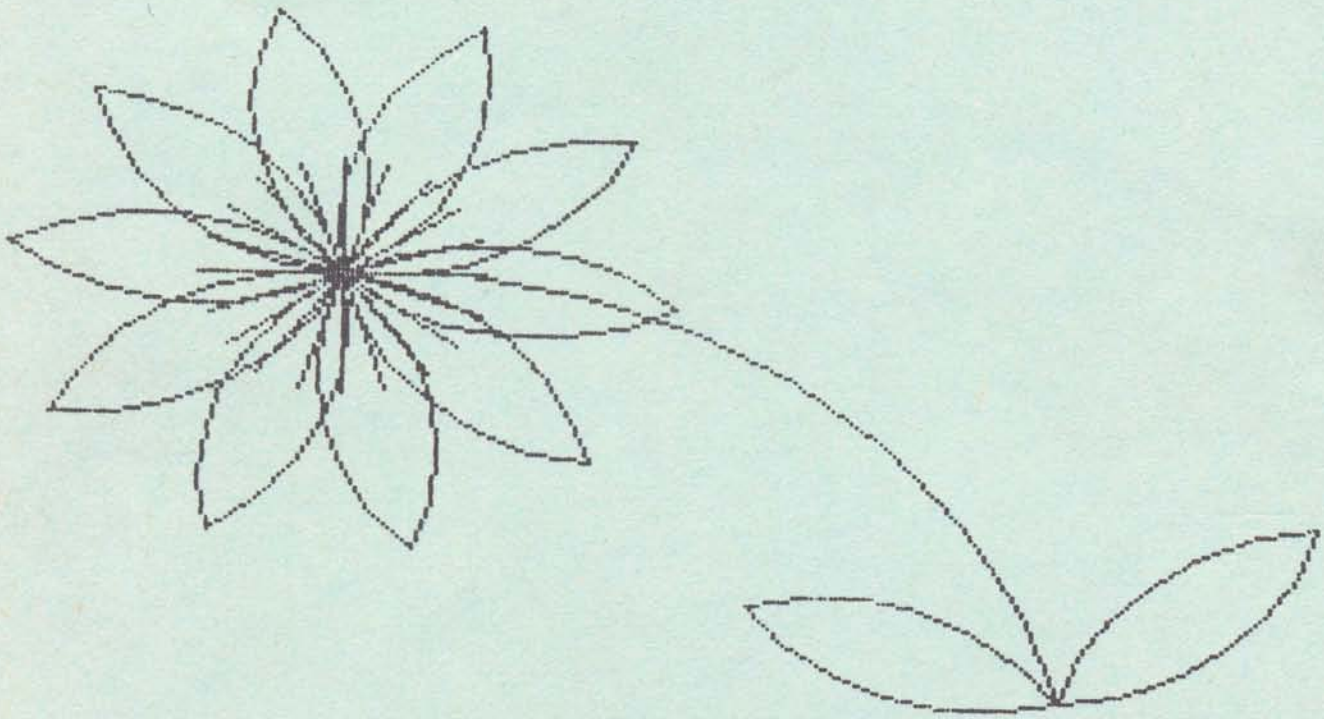
DRAW



CONTENTS

The author and Micro Power Ltd. would like to thank Leeds Education Authority for their co-operation during the development and testing of the program.

Introduction

DRAW has been written to provide a simple, but complete, system for the introduction of certain programming skills. It is based on a subset of the LOGO language developed by Papert, and others, in the USA. The system allows the following programming concepts to be experimented with:

A program as a linear sequence of instructions.
Incorrect programs give incorrect results.
Repetition of a program segment.
Nesting of loops.
Calling of subprograms, or subroutines.
Passing a parameter to a subroutine.

This system only allows certain of the 'Turtle Graphics' processes to be used but emphasis has been placed on producing an easy to use, secure system for the 'uninitiated' to use. To this end the system depends on 'menus' (lists of options) and single key entry for most of the frequently used commands and the system provides full detection of errors and on-screen prompting.

Machine Requirements

The program requires a Model B BBC microcomputer, or a Model A to which additional memory has been added, or an Acorn Electron computer. A cassette backing store has been assumed and, because of the additional memory requirements of the disc operating system, this program will NOT operate on a disc-based system. The program supports program listing, and a print out of the final screen, to an Epson MX80 high resolution printer.

