

When the string is found, the last page of text searched is displayed with the line containing the string at the bottom and the cursor positioned at the start of the requisite text.

If the string prompt is answered by entering a number terminated by (LINEFEED), then LEX moves forward by the number of lines specified.

If the required text is not found in the document the user is left at the end of the file and ERROR-EB signified with a (BELL).

There are three Get enhancements to handle multiple strings. The character between the strings can be redefined in **MONT.

a. Alternative Strings

   To find the first occurrence of string1 or string2 the following syntax can be used.

   (ESC)G string1 / string2

b. Both Strings

   To find the first occurence of string1 or string2 if they are within a screen's width of each other use the following syntax.

   (ESC)G string1 & string2

c. String Starting in a Column

   The following syntax can be used to find the first occurence of string1 that starts in a column number defined by string2. The left most column is column 1 and a non numeric value in string2 will default to 1.

   (ESC)G string1 \ string2

C.38 (ESC)H DISPLAY ERROR HELP MESSAGE

Whenever LEX encounters an error condition, a (BELL) is sent to the VDU. The error message may then be viewed by entering (ESC)H. The message is displayed in background mode from the left hand side of the screen and LEX pauses for a (RETURN), (LINEFEED) or (ESC)E when the screen is restored to its earlier state. The cursor is left at the position it occupied when the sequence was called.

An option in **INST allows the error message to be displayed automatically when an error is encountered. On some VDUs it is possible to put the message in the scroll region of the screen.

A full list of error messages is given in Appendix E.

C.39 (ESC)I ENTER INSERT MODE

LEX normally operates in overtype mode where a character replaces whatever character is at the cursor position.

The (ESC)I sequence causes LEX to enter insert mode, where space is opened up for characters as they are typed in. Word wrap occurs whenever a word is pushed over the right hand margin of the current column of text so the overall shape of the text is modified.

When used in conjunction with vertical mode LEX makes space in the normal way, adds the character and then performs a (LINEFEED).

To leave Insert Mode and return to normal overtype mode, (ESC)-I may be used or the paragraph can be rejustified using (ESC)J which includes an implicit (ESC)-I. Alternatively the insert mode toggle switch (ESC)QD is available.

The insert mode will not push characters beyond a J, F or R in a ruler and, if there are no margins present, beyond a T setting. L settings are ignored.

C.40 (ESC)J JUSTIFY

This sequence reformats the current paragraph of text according to the ruler in force and, if in insert mode, returns LEX to normal overtype mode. It will only act on text between J/F...M margins. A paragraph is terminated by a completely blank line, a print control line, a ruler or a line terminated by the suppress linefeed character (usually ~).

The cursor must be positioned within or to the left of the paragraph to be justified before executing the sequence. This is of particular importance when there is more than one column on the screen. LEX moves to the start of the paragraph and blanks line by line, scrolling if necessary, until the end of the paragraph is reached. The cursor is then repositioned at the start point of the paragraph and the reformatted text displayed. The cursor is left at the end of the paragraph.

When deciding where to embed the extra spaces under a justified type of ruler, LEX tries to avoid rivers of spaces. LEX also places extra spaces after a full stop as defined in **MONT. It is usual to terminate sentences with two spaces.

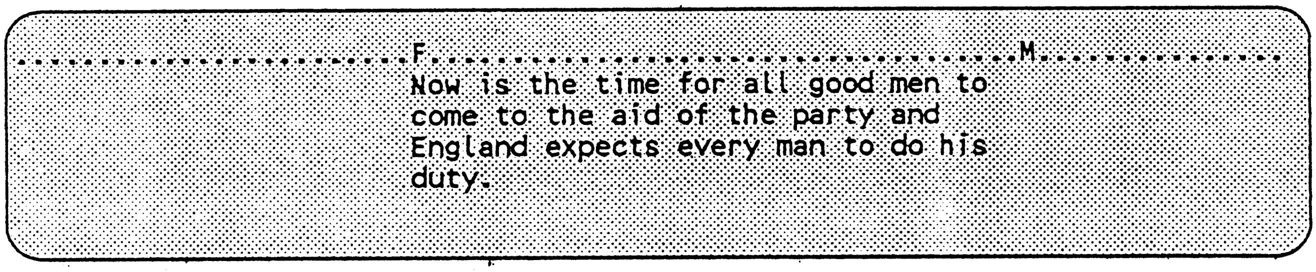
Hyphens in the text remain unless there is an H in the ruler to force the application of hyphenation rules. A - as the last character of the line is stripped. LEX then looks for a space character between the H and M in the ruler and wraps to a new line at that point. If a space is not found the word is hyphenated. Permanent hyphens should be indicated by = and these will be translated to - at print time.

Justifying text into wider and narrower columns is explained in the following examples.

a. Changing from ragged to justified right margins (or vice versa)

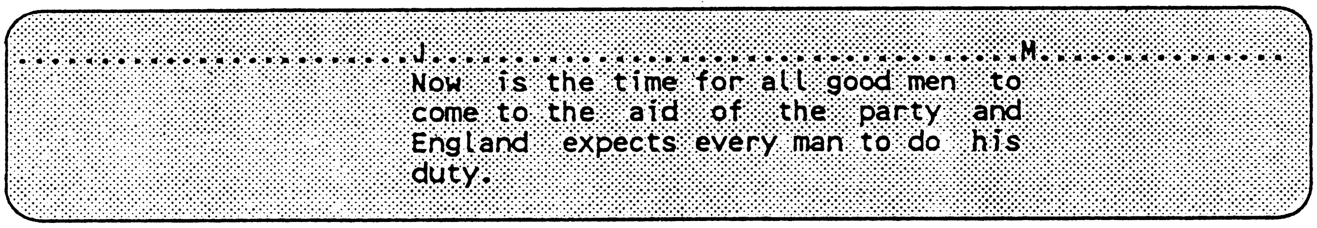
   To change the form of text from a free to a justified right margin, exchange the F to a J in the ruler and execute (ESC)J.

   Thus the text



```
.........................F..............................M..............
                         Now is the time for all good men to
                         come to the aid of the party and
                         England expects every man to do his
                         duty.
```

   becomes



```
.........................J..............................M..............
                         Now  is the time for all good men  to
                         come to the  aid  of  the  party  and
                         England  expects every man to do  his
                         duty.
```

b. Widening a Column of Text

   To widen a column of text move the relevant margin indicators to the new position and execute the sequence. The text above may be widened as shown.

```
.......................J............................................M.......
                       Now  is the time for all good men to come  to
                       the aid of  the  party  and  England  expects
                       every man to do his duty.
```

c. Narrower Margins

To make the current margins narrower, the old margins are left in place and the new ones added to the ruler. Use of the (ESC)J sequence then formats the text to the inner margins. Taking the text above and applying narrower margins results in:

```
.......................J........J.........................M.........M.......
                                Now  is the time  for  all
                                good men  to  come  to the
                                aid   of  the  party   and
                                England  expects every man
                                to do his duty.
```

d. Setting an Indent

If there is a tab point between the margins this is used on justification to set an indent. In the above example the following results:

```
.................J.....T............................................M.......
                       Now is the time for  all  good men to come to
                 the aid of the party and England expects every  man
                 to do his duty.
```

e. Justifying Multiple Columns

In multiple columns of text the justification occurs in the first column right of the cursor. Consider the following text:

```
.................F....................M.....F.......................M..
                 This is text in the        This is text which will
                 first column which         not be changed as the
                 will be changed to         rejustification will take
                 follow the modified        place in the first column
                 layout.                    of the text and not this
                                            one.
```

If the ruler is changed as shown below and ESC J executed with the cursor in column one then the following can be expected.

```
.............J........................M.....F.......................M..
             This is text in the first      This is text which will
             column   which   will   be     not be changed as the
             changed  to   follow   the     rejustification will take
             modified layout.               place in the first column
                                            of the text and not this
                                            one.
```

C.41 ESC K — DATE - YYMMDD

This produces the date in the numeric form YYMMDD. It is useful as a data item in records to allow searches on and sorts on dates. See ESC Y and ESC T for other forms of the date.

C.42 ESC L — DISPLAY THE EDIT STATUS LINE

If this is entered during document preparation, the current status is displayed in background mode temporarily overwriting text on the screen. On entering a RETURN, LINEFEED, ESC E or ESC Z the display is removed from the screen and LEX returns to normal edit mode. All other key depressions made when the status line is on the screen have no effect.

When called from a system record or keystore, typeahead is suspended and the user is prompted for one of the continuation commands before continuing with the next command in the record.

Note the similar commands ESC RL and ESC SL. The former inserts the status line in the text and the latter allows it to be displayed on the screen continuously.

The format of the display is described in Appendix G.

C.43 (ESC)Mn DISPLAY AND ASSERT RULER

The use of this sequence recalls the ruler indicated by n. Whenever this sequence is called, the ruler is entered into the text on a specially inserted blank line; text is not destroyed.

Rulers 1 to 9 can be temporarily set by the sequence (ESC)Wn but on entering a new document default rulers 1 to 5, taken from the document extension details, apply. Until set by (ESC)Wn, rulers 6 to 8 are standard 8 pitch tab settings

```
.......T.......T.......T.......T.......T.......T.......T.......T..
```

and ruler 9 is all dots meaning that no ruler settings are in force.

```
..................................................................
```

Recalling ruler 0 is a special case that displays the ruler currently in force.

When a document is opened the default ruler 1 is shown. If no document extension details are found for this document type then the ruler is taken from the document extension details of the default document type.

C.44 (ESC)Ma MERGING TEXT

A set of sequences allow text to be merged from another document into the current document. They take the form (ESC)Ma where the character a defines the merge option. Note the complementary sequence (ESC)Wa which merges FROM the current document into another. In LEX the latter is referred to as an extract operation to distinguish it from the merge described here.

Before any merge can take place it is necessary to open the merge document using (ESC)MO. If any of the other merge options are used when a merge document has not previously been opened then ERROR-MC is set and a (BELL) sent to the VDU.

(ESC)MA MERGE ADVANCE

This sequence is used to advance through the opened merge document until a specified string is encountered, no text is merged into the current text.

When entered, LEX prompts for a text string to find and displays any previously entered string in the background format. An exact match of the specified text is required if the string is terminated with (LINEFEED), whereas the search is caseless if the string is terminated with (RETURN).

LEX displays the line number of the merge document in background format whilst the search is in progress. If the text is not found then ERROR-ME is set and a (BELL) is sent to the VDU. The file must be re-opened if any further text is to be found or inserted.

If a second search request is made then LEX ignores the text currently in the merge buffer and looks for the next occurrence of the string.

