Malcolm Bull
Training and Consultancy Publications

MB-Guide to

Access on Advanced Pick

Malcolm Bull

MB-Guide to Access on Advanced Pick

MB-Guide to Access on Advanced Pick

by

Malcolm Bull

Malcolm Bull
Training & Consultancy Publications
19 Smith House Lane
BRIGHOUSE
HD6 2JY
West Yorkshire
United Kingdom

Telephone: 0484-713577

ISBN: 1 873283 59 8

Edition 1: September 1994

Contents

Preface

The *MB-Guide to Access on Advanced Pick* is produced for those who need an introduction to the use of the **Access enquiry language** on the Advanced Pick system. It is aimed primarily at those users who need to be able to develop Access sentences and make enquiries about their database, and is also suitable for anyone who needs to brush up their knowledge of Access.

The *MB-Guide* contains:

- A general introduction to the construction and features of Access sentences;
- A look at the various elements which go to make up an Access sentence.
- Guidance on typing Access sentences;
- A description of the way in which Access sentences can be developed and saved as macros and menus.

The floppy diskette accompanying this *MB-Guide* contains all the files which are used in the examples in the text and can be used to practise the commands presented here.

Most of the examples and illustrations used in this *MB-Guide* use a simple STOCK file, holding details of the description, quantity in stock, warehouse location, minimum stock-holding, unit price, supplier reference code, and the date when the stock-holding was last used. Nevertheless, everything that we do with this STOCK file applies equally well to any other Pick file, be it STAFF, INVOICES, ORDERS, SALES-FIGURES, or whatever you are recording on your Pick system.

When you have mastered all the information presented here, you may find it interesting to read the *MB-Guide to Access definitions and dictionaries* which tells you have to create your own datanames to pull data off your files in precisely the form that you want.

You may find the following titles in the *MB-Guide* beginner's guide series useful in conjunction with the present volume:

Advanced Pick / AP/DOS / AP/Native
Access definition and dictionaries
Procs
File design
Files: monitoring and sizing
Advanced Pick fundamentals

You may also find the following *MB-Master* self-tuition courses of interest in conjunction with the material presented in this *MB-Guide*:

Starting Access
Access techniques

For a more detailed coverage of the material presented in this *MB-Guide*, you may be interested in reading the book ACCESS, THE PICK ENQUIRY LANGUAGE by Malcolm Bull which is published by Chapman and Hall.

This *MB-Guide* is not intended to present an exhaustive description of the subject but merely to place it in context and give the reader enough information to use the facilities described and to survive.

Best use can be made of this *MB-Guide* if it is read in conjunction with the reference literature which is provided for your system. You should amend your copy of this guide so that it accurately reflects the situation and the commands which are used on the implementation which you are using. By doing this, your *MB-Guide* will become a working document that you can use in your daily work.

I hope that you enjoy reading and using this *MB-Guide* and that you get a chance to read the other titles in the series, and I welcome your comments.

Malcolm Bull

The floppy diskette

This *MB-Guide to Access on Advanced Pick* is accompanied by a floppy diskette. This contains a number of files which will allow you to practise the Access sentences which are presented in the *MB-Guide*.

To use the diskette, you should do the following:

- Log to the DM account.
- Mount the diskette in the drive.
- Issue a command such as:

```
SET-FLOPPY (AS
```

- Issue the command:

```
ACCOUNT-RESTORE MB-AP-ACCESS
```

and in response to the questions, enter these responses:

```
Account-name: MB-AP-ACCESS

Restore from incremental file-save? N

Restore from transaction log tape? N
```

You may then log on to the account MB-AP-ACCESS and practise the Access sentences shown in the *MB-Guide*.

There is a README item on the MB-AP-ACCESS account and this contains any last-minute news and updates to the *MB-Guide*. You can inspect this by either of the commands:

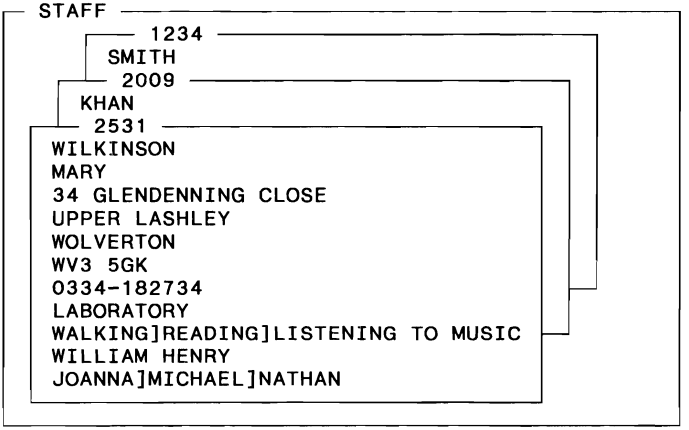
```
OP TEXT README

OP TEXT README (P
```

1 An organisation and its files

The most valuable resource of any commercial organisation is the *data* and the *information* which it owns.

All the information which you and your users process is organised as **files**. There may be a STAFF file to hold details of all the company's employees; a CLIENT file to hold details of all the company's customers, and many more. There may be any number of files on your system.

Each file comprises a number of separate **items** or *records*. Thus, the STAFF file may have one item for employee number 1234 (this may be John Smith), one for employee 2009 (Naren Khan), another for employee 2531 (Mary Wilkinson), and so on. There may be any number of items on a file. You can visualise a file like this:



Within each file, each item is identified by its **item-id** or *record key*. This is some unique identifier for that item. Typically, the items on the STAFF file would use the employee number as their item-id, so the item-id of Mary Wilkinson's item is 2531. Similarly, the items on the CLIENT file may use the account number of client reference number as their item-id.

Each item is made up of a number of data **attributes** or *fields*. Thus, the first attribute of a STAFF item may be the employee's surname; the second might be the employee's given names, and so on. There may be any number of attributes in an item. Normally, all the items on, say, the STAFF file will have the same number of attributes and these will always be in the same order. The nature, form and order of the attributes is determined by the Systems Analyst when he/she designs the file.

If a certain employee did not have, say, a telephone number, then this attribute (attribute 7 in our STAFF file shown here) would

be empty. Some people call this a **null** attribute.

Some attributes, such as the hobbies (attribute 9) or the names of the children (attribute 11), may have more than one value. Such an attribute is said to be a **multivalued** attribute. In the diagram, I have separated these **multivalues** by a square bracket.

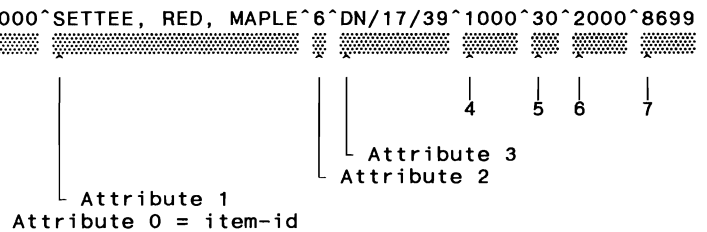
For security reasons, it is unwise to let every user have access to every file; otherwise, anyone could look at, and change, any record on, say, the STAFF file. Therefore, the files of your system are grouped into **accounts**. Thus, the STAFF file, the PAY.RATES file may be held on the WAGES account; the CLIENT file, ORDER file, PRICES file and the STOCK file may be held on the SALES account, and so on. To use the STAFF file, you must log on to the WAGES account, and to use the CLIENT file, you must log on to the SALES account.

We shall look at these points again later.

Obviously, it is very important for us to be able to look at the data which is held in the items on our files, and to know what information is held on the files. This is where **Access** comes in.

1.1 Data format

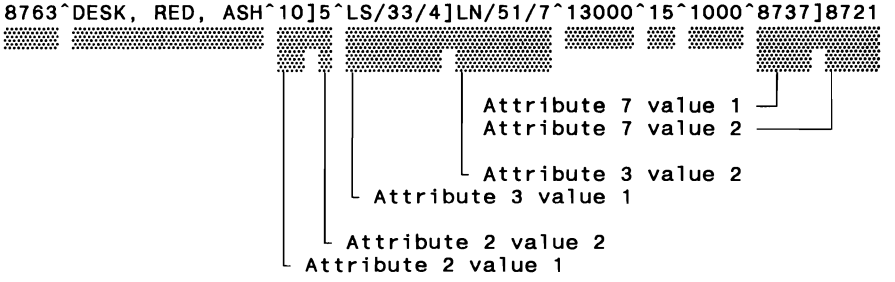
Let's look at the way in which the data is actually held on the computer. All the data which you will use is organised as items on Pick files on the computer's hard disk. An item is *read*, *written* and *held* on disk as a *string of characters*. If we were to look at the disk, we might find that a typical item looks like this:



Each item is uniquely identified within the file by its **item-id**. The item-id is allocated when the item is created by the programs of your application system.

The data within a single item may comprise none, one or more **attributes**. The individual attributes are separated by the **attribute-mark** character which is conventionally represented by the ^ character. Logically, an attribute represents a piece of data about that particular item. For example, the stock items shown in our illustration have attributes representing the Description, the Quantity in Stock, the Location in the warehouse

where that stock is held, the Price and so on.



The data within a single *attribute* may comprise none, one or more **values**. The individual values are separated by the **value-mark** character which is conventionally represented by the] character. An attribute which logically has - or may have - several values is referred to as a **multivalued** attribute. Logically, a set of multivalues represents several occurrences of that particular attribute. For example, in stock item 8763 shown in our illustration, the Quantity (attribute 2), the Location (attribute 3) and the delivery date (attribute 7) are multivalued, indicating that there are several stocks of this particular product (the stock levels shown in attribute 2) and each stock is held at a separating location in the warehouse (the location shown in attribute 3).

It is further possible for each *value* to be made up of none, one or more **subvalues**. Within the value, the individual subvalues are separated by the **subvalue-mark** character which is conventionally represented by the \ character. However, these subvalues are rarely used and we shall not discuss them in this *MB-Guide*.

1.2 Introducing Access

Access is a very powerful feature of the Pick system which allows you to make enquiries about the data on your files and to produce reports simply by typing in English language sentences.

Once you have logged on to the Advanced Pick system, and you have reached the prompt:

```
:
```

you can type in any Access sentence and this will be processed and the answer - or the report - will be *displayed on your screen* or *printed on the printer*.

For example, if you are interested in the information held on your STOCK file, a sentence such as:

```
LIST STOCK DESCRIPTION PRICE
```

might display a report starting like this:

```
Page   1        STOCK                          09:01:00 29 Jul 1994

STOCK..... DESCRIPTION.............  PRICE

9000       SIDEBOARD, YELLOW, ASH    130.00
5500       TABLE, YELLOW, PINE        50.00
8888       DESK, GREEN, ASH           56.00
2000       SETTEE, YELLOW, OAK       100.00
1111       STOOL, GREEN, PINE         20.00
4500       CHAIR, RED, LAMINATE       25.00
```

showing the item-ids of the items on the STOCK file, together with the DESCRIPTION and the PRICE of each item. As we shall see later, words such as DESCRIPTION and PRICE are the **datanames** which identify the various pieces of the data on the STOCK file. Your Systems Analyst will tell you which files and which datanames are available to you. These datanames will have been set up for you on a **dictionary** for the file.

Each file has its own dictionary. For example, the dictionary for the STOCK file might hold datanames such as DESCRIPTION, PRICE, QUANTITY, MINIMUM-LEVEL, VALUE, LOCATION, and SUPPLIER, whilst the dictionary for the STAFF file might have datanames such as NAME, SURNAME, INITIALS, ADDRESS, TELEPHONE-NUMBER, START-DATE, DATE-OF-BIRTH, and so on.

One of the beauties of Access is that it is *flexible*; you can ask for the datanames in any order, and you can ask for as many datanames as you want. If we had typed the sentence:

```
LIST STOCK PRICE DESCRIPTION QUANTITY
```

we might have produced the report:

```
Page   1        STOCK                          09:02:00 29 Jul 1994

STOCK.....   PRICE DESCRIPTION.............. QTY

9000       130.00 SIDEBOARD, YELLOW, ASH      99
5500        50.00 TABLE, YELLOW, PINE         15
8888        56.00 DESK, GREEN, ASH            64
2000       100.00 SETTEE, YELLOW, OAK         18
1111        20.00 STOOL, GREEN, PINE          63
4500        25.00 CHAIR, RED, LAMINATE        18
```

Each sentence is processed immediately, so if you have made a mistake, the Access processor will tell you and reject the sentence without any fuss.