Getting Started

New users are advised to load the system (Page 3) and then to follow through the Tutorial section of this Guide. (Page 14)

WARNING

The program checks to see if any edits have been carried out to the BASIC listing. Attempts to run an edited program will almost certainly fail. Backup copies can be made by LOADing and SAVEing in the usual way but see details on Page 3

N.B.

Due to the incompatibility between Basic 1 and Basic 2 when using the OPENIN statement, please check your copy of DRAW1 (the second program on your cassette) and ensure that line 360 reads
360channel%=OPENIN(file$)
There must be no spaces in the line and you should then resave DRAW1 along with DRAW onto another cassette and then use this as your master.

Loading the System

The cassette supplied carries three separate files:

DRAW	:	a short 'pre-loader'
DRAW1	:	the main system program
DEMO	:	some demonstration DRAW programs

To load the system simply type

```
>CHAIN "DRAW" <return>
```

The preloader will be loaded and run. It produces a message on the screen:

```
******************
* L O A D I N G *
*    D R A W     *
******************
        Searching
```

This program then automatically loads and runs the main system program - DRAW1. This second program produces a Title Page and then asks whether or not it is intended to use a printer. This question simply requires a YES or NO answer followed by pressing the <return> key. The screen will then show the 'Main Menu' see Page 4.

Making Back-up Copies

Making back-up copies is relatively simple. First LOAD (not CHAIN) the DRAW preloader and copy it to the back-up tape:

```
>SAVE "DRAW" <return>
```

Now run the preloader to load the main system but do not rewind the back-up tape. Get through to the main menu and exit to BASIC via the 'F' command PAGE and then SAVE the system program:

```
>SAVE "DRAW1" <return>
```

N.B. Do not press the BREAK key in an attempt to exit from the system as this will corrupt the main program making the back-up copy useless.

Printer Support

The basic program supplied contains support routines for an Epson MX80 Type 3 printer. It expects the printer to be connected via the parallel interface (not the RS 432 interface) and the computer sends linefeed codes after carriage return codes. If your printer supplies its own linefeeds, then you will either have to alter the printer (see the Epson Manual) or change the line in the Preloader program that changes the 'printer ignore' character.

```
i.e.  change   *FX 6,12
        to     *FX 6,10
```

When the program first executes you will be asked if you have a printer connected and you must reply "YES", any other answer is assumed to be "NO".

The 'P' option from the main menu (Page 6) will allow you to list any program to the printer, a "No printer" message will be printed if you have not selected a printer. This feature should work with any 'Centronics' type printer.

When the program has run and produced a pattern on the screen a message:

Press SPACE or P

will appear at the top right of the screen. Pressing the SPACE bar will return you to the Main Menu; pressing P will 'dump' the whole of the screen to the printer before returning you to the Main Menu. Note: this feature will only be supported on a high resolution Epson Printer.

Running the System

This program is based on 3 menus - lists of options:

```
Main menu    : selecting mode of operation
Command menu : for entering program steps
Edit menu    : used when editing programs
```

The computer uses a flashing 'underline' character to indicate where the next character will appear on the screen - this is called the CURSOR.

WHENEVER THE CURSOR CHARACTER IS PRESENT ON THE SCREEN, THE COMPUTER IS EXPECTING ONE, OR MORE, CHARACTERS AND THE 'RETURN' KEY MUST BE PRESSED TO TERMINATE THE INPUT. THE 'DELETE' KEY MAY BE USED TO RECOVER FROM ERRORS ON THIS LINE. IF THERE IS NO CURSOR ON THE SCREEN THEN THE PROGRAM ONLY REQUIRES THE SELECTION KEY TO BE PRESSED. DO NOT PRESS RETURN.

The response to the menu request:

Select from:

is to simply press the letter given at the beginning of the line. In general, further information will be required in order to carry out the function selected and a message requesting the information will appear at the bottom of the left hand side of the screen.

The Special Keys : ESCAPE and BREAK

The functions of these two keys are changed when the system program is running. They will NOT provide a return to BASIC.

The ESCAPE key will provide a means of returning to the main menu at any time the system is in use. It can be considered to be the 'panic' button but there can be complications if it is used when entering or editing programs as these may be left incomplete. Pressing the ESCAPE key during program entry when using an ECONET system can cause 'garbage' to be entered into the program. If this happens, delete the program in question and re-enter it.

The Break key has been programmed so that if it is only pressed once, the program will recover itself and restart. You will be returned to the stage that asks if a printer is connected but programs held in the memory should not be lost.