An alternative use of this sequence is to enter a number followed by (LINEFEED). In this case LEX advances through the merge document the specified number of lines.

(ESC)MC **MERGE CLOSE**

(ESC)MC closes the current merge file.

(ESC)ME **MERGE TO END**

This sequence merges from the current position within an opened merge file until the end of the merge file. All the text lines are inserted before the line containing the cursor.

When merging comes to an end ERROR-ME is set and a (BELL) sent to the VDU.

(ESC)MI **MERGE LINE**

Use of this sequence enables the insertion of one line of text from the merge file. The text is merged into the document on the line before that in which the cursor is positioned. The pointer in the merge file is advanced by one line after the execution of this sequence.

(ESC)MJ **MERGE SELECTED LINES**

This sequence merges lines of text which contain a specified string. The lines are inserted above the line on which the cursor is positioned.

When the sequence is typed, LEX prompts for a text string which may be terminated with RETURN or LINEFEED depending on whether or not the case of the string and embedded blanks are significant. Only those lines in the merge file containing the specified text are inserted into the current file.

ESC MM — MERGE ENTIRE DOCUMENT

This sequence merges an entire document into the current document. It is equivalent to using ESC MO document name RETURN followed by ESC ME and therefore does not require a specific ESC MO prior to its use.

At the Document name prompt a LINEFEED will bring up the next file name in the document index.

ESC MO — MERGE OPEN

Before any merge it is necessary to open a merge document. ESC MO opens an existing file for merging into the document file currently being edited. It then prompts for the required file name. A LINEFEED at this prompt will bring up the next file name in the document index.

If no device is specified with the file name, then LEX searches for the file on the default device as specified in **INST. If the free format indicator is set in **INST, then it is possible to include device and/or account specifications (depending on the operating system) for the required file.

If the specified file is not found, ERROR-MF is set and a BELL sent to the VDU. If an error occurs when opening a file from within a system record or keystore, no further processing occurs in that record and LEX returns to its native edit mode.

A repeat of the sequence causes the current file to be closed and another to be opened. ESC RG displays the name of the merge file opened with this sequence.

ESC MS **MERGE SKIP**

When a document has been opened for merging into the file currently being edited, the current line of the merge document is held in the merge buffer. This escape sequence allows this buffer to be skipped so that any further merging activities will start on the next line of the merge document.

ESC MU **MERGE UNTIL**

When this sequence is entered LEX pauses to allow the user to enter a string. LEX then merges from the currently open merge file until the specified string is found. If the end of the file is met before the specified string, then ERROR-ME is set and a BELL sent to the VDU.

If the user enters a number followed by LINEFEED, instead of the string then LEX merges the relevant number of lines from the merge document.

ESC MV **MERGE VIEW**

This sequence causes the next line from the merge document to be displayed without it being inserted into the user's document. The user can then verify that the line of text in the merge file is the required line for merging into the current document.

The line is a temporary display in background mode and is removed by RETURN, LINEFEED, ESC E or ESC Z.

C.45 ESC MZ **INSERT TAB IN RULER**

This command modifies the ruler in force by placing a new tab mark in the ruler at the cursor's current column. The tab mark put at this position is the same as the next mark to the left. Note that the modified ruler is not displayed or stored but can be viewed by ESC MO.

The reverse, ESC -MZ replaces the nearest tab to the left of the cursor with a . (dot).

C.46 ESC N — NEXT KEY

ESC N causes LEX to display the next key on the VMF after the last key referenced. If ESC N is used within a repeat loop then the loop stops when there are no more keys. Note the previous key can be obtained by using ESC -N.

C.47 ESC Ox — OPEN DOCUMENT FROM PLAYGROUND

These sequences can only be called from the playground. They open a document into which the current contents of the screen buffer, plus any further additions to that buffer, are to be saved. The character x can be D, F or X and define the mode of opening the file.

When the sequence is entered, LEX prompts, in background mode, for the name of the file to be created. Any valid filename may be used terminated with a RETURN. The file will be created on the default device as specified in **INST unless free form file names have been specified in which case such details may be included. Note that default extensions for filenames are not applied to the name.

If an error occurs when opening a file from within a system record or key store, no further processing occurs in that record and LEX returns to its native edit mode.

Note that the entire buffer is passed to the file; if only certain lines of text are required then the ESC Wa sequences should be used.

ESC OD — OPEN DOCUMENT

This option creates a file unconditionally and an existing file of the same name will be overwritten. The index record in the VMF is created or its details updated as appropriate. If write access to the new file is denied ERROR-MW is set and a BELL sent to the VDU.

ESC OF — OPEN DOCUMENT

This sequence has the same effect as ESC OD. It was introduced as an alternative to ESC OD which is generated by the left arrow key on a VT100. If write access to the new file is denied ERROR-MW is set and a BELL sent to the VDU.

ESC OX **CREATE NEW DOCUMENT**

The ESC OX option works in a similar way to ESC OO except that an existing file will not be overwritten. If the file already exists, ERROR-MA is set and a BELL sent to the VDU. ERROR-MW is set and a BELL is sent to the VDU when access to the file is denied. Successful opening of the file causes an index record in the VMF to be created with the appropriate details. If ESC OX is executed from a program in a system record or keystore and an error occurs then the program terminates.

C.48 **ESC P** **PAGINATION - NEXT PAGE**

The ESC P sequence is be used during edit, to insert a new soft page break (.PP line) at the place in the document where the automatic print-time pagination would put it.

When the function is invoked, LEX looks back for the previous .P or .PP line. It then counts forward the appropriate number of lines as defined in the document extension details and applies orphan/widow factors. When counting it takes into account the line spacing defined in .S lines and non print lines such as rulers, lines ending with the suppress line feed character (~), and print control lines (.C, .T, .F etc).

During its downward count, if it comes across any .PP lines already in the document, it deletes them and, if it encounters a .P line before reaching the end of the count then it stops at that point. At the point where LEX considers a page break should occur it inserts a new .PP line.

The .PP line can be Cut & Paste to move it up or down to a more desirable position. If the .PP is moved down then care must be taken not to move it too many lines down beyond the physical size of the paper.

The .PP line is described in Appendix H.

Until a .PP line is found in a document, LEX assumes automatic pagination with breaks determined by the page length and orphan/widow factors specified in the document printing details. For manual pagination it is therefore essential that a .PP line appears as the first line of the document. Further, if a document is to be manually paginated then, once a .PP line appears in the document, it is essential that the remainder of the document be paginated.

C.49 ESC QA — COPY LINE ABOVE

This is used to overwrite the current line from the cursor position to the end of the line, with the characters from the line above. In protect mode it copies into unprotected areas both protected and unprotected characters from the line above. Any characters overwritten using this technique may be recalled by the sequence ESC -V as though they had been deleted by CTRL V.

C.50 ESC QB — COPY LINE BELOW

ESC QB overwrites the current line from the cursor position to the end of the line with the characters from the line below. In protect mode it copies into unprotected areas both protected and unprotected characters from the line below. Any characters overwritten using this technique may be recalled by the sequence ESC -V as though they had been deleted by CTRL V.

C.51 ESC QC — SWAP CHARACTERS

This is used to swap the character at the current cursor position with the next character on the line. For example, eg would become ge.

C.52 ESC QD — INSERT MODE

ESC QD is a toggle switch to enter and leave insert mode. It is complementary to ESC I and ESC -I which always leave LEX in and out of insert mode respectively.

C.53 ESC QE — END OF DOCUMENT

ESC QE causes LEX to advance to the end of the document currently being edited. The last screenfull of text in the document is displayed. It is equivalent to entering ESC 9999 LINEFEED but differs in that whilst searching it displays the **TEXT message `End page number`.

C.54 ESC QF — NEXT PARAGRAPH

This sequence moves the cursor forward to the start of the next paragraph.

C.55 ESC QG BACK PARAGRAPH

This sequence moves the cursor back to the start of the current paragraph or to the start of the previous paragraph if the cursor is already at the start of the current one.

C.56 ESC QH MANUAL HYPHENATION

The sequence ESC QH is used if manual hyphenation is required. When the sequence is typed under a J or F and M ruler, LEX positions the cursor at the end of the current line and awaits input.

The following sequences and functions are available at this point:

↑ CTRL U	Move up one line
↓ CTRL J LINEFEED	Move down one line
ESC J	Justify from this position - note that the sequence ESC QJ does not work at this point.
CTRL N	Push text onto the next line. The text is pushed over character by character and a hyphen is introduced whenever a word is split. If the following line is already full ERROR-HF is set and a BELL sent to the VDU.
CTRL E	Similar to CTRL N, this pushes text onto the next line character by character. If however the following line is full instead of registering an error, the remainder of the paragraph is re-justified.
CTRL P	Pull text back from the next line. The text is pulled up character by character and a hyphen introduced whenever a word is split. If there is no more room on the current line ERROR-HC is set and a BELL sent to the VDU.
I	This is used to show where the system would break the word according to its inbuilt algorithm. This only works if one of the relevant algorithms have been chosen in **INST.
RETURN ESC E	These are both used to exit from hyphenation mode. The cursor is left at the start of the line on which the command was issued.

On rejustification, LEX attempts to remove soft hyphenation marks by removing the hyphen and closing the word up. If a hyphen is essential then a hard hyphenation mark may be specified in **INST or, more usually, in a .C command line. In either case, the ninth position will specify the character to be used. For example,

```
.C ........=..
```

causes the = to represent a hard hyphen. This character is translated to a normal hyphen at print time. WARNING: do not use a - (hyphen) to represent hard hyphen character, it may be lost on rejustification.

C.57 (ESC)QI SHOW HYPHENATION

This command shows, in background mode, the hyphenated part of a word following the **TEXT message Hyphened - . The display is removed by a subsequent (RETURN).

C.58 (ESC)QJ PARTIAL JUSTIFY

(ESC)J justifies the entire paragraph in which the cursor is lying. (ESC)QJ on the other hand justifies a paragraph from the current line to the end of the paragraph.

C.59 (ESC)QK LEFT EDGE OF CURRENT LINE

This sequence moves the cursor to the left most position on the current line.

C.60 (ESC)QL LAST CHARACTER ON SCREEN

(ESC)QL moves the cursor backwards or forwards to the first space beyond the last character on the screen. In forms mode it moves to the last unprotected character.

C.61 (ESC)QM TOP DOCUMENT

This sequence takes the cursor back to the top of the document. If the entire document does not fit into the work file, then the command goes back as far as possible within the work file.

In a protected screen, the protected areas below the cursor are reset to the underline paint character. This is a rapid way of clearing a protected screen after recalling a record or after entering unwanted information.