If we had typed the sentence:

```
LIST STOCK DESCRIPTION PRICE LPTR
```

adding the word LPTR to the sentence, then the same report would have been produced on the *printer*.

Because the way in which you use Access to communicate with the computer, Access is sometimes known as a **query language** or an **enquiry language**. An enquiry language makes it much easier to produce the reports and to answer the questions which you need. Before the advent of enquiry languages, the only solution to these problems was to have a programmer write one or more special report-production programs for each user; this would take many days (or even weeks) and would not be as flexible as the Access enquiry language.

1.3 Multivalues on the report

If any of your data has *multivalues*, then these will appear one beneath the other. So, if our report included any items with multivalues for - in the case of our STOCK file - the QUANTITY or the LOCATION or the SUPPLIER or the DATE - then the:

```
LIST STOCK DESCRIPTION QUANTITY LOCATION
```

might look like this:

```
Page   1        STOCK                          09:03:00 29 Jul 1994

STOCK..... DESCRIPTION.............. QUANTITY LOCATION..

9000       SIDEBOARD, YELLOW, ASH          99 DN/14/70
5500       TABLE, YELLOW, PINE             15 LS/15/77
8888       DESK, GREEN, ASH                64 LS/2/23
2000       SETTEE, YELLOW, OAK             18 DN/1/69
1111       STOOL, GREEN, PINE              63 DN/6/3
4500       CHAIR, RED, LAMINATE            18 LN/3/29
8763       SIDEBOARD, MAROON, ASH          10 LS/33/4
                                            5 LN/51/7
                                            0 MN/3/40
4444       SETTEE, BLACK, ASH                 DN/6/81
1000       DESK, GREEN-BLUE, ASH            8 MN/17/81
4200       SETTEE, GOLD, ASH               55 XN/19/35
                                           10 LS/6/6
7245       BUREAU, GOLD, ASH               55 LN/19/35
                                           10 LN/6/6
7777       STOOL, YELLOW, LAMINATE         25 DN/7/23
```

Here, we see that item 8763 has three values for the QUANTITY and three for the LOCATION, item 4200 has two values for the QUANTITY and two for the LOCATION. Since I am familiar with the data on my file, I know that this means that - in the case of item 8763 - we have got ten of these sideboards in storage bay LS/33/4,

another five in storage bay LN/51/7, and none in storage bay MN/3/40 although there are usually some held there.

1.4 The floppy diskette

The floppy diskette accompaying this *MB-Guide* is an account-save of the *MB-AP-ACCESS* account and contains all the files which were used to produce the reports and examples shown in the text here. If you wish to use these files, you should restore the account to your system by means of a sequence of commands such as:

```
SET-FLOPPY (AS
ACCOUNT-RESTORE MB-AP-ACCESS
```

and, having restored the account, you can log to the MB-AP-ACCESS account to issue the Access sentences shown here. If you do this, you will find that there are many more items on the STOCK file than I have shown in my examples here.

2 Format of Access sentences

Each Access sentence is made up of a sequence of grammatical elements - or *building blocks*. The general structure of any Access sentence is the same, no matter what the action of the sentence:

```
verb filename {item list} {selection criteria}
     {sort specification} {output list} {modifiers}
     {(options}
```

I've shown some elements in *curly brackets* to indicate that these are *optional* and can be included or omitted as you require; you don't type in the curly brackets. If you omit all the optional elements, you produce the simplest possible Access sentences:

```
LIST STOCK
COUNT STAFF
```

A summary of the *grammar* of an Access sentence is shown in the reference section at the end of this *MB-Guide*.

When you are developing a complicated Access sentence such as:

```
SORT STOCK BY COLOUR BY DESCRIPTION BREAK-ON BRCOLOUR "
     'UPB'" WITH PRICE > "50" DESCRIPTION LOCATION
     QUANTITY PRICE TOTAL VALUE HEADING "STOCK REPORT :
     COLOUR '''B''' DATED 'DL'============'L'" FOOTING
     "'LC'PAGE 'PL'" GRAND-TOTAL "FINAL TOTALS'U'" LPTR
```

you will find it much easier to start with the simplest form and then build it up - step by step - adding more and more of the optional elements - the *building blocks* - until you finally produce the report which you need. This is particularly easy if

you use the TCL **stacker** which stores all the TCL commands and Access sentences which you type in and lets you recall and change them.

In some of my examples, I've had to show the sentences spread over several lines, but you will type the entire sentence as a single line.

Provided that you put the **verb** at the beginning of the sentence, the rest of the building blocks can come in any sequence. So if we start with the simple sentence:

```
SORT STOCK
```

we could type it again, adding the block BY COLOUR to produce:

```
SORT STOCK BY COLOUR
```

and then again adding the block WITH PRICE > "50"

```
SORT STOCK BY COLOUR WITH PRICE > "50"
```

We could have typed this same sentence with the blocks in a different sequence, such as:

```
SORT STOCK WITH PRICE > "50" BY COLOUR
```

and both would have produced the same report.

Some of the blocks can be repeated several times, for example:

```
SORT STOCK WITH PRICE > "50" BY COLOUR BY PRICE
```
or:
```
SORT STOCK BY COLOUR WITH PRICE > "50" BY PRICE
```
or:
```
SORT STOCK BY COLOUR BY PRICE WITH PRICE > "50"
```

sorting first BY COLOUR and then BY PRICE. However, you must not split up any single block; so the sentence:

```
SORT STOCK WITH BY COLOUR PRICE > "50"
```

with the *BY COLOUR* splitting up the *WITH PRICE > "50"* block, would be wrong and the Access processor would reject it.

3 Access verbs

The **verb** is the first word of an Access sentence. There are a several Access verbs available, as shown below, and each one has a slightly different action from the others.

By far the greatest use is made of the LIST and SORT verbs. In alphabetical order, some of the more useful Access verbs are:

COUNT count the number of items which satisfy any specified selection criteria.

LIST produce a report based upon specified selection criteria and containing specified datanames.

SORT-ITEM
LIST-ITEM list the items in a file according to any specified selection criteria.

SORT-LABEL
LIST-LABEL produce a report in label format.

SORT produce a report sorted in a specified sequence based upon specified selection criteria and containing specified datanames.

SELECT
SSELECT create a select list of the item-ids on a file sorted into any specified sequence and according to any selection specification.

Most of the examples in this *MB-Guide* will use the LIST and SORT verbs.

4 File names

After the verb, the most important part of an Access sentence is the **file name**. This must be present so as to tell the Access processor which file you are interested in.

Typically, you will specify the file name in a form such as:

```
LIST STOCK
COUNT STOCK
```

In order to use Access sentence such as these, the file STOCK must be accessible on your account; this will be by means of a D-pointer, because it is one of your own files, or by means of a *pathname* or a *Q-pointer* if it belongs to another account. A Q-pointer is an item on your MD which points to a file on another account; we shall not discuss Q-pointers here, but your Systems Analyst will create any which you may need.

4.1 Pathnames

You may use **pathnames** instead of an actual file-name if you want to access a file on another account. Thus, either of the sentences:

```
LIST WAGES,PAYROLL,PAYROLL NAME AGE ADDRESS
LIST WAGES,PAYROLL, NAME AGE ADDRESS
```

can be used to list the name, age and address attributes of items on the PAYROLL file which belongs to the WAGES account.

4.2 Looking at the DICT of a file

If you want handle the DICT - the **dictionary** - section of a file, for example, you may want to produce a report showing the dictionary definitions which are available for a file, then you will use the form:

```
LIST DICT STOCK
SORT DICT STOCK
```

When you list the DICT section of a file - or even some of your own files - you may get a lot of information printed out. This is because the Access sentence is using a standard set of dictionary definitions. If you don't want this and are only interested in the **item-ids** of the items, then you can issue a sentence using the word ONLY before the filename, like this:

```
SORT ONLY DICT STOCK
LIST ONLY STOCK
LIST ONLY MD
```

4.3 LISTFILES - what files are there?

What can you do if you don't know the name of the file which you want to use? There is a special command which will list the files which are available on the account which you are using. The command:

```
LISTFILES
```

will display the details of all the files which are available to you on this account. The command:

```
LISTFILES (P
```

will produce the same list on the printer.

There are two parts to the LISTFILES report. The first shows all those files which belong to your account (that is, the one you are currently logged on to), these will normally have a letter D in the second column of the report; the second part of the report shows all those files which belong to another account, but which are accessible from your account, these will normally have a letter Q in the second column.

If you cannot find the file that you want on that list, it may be that you are logged on to the wrong Advanced Pick account.

5 Output list

Probably the most powerful - and valuable - use of Access is in producing displayed and printed reports showing the contents of your data files.

The datanames which appear in an Access report are:

- The item-ids,
- Any TOTALled fields,
- Any BREAK-ON fields,

together with any other datanames which you include in the sentence. The datanames for the pieces of information which are to appear in the report make up what is sometimes known as the **output list**.

So the sentence:

```
LIST STOCK PRICE
```

will contain two columns of data: the item-ids and the PRICE. That is, the output list consists of just the PRICE. The sentence:

```
SORT STOCK DESCRIPTION BY PRICE
```

will contain two columns of data: the item-ids and the DESCRIPTION. The use of BY PRICE to specify the order of the items in the report does not automatically include the PRICE in the output list for the report. The sentence:

```
LIST STAFF NAME TOTAL SALARY
```

will contain three columns of data: the item-ids, the NAME and the SALARY. The use of TOTAL to add up the individual SALARY fields does include the SALARY in the output list for the report.

5.1 Non-columnar output

There is no limit to the number of datanames which you can ask for in your output list, and you may ask for any dataname as many times as you wish:

```
LIST WAGES SURNAME SURNAME RATE CODE RATE
```

The Access processor will attempt to produce a report with a separate column for each dataname which you included in your output list. If the resultant report is too wide for the screen - or for the printer if you've asked to print the report - then the Access processor will abandon the normal column format and

display (or print) the report in a *non-columnar* fashion, going *down the page* instead of going across in columns. For example, the long sentence:

```
LIST STOCK DESCRIPTION LOCATION SUPPLIER.NAME
     SUPPLIER.ADDRESS SUPPLIER.TELEPHONE
```

might produce a report which begins like this:

```
Page   1       STOCK                      09:04:00  29 Jul 1994

STOCK                 9000
 DESCRIPTION           SIDEBOARD, YELLOW, ASH
 LOCATION              DN/14/70
 Supplier name         ROGER PETERSEN PLC
 Supplier address      98 WALTHAM CROSS WALTHAM FOREST
 Supplier telephone    081-999 4767

STOCK                 5500
 DESCRIPTION           TABLE, YELLOW, PINE
 LOCATION              LS/15/77
 Supplier name         A.B.ATKINS AND CO
 Supplier address      PO BOX 134 ALTRINCHAM CHESHIRE
 Supplier telephone    061-999 2375
```

5.2 FILL

The FILL modifier (and the equivalent R option) may be used with an Access sentence which is too wide for the screen (or printer) and which would otherwise produce *non-columnar* output. When this is used, the data is output with the headings and the data continuing across the page in a text-like manner.

```
STOCK 9000  DESCRIPTION SIDEBOARD, YELLOW, ASH  LOCATION
DN/14/70  Supplier name  ROGER PETERSEN PLC  Supplier
address 98 WALTHAM CROSS WALTHAM FOREST  Supplier telephone
081-999 4767

STOCK 5500  DESCRIPTION TABLE, YELLOW, PINE  LOCATION
LS/15/77  Supplier name A.B.ATKINS AND CO  Supplier address
PO BOX 134 ALTRINCHAM CHESHIRE  Supplier telephone 061-999
2375
```

If you display this report on the screen, the data values will be highlighted and the headings will be faint.

