# 10 Macros

One of the main reasons for using word processors is that you can make the computer do all the really boring jobs.

Suppose you have to type the same letter or invitation 30 times over. The chances are that you will make half a dozen mistakes in your letters which you will only discover when it is too late.

VIEW offers a facility which avoids all such problems. It allows you to make up any collection of text and commands into a package. You give the package a name, and you can then cause the whole package to be printed as often as you like, merely by entering that name in the margin like a stored command.

Such packages are known as 'macros'. To show how they work, suppose you have to send out innumerable copies of the following invitation.

```
          ABC Computers Ltd
request the pleasure of your company
      at the launching of the

              LOGOMANIAC

     Word Processing Package
       at the Queen's Hotel
 on Friday 13 February at 12.00pm
```

Obviously you would use the CE command to centre all the lines, like this:

```
CE ABC Computers Ltd
CE request the pleasure of your company
CE at the launching of the

CE LOGOMANIAC

CE Word Processing Package
CE at the Queen's Hotel
CE on Friday 13 February at 12.00pm
```

To make this into a macro so that you could print out many copies with the least possible trouble you need to:

- Give it a name.
- Signal to VIEW that it is a macro.
- ' Call' it, ie to tell VIEW to use it.

Names of macros can consist of any combination of two letters that is not already used as a stored command. The stored command to define (give a name to) a macro is DM, so to name the macro AZ we simply place before it the stored command: `DM AZ` and after it the stored command `EM` (which of course means ' end macro' ).

The finished macro therefore looks like this:

```
DM AZ
CE ABC Computers Ltd
CE request the pleasure of your company
CE at the launching of the

CE LOGOMANIAC

CE Word Processing Package
CE at the Queen's Hotel
CE on Friday 13 February at 12.00pm
EM
```

When you have written, defined, named and ended your macro you have only to use it. As it stands it will not print out at all. To make it print you have to call it, ie to enter its name in the margin as if it were a stored command. Press EDIT COMMAND (**SHIFT** **f6**), type `AZ` and press **RETURN**.

Try entering it several times as a test. When you set up the printer and type `PRINT`, the text will be printed out as many times as you have entered the macro' s name in the margin.

## 10.1 Modified macros

When composing standard letters, invitations and other documents, we frequently need a piece of text which is essentially the same in each example, but has to be modified so as to ' personalise' it. Normal copying techniques fail here, but VIEW has a special facility which makes its macros even more effective.

Suppose we want to make up a standard, all-purpose invitation, which can be used to invite anybody to anything.

To start with we shall try a macro with just one modified item. Suppose we want to design an invitation into which any name can be inserted, an invitation that begins like this:

```
          ABC Computers Ltd
request the pleasure of the company of
            Mr Bill Brewer
```

In cases like this, VIEW allows you in effect to leave blanks in the macro into which you can fit any name you like at the printing stage. The method is to replace the words which are to be changed with the symbols `@0`, `@1`, `@2` . . . `@9`. Each time the macro is called for printing, these parameters are filled in by typing the words on the same line as its name.

In this case we would simply place `@0` in place of the name:

```
DM BY
CE ABC Computers Ltd
CE request the pleasure and company of
CE @0
CE at the launching of the

CE LOGOMANIAC

CE Word Processing Package
CE at the Queen's Hotel
CE on Friday 13 February at 12.00pm
EM
```

Later in the text, you enter the macro as if it were a stored command, with the appropriate name beside it.

```
BY Mr Bill Brewer
BY Mr Jan Stewer
BY Mr Peter Gurney
```

After that, all you have to do is to issue a print command and the macro will be printed, its blanks filled in, as many times as you have entered its name in the margin.

Developing this into a genuinely all-purpose invitation is merely a matter of extending the number of parameters, so as to leave blanks for almost everything.

```
          .....................
requests the pleasure of the company of

          .....................

           on the occasion of

          .....................
          .....................
     at ..........................
  on ................at.............
```

Try it yourself, before looking at the finished macro below.

```
DM CX
CE @0
CE requests the pleasure of the company of

CE @1

CE on the occasion of
CE @2
CE @3
CE at @4
CE on @5 at @6
EM
```

When you come to use it the method is much the same as before:

```
CX  XYZ Computers Ltd,Jim Smith,the lanuching, of ASTROCALC,
```

. . . and so on.

One variation is worth noting. Suppose we want to invite Tom, Dick and Harry, who naturally go everywhere together. What about the comma after ' Tom' ? Commas are used as spacers between parameters, so if we put it in it would have the wrong effect. To avoid this problem VIEW has the rule that if you want to include commas in a parameter you place it in angle brackets, like this:

```
CX  XYZ Computers Ltd,<Tom, Dick and Harry>,the launching ...
```

Apart from this there is virtually no restriction in the use of macros. The macro below, for example, consists almost entirely of commands. Its purpose is to provide an automatic layout for the top of a letter.

```
DM LH
RJ @0
   @1
   @2
   @3
   @4
RJ @5

   Dear @6

CE @7

EM
```

Try for yourself setting out the parameter line to make the beginning of the letter read:

```
                                              25 June 1984

John Smith Esq
25 Alpha Road
Middletown
Loamshire

                                                 Ref. 25/3

Dear Mr Smith

              Hire Purchase Agreement
```

## 10.2 Macros for mail shots

Using macros in this way, you can develop quite a sophisticated system for standard letters, all of which can be individually addressed and have a number of different key words and numbers in their contents.