On some systems ESC CF blanks the entire screen. In such cases ESC QR will refresh text above the cursor position.

C.31 ESC CP CLEAR PROTECTED AREA AND PAINT CHARS

This is used in a protected screen to clear all the protected areas of a screen from the start of the line containing the cursor. The protected area is replaced with blanks as are any paint characters, usually the underline, in the unprotected areas. The protection is then removed, enabling the user to enter text anywhere on the screen from the cursor onwards.

The purpose of ESC CP is to allow text to be transferred from a fixed record format into a layout suitable for printing onto pre-printed stationery where the stationery has the same format as the screen.

If this sequence is used when not in protect mode then all paint characters are replaced with spaces.

C.32 ESC CS CLEAR AUTOMATIC EDIT STATUS LINE

The edit status line can be continuously displayed on ANSII standard screens that have a scrolling region specification. In such cases this command removes the status line from the screen until the next edit or ESC SL. LEX operates faster when the status line is not displayed.

C.33 ESC CTx CONDITIONALS

In LEX programs it is possible to make LEX branch depending on the outcome of certain tests. The conditions that can be set are defined by x and described below. The general syntax is:

ESC CTx string1 SPACE prog1 RETURN prog2 RETURN

The sequence can only be used when performing a VMF record or keystore and the input of parameters is always done in shush mode. String1 and prog1 combined can be up to 78 characters long and prog2 can be up to 78 characters long.

C.62 ESC QN — TIME

Provided that the time was set up on initialisation, this sequence displays the current time in the form HH:MM:SS. The separator : is the default but can be re-defined in **MONT.

C.63 ESC QR — REFRESH SCREEN

After using VDU local modes, or, in some instances, partial clear screen options, it may be necessary to re-display the screen.

This sequence causes the entire screen to be rewritten and the cursor returned to its starting position. In protect mode relative positioning is correct but no attempt is made to reset any foreground/background attributes. Characters entered in shush or shut up modes are displayed.

C.64 ESC QS — SPAWN

The sequence is used to call other programs from within LEX and to pass information through to those programs. It starts with a **TEXT prompt QS command - where a system record key should be entered. On some systems LEX is then dumped as a memory image onto disk and the relevant program is loaded into memory for execution within the same environment as LEX itself. On other systems a fork and execute is performed on termination of the associated program, LEX is reactivated at the same point as it was before calling this sequence.

Before the call is activated, it is necessary to set up a system record containing information to be passed through to the called program. This system record must contain the full filename of the program to be called in the first 40 bytes of the record following the key. The remainder of the record may contain information used by the called program.

Where a dumping version is used, sufficient space must exist on the disk for the LEX dump or the call will fail. On reinstating LEX the temporary file is removed from the disk and its directory.

In the dumping version of LEX programs called using ESC QS must either be written in D, the language in which LEX is written or, if they are written in other languages they must conform to a strict format. Programs not in these categories should be called using the chain menu switch function, /G.

C.65 (ESC)QT FORK TO EXTERNAL PROGRAM AND RETURN

This is used as an on line forking option. LEX prompts for a string which the operating system will recognise, via the **TEXT prompt QT command -. LEX is then dumped on disk, where relevant and control is transferred to the operating system where the string is executed. On completion LEX issues the **TEXT prompt Press any key to continue after which LEX is reloaded to continue from where the (ESC)QT call was made.

It is a useful function to run such operating system utilities as Directory from a menu.

Note that (ESC)QW is identical except that it does not prompt Press any key to continue before returning to LEX.

C.66 (ESC)QU END OF PREVIOUS WORD

This causes the cursor to be positioned at the end of the previous word. If the cursor is on the first word on the screen when the sequence is executed, then the cursor is left at the top left hand corner of the screen; no attempt is made to scroll the screen back to the previous line.

C.67 (ESC)QV END OF NEXT WORD

This moves the cursor forward to the end of the next word on the screen. If the cursor is currently at the end of the last word on the screen then it does not move.

C.68 (ESC)QW FORK TO EXTERNAL PROGRAM AND RETURN

This is used as an on line forking option. LEX prompts for a string which the operating system will recognise, via the **TEXT prompt QT command -. LEX is then dumped on disk, as above, and control transferred to the operating system where the string is executed. On completion LEX is reloaded to continue from where the (ESC)QW call was made.

Note this function is identical to (ESC)QT except that (ESC)QT prompts with a pause message before returning to LEX.

C.69 ESC QX RE-EDIT TO CURRENT POSITION

ESC QX effectively performs an ESC V followed by an ESC Gnnn to the line on which it was activated. It is therefore a useful means of storing a partially completed document with the current update and then continuing from the same place.

C.70 ESC QY NEXT DELETED KEY

When a record is deleted from the VMF, LEX simply flags the key but does not physically delete it until the next reorganisation of the VMF. The function ESC QY searches the keys in sequence and displays the key of the next deleted record. It is possible to reinstate deleted records using the ESC]S sequence.

C.71 ESC Rn LOAD KEYSTORE WITH LAST KEYSTROKES

This sequence recalls the last characters typed on the keyboard and stores them in the key store n. If control characters or escape sequences are to be included in the string then these are converted to their visible equivalents and the actual number of keystrokes stored is consequently reduced.

Whilst this function is a useful technique for rapidly generating simple programs in the keystores, care must be exercised as it is usually necessary to modify the string displayed. For example it may be necessary to reduce the size of the keystore content, and characters such as] may need conversion to >.

C.72 ESC RC CLEAR CHARACTER STORES

ESC RC clears the character store as used in ESC RS, ESC Rn, etc.

C.73 ESC RD DOCUMENT NAME

ESC RD displays the current document name in full including all device/account details. e.g. B:TECH44.MAN

C.74 (ESC)RG MERGE FILE NAME

If a merge file has been opened, using (ESC)MD, this sequence displays the Filename in background mode. (RETURN) removes the display from the screen. Note that it does not display the name of any extract files opened with (ESC)WD.

C.75 (ESC)RI INSTALLATION PARAMETERS

(ESC)RI displays, in background mode, the **TEXT prompt Installation Parameters followed by the record names of the current **MONT, **INST and **PNST keys respectively. (RETURN) removes the display from the screen.

C.76 (ESC)RK PERFORM SYSTEM RECORD

The usual way to recall a system record activating any visual equivalents is to enter [kkkkkkkkkk followed by], where kkkkkkkkkk represents the key. All characters appear on the screen and may overwrite characters already on that line. Even in shush mode the underlying text is changed. Silent activation of a record may be achieved by use of (ESC)RK with the following syntax.

(ESC)RKkkkkkkkkkk(RETURN)

It also works with partial memory file reads.

(ESC)RKkkkkkkkkkk+1/36(RETURN)

The sequence (ESC)RK prompts for the key with the **TEXT message Key is...

C.77 (ESC)RL INSERT EDIT STATUS LINE IN DOCUMENT

(ESC)L puts the status line on the screen as a temporary display in background mode and it is removed on a subsequent (RETURN). (ESC)RL has a similar function except that the status line is inserted permanently in the text on the screen. The contents of the line are as described in Appendix G.

C.78 (ESC)RM VMF NAME

(ESC)RM displays the current VMF filename including all device/account details. The filename is displayed at the cursor position overwriting existing text.

C.79 ESC RN — DOCUMENT KEY

ESC RN displays the normalised document name as used in the key for index entries. eg. :TECH44MAN

The details are displayed at the cursor position overwriting existing text.

C.80 ESC RO — CURRENT MENU SWITCH OPTION

ESC RO displays the current menu switch option and any associated key. The option letter and key are displayed at the cursor position overwriting existing text.

C.81 ESC RS — RECALL LAST KEYSTROKES

This sequence displays on the screen the last characters typed at the keyboard. The recall starts on the left hand side of the line from which the sequence was called and overwrites the characters already on that line. A maximum of 78 characters can be displayed so, if pure text has been typed in, then the last 78 characters are shown. If any control characters or escape sequences have been entered then these are converted to their visible equivalents and the actual number of keystrokes stored is consequently reduced.

The sequence is similar to ESC Rn which loads an identical string into keystore n.

C.82 ESC RV — VDU KEY

ESC RV displays the current VDU key, e.g.)PCUK

The information is entered at the cursor position overwriting existing text.

C.83 ESC RW — WORK FILE NAME

ESC RW displays the current work file name. The filename is displayed at the cursor position overwriting existing text.

C.84 ESCs(n MARK POSITION ON SCREEN

This sequence marks a point in the text for subsequent use of ESC<n to return the cursor to this position. Ten marks are possible where n is 0 to 9. Position 0 is automatically set by using the MARK for cut and paste. i.e. ESC(.

ESC<n will only find a marked position if it is on the screen; LEX will not scroll backwards or forwards to find a marked position. It is, however, possible to scroll a marked position off the screen and then back on the screen before returning to it.

Note that it is the position which is marked and not an item of text. If lines are deleted or the text re-justified before the return call is executed, the cursor returns to the marked position on the screen, not necessarily the same position in the text.

The cursor will return to the home position if the number specified in the return call has not been set.

C.85 ESCSn SET KEYSTORE

This sequence allows the contents of keystore n to be viewed or changed. n denotes the keystore in the range 0 to 9.

On executing the sequence the contents of the keystore are displayed in background format. If a character is entered over the first displayed character, then the entire keystore is cleared and a new string may be entered. Overtyping any other character will alter the one character only and thereafter the contents of the keystore may be changed piecemeal.

All entries to the keystore are terminated by RETURN when the display will be removed from the screen.

C.86 ESCSB SET BACKGROUND

The use of this sequence sets the VDU into background mode as specified in *VDUFORM. This is achieved by sending to the VDU the codes specified in the Set BG field of *VDUFORM. It is normally used in form layouts for record creation and has no significance in an ordinary document.

LEX normally operates in foreground except when in protect mode when the protected part is in foreground and unprotected part in background.

To return to foreground mode the command ESCUB is used.

C.87 ESCSD — SET DELAY FACTOR

If the sequence ESCSD is entered LEX displays the current value of the VDU delay factor as set in the field Delay in *VDUFORM. RETURN leaves the delay factor unchanged whereas any number followed by RETURN resets the delay factor.

On heavily laden systems a temporary change of delay may prevent the appearance of spurious characters due to missed escape sequences from function or arrow keys. Excessive values for the delay factor will noticeably slow the response, particularly when using arrow keys; it is unwise to put the value much above 100.

C.88 ESCSE — SHOW ERROR MODE

When an error occurs LEX sends the BELL character to the terminal.