5.3 LISTDICT: what datanames are there?

The LISTFILES command tells you which files are available to you; but what can you do if you don't know the datanames which are available for the file which you want to use? There is a command on the system which will list the datanames which are available for use with a particular file. This is used in a command such as:

```
LISTDICT STOCK
```

and will display the details of all the attribute names which are available for the STOCK file, and:

```
LISTDICT STOCK (P
```

will produce the same report on the printer.

The LISTDICT command will produce a list of the **datanames** - or the **attribute definitions** - which are available for a particular file. The report will show one entry for each dataname which is available on the **dictionary** of the file. Each entry will include some or all of these pieces of information about the definitions which are available for use with the file:

- The **dataname** is shown in the first column: this is the name of the dictionary definition, and it is this name which you will use in your Access sentences.

  You needn't concern yourself with the remaining information. This is discussed in the *MB-Guide to Access dictionaries & definitions:*

- The *dictionary-code* or *code* - this is an A (or possibly an S) showing that this dataname defines an **attribute** of the data file. If a definition has anything other than an A or an S here, then you should not use the dataname in an Access sentence.

- The *attribute-count* or *A/AMC* - this tell the Access processor which field of your data records is to be identified by this dataname.

- The *substitute-header* or *tag* - the column heading when this datanames is used in a report. If this is blank, then the column heading will be the same as the dataname itself.

- The *output-conversion* or *conversions* - this specifies any conversion code(s) which are used to change the format of the data as it is displayed. For example, a numeric field such as PRICE or VALUE may have a conversion code such as:

  ```
  MD2
  ```

  indicating that this is to be printed with 2 places of decimals.

- The *correlative* - any correlative(s) which are to be used to calculate the result from the data on the file. For example, the VALUE is not taken directly from the STOCK file; instead, it is derived by multiplying attribute 2 (the quantity) by attribute 4 (the price).

- The *attribute-type* or *tp* - the justification of the data: this

will be Left or Right or Text. Numbers will normally have an R here, text and other information will have an L or T or LX or TX or W or WW.

- The *column-width* or *max* - the width of the column allowed for that data on the report. This is the information that the Access processor uses to work out the width of the report.

The datanames shown in the first column of this report may then be freely used in Access sentences which handle that file.

5.4 Temporary attribute items

If you don't know what datanames are available to you - or even if there are *no* datanames for a particular file - there is a set of standard datanames - known as **temporary attribute items** - which you can use with *any* file. These have the form:

```
A0  to refer to the item-id,
A1  to refer to attribute 1,
A2  to refer to attribute 2,
```

and so on. These datanames can be used just like the datanames mentioned above in sentences such as:

```
SORT STOCK A0 A1 A2 A3 A4 A5 A6 A999
SORT STOCK A1 A2 A3 A4 BY A7
SORT STOCK A1 TOTAL A2 A3 TOTAL A5
SORT STOCK BY A8 DESCRIPTION TOTAL PRICE
LIST MD WITH A0 = "T-]" AND WITH A1 = "P]"
```

You may also come across some older forms of this notation which uses datanames *A0, *A1, and so on, but these only go up to *A13. The A123 forms go on indefinitely.

5.5 Implicit output list

We have seen that you will usually specify an **output list** for your Access sentences, indicating which datanames are to appear on the output report:

```
SORT STOCK DESCRIPTION LOCATION PRICE
```

or:

```
SORT STOCK BY COLOUR DESCRIPTION QUANTITY TOTAL VALUE
```

There is a useful Access facility which allows you produce a report without specifying an output list in the Access sentence. This is known as the **default output list** or the **implicit output list**, and means that, if you use sentences such as:

```
LIST STOCK
SORT STOCK BY PRICE
SORT STOCK BY LOCATION BY PRICE BY-DSND QUANTITY
```

```
LIST STOCK WITH COLOUR "RED"
```

without specifying an output list, then the Access processor will look to the dictionary of the file - STOCK, in this case - to see whether there is a dataname *1*. If dataname 1 exists, then the processor will assume that you have entered:

```
LIST STOCK 1
```

and it will then look for the dataname *2*. If this exists the processor will assume that you have entered:

```
LIST STOCK 1 2
```

It will continue this procedure, looking for datanames *3*, *4* and so on, until the sequence breaks.

This means that you have to do less typing in order to produce a useful report. If you do not have such a facility for the files which you will use and would find it of value, you should ask your Systems Analyst to construct such an *implicit output list* for the most significant fields of your data items.

The datanames 1, 2, 3 and so on, may refer to any of the attributes in the data items. The dataname 1 may identify what seems to be the most important attribute, dataname 2 the next most important, and so on. There may be any number of these datanames, and you may use them just like any other datanames:

```
SORT STOCK 1 3 BY 1 TOTAL 5 TOTAL 4
```

or:

```
SORT STOCK WITH 1 = "[CHAIR]" 1 TOTAL VALUE
```

The keyword ONLY can be used to suppress these implicit datanames:

```
LIST ONLY STOCK
```

The implicit output list will also be ignored if you include any explicit datanames in the sentence:

```
LIST STOCK DESCRIPTION
```

5.6 Implicit output list on the demonstration files

If you want to create an implicit output list for the demonstration account, then you should issue the TCL command:

```
SETUP-MB-AP-ACCESS
```

and select option 1. This will set up numerical datanames which enable you to use the implicit output for the STOCK FILE. You can choose option 2 to remove the numerical datanames.

5.7 Macro and @ phrases

Besides these numerical datanames, Advanced Pick offers other ways of specifying an implicit output list. You may use:

- A **macro**: Attribute 15 of the *data-level identifier* for the file may hold a list of datanames which are to be used in the absence of any other output specification. This **macro** specification is also used by the Update Processor. Any list held here is used in preference to the numerical datanames 1, 2, 3, 4 and so on. If we were to look at the data-level identifier for the STAFF file, we might find that it looked like this:

```
015 macro     NAME INITIALS TELEPHONE DEPARTMENT
```

Your Systems Analyst will put the required list of datanames in the macro for you.

- An **@ phrase or an @filename phrase**: these are items which may be held on the dictionary of the file and have the item-ids @ or @*fffff* (where *fffff* is the name of the file). The contents of a typical phrase item might be of the form:

```
    @STOCK
01 H
02 DESCRIPTION QUANTITY PRICE TOTAL VALUE
```

The first line of a phrase contains just the letter H. The second line contains any keywords and/or field names. Thus, this phrase will be used if you type a sentence such as:

```
LIST STOCK
```

and will be equivalent to the sentence:

```
LIST STOCK DESCRIPTION QUANTITY PRICE TOTAL VALUE
```

Phrases may also be constructed with specific output lists, sort specifications and/or selection criteria, and are used exactly as described in the following Section.

5.8 General phrases

The @ and @filename items which we met in the previous section are special cases of Access phrases. A phrase is an item which is created on the dictionary of the file and has a form such as:

```
    FULL.LIST
01 H
02 DESCRIPTION QUANTITY PRICE TOTAL VALUE
```

The phrase is identified by name (FULL.LIST in this instance) and is used in a sentence. The first line consists of the letter H. The second line contains any keywords and/or field names. Thus, this phrase may be used in a sentence such as:

```
LIST STOCK FULL.LIST
```

and will be equivalent to the sentence:

```
LIST STOCK DESCRIPTION QUANTITY PRICE TOTAL VALUE
```

The phrase may include any keywords. So a phrase such as:

```
   REDS
01 H
02 WITH COLOUR = "RED"
```

```
   PRINTED
01 H
02 DESCRIPTION LOCATION QUANTITY TOTAL VALUE LPTR
```

may be used in sentences such as:

```
LIST STOCK REDS FULL.LIST
SORT STOCK REDS PRINTED
```

6 Specifying the item-ids

Each **item** on a Pick file is identified by its **item-id**. This will be a unique piece of information which identifies that item. For example, each item on the STOCK file may be identified by the *part-number* of the product concerned, each item on the STAFF file may be identified by the *employee-number* or the *clock-number* of the person to whom the information refers, and so on.

If you know any of these item-ids, you can include them in the Access sentence and only those items will appear on the report, the rest of the items on the file will be ignored. Here are some examples:

```
LIST STOCK '1000' DESCRIPTION PRICE VALUE
     list only the details of product number 1000.

LIST STOCK '1000' '8763' '1111' COLOUR MATERIAL TYPE
     list the details of products 1000, 8763 and 1111.
```

Notice that the item-ids are enclosed in **apostrophes**, although you may omit the apostrophes, provided that the sentence is not ambiguous.

If you want a *range* of item-ids, then it may be simpler to use a form like this:

```
LIST STOCK > '1000' DESCRIPTION PRICE LOCATION
```
list details of all products whose part-number is greater than 1000.

```
LIST STOCK < '2000' DESCRIPTION TOTAL VALUE
```
include only the items with item-ids lower than 2000.

```
LIST STOCK > '1000' AND < '2000' DESCRIPTION
```
list details of all products with a product number in the range 1001 to 1999 inclusive.

```
LIST STOCK = '11]' DESCRIPTION
```
list the descriptions of all products whose product number (that is the item-id) starts with 11. The square bracket means *ignore anything beyond the bracket.*

7 Selection criteria

An Access sentence will normally include *all* the items on the file when it produces the report. If you indicate that you only want to look at specific items, for example:

```
LIST STOCK '1000' DESCRIPTION
SORT STAFF = "AB]" DESCRIPTION
```

then you can restrict the report, as we saw a moment ago.

You may also want to select only those items which have a certain value for the PRICE or a certain MATERIAL. This is done by using the:

```
WITH dataname = "value"
```

clause, or the equivalent;

```
IF dataname = "value"
```

clause, like this:

```
LIST STOCK WITH MATERIAL = "PINE" DESCRIPTION PRICE
SORT STOCK WITH PRICE = "50" PRICE DESCRIPTION
SORT STAFF WITH AGE > "60" DESCRIPTION AGE
```

You can combine several selections, using the word OR:

```
LIST STOCK WITH COLOUR = "RED" OR WITH COLOUR "RED"
     DESCRIPTION PRICE LOCATION SUPPLIER
```

to show only those items which are red or blue, or you can combine several selections, using the word AND:

```
LIST STOCK WITH PRICE > "50" AND WITH COLOUR "RED"
     DESCRIPTION PRICE LOCATION SUPPLIER
```

to show only those items which cost more than £50 and which are red.

Notice that the value you are seeking is enclosed in *quotation marks*, and in place of the = and the < signs, you can use any of the following symbols:

Operator	Meaning
=	Equal to
EQ	Equal to
EQUAL	Equal to
IS	Equal to
#	Not equal to
NO	Not equal to
NOT	Not equal to
NE	Not equal to
<=	Less than or equal
LE	Less than or equal
<	Less than
LT	Less than
BEFORE	Less than
>	Greater than
GT	Greater than
AFTER	Greater than
>=	Greater than or equal
GE	Greater than or equal

Some of these forms are worth special mention:

```
WITH QUANTITY = "0"
WITH QUANTITY "0"
```

will show only those items for which the QUANTITY is 0. If the QUANTITY was null or 00 or -0 then these would not satisfy this test.

Note that, if you omit the symbol, then = is assumed.

```
:SORT STOCK DESCRIPTION QUANTITY WITH QUANTITY "0"

STOCK..... DESCRIPTION.............QUANTITY

6000       SIDEBOARD, GOLD, OAK             0
8117       SIDEBOARD, GREY, OAK             0
8763       SIDEBOARD, MAROON, ASH          10
                                            5
                                            0
                                            0
```

```
WITH NO QUANTITY
WITHOUT QUANTITY
IFNO QUANTITY
```

will show only those items which have a null QUANTITY. That is, which have no value in the QUANTITY field. If you compare this with the previous report, you will see that *not having a QUANTITY* is different from *having a QUANTITY of zero*.

```
:SORT STOCK DESCRIPTION QUANTITY WITH NO QUANTITY

STOCK..... DESCRIPTION.............. QUANTITY

4444       SETTEE, BLACK, ASH
```

```
WITH QUANTITY NE "0"
WITH QUANTITY # "0"
WITH QUANTITY NOT "0"
WITHOUT QUANTITY "0"
WITHOUT QUANTITY = "0"
IFNO QUANTITY "0"
```

will show only those items for which the QUANTITY is not 0. This will include any which have a QUANTITY of more than 0, less than 0 or a null QUANTITY.

