nascom

Nascom Microcomputer DOCUMENTATION

USER'S GUIDE

COLOUR GRAPHICS SOFTWARE

FOR THE

ADVANCED VIDEO CARD

Issue 1.1.

Nascom Microcomputers
Division of Lucas Logic Limited
Welton Road Wedgnock Industrial Estate
Warwick CV34 5PZ
Tel: 0926 497733 Telex: 312333

Lucas Logic

TABLE OF CONTENTS

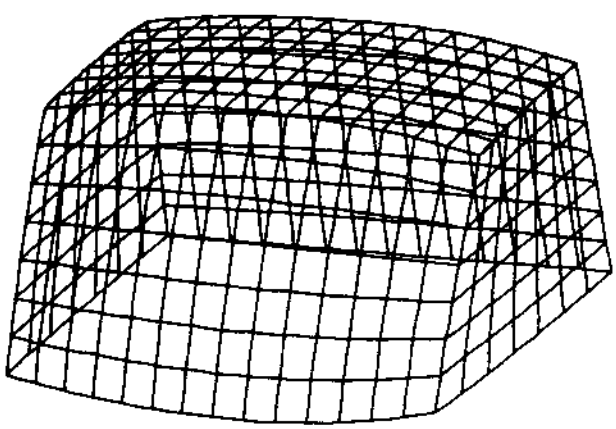
CHAPTER 1

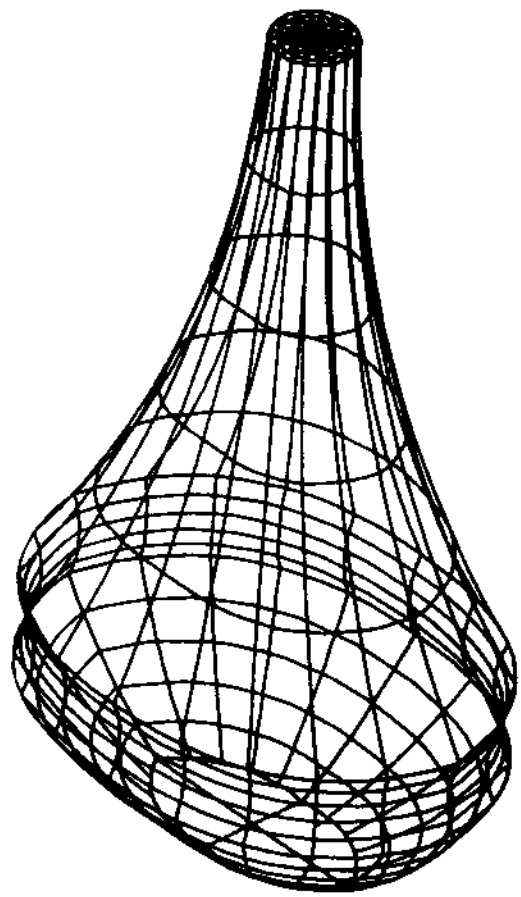
INTRODUCTION

"A PICTURE IS WORTH A THOUSAND WORDS"

The above may not always apply, but when dealing with the computer / man interface the ability to convey information pictorially greatly increases "useability". Sometimes information is difficult or impossible to describe using only numbers and words but becomes comprehensible in graphic form. The application of graphics to scientific,engineering,business and other problems helps the computer user to gain maximum efficiency.

The NASCOM ADVANCED VIDEO CONTROLLER and it's versatile firmware can create and display graphics in a wide range of colours, this adds the ability to emphasize certain areas or to show relationships between sets of data.





USING THIS MANUAL

It is assumed that you are already familiar with the concepts and material presented in the manuals for your NASCOM microcomputer. For details on installation of the AVC refer to the AVC HARDWARE MANUAL.

To gain maximum benefit from this manual it is suggested that it is read completely making full use of the examples and tutorial problems. Once familiarity has been gained, use of the programming card or syntax reference can be used as a memory jogger, with reference to the main manual for more specific details.

A demonstration program is included on disk versions, to run this demonstration program type the following;

```
JE: RUN ME
```

This will give a brief insight into colour graphics, now read on and enjoy the wonderful world of computer graphics.

IMPORTANT NOTE

If the NAS-SYS:3 monitor is not being used then after every system power up the AVC must be paged out. To do this type:

```
O B2 8Ø
```

This does not apply to CP/M users.

SYSTEM PREPARATION

IT IS IMPORTANT THAT A BACKUP COPY IS MADE NOW OF THE AVC SOFTWARE.

Installation of the AVC is covered in the AVC HARDWARE MANUAL, this should be consulted with reference to the NASCOM MANUALS.

This manual covers the colour graphics associated with;

NASCOM ROM BASIC
NASCOM ENHANCED BASIC
MACHINE CODE INTERFACE
operating under NAS-DOS

NASCOM ENHANCED BASIC (See separate manual)
MACHINE CODE INTERFACE (" " ")
operating under CP/M

The explanation of the graphics commands in this manual are limited to a NAS-DOS system, for details of the other system types refer to APPENDIX C Also the commands relate to use under NASCOM ROM BASIC for details of the differences between basics refer to APPENDIX F.