Then, if show error mode has been set using ESCSE, the appropriate error message is displayed. The message is displayed at the current line containing the cursor or, if enabled in *VDUFORM3, in the status line.

If show error mode has not been enabled then ESCH must be input to cause error messages to be displayed.

C.89 ESCSF — SET FOREIGN FILE

This sequence enables LEX to access records on files other than the current VMF. The format of the sequence and special characters associated with it is given below. Note that the filename may be any VMF file and may include account/device details.

ESCSFabcdFilename

The characters at positions abcd have the following significance.

a The first associated character represents the type of record. That is the first character of the record key.

b The next character gives the file type. It must be a V as the option only works on a VMF type data base.

c The third character can be set to R if read only access is required. Anything else assumes read/write access.

d Finally the last character may be M for multi user or N for single user resulting in the file being opened in the appropriate mode. If the character is neither N nor M then the system standard taken from **INST is applied.

After the sequence is entered any request to read or write a record of type a causes LEX to read the record from the foreign file and not from the current VMF. This applies to all explicit requests such as [key] or use of the (ESC)]a sequences.

There are a number of safeguards to prevent problems when writing records to foreign files. Use of an R for position c allows read access only. In single user mode only one user at a time is allowed write access to a file. When the character at position c is other than an R then an error results if the foreign file is already in use. With multiuser systems there is no restriction on gaining access to a foreign file but before a record can be written it must first be locked against other users with (ESC)]L.

Use of this function causes buffer space to be allocated within the user's work file and may lead to confusion if the foreign and current VMFs have records of the same type. It is therefore recommended that the foreign file be closed as soon as it is no longer required. Sequence (ESC)UF and menu switch option /2 both close foreign files.

If an error occurs when opening the file from within a system record, no further processing will occur from that record and LEX returns to its native edit mode.

C.90 (ESC)SG IGNORE RULERS

If a ruler is inserted in a document, it normally applies to all text below it until another ruler is reached. When a new ruler is encountered it then automatically becomes the current ruler.

This sequence forces LEX to ignore the automatic assertion of rulers with the current ruler remaining in force. Note the complementary functions:

(ESC)-SG	is used to leave ignore rulers mode
(ESC)Un	is used to activate ruler n. The ignore rulers setting is not changed. The new ruler is not displayed.

ESC UUn is used to activate ruler n. The ignore rulers setting is switched on, by an implicit ESC SG. The new ruler is not displayed.

ESC Mn is used to display and activate ruler n at the same time forcing LEX out of ignore automatic assertion of rulers state by performing an implicit ESC -SG

ESC MO can always be used to display the current ruler.

C.91 ESC SH SHUSH MODE

In shush mode LEX does not display characters as entered and the cursor remains in its original position. LEX accepts the characters and functions as normal, they do not appear on the screen. The sequence is usually used to prevent the display of text such as passwords in the middle of a system record.

LEX automatically leaves shush mode on returning to a menu but the sequence ESC -SH may also be used. In the latter case, the cursor catches up and jumps to where it would have been if the characters had been displayed. The invisible characters may be reinstated after leaving shush mode by refreshing the screen with ESC QR

Note the similarity with sequence ESC SU. In shutup mode, however, all menus and prompts are shushed.

C.92 ESC SL SET AUTOMATIC EDIT STATUS LINE

This feature is only available on VDUs with a scroll region feature.

The status line described in Appendix G can be displayed on the screen permanently by setting up the option in **INST and *VDUFORM. Use of ESC SL sets the feature on if not set in **INST.

As this feature slows LEX considerably it is wise to use it only when necessary. ESC -SL turns the permanent display off.

C.93 (ESC)SM SET MASK

Normally when reading data from a record the entire record is displayed sequentially from the key to the end. Use of the set mask feature allows parts of a record to be displayed and updated. It is particularly useful for records which occupy more than one screen or when only part of the record is to be updated.

Before a mask can be set a special record must be created using *MASKFORM. (See Chapter 12) The contents of this record describe the parts of the data record that are to be displayed, the order in which they are to appear, and whether any fields are read only.

A mask is activated by the sequence (ESC)SMmaskname. Reading and writing is then in accordance with the mask set until returning to a menu or the sequence (ESC)-SM is sent. Note that masks are for data only so visible equivalents are never translated.

Masking is designed to work in protected forms so it is usual to activate the the appropriate form before the (ESC)SM. Masking may be used in normal edit mode but great care must be exercised as trailing spaces are removed from each line; it is therefore advisable to ensure the squash character is set to (UNDERLINE) and that this is done only if absolutely necessary.

C.94 (ESC)SP SET PROTECT

This sequence is used to initiate a protected area on the screen. Its prime use is in setting protected areas within forms.

The sequence protects all areas of the screen which do not contain an underline character so that text may only be entered on the unprotected - underlined - part of the screen.

Records can be read onto the unprotected screen and when creating or updating records only text in unprotected areas is written to the VMF.

On activating the sequence an implicit (ESC)SB is performed so, whilst the protected area is in foreground mode, data entered on the unprotected screen is displayed in background mode.

Protect mode can be unset by returning to a menu with (ESC)E or by use of (ESC)UP.

C.95 ESCsQa SET SQUASH CHARACTER

To save space in the VMF, records are stored with certain characters squashed. Strings of the nominated squash character are replaced by a byte containing a count of the number of characters squashed in the range 1 to 32. No more than 32 characters may be squashed into a byte, longer strings require more than one squash byte. When a record is read from the VMF bytes with the values less than decimal 32 are replaced by the requisite number of squash characters.

The default character for squashing is the underline and this is re-established on returning to a menu. To specify a different squash character ESCsQa may be used where a is the character to squash. Note that ESCsQ at the end of a line will set the squash character to RETURN and not space.

The character used for squashing when writing a record need not be the same when the record is recalled. For example, if

```
Ace Microsystems Ltd____________________
```

is written into a record with the default squash character and ESCsQ+ performed, then the recalled display will be

```
Ace Microsystems Ltd ++++++++++++++++++++
```

As records are normally written with underline as the squash character it is common to use ESCsQSPACE prior to reading records into a document. In this way the underline characters do not appear in the final text.

When a record is activated, e.g. [kkkkkkkkk], visible equivalents are interpreted and LEX always applies underline as the squash character. It is therefore essential that system records are stored with the default squash character underline.

C.96 ESCssn SEND SEQUENCE

On *VDUFORM2 there are ten definable display sequences, 0 to 9. ESCssn causes the sequence n as shown in *VDUFORM2 to be sent to the VDU. They can be used, for example, to switch to and from graphic character sets where the VDU has such a facility.

C.97 ESC STn — SEND SEQUENCE

This sequence causes one of the escape sequences defined in *VDUFORM4 to be sent to the VDU. Up to 10 escape sequences may be defined and they are sent automatically at different points in LEX, for example when entering edit mode or when exiting from LEX. The sequences may also be sent by issuing ESC STn where n is 0 to 9.

C.98 ESC SU — SHUTUP MODE

Shutup is a form of super shush mode. With shush mode, ESC SH, edit functions and text can be prevented from appearing on the screen. Shutup goes further; making menus and prompts quiet as well.

ESC -SU can be used to turn the mode off. Alternatively menu switch option /2 includes a shutup reset. Note that /2 can be used even if the menu is not visible due to being in shutup mode.

C.99 ESC SV — SEND TO VDU

This sequence results in a string of characters being sent directly to the VDU. The format is:

ESC SV DELIMITER string DELIMITER

where DELIMITER may be any character. After the first delimiter every character entered in the string is sent to the VDU until the same delimiting character is received again. For example, if the string

ESC SV. ESC [?5h.

is sent to a VT100 then all subsequent characters will appear in inverse video.

C.100 ESC SW — SET PAGE WARNING MODE

This sequence puts LEX into page warning mode. On every RETURN or LINEFEED LEX decides whether the next line could fit on the current page or not. It does this by looking back for the last .P or .PP line and counting forward allowing for control lines that are not printed, line spacing, and suppress line feed characters. If this count is greater than the page length set in the document extension details then ERROR-EP is set and a BELL sent to the VDU.

Note that this facility slows the operation of LEX. The mode is turned off by ESC-SW. It can also be turned on automatically on entering a document by selecting the option in **INST.

C.101 ESC SX — DISPLAY NUMERIC VALUE

This displays the numeric value of any character typed in at the VDU. Its main use is to analyse the escape sequences generated by function keys and arrow keys. The format is:

ESC SX DELIMITER KEY KEY ... DELIMITER

The delimiter may be any character and is entered immediately following the sequence. Any subsequent key depression until the same delimiter is received is displayed on the left hand side of the screen as the decimal value of the character typed. If a key generates a sequence of characters then the appropriate sequence of decimal numbers is displayed.

The characters are not stored in the edit document and are removed from the screen by ESC QR.

C.102 ESC T — DATE - SHORT FORM

Use of this sequence displays the system date in the form 01-Nov-83. It is the abbreviated form of the date, the exact format of which may be specified in **MONT.

C.103 ESC Un — USE RULER

If a ruler is inserted in a document, it normally applies to all text below it until another ruler is reached. When a new ruler is encountered it becomes the current ruler.

There are a number of commands, including ESC Un, which affect the normal assertion of rulers. ESC SG and ESC-SG allows LEX to ignore the automatic assertion of embedded rulers and to turn the facility back on respectively. ESC UUn asserts ruler n and at the same time performs an implicit ESC SG.

ESC Un forces LEX to use ruler n without the ruler being displayed and without affecting the ignore ruler state. Therefore, if the ignore rulers command is NOT SET, an embedded ruler will take over automatically as soon as the cursor is moved through the ruler. Conversely, if ignore rulers is SET then ruler n applies and the only way to change it is to leave the ignore rulers mode or by another ESC Un.

The number n can be any ruler number 1 to 9. ESC U0 has no effect as the ruler 0 is always the current ruler.

Besides setting rulers where the ESC SG option has been used, the ESC Un facility is useful when it is important that the ruler does not appear in the source text, for example in computer programs.

C.104 ESC UB — UNSET BACKGROUND

ESC UB unsets background mode by sending a sequence of commands to the VDU which are defined in *VDUFORM. See ESC SB for set background.

C.105 ESC UF — UNSET FOREIGN FILE

This sequence is used to close the foreign file and release the internal buffer space for other usage. It is the inverse of the ESC SF sequence to access foreign files. The menu switch option /2 includes an implied ESC UF.

C.106 ESC UM — UNSET MASK

The unset masking sequence clears any mask in force. No error is given even if masking was not previously in force. Masks are also unset on an ESC E returning to a menu. See ESC SM for set mask and the system record *MASKFORM in Chapter 12.