Incidentally, you may have some difficulty finding the key which displays the character #. On some keyboards, it is represented by the *pound sign* £ and on others by the *hash sign* or *sharp sign* #. It's usually the character above the number 3 on the top row of the keyboard.

7.1 WITH EVERY / WITH EACH

As the WITH QUANTITY "0" report shows, if QUANTITY is a multivalued attribute and there is at least one value which satisfies the condition, then the item *and all its multivalues* will appear on the report. If you only want those items for which every one of the multivalues satisfies the condition, then you can use the forms:

```
WITH EVERY QUANTITY = "0"
WITH EVERY QUANTITY "0"
WITH EACH QUANTITY "0"
```

```
:SORT STOCK DESCRIPTION QUANTITY WITH EVERY QUANTITY "0"
:SORT STOCK DESCRIPTION QUANTITY WITH EACH QUANTITY "0"

STOCK..... DESCRIPTION.............. QUANTITY

6000       SIDEBOARD, GOLD, OAK             0
8117       SIDEBOARD, GREY, OAK             0
                                            0
```

If we wish to see those items which have a QUANTITY of zero but suppress any non-zero multivalues (as item 8763), then we can use the unusual form:

```
SORT STOCK BY-EXP QUANTITY "0" DESCRIPTION QUANTITY
```

```
:SORT STOCK BY-EXP QUANTITY "0" DESCRIPTION QUANTITY

STOCK..... DESCRIPTION.............. QUANTITY

6000       SIDEBOARD, GOLD, OAK              0
8117       SIDEBOARD, GREY, OAK              0
8117       SIDEBOARD, GREY, OAK              0
8763       SIDEBOARD, MAROON, ASH            0
```

7.2 Partial strings

If we were to use a sentence such as:

```
LIST STOCK WITH DESCRIPTION = "PINE" DESCRIPTION
```

then we should not find *any* items which satisfy this requirement, because our STOCK items have descriptions such as:

```
CHAIR, RED, PINE
DESK, GREY, ASH
CHAIR, ORANGE, PINE
SETTEE, YELLOW, PINE
SETTEE, BROWN, PINE
```

and there is no item with a description which consists just of the four letters PINE. In these examples, the value in quotation marks must match precisely with the value of the dataname on the file. We can produce the report we require by means of the sentence:

```
LIST STOCK WITH DESCRIPTION = "[PINE" DESCRIPTION
```

which displays details of any item of which the description *ends* in the letters PINE.

```
:LIST STOCK WITH MATERIAL = "PINE" DESCRIPTION
:LIST STOCK WITH DESCRIPTION = "[PINE" DESCRIPTION

STOCK..... DESCRIPTION.............

5500       TABLE, YELLOW, PINE
1111       STOOL, GREEN, PINE
3500       STOOL, WHITE, PINE
1500       CHAIR, ORANGE, PINE
7500       CHAIR, BLACK, PINE
9999       CHAIR, BLUE, PINE
```

Here are some more examples:

LIST STOCK WITH DESCRIPTION = "S]" DESCRIPTION
display details of any item whose description *begins* with the letter S and:

LIST STOCK WITH DESCRIPTION = "[BLUE]" DESCRIPTION
display details of any item whose description *contains* the letters BLUE in *any* position.

We are lucky because we have a dataname MATERIAL which extracts the last word of the DESCRIPTION field, so we could use the sentence:

```
LIST STOCK WITH MATERIAL = "PINE" DESCRIPTION
```

but, even without such a dataname the [and] allow us to search for just a part of a piece of data.

7.3 SAMPLING

In some situations, such as when you are testing your Access sentence on a very large file - you may wish to restrict your report to, say, the first 50 items on the file. This is done by means of the SAMPLING keyword:

```
LIST STOCK DESCRIPTION TOTAL PRICE SAMPLING 50
SORT STOCK DESCRIPTION WITH COLOUR "RED" SAMPLING 10
```

If selection criteria are included in the sentence, as in this last example, then these are evaluated before the sampling of the first twenty items takes place, reporting 20 red items.

8 Sort specifications

The LIST verb produces a report showing the items in the order that they appear on the file. On the Pick system, this is not any special order, it's just the order that the items were added to the file and changed. If we were to issue the sentence:

```
LIST STOCK DESCRIPTION
```

one day, we might get a report which starts:

```
STOCK..... DESCRIPTION.............

9000       SIDEBOARD, YELLOW, ASH          99 DN/14/70
5500       TABLE, YELLOW, PINE             15 LS/15/77
8888       DESK, GREEN, ASH                64 LS/2/23
2000       SETTEE, YELLOW, OAK             18 DN/1/69
1111       STOOL, GREEN, PINE              63 DN/6/3
4500       CHAIR, RED, LAMINATE            18 LN/3/29
```

and if we issue the same sentence another day, we might get a report with the items in a different order:

```
STOCK..... DESCRIPTION.............

6666        SIDEBOARD, ORANGE, OAK
4545        SETTEE, BLUE, MAPLE
9999        CHAIR, BLUE, PINE
9000        SIDEBOARD, YELLOW, ASH
3333        CHAIR, WHITE, LAMINATE
4500        CHAIR, RED, LAMINATE
```

If you want the report to show the items in a particular sequence, so that the product numbers (that is, the item-ids) come out in numerical order, for example, then you must use the SORT verb. Like this:

```
:SORT STOCK DESCRIPTION

STOCK..... DESCRIPTION.............

1111        STOOLS, GREEN, PINE
1500        CHAIR, ORANGE, PINE
3500        STOOL, WHITE, PINE
5500        TABLE, YELLOW, PINE
5555        TABLE, RED, PINE
7500        CHAIR, BLACK, PINE
```

If you want the report to be sorted in any other sequence, such as PRICE or DESCRIPTION, then you must use the keyword BY.

```
:SORT STOCK WITH MATERIAL = "PINE" BY PRICE DESCRIPTION PRICE

STOCK..... DESCRIPTION.............  PRICE

1111        STOOLS, GREEN, PINE        20.00
3500        STOOL, WHITE, PINE         20.00
1500        CHAIR, ORANGE, PINE        25.00
7500        CHAIR, BLACK, PINE         25.00
9999        CHAIR, BLUE, PINE          25.00
5500        TABLE, YELLOW, PINE        50.00
```

```
:SORT STOCK WITH MATERIAL = "PINE"
      BY DESCRIPTION DESCRIPTION PRICE

STOCK..... DESCRIPTION.............  PRICE

7500        CHAIR, BLACK, PINE         25.00
9999        CHAIR, BLUE, PINE          25.00
1500        CHAIR, ORANGE, PINE        25.00
3500        STOOL, WHITE, PINE         20.00
1111        STOOLS, GREEN, PINE        20.00
5555        TABLE, RED, PINE           50.00
```

The BY keyword outputs the data in *ascending order*. If you are sorting by a right-justified field (typically, with numeric fields such as the PRICE), then the prices will appear in ascending order. If you are sorting by a left-justified field (typically, with text fields such as the DESCRIPTION), then the descriptions will appear in ascending alphabetical order with all the A's will come before all the B's, and so on.

Incidentally, even though you are sorting the report in price sequence, the PRICE won't automatically appear in the report; you have to include PRICE in the output list, as we did.

If you want the highest prices first, then you can sort the report in *descending* order, like this:

```
SORT STOCK BY-DSND PRICE WITH PRICE > "20" PRICE COLOUR
```

Take care that you spell BY-DSND correctly.

If some of your attributes contain more than one piece of data - these are known as **multivalues** - then you can use the BY-EXP keyword to **explode** the dataname into its separate multivalues, like this:

```
SORT STOCK BY-EXP QUANTITY DESCRIPTION QUANTITY
```

```
:SORT STOCK BY QUANTITY DESCRIPTION QUANTITY

STOCK..... DESCRIPTION.............. QUANTITY

1000       DESK, GREEN-BLUE, ASH             8
8763       SIDEBOARD, MAROON, ASH           10
                                             5
                                             0
1200       DESK, GREY, ASH                  16
                                            32
2000       SETTEE, YELLOW, OAK              18
4200       SETTEE, GOLD, ASH                55
                                            10
[405] 5 items listed out of xxx items.
```

```
:SORT STOCK BY-EXP QUANTITY DESCRIPTION QUANTITY

STOCK..... DESCRIPTION.............. QUANTITY

8763       SIDEBOARD, MAROON, ASH            0
8763       SIDEBOARD, MAROON, ASH            5
1000       DESK, GREEN-BLUE, ASH             8
4200       SETTEE, GOLD, ASH                10
8763       SIDEBOARD, MAROON, ASH           10
1200       DESK, GREY, ASH                  16
2000       SETTEE, YELLOW, OAK              18
1200       DESK, GREY, ASH                  32
4200       SETTEE, GOLD, ASH                55
[405] 9 items listed out of xxx items.
```

Without the BY-EXP keyword, only the first multivalue is used for sorting purposes, although the other multivalues appear in the order in which they are held on the file. Look at the entries for item 8763 and compare these two reports.

In the first of these reports, only the first multivalue is used for the sort. In the second report where we use the BY-EXP keyword, the individual multivalues are taken apart before they are sorted independently.

You may even sort the exploded multivalues into descending order:

```
SORT STOCK BY-EXP-DSND QUANTITY DESCRIPTION QUANTITY
```

Normally, any sort specifications are ignored if they are used with:

```
LIST
SELECT
LIST-LABEL
```

or other non-sorting verbs. Exceptions are [M] and [AP] where the use of a BY keyword automatically uses the sorting form of the verb, so a sentence such as:

```
LIST STOCK BY COLOUR PRICE DESCRIPTION
```

is identical to:

```
SORT STOCK BY COLOUR PRICE DESCRIPTION
```

8.1 BREAK-ON and control-breaks

```
:SORT STOCK BY COLOUR DESCRIPTION TOTAL VALUE BREAK-ON COLOUR

STOCK..... DESCRIPTION.............    VALUE COLOUR....

4000       DESK, BLACK, MAPLE         3808.00 BLACK
4444       SETTEE, BLACK, ASH            0.00 BLACK
7500       CHAIR, BLACK, PINE         2125.00 BLACK

                                      5933.00 ***
:
2000       SETTEE, YELLOW, OAK        1800.00 YELLOW
5500       TABLE, YELLOW, PINE         750.00 YELLOW
7777       STOOL, YELLOW, LAMINATE     443.75 YELLOW
9000       SIDEBOARD, YELLOW, ASH    12870.00 YELLOW

                                     15863.75 ***

***                                  91437.00
```

When you use one of the BY keywords to sort your report into a

special sequence, such as:

```
BY COLOUR
```

or:

```
BY DEPARTMENT
```

you may find that the report looks better if you skip a few lines between the various colours or the various departments, thereby making it easier to identify the various sections of the report. Such a visual effect is known as a **control-break** and is produced by using a clause such as:

```
BREAK-ON COLOUR
```

or:

```
BREAK-ON DEPARTMENT
```

in your Access sentence, like this:

```
SORT STOCK BY COLOUR DESCRIPTION PRICE BREAK-ON COLOUR
```

or:

```
SORT STAFF NAME START-DATE BY DEPARTMENT BY NAME BREAK-ON
     DEPARTMENT
```

Note that, when you use the BREAK-ON keyword, the break field, that is, the dataname used in the BREAK-ON clause - COLOUR in this last example - will automatically be included in the output list for your report. This means that, when you ask for:

```
BREAK-ON COLOUR
```

the COLOUR will appear on the report with the control-break signal each time the COLOUR changes from BLACK to BLUE, from BLUE to ..., and so on. If you have a sentence such as:

```
SORT STOCK BY COLOUR DESCRIPTION COLOUR BREAK-ON COLOUR
```

then you will get the COLOUR column appearing *twice* in your report.