When executing colour graphics commands from NASCOM ROM BASIC all commands must be preceded with the "SET" command, this in no way affects the normal use of the "SET" command using the standard NASCOM VIDEO.

For details of interfacing the AVC to NASCOM 1 microcomputers and to other computer systems refer to the applicable manual and to the AVC HARDWARE MANUAL. Information is also provided for non "NASCOM" perpiherial cards.

LINKING NASCOM ROM BASIC TO THE COLOUR GRAPHICS SOFTWARE

Enter NASCOM ROM BASIC using the J command, then execute the colour graphics software which is stored on disk in a file called G48.
(Note for a 48K byte system use G48, for a 32K byte system use G32).
The system should respond with;

```
   NASCOM AVC / ROM basic linked
   Copyright (c) LUCAS LOGIC 1982
-- NAS-DOS 1 -->D?????<-- NAS-SYS 3 --
```

Then re-enter basic using the Z command.

CHAPTER 2

BASIC PLOTTING

To proceed with the examples in this chapter link the COLOUR GRAPHICS software, see previous chapter.

The principles of operation of displaying a graphics picture are transparent to the graphics programmer, but to gain a better understanding of graphics a brief explanation is given of the display system.
The NASCOM ADVANCED VIDEO CONTROLLER (AVC) works on the BIT MAPPED RASTER SCAN principle, this means that the computer can directly address and set the colour of any graphics point (called PIXELS) on the display. This provides a much more flexible system than programmable character generator graphics systems. While these PCG systems offer faster graphics generation in certain applications they suffer from programming difficulties and are restricted by the number of different shapes they can generate, normally only 128 or 256. The "SET" command from NASCOM ROM BASIC is an example of PCG.
There are other graphics systems, VECTOR SCAN being the best known. But vector scan display devices and controllers offer less colour shades and are considerably more expensive.
The display produced by the AVC can be best described as a two dimensional grid, the grid is divided up into small squares (called PIXELS, Picture Elements) each of which can be independently controlled by the computer. When viewed from a distance these combinations of pixels merge together to form shapes. This is a similar effect that newspaper photographs use, if you look closely at a newspaper photograph the individual picture elements can be seen.
The AVC uses the raster scan principle, an electron beam scans across the face of the display (monitor, television) from left to right, then moves down a line and repeats the left, right, down sequence till it reaches the bottom of the screen. This is similar to the way a person reads a page of text. As the beam traverses the screen it's energy is converted into light by the special coating on the inside of the screen. The intensity of the light produced can be varied by changing the energy of the beam. Colour displays are produced by having three coatings (phosphors) to give the three primary colours. The whole screen is scanned fifty times a second which is just fast enough for the human eye to average out the light levels, thus a stable picture is viewed and not just a moving spot of light. This is exactly the same way in which a television picture is formed.
Whereas with raster systems all the possible pixels are traversed by the beam in vector scan systems the beam is moved only to the pixels which require illumination. This provides very fast line generation but does not give constant illumination.
The larger the number of pixels then the better the definition of the picture formed, the AVC has two resolution modes;
SINGLE DENSITY: which produces 392 horizontal and 256 vertical pixels and
DOUBLE DENSITY: which produces 784 horizontal and 256 vertical Pixels.

PREPARING THE PLOTTING AREA

Just as an artist requires a clean "canvas" to start a new painting, the graphics programmer (computer artist) requires a clear display screen to begin. To clear the screen on the AVC the BACKGND command is used (background). However before clearing the screen or beginning to paint the canvas must be on the easel, to achieve this with the AVC we must define the AVC's operating MODE.
Try the following;

```
10 REM For commands in ROM BASIC use the prefix of SET
20 REM
30 REM CLEAR THE SCREEN
40 MODE
50 BACKGND
```

The display will now be clear, to display a coloured canvas try the following;

```
10 REM Primary/secondary background colours
20 REM
30 MODE
40 FOR COUNT = 0 TO 7
50 FOR KOUNT = 1 TO 1000 : NEXT KOUNT
60 BACKGND COUNT
70 NEXT COUNT
```

The display will be set to all the primary and secondary colours, the MODE and BACKGND commands are explained fully further on.