The Main Menu

On the next 2 pages there follows a description of each of the 9 options which are available from the main menu. In several of the commands the user will be asked to either choose a program or to supply a program tape file name:

Programs are selected by the NUMBER which is assigned to that particular program by the system. In each case the programs available, together with their associated numbers will be displayed on the right hand side of the screen.
Names, either for programs or tape files, must start with a letter, and must not be longer than 10 characters. The system will convert any lower case entries to upper case.
Valid names would be:
SQUARE DEMO TESTPROG1 TESTPROG2 MYPROG
The following would be invalid:
THISISTOOLONG 6NUMBER

If the user types in wrong numbers or names the system will usually point out the error allowing another try.

Main Menu Options

N - new program

This option must be used before any new DRAW program is written. The system will ask for a name for the program and it will assign a code number to it, the next available number. The system allows 20 programs, each with 30 lines.
The system will display, on the right hand side of the screen, the program name and a set of line numbers ready for the program lines. The left hand side of the screen will be replaced with the Command menu (see Page 9)

R - run program

Use this option to tell the computer to follow the instructions in a particular program. You will be asked which program you wish to run and you choose the program by number. When the program has finished the pattern will remain on the screen and a message:

Press SPACE bar

will appear at the top right of the screen. Pressing the space bar will return you to the Main Menu.

E - edit program

It very often happens that your programs do not produce the correct results at the first attempt and you will therefore need to alter them. From the main menu it is possible to request to edit them and this command gives access to the EDIT menu (see Page 11). Enter the number of the program you wish to edit.

D - delete program

This option allows you to remove any program from the computer's memory. Choose the program by its number.

L - list program

This option allows a particular program to be listed to the right hand side of the screen. This is not the same as preparing to edit and in order to change the program you will need to type 'E'. Again, choose the program by its number.

P - print program

This option allows the listing of a particular program on an attached printer. If a printer was not selected (by responding 'YES' to the opening question) then the system simply replies with 'No printer'.

S - save programs

It is possible to save ALL the programs that are held in the computer's memory by using this option. It is not possible to save individual programs, the computer will only save all of the programs currently in the memory. You will be asked for a filename for the group of programs (for details, see above) and then prompted to switch on the tape recorder. DON'T record the programs over your DRAW master cassette!

I - input programs

This option is the logical counterpart to the last one. It allows programs previously stored on tape to be reloaded into the computer's memory. N.B. As you can only 'save' and 'load' ALL the programs at once, using the 'I' option will effectively delete all the programs in the memory. The system will ask for the filename which was used when the programs were saved.

F - finish

This is the command used to finish using the DRAW system and it is the only way to exit from the system into BASIC. Because exiting to BASIC will necessitate a 'rerun' of the system program it effectively deletes all the programs in the memory. You will be asked if you are sure you want to exit from the system and whether or not you have saved your programs to tape. A reply of 'YES' will exit and any other answer will return to the main menu.