8.1.1 BREAK-ON options

You can replace the asterisks which are displayed at each control-break by using a form such as:

```
BREAK-ON COLOUR " "
```

which will suppress the asterisks (it actually replaces them by a space), or you could use:

```
BREAK-ON DEPARTMENT "==============="
```

which will print a string of = signs (or any other message you wish) at each control-break.

Within the "xxxxx" message after the BREAK-ON keyword, you can call up other special facilities. These include:

'' to output a single apostrophe.

B to tie up with the B option of an associated HEADING or FOOTING, and inserts the break value into that HEADING or FOOTING. We discuss this in a separate section later.

D to suppress the control break indicator when there is only one detail line.

L to suppress the blank line before the control break message. Not used with U option.

N to set the page-number to 1 at each control break.

P to skip to a new page at this break.

R to suppress roll-over and output two or more simultaneous control breaks on the same page.

U to underline any TOTAL fields at the control break.

V to output the break-value at the control break.

Here are some examples:

```
BREAK-ON DEPARTMENT "====='P'"
     will skip to a new page at each control-break.

BREAK-ON DEPARTMENT "SUMMARY'U'"
     will display the word SUMMARY and underline any subtotals
     produced at the control-break.
```

These same forms are available with the ROLL-ON keyword.

You may have several control-breaks in a sentence:

```
SORT STAFF BY SITE BY DEPARTMENT BY NAME BREAK-ON DEPOT
     BREAK-ON DEPARTMENT
```

Take care if you do this, because the Access processor will produce a control-break whenever there is a change of DEPOT and whenever there is a change of DEPARTMENT: you may get more control-breaks than you expect.

8.1.2 ROLL-ON

If you want to BREAK-ON COLOUR but you don't want the COLOUR to appear, then you can use the ROLL-ON keyword (or the alternative TOTAL-ON keyword):

```
SORT STOCK BY COLOUR DESCRIPTION ROLL-ON COLOUR
```

or, better (for reasons we'll see in a moment):

```
SORT STOCK BY COLOUR DESCRIPTION ROLL-ON COLOUR " "
```

The ROLL-ON modifier **rolls** the output data into the following column, the previous column or as a new column. The format of the ROLL-ON modifier is the same as the BREAK-ON modifier:

```
ROLL-ON attribute {"message"}
```

where the optional *message* consists of any string of text and/or options - the options being enclosed in apostrophes.

The annotated output below illustrates how several values may be rolled into one column, each being identified by a string of asterisks.

```
:SORT STOCK DESCRIPTION ROLL-ON PRICE ROLL-ON MINIMUM

STOCK..... DESCRIPTION.............

1500       CHAIR, BLUE, LAMINATE        <== DESCRIPTION
           * 60                         <== MINIMUM
           ** 25.00                     <== PRICE

2234       CHAIR, GREEN, OAK
           * 30
           ** 10.00

9555       TABLE, RED, PINE
           * 25
           ** 50.00
```

The ROLL-ON modifier also has the effect of BREAK-ON without displaying the break-field data:

```
:SORT STOCK BY COLOUR DESCRIPTION ROLL-ON COLOUR " "

STOCK..... DESCRIPTION.............

4000       DESK, BLACK, MAPLE
4444       SETTEE, BLACK, ASH
7500       CHAIR, BLACK, PINE


2222       DESK, BLUE, OAK
3000       SIDEBOARD, BLUE, ASH
4545       SETTEE, BLUE, MAPLE
9999       CHAIR, BLUE, PINE
```

8.1.3 DET-SUPP and BREAK-ON

If you use DET-SUPP and BREAK-ON in the same sentence, the detail lines will be suppressed and - at each control-break - there will appear any subtotals for any TOTALled fields, and for other fields there will appear the previous entry in that column (that is the last entry just before the control-break); the item-ids will not appear. Compare this report with that in the BREAK-ON Section above which omitted the DET-SUPP keyword.

```
:SORT STOCK BY COLOUR DESCRIPTION TOTAL VALUE
      BREAK-ON COLOUR DET-SUPP

STOCK..... DESCRIPTION.............     VALUE COLOUR....

           CHAIR, BLACK, PINE          5933.00 BLACK

           CHAIR, BLUE, PINE          13408.00 BLUE
:
           SIDEBOARD, YELLOW, ASH     15863.75 YELLOW

**                                    91437.00
```

9 TOTAL - totals and subtotals

Some of your datanames will represent numbers - PRICE, QUANTITY, VALUE and so on. If you want to add up these amounts as they are displayed and show the totals at the end of the report, then you just include:

```
     TOTAL VALUE
or:
     TOTAL QUANTITY
```

in your Access sentence. Here are some examples.

```
     LIST STOCK DESCRIPTION PRICE TOTAL VALUE
     LIST STOCK DESCRIPTION TOTAL QUANTITY PRICE TOTAL VALUE
```

If you use the BREAK-ON keyword in your report, then the subtotals of your TOTALled fields will be shown at each control-break. Some typical output - including the subtotals generated at the control-breaks - can be seen in the previous section.

9.1 GRAND-TOTAL

This last example demonstrates that there is a grand-total shown at the foot of the report beneath the QUANTITY and the VALUE columns. This grand-total line is highlighted by having two asterisks at the left of the page. If you want some special message to appear at the foot of the report alongside the grand-total instead of the asterisks, then you would use the:

```
GRAND-TOTAL "text"
```

clause in your sentence, like this:

```
SORT STOCK BY COLOUR DESCRIPTION QUANTITY PRICE TOTAL
     VALUE GRAND-TOTAL "Final figures"
```

to produce a report like this:

```
STOCK..... DESCRIPTION.............. QUANTITY  PRICE    VALUE

4000       DESK, BLACK, MAPLE              68  56.00  3808.00
4444       SETTEE, BLACK, ASH                  10.00     0.00
7500       CHAIR, BLACK, PINE              85  25.00  2125.00
2222       DESK, BLUE, OAK                 98  56.00  5488.00
3000       SIDEBOARD, BLUE, ASH            58 130.00  7540.00
4545       SETTEE, BLUE, MAPLE              8  10.00    80.00
:
7777       STOOL, YELLOW, LAMINATE         25  17.75   443.75
9000       SIDEBOARD, YELLOW, ASH          99 130.00 12870.00
Final figures                                       93437.00
```

9.1.1 GRAND-TOTAL options

As with the BREAK-ON clause, you can call up other special facilities within the "xxxxx" message after the GRAND-TOTAL keyword. For example:

GRAND-TOTAL "End of report'P'"
: will skip to a new page before printing the final figures.

GRAND-TOTAL "SUMMARY'U'"
: will display the word SUMMARY and underline the grand-total figure.

GRAND-TOTAL "Final figures'UP'"
: will skip to a new page before printing the final figure and then underline the grand-total.

EPick describes these in detail.

10 LPTR - printed or displayed?

We have seen that you will include the word:

```
LPTR
```

if you want to produce the report on the printer instead of displaying it on the screen. The word is actually an abbreviation for *Line-PrinTeR*.

Here are some examples of LPTR in action:

```
LIST STOCK DESCRIPTION PRICE LPTR
LIST STOCK LPTR DESCRIPTION PRICE
```

Like the other building blocks which make up an Access sentence, LPTR can come anywhere in the sentence after the verb.

10.1 LPTR or (P?

A shorthand way of achieving the same effect is by using the option:

```
(P
```

at the *very end of the sentence:*

```
LIST STOCK DESCRIPTION PRICE (P
```

Strictly speaking, LPTR is an **Access keyword** and may appear *anywhere* within an Access sentence; LPTR cannot be used in TCL commands. The (P element in just one example of a TCL **option** and can appear in any TCL and any Access sentence but it must appear *at the very end*. Thus, the TCL commands:

```
WHO (P
COPY STOCK * (P
```

are acceptable, but the forms:

```
WHO LPTR
COPY STOCK * LPTR
```

are not acceptable because WHO and COPY are not Access verbs.

We shall look at Access options in a moment.

10.2 LEGEND

Advanced Pick offers the LEGEND facility which allows the System Manager to specify a piece of text which is to appear at the foot of every Access report which is sent to the printer. The text of this system-wide legend is specified (in Output Processor format) in an item with the item-id LEGEND which is held on the MESSAGES file. A legend for a specific user may be placed on the MESSAGES file with the item-id:

```
LEGEND*user.id
```

A simple legend might include your company name, like this:

```
01
02 ------------------------------------------------------------------
03       Malcolm Bull / Training & Consultancy Services
```

The inclusion of this message in Access reports is enabled/disabled by the LEGEND-ON / LEGEND-OFF commands at TCL and by the LEG-SUPP keyword (or the equivalent K option) in Access sentences.

10.3 TCL-SUPP

You've probably noticed that, when you use an Access sentence to produce a report *on the printer*, the sentence is shown at the head of the report. The action may be disabled/enabled for all Access reports by means of the TCL-HDR-ON and TCL-HDR-OFF commands and, for an individual report, by the TCL-SUPP keyword. The simple TCL-HDR will display the current status of the TCL headers.

11 HEADINGs and FOOTINGs

Access will automatically produce a **heading** for your report, showing the page number, the date and the time. But you may specify your own heading, if you wish. Like this:

```
SORT STOCK DESCRIPTION PRICE HEADING "Stock Report"
```

and the words:

```
Stock Report
```

will replace the standard heading which you get on your reports showing the date and the time.

If you wish, you can have your own heading at the top of each page and/or your own **footing** at the bottom of each page, as we see in a moment.

11.1 HEADING/FOOTING options

Within the text of the heading or the footing, you can enclose special codes in apostrophes, as I did in the last example. Some of these codes are:

B to output the BREAK-ON (or ROLL-ON) value, as discussed later.

C to output the text in the centre of the line.

D to output the current date in the format: 01 JAN 1999

L to start a new line.

PN to output the page-number with no surrounding spaces.

T to output the current date and time in the format: 09:30:59 01 JAN 1999

If there are two adjacent codes, then you will enclose the pair in one set of apostrophes:

```
:SORT STOCK BY PRICE DESCRIPTION PRICE
     HEADING "'C'Stock Report'LC'============'L'"
     FOOTING "End of page 'PN' of the report"
```

```
                                   Stock Report
                                   ============

STOCK..... DESCRIPTION.............  PRICE

2936       SETTEE, GREY, OAK           7.00
1234       CHAIR, RED, OAK            10.00
4545       SETTEE, BLUE, MAPLE        10.00
7777       STOOL, YELLOW, LAMINATE    17.75
:
1111       STOOLS, GREEN, PINE        20.00
7500       CHAIR, BLACK, PINE         25.00
9999       CHAIR, BLUE, PINE          25.00

End of page 1 of the report
```

If your text is to include an apostrophe, then you will represent this by two apostrophes:

```
HEADING "'C'** Mary O''Cahill''s report **'L'"
```

A list of the options which are available can be found in the reference section at the end of this *MB-Guide*.

11.2 The B option

The B option is handy when used in both the HEADING/FOOTING clauses and in the BREAK-ON (or ROLL-ON) clause because it allows you to incorporate the break value into the heading (or footing). So the sentence:

```
SORT STOCK BY COLOUR ROLL-ON COLOUR " 'BP'" DESCRIPTION
     QUANTITY PRICE TOTAL VALUE HEADING "'C'** Report for
     'B' products **"
```

might produce a report which looks like that shown opposite.