The previous program used with NASCOM ROM basic would have lines 30 and 60 changed as follows;

```
30 SET MODE

60 SET BACKGND COUNT
```

COLOUR ON THE AVC

Before continuing with colour graphics the concept of 'colour' and how it is implemented on the AVC must be understood.
Colour is a complex adjective used often in everyday life, but it is difficult to describe objectively. The scientific definition of colour is the wavelength of electromagentic energy in the visible spectrum, to the casual observer it is red,green,pink etc. There are two main subjective methods of defining colour;

HSL: HUE,SATURATION,LUMINOSITY

With this method the 'HUE' defines the hue of the colour, hue is normally confused with colour but is a different concept. The amount of hue within a colour is defined by the 'SATURATION', and is defined as the percentage deviation from an achromatic standard (a non coloured standard) such as black,white or some grey scale. The amount of transmitted (or reflected) energy is the intensity or 'LUMINOSITY' of the colour. The three components of colour H,S and L are shown in fig 1 on the coloured insert.

PRIMARY VECTORS: RED,GREEN,BLUE

With this method colour is defined by the three primary colour vectors RED,GREEN and BLUE, these vectors are equally spaced on a circle, the colour being defined as the relative lengths of the vectors. Thus each vector defines the intensity of the primary colours. Note that the three primary colours RED,GREEN and BLUE are the transmitted colour primaries not the reflected colour primaries. The three components of the PRIMARY VECTOR method (called the R,G,B method) are shown in fig 2 on the coloured insert.
The method used on the AVC is the R,G,B method.
The colour is defined as the intensities of the three primaries;

```
RED intensity, GREEN intensity, BLUE intensity
```

Try the following;

```
10 REM 4913 COLOUR SHADES
20 REM
30 FOR COUNT = 1 TO 10
40 BACKGND 17*RND(1),17*RND(1),17*RND(1)
50 NEXT COUNT
```

The range of each intensity is from 0 (zero colour) to 16 (full colour), note these are integer values which explains why

```
17*RND(1)
```

gives a number in the range 0 to 16. The basic internally truncates the number, an alternative could be;

```
16*RND(1)+0.5
```

With a 17 step range for each of the primary colours there are 4913 (17^3) colours available. Although the BACKGND command has been used in the examples the command parameters apply to foreground colours as well, using the COLOUR command.
You will have already have noticed that when non primary or secondary colours are displayed there is a pattern effect, this is due to the way the extended colours are obtained. Each pixel in the display can only be set to one of the eight standard colours, the extended 4913 colours are obtained by arranging groups of pixels together and forming larger pixel units. As each individual element of this larger pixel can be one of eight colours there are many colour combinations. This method of obtaining more colours is called 'dithering'.
The entire 392 by 256 (single density) graphics raster is divided up into 4 by 4 arrays of pixels, each containing 16 elements, each element a pixel.

Y address					
	3	13	1	16	4
	2	5	9	8	12
	1	15	3	14	2
	Ø	7	11	6	1Ø
		Ø	1	2	3

COLOUR MATRIX

X address

Using colour in this way does not reduce the available horizontal or vertical resolutions, but if used at a single pixel level the colour of a pixel will depend on both it's colour and it's position. This is not as such a serious limitation as it first seems as the extended colours are normally used to give shade effects. The software checks the two lowest address bits of the pixel position and with reference to the colour matrix set that pixel if the matrix number is lower than or the same as the primary colour intensity. This sequence is repeated for the three primary colours, RED,GREEN and BLUE.
Thus for a colour of;

RED = 7 GREEN = 3 BLUE = 12

The three primary colour maps are;

```
13   *  16   *      13   *  16   4      13  *  16  *
 *   9   8  12       5   9   8  12       *  *   *  *
15   *  14   *      15   *  14   *      15  *  14  *
 *  11   *  1Ø       7  11   6  1Ø       *  *   *  *

    RED                 GREEN               BLUE
```

The three colour matrix maps are superimposed to give the final colour.
By chosing even intensities the effective matrix pixel block is

reduced to a 4 by 2 matrix, if multiples of 4 are used then the matrix block is further reduced to a 2 by 2 matrix, this enables the reduction of the pattern effect. Mathematicians will have noticed that a block of 4 by 4 elements each of which can have 8 states (colours) produces more than 4913 colours. The limitation to the 4913 colours reduces the aliasing effect. Aliasing is the overall pattern effect that is sometimes noticed on television pictures, especially on news presenters suit jackets and on western "wagon" wheels which rotate backwards.
After the slight deviation into colour let's return to graphics with the BACKGND command.

format;

```
     BACKGND RED intensity, GREEN intensity, BLUE intensity
or   BACKGND colour,carriage
```

If only one parameter is included after the BACKGND command then this is taken to be a primary/secondary colour and is selected by a number in the range Ø to 7, any value outside this range will cause a basic OV error (overflow) as will all commands with invalid parameters.