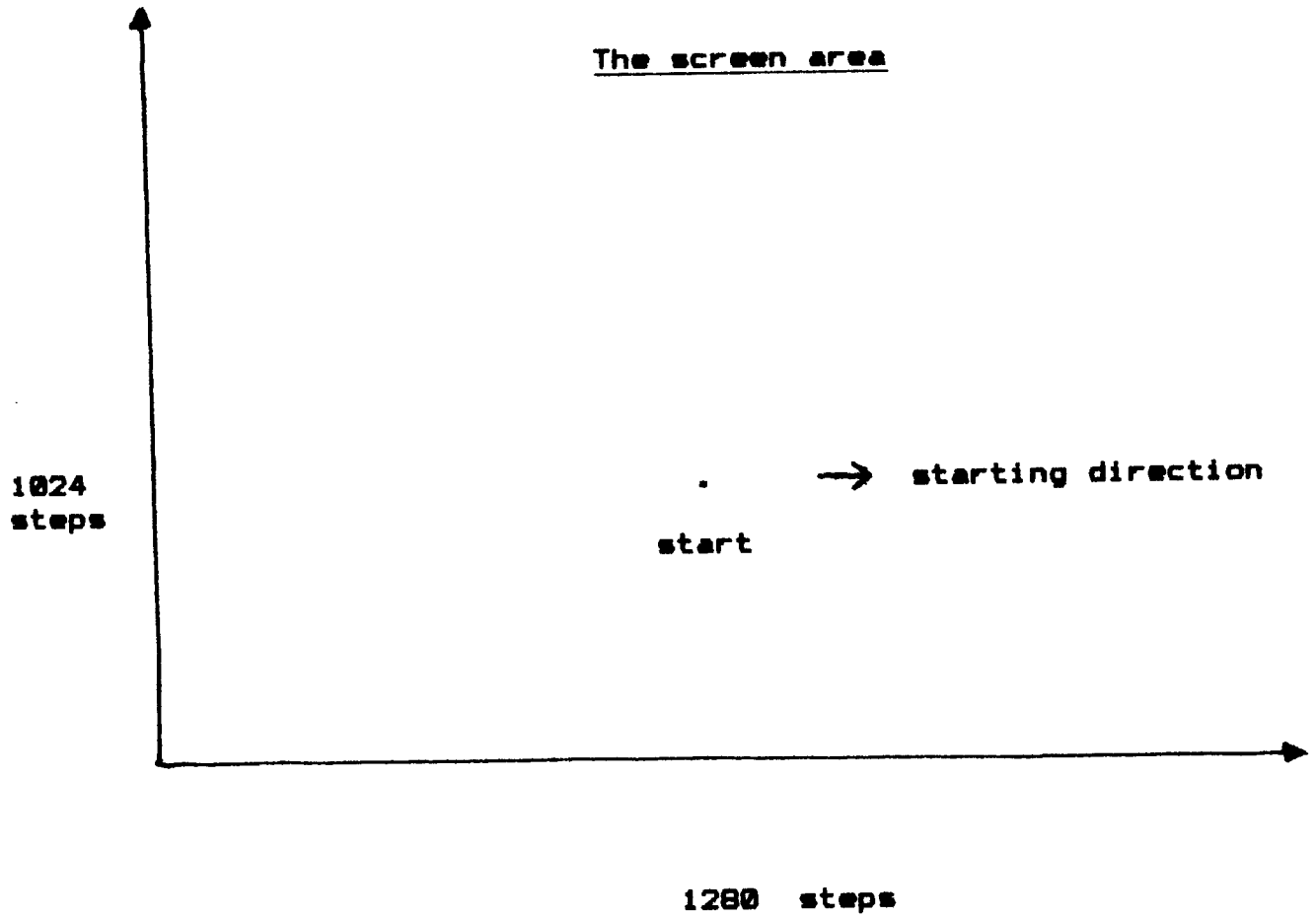
Recovering from errors

If, at any time, you have made an incorrect selection then the ESCAPE key will allow you to return to the main menu.

Program statements

This section gives details of the program statements available in the DRAW language together with any limitations that may apply to them. The commands control the movement of a 'dot' (equivalent to the Turtle in LOGO) on the graphics screen. You control the amount by which the 'dot' moves, and by how much it turns between forward movements. It moves on the screen which is 1280 steps wide by 1024 steps high. It starts in the middle of the screen 'facing' towards the right. Because of the way in which the computer handles its graphics, it takes 4 steps for the appearance of ONE dot on the screen. In other words, the actual visual screen is 320 x 256 dots.

The screen area



If you move the 'dot' off the screen it simply disappears and it will reappear when you give the correct commands to bring it back to the starting area.

When you are entering program statements the left hand side of the screen shows the possibilities available in the form of the Command Menu. The system will prompt for any further information needed and then it will list the program step by step on the right hand side of the screen.

The Command Menu

M - move

This is the basic command used to make the 'dot' move on the screen. Depending on whether the 'pen' is up or down (see below) a line will be drawn on the screen in the direction that the 'dot' moves. The system will prompt for the number of steps that you wish the 'dot' to move. Positive numbers will cause the 'dot' to move forwards in the direction it is currently facing and negative numbers (you will need to type in the - sign, found next to the 0 key on the top row of the keyboard) will cause it to move backwards without changing the direction it is facing.

An entry of zero, or any characters other than a number, will produce the response on the screen of:

move SIZE

which simply means that the computer does not know what value to use and it will assume a value of zero. Early on this response will often be produced by typing O (capital 'o') instead of 0 (zero). The SIZE parameter is used in subprograms for a special purpose (see below) but the line can be conveniently be removed, if it was not intended, by using the 'K' option.

T - turn

This is the command which controls the direction in which the 'dot' will move next. The system will ask for the number of degrees through which the 'dot' is to turn. Note: the 'dot' turns 'on the spot' rather than to a new heading so you need to keep track of the current direction at any time when planning your program.

Positive values for the angle will result in a turn in an anticlockwise direction whilest negative values turn clockwise. Zero, or non-numeric, values will result in a SIZE response (see above: under 'M')

R - repeat

This statement, used in conjunction with 'end repeat' is used to define a number of program steps which you would like the computer to obey more than once. This program 'structure' is called a 'Repeat Loop'. The system asks for the number of times that you wish the loop to be repeated. A positive number is required here and zero, or non-numeric, entries give the SIZE response.