The B option used in the ROLL-ON clause ties up with the B option used in the HEADING, and whenever the colour changes the new colour is inserted into the heading (and we skip to a new page because of the P option in the ROLL-ON clause).

```
                          ** Report for BLACK products **

STOCK..... DESCRIPTION..... QUANTITY  PRICE    VALUE

4000       DESK, BLACK, MAPLE              68  56.00  3808.00
4444       SETTEE, BLACK, ASH                  10.00     0.00
7500       CHAIR, BLACK, PINE              85  25.00  2125.00

                                                      5933.00
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                          ** Report for BLUE products **

STOCK..... DESCRIPTION..... QUANTITY  PRICE    VALUE

2222       DESK, BLUE, OAK                 98  56.00  5488.00
3000       SIDEBOARD, BLUE, ASH            58 130.00  7540.00
4545       SETTEE, BLUE, MAPLE              8  10.00    80.00
9999       CHAIR, BLUE, PINE               12  25.00   300.00

                                                      13408.00
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
:
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                          ** Report for YELLOW products **

STOCK..... DESCRIPTION..... QUANTITY  PRICE    VALUE

2000       SETTEE, YELLOW, OAK             18 100.00  1800.00
5500       TABLE, YELLOW, PINE             15  50.00   750.00
7777       STOOL, YELLOW, LAMINATE         25  17.75   443.75
9000       SIDEBOARD, YELLOW, ASH          99 130.00 12870.00

                                                      15863.75

**                                                    93437.00
```

11.3 Other modifiers

There are a number of other keywords - or **modifiers** - in addition to those we have already mentioned, and these will change the normal form of an Access report or modify the normal action of the Access processor. We have already met GRAND-TOTAL and BREAK-ON and BY. Other modifiers which you may find useful are:

ID-SUPP
: suppress the item-id which is normally shown in the first column of the report. When ID-SUPP is used, only the datanames which you request (or which are implied by the TOTAL and BREAK-ON modifiers) will appear.

HDR-SUPP
: cancel the normal page heading. You don't need this if you are using the HEADING modifier.

COL-HDR-SUPP
: stop both the page heading and the column headings being displayed.

DET-SUPP
: As we have seen, this is useful when you are using BREAK-ON modifier and wish to stop the display of the detail lines within each subsection of your report.

DBL-SPC
: print a double-spaced report, skipping a blank line between each item of the report.

NI-SUPP
: suppress the count of the number of items shown at the end of the report.

NOPAGE
: override the normal action of pausing at the foot of each page when displaying a report on the screen. You might use this if you are only interested in the last page of a report.

As you might expect, you can put these keywords anywhere in your Access sentence. Here are some examples of their use:

```
LIST STOCK DESCRIPTION PRICE ID-SUPP
LIST STOCK DBL-SPC NI-SUPP
SORT STOCK BY TYPE BREAK-ON TYPE TOTAL PRICE DET-SUPP
SORT STOCK COL-HDR-SUPP ID-SUPP DBL-SPC
LIST STOCK ID-SUPP DET-SUPP
ISTAT STOCK NOPAGE
```

11.4 Throwaway modifiers

There is another set of keywords - known as **throwaway modifiers** or **throwaway connectives** - which are provided in order to enable your users to write more natural English sentences. Thus, the sentence:

```
LIST THE STOCK FILE WITH A QUANTITY OF "50"
```

(where I've underlined the *throwaways*) is equivalent to:

```
LIST STOCK WITH QUANTITY "50"
```

The list of standard throwaways includes:

A	ANY	ARE	DATA
FILE	FOR	IN	ITEMS
OF	OR	THE	

and you can use these anywhere within a sentence to make it read

more naturally. Throwaways are ignored by the Access processor. The command:

```
LISTCONN
```

or:

```
LISTCONN (P
```

will produce a list of the *modifiers* (or *connectives*) and other keywords which are available with Access sentences.

12 Access options

A range of options are available for use with Access sentences. Some of these are abbreviations for some of the keyword modifiers which we met earlier.

C to suppress the default page headings and column headings. [This is equivalent to the COL-HDR-SUPP modifier]

D to suppress the detail lines between control breaks. [DET-SUPP]

H to suppress the default page heading. This will also suppress the creation of labels on the diskette/tape when used with T-DUMP. [SUPP and HDR-SUPP].

I to suppress the output of the item-id. [ID-SUPP]

N to suppress the end-of-page pause when displaying on the screen. [NOPAGE]

P to send the output to the printer. [LPTR]

A full list of the available options is shown in the reference section at the end of this *MB-Guide*.

Like all TCL options, Access options are specified *at the end of the sentence*, for example:

```
LIST STOCK (P
LIST STOCK (P)
LIST STOCK WITH COLOUR "RED" (P
SORT STOCK BY PRICE DESCRIPTION TOTAL VALUE (CI
```

The options may appear in any order, with or without separating commas and with or without the closing parenthesis.

13 Lists and SELECT / SSELECT

When you issue a sentence such as:

```
LIST STOCK WITH EVERY QUANTITY <= "0" DESCRIPTION MINIMUM
     LOCATION
```

the Access processor will read every item on the file to discover whether every quantity for that particular product is zero. If you need to process these same items again with a second sentence:

```
SORT STOCK WITH EVERY QUANTITY <= "0" BY DEPOT
     DESCRIPTION BREAK-ON DEPOT
```

then the Access processor will have to read all the items on the file once more. It will have to read all the items on each occasion, even though only a small percentage of the items may be eligible for inclusion in the report. If you are going to process the same subset of the file - the out-of-stock items, in our case - then you can save yourself a lot of time by building a list of the item-ids of eligible items, saving this list and then recalling it when you need to handle just those out-of-stock items.

You will create and save the list of item-ids by a sequence such as:

```
SELECT STOCK WITH EVERY QUANTITY <= "0"
```

and then immediately issuing a command such as:

```
SAVE-LIST DANGER
```

to save the list of eligible item-ids with the name DANGER. You can use any name here. The lists are saved on a special file called POINTER-FILE which is provided just to hold such lists.

You may then recall and use this **saved-list** whenever you need it. This is done by means of a sequence such as:

```
GET-LIST DANGER
```

followed by the Access sentence which is to process these items:

```
LIST STOCK DESCRIPTION MINIMUM
```

or by a sequence such as:

```
GET-LIST DANGER
SORT STOCK BY MINIMUM DESCRIPTION BREAK-ON MINIMUM
```

The list DANGER is now permanently saved - although it won't change if the stock levels of any other items fall or if new deliveries are received of any out-of-stock items - so from time to time you may need to repeat the:

```
SELECT STOCK WITH EVERY QUANTITY <= "0"
SAVE-LIST DANGER
```

sentences to create the saved-list again.

There is also the SSELECT verb (note the double S). This produces a list of item-ids which are sorted according to any BY specifications in the SSELECT sentence:

```
SSELECT STOCK BY DEPOT WITH EVERY QUANTITY <= "0"
```

Other verbs associated with such lists are:

EDIT-LIST DANGER
: create a new list or to change a list which you saved earlier.

COPY-LIST DANGER (T
: display the contents of a list.

COPY-LIST DANGER (P
: print the contents of a list.

COPY-LIST DANGER
: copy an existing list and saved it under another name, or save it as an item on an ordinary file.

DELETE-LIST DANGER
: delete an existing list. It is good practice to delete any lists when you no longer need them. thereby saving space on the system.

LIST-LISTS
: display the names of the lists which are currently saved.

The saved-lists are normally saved on the POINTER-FILE. But if you want to save them on one of your own files, you simply include the name of the file in the command:

```
SAVE-LIST MYLISTS DANGER
GET-LIST MYLISTS DANGER
DELETE-LIST MYLISTS DANGER
```

and on so. It's a good plan to save the lists on your own file if you want to keep them long-term.

13.1 QSELECT

The special verbs:

```
QSELECT filename itemname
```

or the equivalent:

```
GET-LIST filename itemname
```

will fetch a specified item from an ordinary file and use the individual attributes to produce a select list, in much the same

way as the SELECT and SSELECT verbs create a select list.

You can create and save as many saved-lists as you need.

Whenever you save a list, you should make a written note of the name of the list, when you created it, and what it is used for. You should also be sure the use the DELETE-LIST to remove any lists which are no longer needed.

14 Typing Access sentences

Typically, Access is used by typing the sentences in at the keyboard. In general, you may type all the words in UPPER-CASE letters:

```
LIST STOCK WITH COLOUR "RED" TOTAL VALUE DESCRIPTION
```

or in lower-case, such as:

```
list stock with colour "red" total value description
```

You will probably be familiar with most of the features of the keyboard, especially <Return> or <Enter> or ^M to send the sentence to the computer.

There are a number of standard features which you can use to correct the sentence before you hit <Return>:

```
^J   cursor left one character
^K   cursor right one character
^L   delete a character
^Y   cursor left one word
^U   cursor right one word
^O   delete a word
^W   insert a blank space
^H   backspace and delete (the <Backspace> can be used)
^R   enable/disable text insertion
```

14.1 <Ctrl>X to abandon a report

When you are displaying a long Access report and this pauses at the bottom of the page waiting for you to press <Return>, you may terminate the report at this point by entering:

```
<Ctrl>X
```

instead of <Return>.

14.2 TCL stacker

When you are typing in TCL commands and Access sentences, you will soon realize how easy it is to make a mistake, and there

will be many occasions when you have to re-type a very long sentence because you made a mistake or because you want to add another clause to it.

Advanced Pick overcomes this by providing a **TCL stacker**. This is a piece of software which stores each TCL command and Access sentence which you type in at TCL, and holds them in a stack, allowing you to recall these commands and sentences later - correcting them, if necessary - and issue them again. The commands in the stack can be inspected by the following keys and key sequences:

- You can browse through the stack:

  ^D recall the last command. You can repeat this to browse down the stack.

  ^F recall the next command. You will use ^D and ^F to browse up and down the stack.

- When you reach the command you want, you can just hit <Return> or the equivalent ^M key to issue that command.

- You can search the stack looking for a specific substring by means of the ^A key:

  STOCK^A will search for the most recent command which contains the string STOCK

  ^A will repeat the last search, looking for the next most recent command.

- When a command is displayed, you may amend it by using the cursor movement and editing keys:

^J	cursor left one character
^K	cursor right one character
^L	delete a character
^Y	cursor left one word
^U	cursor right one word
^O	delete a word
^W	insert a blank space
^R	enable/disable text insertion

- You can delete a command by means of the ^E key. This is useful for reducing the size of the stack.

- You can leave the inspection of the stack by means of the ^X key.

There is no simple means of inspecting the contents of your stack, although the command:

```
LIST-ITEM TCL-STACK uuuuu
```

will show you the contents of the stack, most recent last (where *uuuuu* is your user-id). Advanced Pick holds all the users' stacks on the TCL-STACK file.

14.3 The ^ keys

Each of these special keys - and the others we'll meet later - is entered by holding the down the <Ctrl> key where I've shown the ^ character and then pressing the next letter whilst the <Ctrl> is held down. So, to enter:

```
^K
```

you will held down the <Ctrl> key, press the K key, and then release both <Ctrl> and K; to enter:

```
^XF
```

you will held down the <Ctrl> key, press the X key, release both <Ctrl> and X, and then press the F key. Don't hit <Return> in either case.

15 Saving Access sentences

We have seen that Access can be used simply by typing in your sentences at the keyboard. However, this can be laborious if it is a long sentence and especially if you are going to use the same sentence over and over again; even more so if you are a slow or inaccurate typist!

We have seen that a TCL stacker is one solution, but a more permanent solution to is save the sentence - giving it some meaningful name such as STOCK.REPORT or MARY.REPORT or FRIDAY.STOCK.LIST - and then you can recall the sentence whenever you want simply by typing in the name:

```
STOCK.REPORT
```

or:

```
FRIDAY.STOCK.LIST
```

There are several ways of doing this:

- You can create a macro, or
- You can create a menu to let you select which of several sentences you invoke, or
- You can create a Proc.

In general, a macro is the simplest way of doing this.

15.1 Creating a macro: CREATE-MACRO

If you have typed in an Access sentence, you can easily convert this into a permanent macro by issuing a command such as:

```
CREATE-MACRO FRIDAY.REPORT
```

This will take the last command which you typed in - it finds this in your TCL stack - and save it on the MD file with the name FRIDAY.REPORT. Whenever you want to issue this command, you simply type in the name which you saved it with:

```
FRIDAY.REPORT
```

The command will be displayed and when you press <Return> it will be executed. Before pressing <Return>, you can use the ^ keys to change it, if you wish.

If you were to look at the macro which has been created on the MD, it would look like this:

```
    FRIDAY.REPORT
01 M
02 LIST STOCK WITH COLOUR "RED" AND WITH PRICE > "50"
```

with line 02 containing the last TCL command or Access sentence which you issued.