Relationship between colour parameter and pixel colour

parameter	colour
Ø	BLACK
1	RED
2	GREEN
3	YELLOW
4	BLUE
5	MAGENTA
6	CYAN
7	WHITE

If no parameters are entered then the default value is Ø (BLACK). The second parameter if entered is a carriage control. The display can best be regarded as a flat bed plotter, the carriage when down (1) will plot as requested but when up (Ø) no plotting takes place. The carriage overrides the pens but more about this later.
If more than two parameters are entered after the BACKGND command then the first three parameters define the RED, GREEN and BLUE intensities. The fourth parameter is the carriage control. The BACKGND command always uses the DOMinant plotting mode (see later on).
The speed of the BACKGND command is dependant on the colour and MODE selected, the order of execution is, fastest first;

a] Default WINDOW, white or black

b] Default WINDOW, other primary/secondary colours

c] Non default WINDOW or non primary/secondary colour

The special AVC MODE 3 which provides both a single and a double density graphics screen modifies the BACKGND command,

When used with BACKGND a] and b] both AVC screens are set to the selected background colour irrespective on the state of the PEN, however the BACKGND c] does take notice of the PEN state (see the MODE command later on).
See the syntax reference for details of all command default values.

TUTORIAL

Write a basic program which will convert colours defined as hue,saturation and intensity into primary vectors (red,green and blue), then display these as BACKGND colours. Hint use the SIN function.

SELECTING THE AVC OPERATING MODE

We have previously seen that to prepare the AVC for graphics plotting the MODE command must be used:
format

```
MODE avcmode,avc output
```

There are four AVC operating modes;

avcmode=Ø Displays the external video input to the AVC. This would normally display the standard 48 column NASCOM screen.

avcmode=1 This is the single density eight colour mode (392 by 256 pixels)

avcmode=2 This is the double density two colour mode (784 by 256 pixels)

avcmode=3 This is the dual density mode providing two two colour planes of 784 by 256 pixels (green) and 392 by 256 pixels (blue). This enables four colours to be displayed blue,green,cyan,black.
The colours obtained by the three planes can be software redefined if AVC model B is used, refer to your AVC hardware manual.
The default value of avcmode is 1, the single density eight colour display.

The second parameter used with MODE selects which combination of the three available planes are displayed;

avc output =	Ø	no colour planes displayed
	1	only red " "
	2	only green " "
	3	red and green " "
	4	only blue " "
	5	blue and red " "
	6	blue and green " "
	7	all planes displayed

The default values for the avc output parameter are;

MODE =	Ø	avc output =	N/A
	1	"	7 (all planes displayed)
	2	"	2 (green)
	3	"	6 (blue and green)

If the mode is changed so as to change the resolution then an automatic DEFAULT is performed, note MODE Ø does not envoke an automatic DEFAULT, this relieves the graphics programmer of remembering to do so. The DEFAULT command is explained further on.

SELECTING A PLOTTING COLOUR

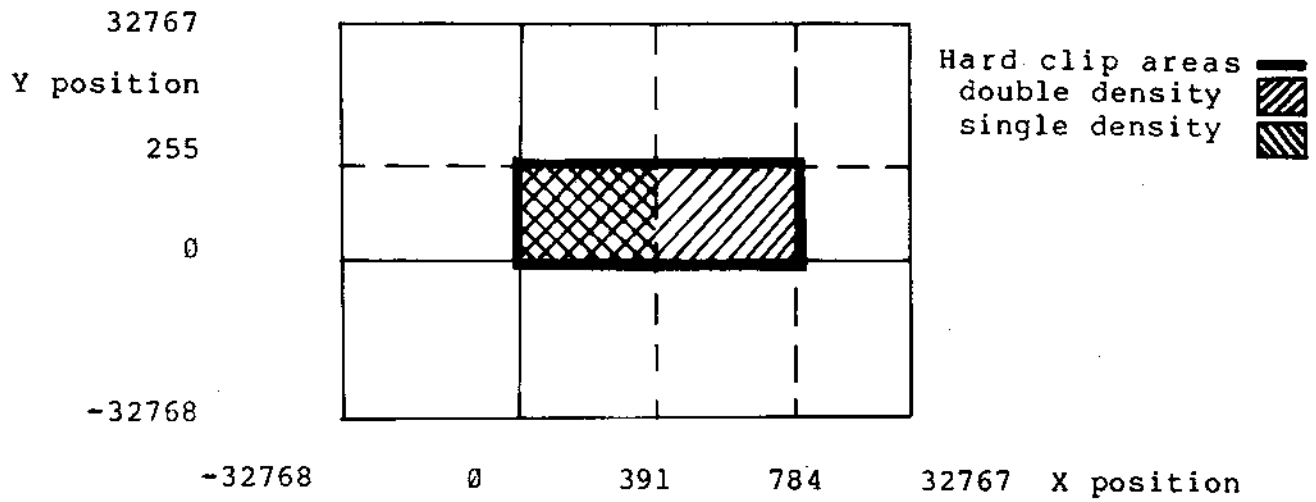
The foreground plotting colour is selected in an indentical way to that of the BACKGND except that there is no carriage parameter.
format

COLOUR colour
COLOUR RED intensity, GREEN intensity, BLUE intensity

The colour parameter default is to the colour white, ie, a colour of 7 or intensities of 16,16,16.
Note The specified colour applies to all plotting commands unless redefined by a COLOUR,DEFAULT or MODE change. All plotting commands can use both types of colours, primary/secondary and extended.