C.107 ESC UP — UNSET PROTECT

The unset protect sequence clears the protection from any areas on the screen thereby allowing the user to type over any part of the screen. It is the inverse of ESC SP.

C.108 ESC UUn **USE RULER**

ESC UUn is complementary to the ESC Un function. It forces LEX to use ruler n without the ruler being displayed and at the same time forces LEX to ignore the automatic assertion of embedded rulers. i.e. it is ESC Un followed by an implicit ESC SG. ESC UU0 will cause the current ruler to be obeyed until countermanded by another ruler assertion sequence.

C.109 ESC V **RE-EDIT TO TOP OF DOCUMENT**

This is used to re-edit a file returning the user to the start of a document currently being edited

In the course of performing the return to the start of the document, LEX closes off the temporary output file and performs the necessary file renaming of the backup files as if the sequence were ESC E followed by a request to edit the same document.

This sequence is most commonly used to save a copy of a document during editing or to return to the start of a document if the work file is too small.

C.110 ESC Wn **TEMPORARILY STORE RULER**

Positioning the cursor anywhere on a ruler and entering ESC Wn causes the ruler to be written to ruler store n. It may then be recalled by the sequences ESC Un or ESC Mn.

This function is useful if the ruler is to be used later in the document being edited. It must be remembered, however, that on leaving the document rulers 1 to 5 changed in this way will be lost as entering a document resets the default rulers set up by the document extension details. Rulers 6 to 9 remain in the new form until a new workfile is used when the system defaults are set.

C.111 ESC Wa **WRITE OUT TEXT**

There are a series of sequences which handle output merging whereby text from the current document may be extracted to another document or device. LEX uses the term write out for this operation to minimise confusion with merge. Merge sequences, ESC Ma, merge text into the current document, write out copies text out of the current document.

When used from the playground this feature is a useful way to take a documentary copy of the normally transient work performed while in the playground.

Before an extract can take place it is necessary to create the extract document using (ESC)WO. If any of the other write out options are used when an extract document has not previously been opened then ERROR-WF is set and a (BELL) sent to the VDU.

(ESC)WC WRITE OUT CLOSE

The use of this sequence will close the output file opened by (ESC)WO. Another file may then be opened.

Note that the output file is closed automatically if (ESC)E is used to return to a menu.

(ESC)WE WRITE OUT TO END

This causes all text from the current cursor position to the end of the document being edited to be sent to the extract file.

(ESC)WJ WRITE OUT SELECTED LINES

This sequence causes lines of text which contain a specified string to be sent to the extract file.

When the sequence is typed, LEX prompts for a text string which may be terminated with (RETURN) or (LINEFEED) depending on whether or not the case of the string and embedded blanks are significant. Only those lines in the current document containing the specified text are output to the extract file.

(ESC)WO WRITE OUT OPEN

Before any extract operation it is necessary to open an extract document with (ESC)WO.

On issuing the command LEX prompts for a document name, the prompt and any input string being shown in background mode. A (RETURN) opens the named file. A (LINEFEED) at this prompt returns the next file name in the document index. If an extract file already exists, ERROR-OF is sent and a (BELL) sent to the VDU.

Note that the default document extension is not applied. The device details are shown after the prompt depending on the options selected in **INST. By changing the device is possible to output directly to a physical device such as a printer.

If an error occurs when opening a file from within a system record or keystore, no further processing occurs in that record and LEX returns to its native edit mode.

ESC WU **WRITE OUT UNTIL**

This sequence copies into the extract document all lines from the current document until a line containing a specified string is reached. The line containing the string is also copied into the document.

When entered LEX prompts for a string. Any previously entered string is displayed in background mode. If the text string is terminated with RETURN, then all lines of text are output until a line containing a caseless match of the specified string is found. If the string is terminated with LINEFEED then an exact match is sought. The line including the string is not output.

If the string is a number terminated with LINEFEED, then that number of lines from the current cursor position are sent to the extract file.

ESC WX **WRITE OUT TO END OF FILE**

The use of this command appends the remainder of the document onto the end of the open extract file.

C.112 ESC X **REPEAT LOOPS**

This sequence introduces various options which repeat a given string a number of times.

It should be noted that misuse of these facilities can cause LEX to enter infinite loops or even to crash because of stack overflow problems. The sequences stop on reaching the end of a document or, when repeatedly reading keys with ESC N, when there are no more keys.

Repeat loops can be terminated using CTRL C

ESC Xnn **REPEAT**

This sequence allows a string to be repeated nn times.

After typing ESC X the **TEXT prompt Repeat- appears and all characters until the execution command, RETURN, are shown in background mode. The number of repeats required is entered followed by the string to be repeated. The string commences on the first non-numeric character and may contain any characters, control functions or escape sequences. The latter are entered as their visible equivalents.

EXAMPLES

1. To write JOE JOE JOE enter ESC X and reply to the prompt:

```
Repeat-3JOE RETURN
```

2. To copy the line above using CTRL Y enter ESC X and reply to the prompt:

```
Repeat-80'Y RETURN
```

3. To copy the line above using CTRL Y but as a program in a keystore, the keystore should be loaded with

```
$X80@'Y\
```

Here the \ represents the end of the string, i.e. RETURN and the @ is necessary to nest the CTRL Y. If it was not so nested CTRL Y would be actioned after the X and not as part of the repeat. The same effect could be achieved by:

```
$X80"'Y"\
```

ESC X0 **REPEAT TO END OF DOCUMENT**

This is a special case of ESC nn where nn is zero. The string is repeated up to a maximum of 9999 times or until the end of the document is reached. This function must be handled with care or infinite loops can result.

On reaching the end of a document ERROR-EB is set and a BELL sent to the VDU. It also stops if an ESC G or ESC F in the repeat string detects the end of the text.

EXAMPLES

1. Replace every occurrence of this with that. The following sequence changes the first occurrence of this with that

`$Gthis\that`

If this is stored in keystore 1 then to execute a global exchange of this for that enter the repeat sequence (ESC)X and the following in reply to the prompt:

`Repeat-0$1` (RETURN)

This will terminate at the end of the document.

2. Note the effect of entering the following after an (ESC)X

`Repeat-0JOE@'H@'H@'H\`   `Repeat-0JOE"'H'H'H"\`

In this example the string JOE is added to the file and the cursor moved back to the beginning of the word. This means that the end of the document cannot be reached so the string will be executed 9999 times. The only way out of this is to wait for the sequence to execute 9999 times or apply the panic command, (CTRL)C.

(ESC)XU and (ESC)XW **REPEAT UNTIL/WHILE**

Two complementary functions are available to enable the conditional execution of a specified string. These are known as Repeat While and Repeat Until.

After typing (ESC)X the **TEXT prompt Repeat- appears and all further characters until the execution command, (RETURN), are shown in background mode. The general format of the character string is:

`WString1\String2`	for Repeat While and
`UString1\String2`	for Repeat Until

Where: String1 is the string to be tested
\ is the visual equivalent for the separator (RETURN)
String2 is the string to be actioned.

Note that String1 is compared with the text on the screen from the current position of the cursor onwards.

Control functions and escape sequences may be included in String2 as their visual equivalents.

As with all LEX functions repeat sequences may either be typed in as required or set up in a system record or keystore. In the latter case they will be nested in the format:

```
$XW"String1\String2"\
```

EXAMPLES

1. Consider the string:

```
AAAAAAAAAAAAAAAAAAAAAABBCCCCCCCCCCCCCCCCCC
```

If the cursor is positioned on the first A, (ESC)X followed by the following in reply to the prompt:

[redacted]

Whe cursor will move right until it reaches the first B. If the cursor was positioned outside of the AAAA, nothing happens.

2. Note that these features must be used very carefully. For example if the cursor is to the right of the AAAA, and the string is changed to the following:

```
Repeat-UA\'R (RETURN)
```

LEX enters an infinite loop which can only be terminated by killing the job or (CTRL)C.

3. A practical application of the use of this function may be found in the issued system record *LISSYS, where the function repeats while the first character of the key recalled is an asterisk:

```
$XW*@\"'R'R'R'R'R'R'R'R'R'R$>R))\\\<$N$1"\
```

C.113 (ESC)Y — DATE - IN FULL

(ESC)Y causes the full form of the date to be displayed. e.g. 28th September, 1985. The format is defined in **MONT and may be modified to suit the language, country or application. See Chapter 5.

C.114 (ESC)Z — ABANDON EDIT

This is used to abort the current edit of LEX. The output temporary file is killed and LEX returns to the relevant menu via, if initiated in **INST, the **TEXT prompt Are you sure (Y/N). The version of the document retained is that left on completion of the last (ESC)E or (ESC)V.

C.115 ESC-\ **EXIT VERTICAL MODE**

This forces LEX out of vertical mode. See ESC\.

C.116 ESC-D **RESTORE LAST DELETED CHARACTER**

This sequence is used to restore the last character deleted using either the CTRL D or the CTRL P combinations. When restoring the character, LEX acts as though the character were being entered in insert mode.

C.117 ESC-G **BACK SEARCH**

The use of this command prompts for a string using the **TEXT message `Search backwards for -`. LEX then searches back through the document for the first occurrence of that string. If the string is beyond the top of the screen the screen is refreshed with the found string placed a third of the way down. If the string is not found by reaching the top of the document ERROR-TB is set and a BELL sent to the VDU.

Note that when the input string is terminated by a RETURN, case and embedded spaces are ignored whereas an exact duplicate is sought if the string is terminated by a LINEFEED. In either case attributes are ignored.

C.118 ESC-I **LEAVE INSERT MODE**

The use of ESC-I will switch LEX out of Insert Mode. It has no effect if the ESC I sequence has not been previously used.

C.119 ESC-N **PREVIOUS KEY**

Keys are stored in the VMF in sequence and ESC-N displays the key on the VMF immediately before the last key referenced. If used within a repeat loop then it will stop when there are no more keys. Note that the next key may be obtained with ESC N.

C.120 ESC-MZ **REMOVE TAB FROM RULER**

This sequence modifies the current ruler by replacing the tab stop to the left of the cursor with a . (dot). The modified ruler is not shown or stored but may be viewed by ESC MO

C.121 ESC-SE — SHOW ERROR MODE OFF

ESCSE exits LEX from show error mode, i.e. if any errors occur a BELL is sent to the terminal but no error message is displayed unitil ESCH is entered.

C.122 ESC-SG — RESTORE AUTOMATIC RULER ASSERTION

If a ruler is inserted, it normally applies to all text below it in the document until another ruler is reached. When a new ruler is encountered it then automatically becomes the current ruler. There are a number of commands which effect the normal assertion of rulers which should be consulted in conjunction with this command. They are ESCSG, ESCUn and ESCUUn.