If we had created the macro with the command:

```
CREATE-MACRO FRIDAY.REPORT (N
```

then the macro would have had N on line 1, like this:

```
    FRIDAY.REPORT
01 N
02 LIST STOCK WITH COLOUR "RED" AND WITH PRICE > "50"
```

We'll discuss the differences in a moment.

15.2 Creating a macro: using the Update Processor

If you wish to create your own macros, they will have this same form. You can use the Update Processor to create and change your macros:

```
UPDATE MD macro.name
```

where *macro.name* is the name you have chosen for your macro. It's good practice to use your own initials as a part of the

macro name. You will then simply type in the sentences which are to make up your macro. Remember the first line must start with M and may be followed by a space and any comments to remind you why you created the macro. The following Update Processor keys will help you change and correct your macro:

```
    ^J   move the cursor left one character
    ^K   move the cursor right one character
    ^L   delete a character
    ^Y   move the cursor left one word
    ^U   move the cursor right one word
    ^O   delete a word
    ^W   insert a blank space
    ^R   enable/disable text insertion

    ^XF  save the macro on the MD
    ^XX  abandon the macro without making any changes.
```

Any macro may include as many Access sentences and/or TCL commands as you wish; each separate command and sentence will be held on a separate line. If a macro contains, say, three Access sentences, like the following example, then these will be executed in turn when you call up the macro.

```
     FRIDAY.REPORTS
01 N
02 LIST STOCK WITH COLOUR "RED" AND WITH PRICE > "50"
03 SORT STOCK BY-EXP LOCATION DESCRIPTION LOCATION LPTR
04 SORT STOCK BY-EXP LOCATION DESCRIPTION LOCATION QUANTITY
          PRICE TOTAL VALUE ROLL-ON DEPOT " " LPTR
```

You might set up a macro to create and save a list:

```
     MAKE.DANGER.LIST
01 N
02 SSELECT STOCK WITH QUANTITY < "10"
03 SAVE-LIST LOWQTY
```

The first line of a macro can be either M or N. If it is M, then each sentence will be presented to you before it is executed; this gives you a chance to change the sentence if you want. If the first line is N, then the sentences will be executed without any intervention from the user.

15.3 A menu

A menu is rather like a macro except that you are invited to choose which of the commands and sentences you want to execute. You create menus on the MD by means of the Update Processor. A typical menu definition might look like this:

```
   STOCK.MENU
01 ME The main stock control menu for the stores
02 Stock Control Processing
03 Display stock items
   Display a list of the records on the STOCK file
   SORT STOCK BY PART.NO
04 Print stock items
   Produce a printed list of the records on the STOCK file
   SORT STOCK BY PART.NO LPTR
05 Add new stock records
   Add new records to the STOCK file
   RUN STOCK.BP ADD.REC
06 Archive stock records
   Archive out-of-date records on the STOCK file
   PROMPT Mount 1.44MB diskette in drive A:
   SET-FLOPPY (AH
   T-REW
   T-DUMP STOCK WITH AGE > "366"
   T-DET
   DISPLAY Remove diskette
07 Delete stock records
   Delete old records from the STOCK file
   RUN STOCK.BP DEL.REC
```

This has the identifier ME on line 1, a title on line 2, and then each of the remaining lines contains three or more *multivalues*, one set of multivalues for each menu choice:

- A description which is to appear on the displayed menu,
- An explanatory message, and
- A sequence of one or more TCL commands and Access sentences which are to be invoked when you select that choice from the menu.

You will use the ^V key to start a new multivalue on any line.

To call up this particular menu, we should type in the name:

```
STOCK.MENU
```

and the following screen would be displayed:

```
                        Stock Control Processing

1) Display stock items              4) Archive stock records
2) Print stock items                5) Delete stock records
3) Add new stock records

Enter number of choice, number? for Help, <Return> to exit
menu, or verb:
```

The users may make any choice, 1, 2, 3, 4 or 5, or they may ask for the help message to be displayed:

```
2?
```

and they may press <Return> to abandon the menu.

15.4 A Proc

The use of Procs is not quite so simple. I only introduce them here because some older system might already have Procs which you may have to master. As with a macro, a Proc is held on the MD and can be created and maintained by means of the Update Processor. If we were to write the above macros as Procs, they would look like this:

```
   FRIDAY.REPORT.2
01 PQ
02 H LIST STOCK WITH COLOUR "RED" AND WITH PRICE > "50"
```

```
   FRIDAY.REPORTS.2
01 PQ
02 H LIST STOCK WITH COLOUR "RED" AND WITH PRICE > "50"
03 P
04 H SORT STOCK BY-EXP LOCATION DESCRIPTION LOCATION LPTR
05 P
06 H SORT STOCK BY-EXP LOCATION DESCRIPTION LOCATION
   QUANTITY PRICE TOTAL VALUE ROLL-ON DEPOT " " LPTR
07 P
```

```
   MAKE.DANGER.LIST.2
01 PQ
02 H SSELECT STOCK WITH QUANTITY < "10"
03 STON
04 H SAVE-LIST LOWQTY
05 P
```

Note that, because the SAVE-LIST command *feeds* the SSELECT sentence in this last example, you must use the STON statement on line 3, and not the P as you might expect.

16 LIST-LABEL / SORT-LABEL - printing labels

The LIST-LABEL and SORT-LABEL verbs are provided to allow you to produce a report in **label-format**. For example, the sentence:

```
SORT-LABEL STOCK DESCRIPTION PRICE
```

might be used to produce output like that shown below, instead of the more familiar column-format. The most obvious application is for printing sticky labels, such as might be used for names and addresses. But you can, of course, use the verbs to output to the printer on plain stationery and to the screen.

```
1000                        1111
DESK, GREEN-BLUE, ASH       STOOLS, GREEN, PINE
56.00                       20.00

1234                        1500
CHAIR, RED, OAK             CHAIR, ORANGE, PINE
10.00                       25.00

2200                        2222
SETTEE, BROWN, OAK          DESK, BLUE, OAK
10.00                       56.00
```

The syntax of the LIST-LABEL and SORT-LABEL sentences is identical to that of an ordinary Access sentence. The LIST-LABEL and SORT-LABEL verbs, however, require additional information:

1) The number of labels across the page. We used the value 2 in the example shown here. This is represented by parameter *a* in the diagram below.

2) The number of lines per label. We used the value 3 in the example shown here. This is the same as the number of columns which would appear on the equivalent LIST or SORT sentence. (Parameter *b*)

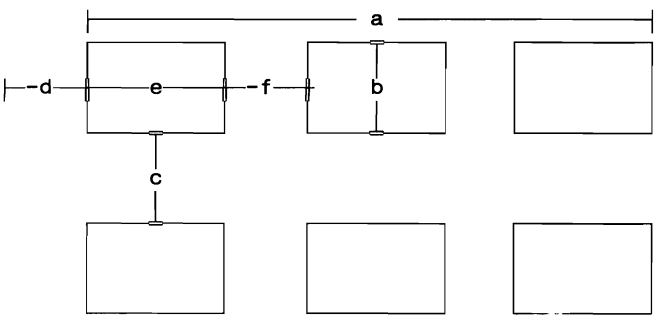
3) The number of blank lines between the labels. We used the value 1 in the example shown here. (Parameter *c*)

4) The width of the left-hand margin, that is, the number of spaces by which the labels are to be indented. We used the value 0 in the example shown here. (Parameter *d*)

5) The number of characters per label width. We used the value 26 in the example shown here. (Parameter *e*)

6) The number of horizontal spaces between the labels. We used the value 0 in the example shown here. (Parameter *f*)

Visually, these *parameters* refer to the follow details of the layout:



These values are entered when invited by the label-processor *after* the sentence has been entered. The parameters must be entered on one line when the processor displays the:

```
?
```

prompt, and are separated by commas:

```
2,3,1,0,26,0
```

to produce the report shown above. If you make a mistake, then the processor will abort with a message such as:

```
THE RANGE OF THE PARAMETER "" IS NOT ACCEPTABLE
```

Because of the complex nature of these parameters, the label processor is generally invoked by a macro, a Proc or a Basic program with the parameters specified correctly. This is particularly important where the facility is to be made available to end-users. A macro to produce the labels shown here might look like this:

```
    STOCK.LABELS
01 N
02 SORT-LABEL STOCK DESCRIPTION PRICE
    2.3.1.0.26.0      <== This is the second value of line 02
```

Note that because the parameters *feed* the SORT-LABEL sentence, these must be entered as a multivalue of line 02 (remember to use ^V to enter multivalues).

If you use the fourth parameter (d) to indicate that you wish to *indent* the output, for example:

```
2,3,1,10,26,0
```

Then you will be asked to enter, in this case, up to ten characters of text for each of the three lines of the label. You will specify the headings one at a time when invited. These headings are only shown in front of the left-most label; labels in the second and later columns do not have the headings.

As another example, the following Access sentence and parameters will produce a four-column listing of the item-ids on the STOCK file:

```
:SORT-LABEL ONLY STOCK
?4,1,0,0,19,0
```

If we wanted to produce this report frequently, it would be more convenient to write it as a macro:

```
   PRODUCE.LABELS
01 N
02 SORT-LABEL ONLY STOCK
   4,1,0,0,19,0                          <== A value of line 02
```

17 Practical Access

There are a few pieces of practical advice which might be of interest to anyone who is coming to use Access for the first time.

1) Be sure that you know which files are available to you.

   You can waste a lot of time by trying to use a file which is not available on the account which you are logged on to.

   Use the:

   ```
   LISTFILES (P
   ```

   command to produce a printed list of all the files which are available. Mark the list clearly with the name of the account on which the files are available.

   Put this list in a loose-leaf binder where you can find it when it's needed.

2) Be sure that you know which datanames are available to you for each file.

   You can waste a lot of time by trying to guess what data names are available on the file which you are using.

   Use the:

   ```
   LISTDICT filename (P
   ```

   command to produce a printed list of all the datanames which are available for each of the files which you will be using.

   Put these lists in a loose-leaf binder where you can find them when they are needed.

3) Be sure that you know the format of the data which is output by each of the datanames which is available to you.

   You can waste a lot of time when you are attempting to use the WITH clause if you don't know the precise format of the data which it output by the datanames on the file which you are using. For example, does the NAME field on the STAFF file display the data as:

```
SIMPSON
Simpson
DAVID SIMPSON
SIMPSON, D
D. SIMPSON
D.SIMPSON
MR D SIMPSON
```

or what?

It is a good plan to print a LISTDICT report and - down the right-hand side of each page - alongside each dataname write an example of the format of the data produced by that dataname.

Put these lists in a loose-leaf binder where you can find them when they are needed.

4) Write down - and check - your Access sentences before you start to type them in.

   If you are not yet confident about using the computer, you might like to ask a colleague to check that you have got the grammar of the sentence correct.

5) Build up your Access sentences bit by bit. It can be disheartening to type in a long Access sentence only to have it rejected out of hand by the Access processor. It is much better to type in a simple sentence:

```
LIST STOCK
```

   and make sure that this works; then add the selection criteria which you want:

```
LIST STOCK WITH COLOUR "RED"
```

   and then add the sort specifications:

```
LIST STOCK WITH COLOUR "RED" BY COLOUR
```

   even though the sorting won't be obvious until you use the SORT verb, and gradually develop the sentence in this way. Then, if the sentence suddenly misbehaves, you will know that your last addition was wrong.

   Finally, change the LIST to SORT. Leaving the SORT until last saves time when you are using a very large file.

6) Remember that the <Ctrl>X key can be used to abandon any Access report when it is waiting at the foot of the screen for you to hit <Return> for the next page.

7) Use the TCL stacker or a macro to build your Access sentences.

18 What next?

There is still much to be done before you become an expert in the use of Access. At this stage, the most important thing is to *practise.* Use Access to produce real reports that are actually going to be useful to you or to your colleagues.

Most of the examples and illustrations used in this *MB-Guide* used a simple STOCK file, holding details of the description, quantity in stock, warehouse location, minimum stock-holding, unit price, supplier reference code, and the date when the stock-holding was last used. Nevertheless, it is important that you remember that everything we have done with the STOCK file applies equally well to any other Pick file, be it SALES-FUGIRES, STAFF, INVOICES, ORDERS, or whatever you are recording on your Pick system.