PLOTTING POINTS

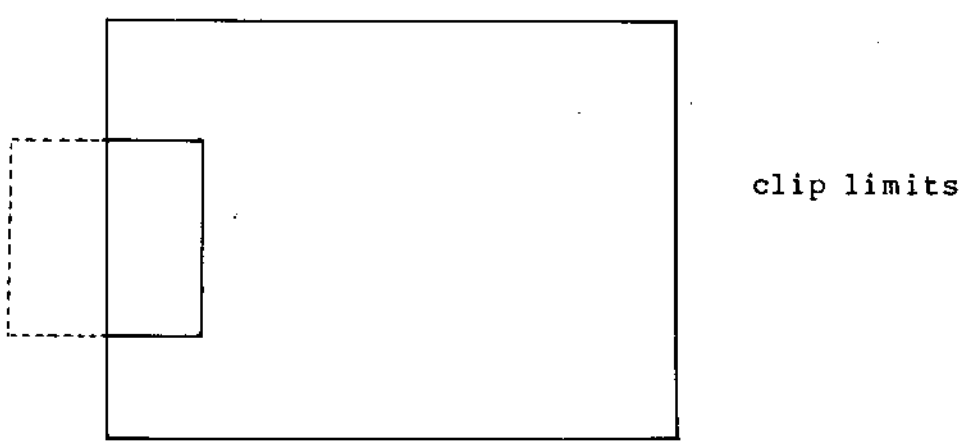
After clearing the screen and selecting an operating mode we can now start serious graphics programming, plotting points.
The plotting area is a two dimensional grid of pixels, each of which can be indivually coloured. Positional information for pixel plotting is provided by entering X and Y coordinates, the available plotting grid being a 65,536 by 65,536 X,Y array.



The 65,536 by 65,536 plotting area is arranged as -32768 to 32767 for both axes, but the visible area is;

Single density	0 >= X =< 391
	0 >= Y =< 255
Double density	0 >= X =< 783
	0 >= Y =< 255

These visible areas are defined as 'HARD CLIP' areas because if a pixel (or pixels) is plotted outside this area they are not displayed. The HARD CLIP area limits may be reduced to 'SOFT CLIP' areas by the use of the WINDOW command.
The following diagram illustrates the effect of HARD CLIP on a plotted square.

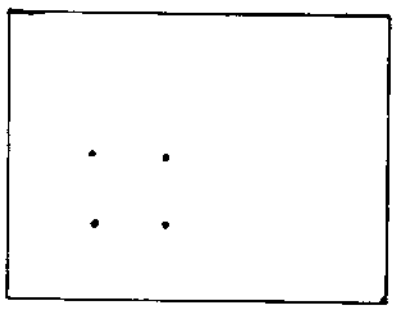


Only the solid portion of the square is displayed. If plotting is attempted outside the plotting matrix (65,536 by 65,536) then a basic "FC" error (function call) will result.

Try the following;

```
10 REM PLOTTING POINTS
20 REM
30 MODE:BACKGND
40 PLOT 100,100
50 PLOT 200,200
60 PLOT 100,200
70 PLOT 200,100
```

The resulting display is;



Four white pixels forming a square on a black background.

TUTORIAL

a] Repeat the example but plot four RED pixels on a CYAN background, try not to refer to the listed answer below.

```
10 REM plotting coloured points
20 REM
30 MODE:BACKGND 6                :REM select colour CYAN (6)
40 COLOUR 1                      :REM select colour RED (1)
50 PLOT 100,100:PLOT 100,200
60 PLOT 200,100:PLOT 200,200
70 PLOT 500,500
```

Note like all other basic statements more than one graphics command can be used on a line if seperated by colons (:).

Why does line 70 not produce a point ?

b] Make a modification to plot a BLUE pixel equidistant from the four red pixels.
c] Now use the MODE command to eliminate the red pixels but leave the CYAN background unchanged.

answer: add the following to your program

```
100 MODE 1,6
```

The red pixels can be restored by typing;

```
MODE     in direct mode or in the program
```

Note all graphic commands can be used either in program or direct mode.
Now try;

```
100 MODE 1,6
110 FOR KOUNT = 1 TO 500 : NEXT KOUNT
120 MODE 1,7
130 FOR KOUNT = 1 TO 500 : NEXT KOUNT
140 GOTO 100
```

You will now have flashing red pixels, press ESC (shift ENTER) to abort the program.

PLOT command

Format PLOT X,Y,[MODE,CARRIAGE]

The parameters enclosed by the square brackets are optional (this applies to all command formats).
The mode parameter of the PLOT command allows the X and Y coordinates to be entered either as absolute (mode=0) or relative to the last pixel plotted (mode=1). The carriage operates as defined previously.