Loops can be 'nested', i.e. loops within loops in order to produce complex patterns. There is a limit to the number of repeat loops which can be nested but these are only detected when the program is being run, not when it is being entered. That limit is 12 on Version 3.1P and the count is cumulative through Program calls (see 'P' command). This is because the system needs to 'keep track' of which line it needs to jump back to and the restriction is imposed by the available memory.

E - end repeat

Use this command at the end of the group of steps that you wish to be repeated. You can think of the 'repeat' - 'end repeat' statements as acting like brackets round the steps to be repeat. If you try to end a repeat loop which does not exist then the system will produce a suitable error message. The lines between the 'repeat' and 'end repeat' lines will be indented one space for each loop started to give an indication of the scope of each loop.

P - program

Once you have a sequence of commands that produce a shape you can use this sequence in another program by simply 'calling' the first program as a subroutine from the new one. You do this by using the 'P' command option. The system will ask you for the number of the program that you wish to execute at this point (entering 0 will give the list of the programs currently held in the memory).

The name of the chosen program will be entered into the program listing on the current line of the program. When the program is run the computer effectively suspends the execution of the current program, finds the program specified as a subroutine, executes that program, and then returns to the next line in the calling program. This is very useful as it allows you to plan a large pattern in a series of much smaller, and testable, units.

Subprograms may call other programs from within them but, for reasons similar to those outlined under 'repeat' above, there has to be a limit to the number of 'nested' calls. This limit is 7 in Version 3.1P and it is only detected when the program is running. A few moments thought would suggest that it is not possible for a program to call itself and an error message is printed if this is attempted.

It is possible for the subroutine to depend on the calling program for one of the values that it uses when it executes. This is explained more fully in the section on 'Passing a Parameter - SIZE' - Page 13

U - pen up

You can think of the 'dot' as a pen which can either be on, or off, a piece of paper. The command 'pen up' causes the pen to be raised and so subsequent movements of the 'dot' will be invisible. This is used so that parts of a pattern may be produced without lines joining them.

D - pen down

This command is the counterpart of the previous one. It causes the pen to be placed on the paper. The 'dot' always starts in the 'pen down' condition at the beginning of a new program.

F - finish

Use this command at the end of your program. It will place the word 'finish' at the end of the program and return you to the Main Menu. You will be warned if there are any 'repeat loops' that have not been correctly closed with 'end repeat' statements and these have to be closed first.

K - kill last line

This provides a simple editing function. It simply deletes the last line of the current program. You can use it several times in succession to remove more than one line.

Recovering from errors

If you make the wrong selection, e.g. 'M' instead of 'T', finish the entry, then use 'K' to delete the line.

Editing a program

Choosing the 'E' option on the main menu will ask you to select the program which you wish to edit and then it will list this program on the right hand side of the screen. The left hand side of the screen will be replaced with a new menu - the Edit Menu - which allows you to insert, delete or change lines within the program. In each case you select the line you wish to alter in response to a request from the system.

The Edit Menu

I - insert a line

Give a line number that lies between those in the program (i.e. it must be an odd number) and you will be presented with the Command Menu for you to choose the statement that you wish to add to the program. The number of the line that you are inserting will be printed on the very top line of the right hand side of the screen - along from the program name. The insert will be carried out and you will be returned to the Edit Menu.

D - delete a line

Give the number of a line within the program and that line will be deleted. The remaining lines will be 'moved up' to replace it.

C - change a line

Give the number of a line you wish to correct. The line number will be placed on the top right of the right hand side of the screen and the Command Menu will allow you to choose the new line.

F - finish editing

Returns to the Main Menu to allow you to run the edited program.

Recovering from errors

If you have made a wrong selection, use the ESCAPE key to return to the Main Menu.

Errors

Any edit which produces an unclosed repeat loop will be indicated by a suitable error message. Correct the error before running the program or an error will be produced when the program is run.

Error messages

Most of the error messages produced by the system are self explanitory but one or two may need further explanation:

Sorry can't
This message will be produced if you select a program number to list, edit, run or delete for which there is no program. Simply enter the correct number.

Error in REPEAT - END REPEATS
This message will be produced if, when editing, you have an unclosed repeat loop. You should correct the program before finishing the editing.