If LEX is operating under a ruler other than that which would normally be automatically asserted, then ESC-SG will return LEX to the default conditions. It switches automatic assertion of rulers back on and activates the ruler to be expected when in automatic assertion mode. It is usually used after a ESCSG command and has no effect in isolation.

C.123 ESC-SH — LEAVE SHUSH MODE

This forces LEX out of shush mode. See ESCSH.

C.124 ESC-SL — SWITCH OFF AUTOMATIC EDIT STATUS LINE

The status line can be set to remain on permanently whilst in edit mode. This sequence switches the facility off. As the status line slows the operation of LEX it is advantageous to switch it off when not required.

C.125 ESC-SU — EXIT SHUT-UP MODE

This forces LEX out of shutup mode. See ESCSU.

C.126 ESC-SW SWITCH OFF PAGE WARNING MODE

Page warning mode can be set in **INST or by ESCSW in which case LEX sets ERROR-EP and sends a BELL whenever a RETURN or LINEFEED is performed below the printable page according to the document extension details. This command switches the facility off. As page warning mode slows the operation of LEX it is advantageous to turn it off when not required.

C.127 ESC-V RESTORE LAST DELETED LINE

This is used to restore the text deleted from the screen using CTRLV, delete to end of line. It is identical to ESC-X and therefore also restores data deleted by CTRLX, ESCQA and ESCQB.

The action taken by LEX is to insert a blank line above the current line and enter the deleted text starting at the left hand column disregarding the margin settings.

C.128 ESC-W RESTORE LAST DELETED WORD

This sequence will restore the last word deleted using the CTRLW combination. It causes the restored word to be put into the text in insert mode.

C.129 ESC-X OR ESCCTRLX RESTORE LAST DELETED LINE

This is used to restore the text last deleted from the screen using CTRLX. It also restores the text previously overwritten or deleted by use of CTRLV, ESCQA or ESCQB.

The action taken by LEX is to insert a blank line above the current line and enter the deleted text starting at the left hand column disregarding the margin settings.

The sequence may be repeated thereby restoring multiple copies of the deleted text and so acting as a single line Cut-and-Paste buffer. Care must be exercised, however, as there are many ways in which this buffer may be overwritten.

Appendix D. CALCULATOR COMMANDS

D.1 INITIAL SETTINGS

The following commands are used on first entering the calculator to set various factors. They remain in force until leaving the calculator or until they are specifically changed.

Pn	Precision	When the calculator is entered by TAB into a ## ruler, the number of decimal places is defined in the ruler. When the calculator is entered using the ESC+ command LEX displays numbers to the number of decimal places specified in **INST but this can be changed to n decimal places by entering Pn.
KSx	Set thousands marker	A thousands delimiter can be enabled in the calculator either by default in **INST or by use of this command. In English versions it would be usual to set x as a , thus giving 12,345.67 for example.
KU	Unset thousands marker	This option unsets any marker at the thousands position, e.g. 12345.67

D.2 NUMBER MODIFIERS

On entering a number into the calculator there are commands which allow that number to be modified before continuing.

C	Clear	Zeros the number currently displayed allowing a new number to be entered before continuing with the calculation. The stored memories are unaffected.
Z	Reset	Clears the current calculation. Stored memories are unaffected.
F	Suppress non-significant figures	For example, 123.40 will be displayed as 123.4 and adding 0.6 to this will result in 124 being displayed. This does not work under a numeric ruler.

I	Integer	Entering I after a number, truncates it to its integral part. It is still displayed to the number of decimal places set but the fractional digits will be zero. This means that (ESC)+123.45I and (ESC)+123.98I will each produce the result of 123.00.
N	Negative	Entering N after a figure will cause the value to be negated. Positive values become negative and vice versa. This is the only way of entering negative numbers.

D.3 MATHEMATICAL OPERATORS

These commands initiate mathematical operations.

+	Add	Justifies the current number to the set precision (default 2 decimal places), and prepares to add the next number.
-	Subtract	As above but prepares to subtract the next number.
*	Multiply	As above but prepares to multiply by the next number.
/	Divide	As above but prepares to divide by the next number entered. A divide by 0 gives 0 as the result and sounds an error (BELL).
%	Percent	A % indicates that the last figure entered should be interpreted as a percentage (the figure is divided by 100).
Q	Square Root	Finds the square root of the current number using an iterative process. Attempts to find the square root of a negative number result in zero.
=	Equals	Displays the current result without leaving the calculator.
(	Open bracket	Brackets may be used in a normal mathematical way to establish the order in which actions should be performed. It is possible to nest up to four sets of brackets within an expression.

)	Close bracket	This marks the end of a bracketed expression.

D.4 MEMORY COMMANDS

There are 36 accumulators into which numbers can be stored identified by the numbers 0 to 9 and letters A to Z. They are not lost on leaving the calculator and can even be transferred between documents. This means that they have to be deliberately deleted either by using the calculator commands described below or the menu switch option /2. The following commands are associated with these memories where the accumulator is shown as x.

Sx	Store in memory x	Store the number being displayed in accumulator x.
Ax	Add to memory x	Add the displayed number to the specified accumulator.
Rx	Recall memory x	Recall the contents of the specified accumulator. The number is recalled to the precision previously specified using the Pn combination.
Ex	Erase memory x	Set accumulator x to zero.
EZ	Erase all memories	Set all accumulators to zero.

D.5 TOTALS

LEX can perform totals on rows and columns.

H	Horizontal total	If H is entered when LEX is in the calculator, all numbers to the left of the cursor position are added together to produce a horizontal total.
V	Vertical total	This function causes all figures vertically above the current position to be totalled. The operation will, if necessary, scroll back beyond the top of the screen until a non numeric field or the top of the document is reached.

D.6 MISCELLANEOUS COMMANDS

Some LEX control functions and escape sequences work when in the calculator:

CTRL B CTRL F TAB	These commands allow movement between fields on the current line. If the cursor is moved to another calculator field then LEX remains in the calculator. If not it exits calculator mode.
CTRL Y	Copies the entire number on the line above into the current calculator position.
ESC n	The contents of the keystore n are treated as though typed in at the keyboard. Visible equivalents are interpreted.
[ ]	System records can be activated from within the calculator. The contents of the record are treated as though typed in at the keyboard. Visible equivalents are interpreted.
# $	Whilst not commands, these characters may be entered in the calculator but they will not be displayed until leaving calculator mode.
( )	Calculations can be nested using brackets.

D.7 LEAVING THE CALCULATOR

RETURN	Performs a RETURN
SPACE	Moves one space right. A # or $ entered whilst in the calculator is deleted.
↑	↑ into a non-numeric field sets ERROR-AA and sends a BELL to the VDU.
ESC E	Exits from the calculator, the cursor is not moved.
CTRL B CTRL F TAB	Exits the current calculator field.

Appendix E. ERROR MESSAGES

SUMMARY

-ERROR-MA	******* Document already exists *******	*E.25*
-ERROR-OD	**** You already have a file open *****	*E.41*
-ERROR-OW	********* Can't open document *********	*E.43*
-ERROR-MW	********* Can't open document *********	*E.31*

SORT

-ERROR-SL	***** Sort columns unequal length *****	*E.46*
-ERROR-SR	******** Sort ruler not all L's *******	*E.47*

REPEAT

-ERROR-XR	** Missing String to Repeat in ESC X **	*E.51*
-ERROR-XS	* Missing Until/While String in ESC X *	*E.52*

RECORD HANDLING

-ERROR-NX	**** Open Square Bracket not found ****	*E.39*
-ERROR-BB	*ESC] not followed by valid character*	*E.3*
-ERROR-BC	********* Key Already Used **********	*E.4*
-ERROR-BK	********** Key Not Found ************	*E.6*
-ERROR-BS	***** Save key no longer on file ******	*E.8*
-ERROR-LK	************ Key is locked ************	*E.24*
-ERROR-NZ	*********** Key not found *************	*E.40*
-ERROR-MR	********** No room for mask ***********	*E.29*
-ERROR-NM	****** No masking record on file ******	*E.36*
-ERROR-NP	********* No paint characters *********	*E.37*
-ERROR-TM	******* Record too big for form *******	*E.49*
-ERROR-FF	***** Unable to open Foreign File *****	*E.19*
-ERROR-FR	*** No write access to Foreign File ***	*E.20*

VMF MAINTENANCE

-ERROR-NF	********** VMF is nearly full *********	*E.32*
-ERROR-NG	************* VMF is full *************	*E.33*
-ERROR-NR	****** No write access to file ********	*E.38*

MISCELLANEOUS ERRORS AND WARNINGS

-ERROR-AC	******* Unknown Escape Sequence *******	*E.1*
-ERROR-DL	* Demonstration version File too Big **	*E.15*
-ERROR-EP	***** Page break should occur here ****	*E.17*
-ERROR-EV	* ESC V/QX not allowed in playground **	*E.18*
-ERROR-OF	************ Invalid file *************	*E.42*
-ERROR-NK	******** Invalid key in ESC QS ********	*E.35*

E.1 ERROR-AC

`******* Unknown Escape Sequence *******`

An escape sequence not recognised by LEX results in this error being set.

E.2 ERROR-AX

`****** Cut too long for keystore ******`

Cut and paste operations into a keystore are limited by the size of a keystore. Therefore the MARK and CUT sequences must be on the same line and limited to 78 characters. Any more results in this error being set.

E.3 ERROR-BB

`*ESC ] not followed by valid character*`

When accessing a record on a VMF the ESC] or ESC> must be followed by A (append), C (create), K (kill), L (read with lock), P (put), R (read), S (restore) or U (update). Any other character results in this error.

E.4 ERROR-BC

`********* Key Already Used *********`

This error results from attempting to create a record on the VMF using a key which already exists.

E.5 ERROR-BF

`***** Cut and Paste buffer full *****`

The cut operation in cut and paste moves text into buffers in the work file. Each cut and paste buffer may be any size but the overall space is limited. The buffers are retained until specifically deleted and if an attempt is made to hold more than the total allowable then this error is set.

E.6 ERROR-BK

`*********  Key Not Found  **********`

Attempting to read or kill a record whose key does not exist gives this error.

E.7 ERROR-BR

`***** Invalid cut & paste block ******`

A block of text is marked at the top left hand corner by the MARK and at the bottom right by the CUT. This error is set if the CUT is to the left or above the MARK or if an attempt is made to CUT without having first specified a MARK.