The concept of the cursor now requires some explanation
The graphics cursor is a non displayed cursor, it is effectively the absolute position of the pen, whether or not the pen(s) are enabled. It is in effect the position of the last plotted pixel if that pixel were to be plotted. Many graphics commands are much simplified if plotting can be achieved relative to the last point plotted, this last point is stored in the cursor. The position of the cursor can be read from basic using the cursor command, but more of this later.

The default for the mode command is to absolute plotting (0).
Absolute plotting as described previously plots the actual X,Y position, relative plotting adds the entered X parameter to the cursor X position and the entered Y parameter to the cursor Y position to get the pixel position.
Example

```
10 REM Relative plotting
20 REM
30 INPUT "Absolute [A] or Relative [B]";A$
40 IF A$="A" THEN AB=1 : REL=0 : GOTO 70
50 IF A$="R" THEN AB=0 : REL=1 : GOTO 70
60 GOTO 30
70 MODE:BACKGND
```

```
 80 REM ABSOLUTE PLOTTING
 90  PLOT  100,100,0,AB:PLOT  200,100,0,AB
100  PLOT  200,200,0,AB:PLOT  100,200,0,AB
110 REM RELATIVE PLOTTING
120 REMEMBER LAST CARRIAGE POSITION WAS 100,200
130 PLOT 100,0,1,REL :  PLOT 0,-100,1,REL
140 PLOT -100,0,1,REL:  PLOT 0,100,1,REL
150 FOR KOUNT = 1 TO 5000 : NEXT KOUNT
160 SET MODE 0 : GOTO 10
```

NB. If dual screens (or split screens) are available lines 150-160 may be deleted.

The following is a simple plot of the polar response of a microphone, note the use of functions within the PLOT function. The plot shows how relative line plotting can produce graphics shapes without using LINE or other shape functions.
The various multiplication factors in the example are used to centralise the plot.

```
 10 REM A cardioid plotting program
 20 MODE : BACKGND
 30 PI = 3.141592654 :rem this line not required in extend.
 40 COLOUR 1                : REM plot in red
 50 REM     R=a(1-COS(a))
 60 FOR COUNT = 0 TO 2*PI STEP PI/100
 70 R=(1-SIN(COUNT))*80
 80  PLOT 80,127,0,0
 90  PLOT -R*SIN(COUNT)*1.5,R*COS(COUNT),1
100 NEXT COUNT
110 COLOUR 2               : REM green for axes
120 REM plot axes
130 FOR X = 0 TO 390
140 PLOT X,127
150 NEXT X
160 FOR Y = 0 TO 255
170 PLOT 80,Y
180 NEXT Y
```

TUTORIAL

a] Modify the above program to include TIC marks on the X and Y axes.
b] The above program draws the axes using PLOT, this is simple because only one axis has to be changed, write a program to join the points (10,50) to (130,200) with a series of points to form a straight line. Start initially with the formula;

$$Y = Mx + C$$

Where M is the gradient and C is the intercept on the Y axis. When this is working try to work out a faster algorithm.

LINE PLOTTING

If you managed the second part of the last tutorial you will have appreciated that a LINE command would be very useful.

Format

```
LINE X,Y,[MODE,CARRIAGE]
ALINE X,Y,X1,Y1,[MODE,CARRIAGE]
```

LINE plots points from the current carriage position to the position X,Y or if mode is set to 1 to the relative position X,Y from the cursor position. The mode parameter operates in exactly the same way as the PLOT command's mode parameter.
ALINE is similar to LINE except that a line is not drawn from the cursor position to the defined end point but from an entered point instead, ie. the line drawn is not affected by the cursor position.

Example

```
 10 REM line plotting
 20 MODE:BACKGND:COLOUR 5
 30 ALINE 100,200,150,150
 40 LINE 200,200
 50 ALINE 200,200,-100,0,1
 60 REM now remove the triangle
 70 COLOUR 0
 80 ALINE 100,200,150,150
 90 LINE 200,200
100 LINE 100,200
110 GOTO 20
```

TUTORIAL

a] Modify the cardioid plot described earlier to join all points to the centre (hint: replace PLOT with LINE)
b] Further modify the cardioid to plot with a solid curve.
c] Using relative line plot program a subroutine to draw polygons, with programmable centre points,colours and number of sides. How many sides are required to produce a circle ?