Recursion is NOT allowed!
Recursion is the 'jargon' term in computing for a subroutine which 'chases its own tail' and in DRAW you may not call a program from within itself

ERROR!!! in Program MYPROG
There are 3 error messages which may be printed on the graphics screen when a program is running. This one means that you have attempted to run a program that has either no 'finish' at the end, or too many 'end repeat' lines. Edit the offending program.

As above + Too many REPEAT's
More then 12 repeat loops have been started. (see Page 9 'R') Note: this error may be referring to a subprogram as the number of loops is cumulative through program calls.

As above Too many PROG calls
You have tried to 'nest' more than 7 subroutine calls.
(see Page 10 'P').

Unreported errors

Should there be an error produced by the BASIC interpreter for any reason then this will not be reported in the usual way. All that will happen is that the Main Menu will reappear - in the same way that happens when the ESCAPE key is pressed.

Passing a Parameter - SIZE

When you have become practiced in using the program calls you will probably want to be able to draw the same shape, but with different sizes, as parts of the same pattern. You do not need to write a whole series of subprograms - a general, variable one is possible.

As an example, consider drawing a square of variable size. First you must start a new program where you add the 4 characters 'SIZE' to the end of the program name. (This is very important as you cannot pass a value to the program otherwise). Now write the program in, but give a zero value to line that you want to be variable:

```
SQUARE SIZE
repeat 4
 move SIZE
 turn 90
end repeat
finish
```

This program will not produce any results as the value of SIZE is taken to be zero initially.

(Note: the SIZE characters are NOT included in the 10 characters allowed for a program name so, for example, RECTANGLESIZE would be a valid entry. You do not need to put the space into the name as the system does this for you. It is a check that the SIZE indication has been recognised.)
If you write another program and call this one as a subprogram then you will be asked to provide a value for SIZE and this will be listed alongside the program name on the right hand side of the screen. When this new program is run and the subprogram called, this value is passed on to the SIZE parameter and the first program wil give some sensible output.
Parameters may be 'passed down' through more than one level of subroutine. Try the following:

```
SQUARE SIZE
repeat 4
move SIZE
turn -90
end repeat
finish
```

```
TRIANGLE SIZE
repeat 3
 move SIZE
 turn 120
end repeat
finish
```

```
HOUSE SIZE
SQUARE SIZE
TRIANGLE SIZE
finish
```

```
HOUSES
HOUSE 100
pen up
move 150
pen down
HOUSE 200
finish
```

Run the program HOUSES and see what happens.

Tutorial

The intention of the next few pages is to lead you through the basic programming ideas that can be illustrated using DRAW. It is strongly suggested that the next section is worked through at the computer with the DRAW system loaded and running. (Refer to pages 3 and 5 on how to load the system program)

After a 'Title Screen' you will be asked if you have a printer connected. If you intend to use a printer (see page 4 for details of the printer) then you should answer with the 3 characters - YES - and then press the RETURN key. Any other reply will be assumed to be equivalent to "NO". You should then have the Main Menu on the screen:

```
Select from:

N - new program

R - run program

E - edit program

D - delete program

L - list program

P - print program

S - save program

I - input program

F - finish
```

Before you can write any program you need to plan out what you want the 'dot' to do in order to produce the required pattern. Let's take a simple example: to draw a square we need to move the 'dot' a certain number of steps, say 250, and then turn through 90 degrees. This will draw one side so the right number of 'moves' and 'turns' should give a square.

In order to put in program commands you will first need to start a new program.

Type in 'N' and you should see the message:

```
Type in the name

for your program

?
```

At this point you should type in a suitable name, eg. SQUARE, and then press the RETURN key. You only need to press the RETURN key if there is a flashing line on the screen. (See page 4 - Running the System) The screen should now look like this:

```
Select from :          SQUARE
                       2
M - move               4
                       6
T - turn               8
                       10
H - home               12
                       14
R - repeat             16
                       18
E - end repeat         20
                       22
P - program            24
                       26
U - pen up             28
                       30
D - down               32
                       34
F - finish             36
                       38
K - kill last line     40
                        .
                        .
                        .
```

The numbers on the right hand side are where the program lines will be listed. The Command Menu has now replaced the Main Menu on the left hand side of the screen.

Now type in the program, line by line. First type 'M' when the system will reply:

```
MOVE ...

how many steps

?
```

Reply with 250 <RETURN> and continue to enter commands in sequence until the program is nearly complete.