E.8 **ERROR-BS** `**** Save key no longer on file *****`

When records are deleted from the VMF they are not initially removed but simply flagged. It is therefore possible to reinstate deleted records. On VMF reorganisation the deleted keys are lost. An attempt to reinstate a record no longer on file results in this error being set..

E.9 **ERROR-CB** `**** Calculator, Invalid Use of ] *****`

When executing a record whilst in the calculator, the key specified could not be found.

E.10 **ERROR-CC** `**** Invalid Calculator Character *****`

When in the calculator any operator or character not recognised by the calculator is indicated with this error being set. LEX remains in the calculator and must be exited before the error message can be viewed.

E.11 **ERROR-CE** `***** Invalid Calculator Syntax *******`

Operand wrong at this stage in calculation.

E.12 **ERROR-CM** `******* You haven't done a cut ********`

Attempting to paste an empty buffer results in this error being set.

E.13 **ERROR-CN** `***** Invalid Number in Calculator ****`

When tabbing into a calculator field defined in the ruler this error is set when a non numeric character is detected in the field. It is also set when entering the calculator using ESC+ if there are not a minimum of four blanks to the right of the cursor. In either case the calculator is not entered.

E.14 **ERROR-CO** `***** Calculator Number Overflow ******`

If a number is entered into the calculator with more digits then there are spaces in the field it is indicated by this error and the field is padded with ***. It also occurs when the result of a calculation contains too many digits to be displayed. When under a calculator ruler the display field is limited to the #### field set. The maximum number of figures, also applicable if the calculator was entered by ESC+, is 10 for the decimal version and 16 with the floating point RTS.

E.15 ERROR-DL

`* Demonstration version file too big **`

When using a demonstration version of LEX the length of a document may be up to 99 lines only.

E.16 ERROR-EB

`*********** End of Document ***********`

This error is set on detecting the end of a document. It can be the result of trying to page down beyond the end with CTRL L or by some automatic process. For example LEX allows infinite loops using the ESC XO sequence. On detecting the end of a document in such a loop LEX stops with this error set.

E.17 ERROR-EP

`***** Page break should occur here ****`

An option can be set in **INST to inform the user where LEX would normally put a page break according to the document extension details in use. In that mode LEX sets this error on every RETURN and LINEFEED beyond the nominated paper size.

E.18 ERROR-EV

`** ESC V/QX not allowed in playground *`

ESC V and ESC QX take a "snapshot" of the document by storing it away and re-entering at the top of the document and where the sequence was initiated respectively. When in playground there is no document to write away so the task is impossible and LEX sets this error.

E.19 ERROR-FF

`***** Unable to open Foreign File *****`

On attempting to open a foreign VMF file this error is set if LEX cannot find the file or cannot open it for any reason. The latter could result if the file is protected. Note that this error must not be confused with a user being limited to read only access as described in the next message.

E.20 **ERROR-FR** `*** No write access to Foreign File ***`

On attempting to open a foreign VMF file in single user mode this error is set if another user currently has access to the file. The VMF can still be read but write access is restricted. Note that this error must not be confused with the last error described where the file could not be opened even for read access.

E.21 **ERROR-HC** `******* Invalid hyphen character ******`

In hyphenation mode, after ESC QH, commands can be issued to control the position of the hyphenation. Characters not recognised result in this error being set. It also occurs when pulling characters from the line below if the current line is already full.

E.22 **ERROR-HF** `****** Next line is already full ******`

In hyphenation mode, after ESC QH, characters can be pushed, one at a time, onto the next line. Once the next line is full this error results.

E.23 **ERROR-IN** `***** Can't do insert character *******`

CTRL N nudges text to the left at the same time obeying tabs in the ruler. On reaching a hard tab mark such as L or R text cannot be nudged further right with this sequence and this error is set. Exactly the same limitation is placed on entering text when in insert mode. Note that the right margin M does not have this effect as text pushed of the end is automatically word wrapped onto the next line.

E.24 **ERROR-LK** `************ Key is locked ************`

If an attempt is made to read and lock a record using ESC]L when the record is already locked then this error is set. It is only used in multi user VMFs.

E.25 ERROR-MA

`******* Document already exists *******`

It is not possible to create a file that already exists. If an attempt is made to open a new file from the playground using ESC OX and a file of that name already exists then the file is not opened and this error results. Note that ESC OD opens a file unconditionally, deleting any existing file.

E.26 ERROR-MC

`*ESC M not followed by valid character*`

ESC M may be followed by a number to read a ruler or the letter Z to alter the position of the margin setting on the ruler. ESC M may also be followed by one of the letters A C E I J M O S U or V when merging in documents. Anything else after an ESC M results in this message.

E.27 ERROR-ME

`*********** End of merge file *********`

During merge operations advancing through the merge document will eventually reach the end of file. On detecting the end of the merge file this error is set.

E.28 ERROR-MF

`***** Can't Find Merge Document *******`

An attempt to open a merge file which does not exist results in this error.

E.29 ERROR-MR

`********** No room for mask ***********`

Not enough memory is available for masking to be used.

E.30 ERROR-MS

`**** Marked position not on screen *****`

The ESC <n sequences allow the cursor to be returned to position n previously marked by ESC S<n. This only works if the marked positions are on the screen. They may be scrolled off and back but if not on the screen when the ESC <n is issued this error is set.

E.31 ERROR-MW

`********* Can't open document *********`

The document specified when using ESC OD, ESC OF or ESC OX could not be opened.

E.32 **ERROR-NF** `********** VMF is nearly full *********`

This warning is set as the space left in the VMF gets below the size set in **INST. It is issued on creating or updating records as a warning that VMF should be reorganised soon.

E.33 **ERROR-NG** `************* VMF is full *************`

This error is set on creating or updating records when the VMF is full. The create or update fails and the VMF should be reorganised before any more record handling.

E.34 **ERROR-NJ** `******* Not a justifiable line ********`

A number of sequences can only take place between justifiable margins. Outside these limits these sequences fail with this error set. Sentence cut and paste is a case in point as both the MARK and CUT must be within such J/F.....M margins.

E.35 **ERROR-NK** `******** Invalid key in ESC QS ********`

The key specified when using ESC QS could not be found.

E.36 **ERROR-NM** `****** No masking record on file ******`

When a set mask ESC SM is used LEX looks for a masking record whose key follows the sequence. If such a record does not exist this error is set.

E.37 **ERROR-NP** `********* No paint characters *********`

An attempt to set protect mode with ESC SP when there are no paint characters on the screen results in this error.

E.38 **ERROR-NR** `******* No write access to file *******`

An attempt to update a VMF record on a VMF which does not have write access results in this error.

E.39 **ERROR-NX** `**** Open Square Bracket not found ****`

In record read/writes LEX goes back to find the last [. This error results if one cannot be found, or in the case of reads, if the [is more than the length of a key away.

E.40 **ERROR-NZ** `*********** Key not found *************`

This error results when an attempt to read a record is made and that record does not exist.

E.41 **ERROR-OD** `**** You already have a file open *****`

When in playground it is possible to open a file and transfer the contents of playground to it. This is achieved with (ESC)OD, (ESC)OF and (ESC)OX. If these sequences are used when in a document then another file cannot be opened and this error message is set.

E.42 **ERROR-OF** `************ Invalid file *************`

Attempting to open an existing file during (ESC)WO operations results in the setting of this error.

E.43 **ERROR-OW** `********* Can't open document *********`

The document specified when using (ESC)WO could not be opened.

E.44 **ERROR-PM** `******* You are in protect mode *******`

Some functions, e.g. (CTRL)W, are not allowed when protect mode has been invoked with (ESC)SP.

E.45 **ERROR-RI** `******* Invalid ruler character *******`

Characters which may appear in a ruler are . T L R J F M H # C. This message is set if an attempt is made to save a ruler (using (ESC)Wn) or to recall a ruler (using (ESC)Mn) which contains other characters.

E.46 **ERROR-SL** `***** Sort columns unequal length *****`

A column sort using (ESC)Bax requires that each column to be sorted is marked by an L in the ruler and the columns are the same width. If the width between each L in the ruler is not constant the sort cannot take place and this error is set.

E.47 **ERROR-SR** `******** Sort ruler not all L's *******`

A column sort using ESC Bax requires that each column to be sorted is marked by an L in the ruler and the columns are the same width. Any character in the ruler other than L to the left of the cursor position results in this error.

E.48 **ERROR-TB** `*** Cursor already at home position ***`

When the cursor is at the top of the document any attempt to move further up, e.g. CTRL T, gives this error. It also results when a string cannot be found in a back search ESC -G. In both cases the cursor is left at the home position.

E.49 **ERROR-TM** `******* Record too big for form *******`

An attempt to read a record onto a form that is smaller than the record causes the first part of the record to be read and this error to be set.

E.50 **ERROR-WF** `******** ESC W - No file open *********`

ESC WO must precede an extract operation so as to open the extract document. If a file is not open when an extract operation attempted then this error is set.

E.51 **ERROR-XR** `** Missing string to repeat in ESC X **`

If the number of times a string is to be repeated is missing or if the string itself is omitted when ESC X is used this error is set.

E.52 **ERROR-XS** `* Missing until/while string in ESC X *`

If the string to be repeated is omitted when ESC XU or ESC XW is used this error is set.

E.53 **ERROR-ZZ** `.......... No Error Message ..........`

In LEX one of the error messages described in this Appendix is always set. Before an actual error is detected this is the message set. After an error it is automatically reset to this value after the message is read with ESC H or on performing an ESC E to return to a menu.

Appendix F. PRINT TIME INSTRUCTIONS

SUMMARY

`.C`	Special print control characters	*F.1*
`.P`	Hard page break	*F.9*
`.PP`	Soft page break	*F.9*
`.G`	Suppress page break	*F.3*
`.T`	Titles (page heading)	*F.11*
`.F`	Footers	*F.2*
`.N`	Footnotes	*F.8*
`.L`	Throw to line number	*F.6*
`.S`	Line spacing	*F.10*
`.M`	Document merging	*F.7*
`.H`	Halt during printing	*F.4*
`.I`	Information (not printed)	*F.5*
`.X`	Fork to external program	*F.12*

F.1 PRINTER COMMAND .C SPECIAL CHARACTERS

The .C line defines characters occurring in a document which are used at print time. Twelve characters may be defined, e.g.

```
.C $~^#_@!|=<>%
```

Note that it is the position in the string that determines the use of a character. The characters used in the example are the recommended ones for English language versions of LEX. The special print control characters may also be defined in the `**INST` installation record. The character sequence sent to the printer for the ones marked * may be defined in the records created using `*PRTFORM` (see Chapter 9).