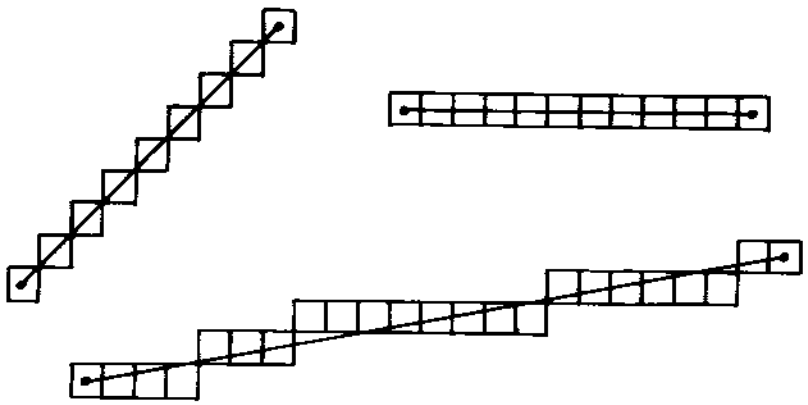
The last question leads on to a discussion concerning the nature of lines drawn using raster scan systems: the dreaded "jaggies".

Try the following;

```
 10 REM jaggie demonstration
 20 MODE:BACKGND
 30 FOR COUNT =1 TO 50
 40 COLOUR 1+7*RND(1)
 50 ALINE 392*RND(1),256*RND(1),392*RND(1),256*RND(1)
 60 NEXT COUNT
```

You will notice that horizontal and vertical lines are perfectly straight, while others have breaks in them, called 'JAGGIES'.

These jaggies are caused by the limited resolution of the screen, the line plotting algorithm tries to find the best line by selecting the closest available pixels to the true line.



Reducing the size of the pixels reduces the jaggie effect, try running the previous example but with double density selected (MODE 3, the colour statement can be removed).
If the output is directed to a pen plotter then usually no jaggies will be noticed as pen plotters have much higher resolutions.

The line plotting algorithm will not always produce jaggies in the same place for a given line plotted in different directions. So if you intend to erase a line it must be plotted in the same direction as it was drawn.

To reduce the jaggie effect try;

a] Draw thick lines. Use adjacent lines or rectangle fill.
b] Use dotted/dashed lines. Make the jaggies fall in the spaces.
c] Move the line endpoint slightly. Forty five degree lines exhibit minimal jaggies.

Having mentioned dotted/dashed lines it is now an appropriate time to mention the DASH command.

Format

DASH [lineon,line off,last pixel]

The DASH command sets the duty cycle of dotted lines, the line on parameter defines the number of 'on' pixels, while the line off parameter defines the number of 'off' pixels. The line on

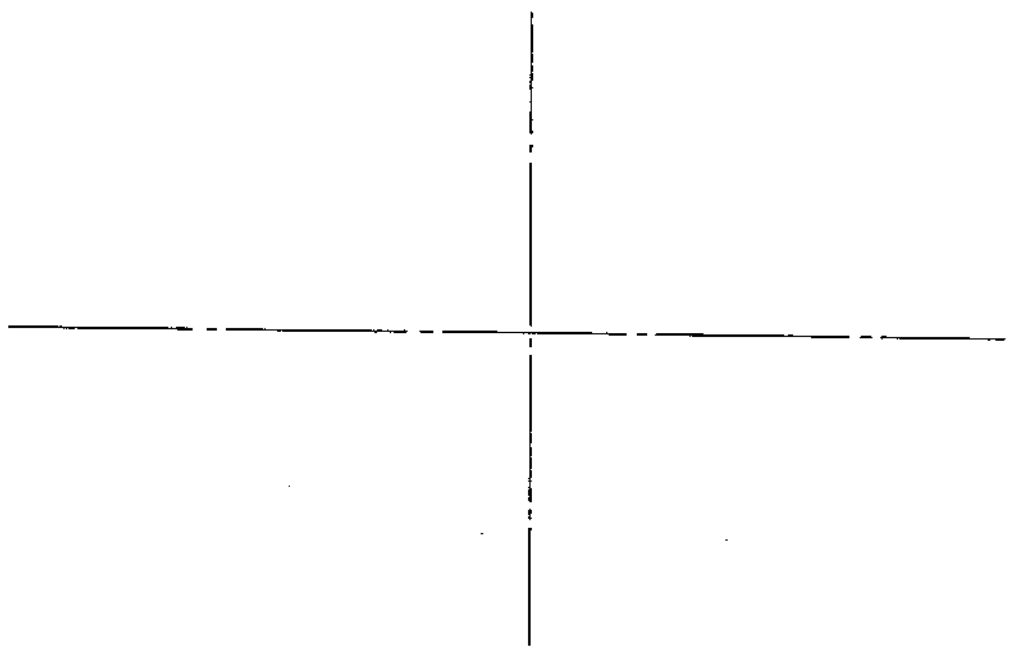
defaults to 1 the line off to Ø. So long as line off is Ø a solid line is drawn. Every time a line is started plotting begins with the solid (on) portion. Dash does not effect the filling of shapes or PLOTted pixels and is not itself affected by any scaling. The dash function does not plot black in the spaces but takes all the pens up during this time (carriage).
Included in the DASH command is a parameter which instructs the line command to plot or not plot the last pixel of a line. The default is to 1 which plots the last pixel, a value of Ø does not plot the last pixel. This may seem a totally useless facility however it comes into it's own when used with plotting using the XOR or XNOR function, but more of that later on.