If you make any mistake in a line then you can use the 'K' option to delete the last line that you entered and then re-enter the line correctly. If you press 'T' instead of 'M', or vice versa then complete the incorrect line, 'kill' it and then enter it properly. If you have typed in the wrong value , eg. 259 instead of 250, then the DELETE key (bottom right of keyboard) will erase it from the entry line before you press RETURN.

The next diagram shows what the screen ought to look like:

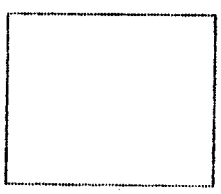
```
Select from :          SQUARE
                       2  move 250
M - move               4  turn 90
                       6  move 250
T - turn               8  turn 90
                       10  move 250
H - home               12  turn 90
                       14  move 250
R - repeat             16
                       18
E - end repeat         20
                       22
P - program            24
                       26
U - pen up             28
                       30
D - down               32
                       34
F - finish             36
                       38
K - kill last line     40
                       .
                       .
                       .
```

Now press 'F' which will add the word 'finish' to the end of the program and return you to the Main Menu. You can now 'run' your program by using the 'R' option. The system will reply with the message:

```
Which program do

you wish to run

?
```

The list of programs, with their code numbers, will be printed on the right hand side of the screen (there is only one at the moment) and you should type 1<RETURN>. The screen should clear and then draw a square:

```
Press SPACE or P
```



Pressing the SPACE bar will return you to the Main Menu.

This simple example shows how a simple sequence of commands can produce a shape. A computer program could be described as a sequence of commands that instructs a computer to carry out a particular task - so our few lines are a program.
Try out these 2 examples:

TEE

```
turn 90
move 250
turn 90
move 100
turn 180
move 200
finish
```

CROSS

```
move 250
move -100
turn -90
move 100
move -100
finish
```

What do the negative values for 'move' and 'turn' do?

Now try to write a program to draw out the intial of your name or some other simple shape.

Unless instructed to do so, a computer will always obey commands in the order in which they are written. DRAW allows two basic variations on this idea:

repeating a set of instructions and
calling an already existing program.

and we shall look at each of these ideas in turn.

Using 'repeat - end repeat'

Go back to the first program, SQUARE. It could be shortened if we realise that another way of expressing the instructions would be to say:

```
repeat 4 times:
     move 250 steps, turn 90 degrees
```

This construction, or program structure, is available in DRAW by using the two commands 'repeat (N times)' and 'end repeat'. These two commands are used rather like brackets and they are placed around the lines of the program that are to be repeated. Look at the program below which will also draw a square:

SHTSQUARE

```
repeat 4
 move 250
 turn 90
end repeat
finish
```

The output of this program is identical to the first but the program is slightly shorter. This structure comes into its own when polygons with greater numbers of sides are attempted. Try these programs:

```
HEXAGON
repeat 6
 move 100
 turn 60
end repeat
finish
```

```
OCTAGON
repeat 8
 move 100
 turn 45
end repeat
finish
```

```
DECAGON
repeat 10
 move 100
 turn 36
end repeat
finish
```

In each case you should end up just where you started.

Now try a circle (= "move a little bit, turn a little bit").

Very attractive patterns can be achieved by having two loops in one program, where a basic shape is repeated a number of times but turning between each repetition. Try these:

```
BOXES
repeat 6
 repeat 6
  move 100
  turn 60
 end repeat
 turn 60
end repeat
finish
```

```
HEXAGON
repeat 6
 repeat 3
  move 100
  turn 120
 end repeat
 turn 60
end repeat
finish
```

Editing a program

At this point it would be worth exploring the way in which it is possible to change programs that have already been written. If there are a lot of alterations that would need to be done it is probably easiest to delete the program completely - by using the 'D' option from the Main Menu - and starting from scratch. However, if you only want to change one or two lines, or add or delete a line then use the 'E' option from the Main Menu and alter the program that way. Read through page 11 for the detailed instructions.

Use the Edit Menu to change the last program, HEXAGON, to produce a different sized pattern by changing the 'move' line. Then try the effect changing the outer repeat- loop to 'repeat 12'. You will also need to change the value of the second 'turn' line.

You will find that you can carry out most of the changes that you are likely to want to make in this way.

Calling Subprograms

When you have written, and stored, a program it is then possible to use it as a part of another program. For example, look at the program you BOXES on the previous page. That could be described as:

```
repeat 6 times -
     draw a square and turn through 60 degrees
```

Assuming that you already have a program which draws a square - e.g. SHTSQUARE - then you can use the 'P' option from the Command Menu to simply 'call' SHTSQUARE instead of typing the lines out again.