As you progress, you will want to look at other areas, such as:

- The other Access verbs, such as COUNT, SUM, ISTAT and any other special facilities which are offered on your system.
- The way in which the USING clause allows you to use the dictionary of one file to interpret the data of another.
- The way in which the TAPE modifier allows you to produce reports from data which has been transferred to a backing storage file - that is, the magnetic tape or diskette - by means of the T-DUMP verb.
- The facilities which are available for using text and special options with the HEADING, FOOTING, BREAK-ON and the GRAND-TOTAL modifiers.
- You will also want to look at the construction and use of attribute definitions for new datanames which can be used with your files. By doing this, you will be able to extend and augment the ways in which the data can be drawn from your files.
- You might alse look at what *Accu/Plot* can offer you. *Accu/Plot* is a business graphics system and allows you to use sentences exactly like those which we have used with Access to draw graphs, bar charts and pie-charts based upon the data on your files.

Information on these and other aspects of Access can be found in the publications and courses mentioned in the Preface.

19 Reference information

This section presents lists of the verbs and options which are used in Access sentences.

Remember that - if it's available on your system - you can use the EPick encyclopedia Pick to find information on any Advanced Pick topic:

19.1 EPick

EPick, *the Encyclopedia Pick*, is the Advanced Pick help facility and provides information on every TCL command, Basic statement, Access component and a wide range of general Pick topics. EPick is supplied in hard-copy form, and is is also available as an on-line reference source called up by the TCL HELP facility using commands of the form:

```
     HELP topic
or:
     ?  topic
```

where *topic* is the subject on which help is required. For example, the TCL command:

```
     HELP GFE
or:
     ? GFE
```

would invoke EPick and display the EPick entry on GFEs. If you ask for help on a topic which is not included in EPick, then the next available alphabetic entry will be offered. Thus, if you asked for:

```
     HELP GRO
```

the EPick entry for GROUP would be displayed.

Having found a particular entry in EPick, you may use the following keys and sequences to browse through the information which is displayed.

```
^F       to cruise forwards through EPick
^D       to cruise backwards through EPick
^G^G     to zoom to a related subject
^F       to cruise forward on a topic
^D       to cruise backward on a topic
^XE      to abandon the item
^XK      to abandon the HELP activity
^XX      to abandon the HELP activity

and the special hotkeys:

^X4      to display the general HELP details
^X5      to print a copy of the displayed information
```

To switch from one EPick entry to another, you should:

- Move the cursor to the *TOKEN* field.
- Type in the topic (or just the first few letters of the topic) on which help is required.
- Press ^E to clear the rest of the current token.
- Press ^F to jump to that topic.

You can terminate the current HELP session at any time by typing any of the ^X sequences shown.

19.2 Summary of sentence format

An Access sentence is made up of a number of *building blocks*. This shows the general form of the building blocks and their contents. Those blocks with a double vertical bar at the left may be repeated several times. The remaining boxes may be included only once. The contents of any block should not be split up by any other information. You may use any operator where I have used the = sign.

```
Verb
LIST
SORT
COUNT
HASH-TEST
ISTAT / GROUP.STAT
LIST-ITEM / SORT-ITEM
LIST-LABEL / SORT-LABEL
REFORMAT / SREFORMAT
SELECT / SSELECT
STAT
SUM
T-DUMP / S-DUMP
T-LOAD
```

```
File specification
  ONLY

  filename
  filename, data.file.name
  DICT filename
  pathname
  DICT pathname
```

```
Alternative dictionary
USING filename
USING filename, data.file.name
USING DICT filename
USING pathname
USING DICT pathname
```

```
Item-list
```

```
= 'item-id'
    OR 'item-id'

= "value"
    AND "value"
    OR "value"
```

```
Selection criteria
WITH dataname
WITHOUT dataname
WITH NO dataname

  WITH dataname = "value"
  WITH EVERY dataname = "value"
  WITH EACH dataname = "value"
  WITHOUT EVERY dataname = "value"
  WITHOUT EACH dataname = "value"
    AND "value"
    OR "value"

  AND WITH dataname = "value"
  OR WITH dataname = "value"
```

```
Sort specifications
BY dataname
BY-DSND dataname
BY-EXP dataname
BY-EXP-DSND dataname
BY-EXP dataname = "value"
BY-EXP-DSND dataname = "value"
```

```
Output list
dataname
dataname = "value"
```

```
Modifiers
GRAND-TOTAL "message"
COL-HDR-SUPP
DET-SUPP
ID-SUPP
HDR-SUPP
NI-SUPP
SUPP
DBL-SPC

LPTR
NOPAGE
```

```
TOTAL dataname
TOTAL dataname = "value"
```

```
BREAK-ON dataname
BREAK-ON dataname "message"
```

```
HEADING / FOOTING modifiers
    HEADING "message"
    FOOTING "message"
```

```
Options (at the end of the sentence)
(options
```

19.3 Access verbs

This list shows the Access verbs which are offered on the Advanced Pick system. Some verbs have a non-sorting and a sorting form (such as REFORMAT and SREFORMAT, T-DUMP and S-DUMP). The use of a BY modifier will automatically invoke the sorting form.

CHECK-SUM calculate the check-sum for items on a file.

COUNT count the number of items which satisfy any specified selection criteria.

HASH-TEST produce item statistics for a file based upon any hypothetical test modulo.

ISTAT produce item statistics for a file.

LIST / SORT produce a report based upon specified selection criteria and containing specified datanames.

LIST-ITEM / SORT-ITEM list the items in a file according to any specified selection criteria.

LIST-LABEL / SORT-LABEL produce a report in label format.

REFORMAT / SREFORMAT create new items from those on a file.

S-DUMP / T-DUMP output the contents of selected items on a file sorted according to any specified criteria.

SELECT / SSELECT create a select list of the item-ids on a file according to any selection specification.

SORT produce a report sorted in a specified sequence based upon

specified selection criteria and containing specified datanames.

STAT output the total and average size of items on a file selected according to any specified criteria.

SUM output the total of a data name of the items on a file selected according to any specified criteria.

T-LOAD restore one, several or all the items on a file from a magnetic tape produced by the T-DUMP / S-DUMP command.

UPDATE invoke the Update Processor, allowing you to change items on the file.

19.4 Access options

This list shows the options which are available for use with Access sentences. Where there is an equivalent keyword, this is shown in brackets.

A skip the tape label handling and assume that media is positioned at the beginning of the desired data area. Used with T-LOAD.

B suppress the number of items listed at the foot of the report. [NI-SUPP]

C suppress the default page headings and column headings. [COL-HDR-SUPP]

D suppress the detail lines between control breaks. [DET-SUPP]

F skip to a new page for each item. Used with LIST-ITEM and SORT-ITEM.

G suppress the row and column totals when using the SS connective.

H suppress the default page heading, and will also suppress the creation of labels on the diskette/tape when used with T-DUMP. [SUPP / HDR-SUPP]

I suppress the output of the item-id. [ID-SUPP]

J suppress the number of items listed at the foot of the report. [NI-SUPP]

K suppress the output of the legend message for list and sort operations. [LEG-SUPP / LEGEND-SUPP]

N suppress the end-of-page pause when displaying on the

screen. [NOPAGE]

O overwrite existing items when used with the T-LOAD verb.

P send the output to the printer. [LPTR]

R output the data in filled text manner across the page. [FILL]

Rn load binary items from a T-DUMP produced on a system which used *n* bytes per frame. Used with T-LOAD.

S suppress the effect of any embedded Output Processor commands invoked by W-type attribute definitions. Used with LIST/SORT.

S activate a secondary select-list. Used with SELECT/SSELECT.

V verify the tape file but do not write the data to the disk file. If the O option is used, the V option is ignored. Used with T-LOAD.

Y display the **parsed** version of all references and attribute names used in the sentence. This includes the F processing code equivalent of any complex conversion or correlative codes.

Z print the **parsed** version of all references and attribute names used in the sentence. This includes the F processing code equivalent of any complex conversion or correlative codes. See also the Y option.

Z to bypass the Access preprocessor and pass the sentence directly to the Access processor.

19.5 HEADING/FOOTING options

This list shows the options which are available for use with HEADINGs and FOOTINGs in Access sentences.

Axx output the value of dataname xx.

'' output a single apostrophe.

B{n} output the BREAK-ON or ROLL-ON break-value left-justified in a field *n* characters wide.

C{n} centre the following text within a field of *n* characters.

D output the current date in the format: 01 JAN 1999

DU output the date, as for D, except that the date of a long

report will be changed if it runs over midnight.

F{n} output the file-name left-justified in a field of *n* spaces.

I{n} output the current item-id left-justified in a field of *n* spaces.

J right-justify the following text.

L start a new line.

N suppress the pause at the end-of-page when displaying on the screen.

P output the page-number right-justified in a field of four spaces.

PN output the page-number with no surrounding spaces.

R output the current page-number in Roman numerals.

S output subsequent text in italics.

T output the current date and time in the format: 09:30:59 01 JAN 1999

TU output the current date, as for T, except that the date of a long report will be changed if it runs over midnight.

U underline subsequent text.

V output subsequent text in boldface.

X combine the effects of XS, XU and XV in disabling all special output effects.

XC cancel centering.

XS disable italic output which was invoked by the S options.

XU disable underlined output which was invoked by the U options.

XV disable boldface output which was invoked by the V options.

20 Glossary

This section presents some of the terms which are used in this *MB-Guide*, and is based upon material to be found in the *MB-Guide to Pick terminology*.

Attribute: The Pick terminology for a field or a data field of a record. Each field of a Pick item is known as an attribute. An item may have zero or more attributes. Each attribute may have zero or more values. Each value may have zero or more subvalues.

Control-break: A visual signal - such as skipping a few blank lines - which is produced by the BREAK-ON modifier to indicate that a control field has changed in value when a report is sorted according to the control field.

Dataname: The name by which each data field is identified in an Access sentence. There must be an attribute definition for each such dataname.

Dictionary: That part of a file which holds a description of the **datanames** which can be used with that file. The dictionary also holds the **data-level identifier** by which the Pick system locates the data section for the file.

Item: The Pick term for a data record. Each file consists of any number of items. Each item consists of the **item-id** which identifies the item, followed by none, one or more **attributes** separated by attribute-marks.

Item-id: The Pick term for a record **key**. The item-id uniquely identifies each item on a Pick file.

Modifier: Any of a number of keywords which can be used to change the normal effect of an Access sentence. Examples are HDR-SUPP, COL-HDR-SUPP, SUPP, NOPAGE, ID-SUPP, LPTR.

Multivalue: Each **attribute** in a Pick **item** may consist of a none, one or more multivalues.

Phrase: A means of saving part of a sentence.

Proc: A sequence of instructions (such as an Access sentences or TCL commands and possibly with other statements in the Proc language), which has been saved under a single name so that it can easily be recalled in future. The name is also used to refer to the language which is used to write such processing routines. The language is peculiar to the Pick system.

Subvalue: Each field of a Pick item is known as an attribute.

An item may have zero or more attributes. Each attribute may have zero or more values. Each value may have zero or more subvalues.

Verb: The first word of an Access sentence or a TCL command.

Index

MB-Guides

The booklets in the ***MB-Guide*** series cover a range of fundamental topics of interest to users and those responsible for developing, implementing and running Pick and associated systems.

Each ***MB-Guide*** deals with a specific aspect of the Pick system and the booklets represent an economical introduction to the various topics and the whole series forms an integrated presentation of the subject matter.

The booklets are intended to be a working document and, for this reason, space is provided for the user's own notes, and the reader is encouraged to amend the booklet so that it applies to his/her own system.

The series of ***MB-Guides*** is of special interest to new users and those migrating from one version of the system to another, and is a convenient source of information for training organisations, software houses and others who are responsible for the instruction and support of their clients and staff in the fundamental aspects of the Pick system.

Malcolm Bull

Training and Consultancy Publications