`$` Escape. At print time any occurrence of this character is replaced by the value set for escape in `*PRTFORM`. *

`~` Suppress linefeed and overtype the following line onto this one. *

`^` Control character. This is used in conjunction with the following character and results in the relevant low-value ASCII character being sent to the printer.

#	Special pound. This is used to signify that a special escape sequence is necessary to produce a pound sign on the printer. At print time any occurrence of this character is replaced by the value set for pound in *PRTFORM. If the print wheel has a pound sign in the position normally occupied by the hash symbol, then this should be left blank.	*
_	Special underline. This character is replaced with the escape sequence necessary to produce special underlining (double underlines for example), as defined in the underline field in *PRTFORM.	*
@	Space. This is used whenever a fixed layout is required and the layout includes spaces. This character is replaced with a space at print time.	
!	Paragraph numbering. This symbol is used to mark the start and end of a paragraph numbering area of text. Characters appearing within two of these symbols are translated into the relevant numbers.	
|	Indexing. Any text entered between two of these symbols is extracted into an index file together with the relevant page number on which it is found. The index file extension must be specified in **PNST for this to work.	
=	Hard hyphen. This is used at edit time to signify a hard hyphen which should not be removed on rejustification. It is translated at print time to a standard hyphen (-).	
<	Special character 1. At print time this is replaced by the sequence defined for Special character 1 in *PRTFORM. This and the next field could be used to cause special features, such as emboldening, to be switched on and off.	*
>	Special character 2. At print time this is replaced by the sequence defined for Special character 2 in *PRTFORM.	*
%	Footnotes. At print time text associated with the footnote number is placed at the foot of that page.	*

If no .C line appears in a document or if a .C line followed by spaces is used then the default setting defined in **INST is used.

A dot (.) is used in place of characters which are not required, e.g.

```
.C $..#.......
```

A .C line followed by one dot will disable all special characters, e.g. .C .

The special printer control characters may be redefined in a document by introducing new .C lines at any point. Note that any special printer control characters in force when a header or footer is defined remain in force each time the header or footer is printed, regardless of any subsequent redefinition of the characters.

The definition of any character in a .C line or in **INST has precedence over the value assigned to that character in petal form or petal function records. For example, if < is defined in the petal form and petal function records and < is also defined in the special character 1 position of the .C line (or in **INST) then for any occurrence of < in a document the value assigned to special character 1 in the *PRTFORM record will be sent to the printer.

F.2 PRINTER COMMAND .F — FOOTERS

A .F line introduces a footer which is to be printed at the foot of each page.

F.3 PRINTER COMMAND .G — SUPPRESS PAGE BREAK

Format: .G nnn

.G has a numeric operand which defines the number of lines before the next page break can occur. It should only be used in unpaginated documents and is useful for ensuring page breaks do not appear part way through tables or diagrams.

F.4 PRINTER COMMAND .H — HALT DURING PRINTING

A .H in a document causes LEX to stop printing at the point at which the .H appears and display any text typed on that line. It is useful when a print wheel or the stationery has to be changed during printing.

F.5 PRINTER COMMAND .I — INFORMATION

A .I line is used to introduce information lines in a document.

F.6 PRINTER COMMAND .M — MERGING DOCUMENTS

The .M control line allows a number of separate documents to be merged and printed as one document. The operand of the .M control line defines the name of a document to be printed. It is used, for example, when a large document such as a manual is stored in separate documents each of which contain one chapter.

F.7 PRINTER COMMAND .N — FOOTNOTES

.N is used to introduce a footnote, a footnote number is printed in the text and the associated footnote is printed at the foot of the page.

F.8 PRINTER COMMANDS .P AND .PP — PAGE BREAKS

When a document is being printed a .P or a .PP line forces a page break at the point at which it occurs. Full details of the use of .P and .PP for pagination are given in Chapter 13 of the User Manual.

Note that once a .PP appears in a document automatic pagination ceases and LEX only forces a page break at .P and .PP lines.

F.9 PRINTER COMMAND .S — LINES SPACING

Normally LEX prints in single line spacing. However .S allows different line spacings to be used, such as; quarter line, half line, one and a half line and double line.

F.10 PRINTER COMMAND .T — TITLE (PAGE HEADINGS)

A .T line introduces a page heading which is to be printed at the top of each page.

F.11 PRINTER COMMAND .X — FORK TO PROGRAM

Format: `.X n,m /XXXXXX YYYYYY.YYY`

.X is used to allow LEX to fork to a different program during printing of a document so that information from other documents may be merged into the existing printing. It is mainly used for printing graphics documents.

When a .X line is encountered in a document LEX will fork to the program xxxxxx and will pass YYYYYY.YYY as a parameter to the program. n will be added to the page count and m will be added to the line count.

Appendix G. THE LEX EDIT STATUS LINE

The edit status line takes the following form:

```
Status- AAAAAA.BBB L=   9.00 A=      10 C= 25 P=      9 .S 0.50 .C=$.....!|.{..
```

Each field has the following meaning:

`Status -`	Literal from **TEXT to announce the purpose of the display.
`AAAAAA.BBB`	The name of the document being manipulated which may include account/device details. This will be blank if the status edit line is requested from the playground.
`L=  9.00`	The number of print lines up to this point in this page of the document. The count takes note of line spacing, titles, footers and any non-printing or over-printed lines. If the document has not been paginated, then very large values of line counts are possible. It is shown to the nearest quarter line.
`A=  10`	The actual number of lines in the document. including lines containing rulers and other control information.
`C=  25`	The column number at which the cursor was positioned when the status line was called.
`P=   9`	The number of the page currently on the screen. This is only incremented when a .P or .PP is encountered in the document.
`.S  0.50`	The line spacing in force at this point in the document. This is shown to the correct quarter line.
`.C $.....!|.{..`	The current values of the special print time characters in force at this point in the document. These will have been set by the last .C line in the document or, if no .C line was encountered, then from the values in **INST. Appendix F includes details of the meaning of each character in the .C string.

Appendix H. THE PAGE BREAK LINE

The page break display line takes the following form:

```
.PP=0123===================L/P=55/FF            .S    1.00 .C $.....!|.(..====
```

The .PP signifies a soft page break that can be moved when manually paginating with (ESC)P. A .P may be entered in a document to force a page break.

Manual pagination with (ESC)P adds the details described below to any .P line encountered or, if a .P is not found before a new page is necessary then a line is inserted commencing with .PP and including the same details.

`0123`	The pseudo page number. Not necessarily the number printed on the page because of print merges, etc. Note that it is possible to use the pseudo page number in a 'Find' operation when editing a document. With a hard page break a page number can be forced by prefixing it with a * e.g. `.P *0003`
`L/P=55/FF`	This is taken from the document printing details and shows the number of lines to print per page and the page length.
`.S 1.00`	This is the current line spacing taken from the last .S command, and is displayed to the nearest quarter line.
`.C $.....!|.(..`	The curient special print time control characters taken from the last .C command encountered in the document, or if no .C was defined the default setting from **INST is used. The meaning of each character in a .C line is described in Appendix F.

Appendix I. VISIBLE REPRESENTATION OF ATTRIBUTES

Character	Attributes Set				Character	Attributes Set				
$\boxed{\text{SPACE}}$	none				$\boxed{\text{UNDERLINE}}$	1				
A	2				a	1	2			
B		3			b	1		3		
C	2	3			c	1	2	3		
D			4		d	1			4	
E	2		4		e	1	2		4	
F		3	4		f	1		3	4	
G	2	3	4		g	1	2	3	4	
H				5	h	1				5
I	2			5	i	1	2			5
J		3		5	j	1		3		5
K	2	3		5	k	1	2	3		5
L			4	5	l	1			4	5
M	2		4	5	m	1	2		4	5
N		3	4	5	n	1		3	4	5
O	2	3	4	5	o	1	2	3	4	5

For example

If a line is underlined using attribute 1 then, in the document, this is stored as:

This is a title$\boxed{\text{SPECIAL-CHARACTER}}\boxed{\text{CR}}\boxed{\text{LINEFEED}}$
______________$\boxed{\text{CR}}\boxed{\text{LINEFEED}}$

Similarly if the last word is underlined AND has attribute 2 set (typically bold) then it is stored as:

This is a title$\boxed{\text{SPECIAL-CHARACTER}}\boxed{\text{CR}}\boxed{\text{LINEFEED}}$
aaaaaaaaaaaaaa$\boxed{\text{CR}}\boxed{\text{LINEFEED}}$

The special character is defined in ****MONT** and is usually 255 (octal 377)

Appendix J. ASCII CHARACTER SET

Decimal Value	ASCII Char(1)	Decimal Value	ASCII Char	Decimal Value	ASCII Char	Decimal Value	ASCII Char
000	NUL	032	SP	064	@	096	'
001	SOH	033	!	065	A	097	a
002	STX	034	"	066	B	098	b
003	ETX	035	# or £(2)	067	C	099	c
004	EOT	036	$	068	D	100	d
005	ENQ	037	%	069	E	101	e
006	ACK	038	&	070	F	102	f
007	BEL	039	'	071	G	103	g
008	BS	040	(	072	H	104	h
009	HT	041	)	073	I	105	i
010	LF	042	*	074	J	106	j
011	VT	043	+	075	K	107	k
012	FF	044	,	076	L	108	l
013	CR	045	-	077	M	109	m
014	SO	046	.	078	N	110	n
015	SI	047	/	079	O	111	o
016	DLE	048	0	080	P	112	p
017	DC1	049	1	081	Q	113	q
018	DC2	050	2	082	R	114	r
019	DC3	051	3	083	S	115	s
020	DC4	052	4	084	T	116	t
021	NAK	053	5	085	U	117	u
022	SYN	054	6	086	V	118	v
023	ETB	055	7	087	W	119	w
024	CAN	056	8	088	X	120	x
025	EM	057	9	089	Y	121	y
026	SUB	058	:	090	Z	122	z
027	ESC	059	;	091	[	123	{
028	FS	060	<	092	\	124	|
029	GS	061	=	093	]	125	}
030	RS	062	>	094	^	126	~
031	US	063	?	095	_	127	DEL

(1) ASCII characters with decimal value 000 to 031 are known as 'control codes' because they are often generated on a keyboard by holding down the CTRL key. They are not displayed on the screen but may cause the terminal to perform some other function. For example LF (linefeed) will cause the cursor to move down one line, BEL (bell) will cause the terminal to emit a bleep.

(2) On some UK terminals this ASCII code is generated by the £ key and displays as £ and on other terminals it is represented by #. Similarly some daisy wheels have a £ where the # would normally be found although others may have the £ in an otherwise unused position.