First call the new program BOXES2 and open up the 'repeat 6' loop. Then press 'P' and you will be asked:

```
Which program

(type 0 for list)

?
```

The system is asking you for the number of the program that you would like to insert at this point - if you have fogotten, or don't know the number then typing 0 will give you the list of those available. If you have been just typing in the programs outlined so far then SHTSQUARE is likely to be number 4. Which ever number it is, type that number and the system will put the name of that program on the current line of the program. Now finish typing in lines so that you get:

```
BOXES2

repeat 6
 SHTSQUARE
 turn 60
end repeat
finish
```

Run this program and the output should look the same as BOXES did before but the way it works is rather different. When the computer reaches the instruction SHTSQUARE, it finds that program, executes all the steps in it and then returns to the first program to execute the line after the 'program call'. The next diagram might help to explain what is happening:

```
BOXES2                    SHTSQUARE

repeat 6                  repeat 4
 SHTSQUARE                 move 250
 turn 60                   turn 90
end repeat                end repeat
finish                    finish
```

The lines are intended to show the 'flow' of the program. The subprogram is, in fact, called 6 times as a part of the repeat loop of the program BOXES2
This technique allows you to write a program in short sections, each of which can be tested, and then string them all together to make the final program. It is also possible to have subprograms which call other subprograms:

```
PATTERN

repeat 12
 LOLLYPOP
 turn 30
end repeat
finish
```

```
LOLLYPOP

move 150
turn -81
SMCIRCLE
turn 81
move -150
finish
```

```
SMCIRCLE

repeat 20
 move 50
 turn 18
end repeat
finish
```

Again the lines are intended to show the program flow. Try these programs out and see if you can follow what is happening. The flower design on the front cover of this manual was produced by a series of such program calls. The program details are listed below:

```
FLOWER

pen up
move -250
pen down
repeat 10
 PETAL
 turn 36
end repeat
CENTRE
STEM
turn 90
PETAL
turn 120
PETAL
finish
```

```
PETAL

repeat 2
 repeat 6
  move 50
  turn 12
 end repeat
 turn 108
end repeat
finish
```

```
CENTRE

repeat 20
 move 120
 move -120
 turn -18
end repeat
finish
```

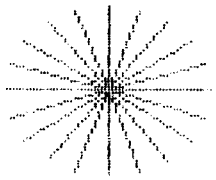
```
STEM

repeat 8
 move 100
 turn -10
end repeat
finish
```

The main program, FLOWER, first of all moves the 'dot' to the left so that the drawing fits on to the screen. It then draws the main flower outline which is made up of 18 PETALS. Running PETAL should give:

Press SPACE or P

The program CENTRE, which on its own gives:

Press SPACE or P



is then added to make the flower more interesting. The STEM is simply an arc of a circle, note the negative angle needed to turn to the right, and finally two more PETALs are added at suitable angles to look like leaves.

Programs with variable SIZE

This is about the most complicated feature of the DRAW language. When you have a basic shape it often happens that you will want to use it at different times but you might want it to give a different size each time. This can be achieved by using the SIZE parameter available in DRAW. Read through page 13 of this manual but, for the time being, ignore the last section.

Enter the program SQUARE SIZE given on page 13 and then type in the following very short program. Note: when you select the subprogram the system will ask:

PROGRAM: SQUARE

SIZE = ?

This value of SIZE is the value that you want to be passed to the subprogram.

```
THREESQS

SQUARE 100
SQUARE 200
SQUARE 300
finish
```

Run this program and see if you can follow how it produces the pattern. The diagram at the top of the next page is an attempt to show what is happening.

```
THREESQRS                    SQUARE SIZE

SQUARE 100 ---1--->          repeat 4
                               move SIZE  :  100  200  300
SQUARE 200                     turn 90
                             end repeat
SQUARE 300                   finish
finish
```

Each time the program SQUARE SIZE is called it collects the value given in the calling program and substitutes that for the word SIZE in the 'move' line.

Here is a rather useful subprogram. It is a simple way of moving the 'dot' a specified number of steps in the current direction:

```
JUMP SIZE

pen up
move SIZE
pen down
finish
```

Use JUMP SIZE and a subprogram written to produce a circle combined together to produce the 5 interlocked rings of the Olympic Symbol.

You should now be in a position to follow what is happening in the programs given at the bottom of page 14.

From now on you're on your own! There are many limitations on this simple system but there are also very many possibilities. If it is not too trite to say it - the possibilities are largely limited by your imagination.