Suppose we wish to send the following letter to applicants for a course, making changes of date, student, course, fee, etc.

```
Mr A B Carter          15 Sept 1984
10 Old Street
Newtown CX3 9JJ

Dear Mr Carter

Thank you for your application. We
confirm your enrolment for the
Intermediate Course C. Your student
number is 552.

You will receive further details in
a few days. Meanwhile will you please
send the acceptance payment as shown
in our brochure of £22.50.

If you have any queries please
address them to Dept 4B.

Yours sincerely
```

The equivalent macro would be:

```
DM LL
@0                    @1 Sept 1984
@2
@3

   Dear @4

   Thank you for your application. We
   confirm your enrolment for the
   Intermediate Course C. Your student
   number is 552.

   You will receive further details in
   a few days. Meanwhile will you please
   send the acceptance payment as shown
   in our brochure of £22.50.

   If you have any queries please
   address them to Dept 4B.

   Yours sincerely

EM
```

and appropriate parameter lines would be as follows:

```
LL Mr A B Carter,15,10 Old Street,Newtown CX3 9JJ,Mr Carter,C,552,£22.50,4B

LL Ms Jane Brown,15,The Cottage,<Stenby, Powys>,Ms Brown,433,£19.25,9A
```

## 10.3 Automatic Layout

If you need to produce reports or books to a standard format, it is possible to use macros in conjunction with number registers to automate the layout in quite a sophisticated way.

Suppose you produce a series of reports, each with many chapters with headings and sub-headings, all numbered as follows:

```
CHAPTER 1

                      Chapter title

1.1 Section heading

1.1.1 Sub-section heading

     Text text text text text text text text text text text text
     Text text text text text text text text text text text text
     Text text text text text text text text text text text text

1.2 Section heading

     Text text text text text text text text text text text text
     Text text text text text text text text text text text text
     Text text text text text text text text text text text text

                                                          Page 1
```

The most efficient way to manage such a series would be to set up a file containing macros to control all the standard features: spacing, bold type, and in particular numbers. For example, if we set register C to 1, and then cause a macro to increase register C by 1 each time a chapter heading is printed, we can print chapter headings completely automatically. Numbers for section and sub-section headings, and page numbers could be dealt with in the same way.

In the case of chapter headings the settings of the register would take the form:

```
SR  C 1
```

and each time a chapter heading was printed a macro would be called containing the line:

```
SR  C |C+1
```

which means that the new value of C is set to 1 greater than the old value of C. Later in the macro the value of C would be printed out by

```
LJ  *CHAPTER |C*
```

which uses the `LJ` (Left justify) command to print out the chapter heading, the asterisks to print it in bold, and the vertical bar to indicate that it is register C that it is being printed, not the letter C.

Using these principles we can construct a set of macros to manage a whole series of reports. The macros would be saved on disc, and used as part of the print command whenever a report was printed. For example if the set of macros is given the name ' BOOK'and the report files are ' A1'to ' A5'the print command would be:

```
PRINT BOOK A1 A2 A3 A4 A5
```

The macros below are an example of how this can be done. The way they do this may not be immediately apparent, but they will repay study.

```
DF  ///Page|P/          (Define footers to display page numbers as
SR  P 0                 register P which is automatically increased
SR  C 0                 by 1 as each page is printed; also set
SR  S 0                 chapter, section and sub-section heading
SR  T 0                 numbers to zero.)
```

Chapter heading macro:

```
DM  CH                  (Define chapter heading macro.)
SR  C |C+1              (Increase chapter number by 1.)
SR  S 0                 (Set section number to 0.)
SR  T 0                 (Set sub-section number to 0.)
PE                      (Eject page - so chapter begins on new page.)
LJ  *CHAPTER |C*        (Print chapter heading.)
CE  *@0*                (Print chapter title, bold, centred.)
EM
```

Section heading macro:

```
DM  SE                  (Define section heading macro.)
SR  S |S+1              (Increase section number by 1.)
SR  T 0                 (Set sub-section number to 0.)
PE  5                   (Eject page if within five lines of bottom.)
LJ  *|C.|S @0*          (Print chapter and section numbers, and
EM                      section heading in bold.)
```

Sub-section heading macro:

```
DM  SS                  (Define sub-section heading macro.)
SR  T |T+1              (Increase sub-section number by 1.)
PE  3                   (Eject page if within three lines of bottom.)
LJ  *|C.|S.|T @0*       (Print chapter, section, and sub-section
EM                      numbers, and sub-heading in bold.)
```

When the text is written in VIEW, the footers appear automatically, showing the page numbers, and the numbers in the chapter, section and sub-section headings are provided by the macros, the words of these headings being parameters to the macros, as follows.

```
CH  Chapter
SE  Section heading
SS  Sub-section heading
    Text text text text text text text text text text text text
    Text text text text text text text text text text text text
    Text text text text text text text text text text text text

SE  Section heading
    Text text text text text text text text text text text text
    Text text text text text text text text text text text text
    Text text text text text text text text text text text text
```

If the macros above are made into a file called ' MACRO'and the text into a file called ' TEXT" then the command

```
PRINT MACRO TEXT RETURN
```

will reproduce the text shown earlier in this chapter. You will notice, however, that it first prints out a page which is blank apart from the footer: `Page 0`. If you can work out why it does this you will have no difficulty in understanding macros.

The reason is that the chapter heading macro causes each chapter to be printed on a new page by doing a page eject. The setting-up sequence sets the page number register P to 0, and the chapter heading macro then ejects page 0, and P is incremented to 1. This gives the correct starting page and chapter (chapter 1 page 1), but also gives a blank 0.