Example

```
10 REM DASHED/DOTTED LINES
12 MODE:BACKGND
20 FOR I=1 TO 255 STEP 2
30 COLOUR 1+7*RND(1)
40 DASH 30*RND(1),30*RND(1)
50 ALINE Ø,I,390,I
60 NEXT I
```

TUTORIAL

Use the DASH command to draw two centre lines.



DIFFERENT TYPES OF PLOTTING

THE PEN COMMAND

We have seen previously that the display can be described as a flat bed plotter with a moveable carriage. On this imaginary carriage are three coloured pens RED, GREEN and BLUE, each of which can be independantly enabled or disabled using the PEN command. Another representation (closer to reality) is that the PEN command can be used to write protect any or all of the primary colour planes. It must be remembered that the AVC operates with radiant energy and hence colour combination is additive rather than subtractive as with a pen plotter.
ie. combining the red,green and blue colours (light) produces white while combining red,green and blue colours (paint) produces black.

Format

```
PEN [red,green,blue,plot mode]
```

The defaults are all pens down and dominant plotting, a value of 1 enables the pen while a value of Ø disables the pen.
Example

```
10 REM PENS
20 COLOUR          :REM select white
30 MODE:BACKGND
40 PEN 1,Ø,Ø       :REM disable green and blue pens
50 ALINE 10,100,300,100
```

The example produces a red line even though the colour was specified as being white, because the green and blue pens have been disabled. Removing the green and blue colour components from white produces red.
It is worthwile noting at this stage that if the program were to be run again but with line 40 and 20 removed the same effect would still result. This is because the RUN command which under normal basic conditions initialises all the basic's parameters does not affect the graphics workspace, this is a definite advantage. The default command has the effect of restoring the graphics workspace.

The final parameter in the PEN command is the plotting mode, this defines how the actual pixels are to be plotted in relation to themselves and the existing colour of the pixel. This defines exactly how the pixels are changed and applies to all the plotting commands except the BACKGND command.
The plotting mode is the logical relationship between the colour of the plot and the colour of the pixel already at the plotting position. The default value is Ø which is the dominant mode, ie the pixel is plotted taking no account of the colour of the pixel already at the plotting position, the dominant mode always writes to all the three primary colour planes unless prevented by the pens being disabled.

PLOTTING MODE	LOGICAL RELATIONSHIP
0	DOMINANT or REPLACE
1	OR
2	XOR
3	AND
4	XNOR
5	NOT
6	NOR
7	NAND

Mode 7 is the inverse of mode 3
Mode 6 is the inverse of mode 1
Mode 5 is the inverse of mode 0
Mode 4 is the inverse of mode 2

The BACKGND command obeys the pen parameters of PEN but always uses the DOMINANT (0) plotting mode, note this does not change the current logged plot mode.
The PEN command is used for colour plane selection when using MODE 3, this is because the two planes are treated as separate displays for plotting purposes. Normally all plotting will be in the double density plane (green 784 by 256), however by deselecting the red and green planes (pens set to 0) plotting can be achieved in the blue single density plane.

Example

```
 10 REM MODE 3 PLOTTING
 20 REM draw a double density green square
 30 MODE 3:BACKGND
 40 ALINE 100,100,300,100
 50 LINE 300,200
 60 LINE 100,200
 70 LINE 100,100
 80 REM now plot a square in the single density plane
 85 PEN 0,0,1
 90 ALINE 150,150,200,150
100 LINE 200,200
110 LINE 150,200
120 LINE 150,150
```

The above example illustrates the effect of ASPECT RATIO, the green "square" is actually formed by drawing a rectangle, wheras the blue square dimensions are correct. The double density plane as it's name suggests has twice the amount of pixels available, but as the horizontal resolution is doubled the vertical resolution remains the same. In effect a scale factor of 0.5 is applied to all X coordinates. Most standard CRT type display devices have an aspect ratio of 4 to 3, this means that the picture is 25% longer than wide, ie. rectangular. The standard output from the AVC matches this ratio, thus a line drawn 100 by 100 relative produces a 45 degree line. The aspect ratio of the double density plane is 2 to 3, so the same line produces a 22.5 degree line to the vertical.

Manufacturing tolerances within the display device although on average producing an aspect ratio of 4 to 3 are not generally linear over the whole screen, this has the effect of shape distortion and is most easily seen on circles.

READING THE COLOUR OF PIXELS

There are many applications where the graphics programmer will require to be able to read the colour of a given pixel, this facility is catered for with the CHECK command

Format

```
CHECK variable,[X,Y]
```

The colour of the pixel is placed in the variable, which must be a numeric variable (arrays are allowed). The optional X,Y parameters specify the position of the pixel to be read, the default for this position is the cursor position.

Values returned in the variable

0	BLACK	colour of a pixel within the soft clip area.
1	RED	
2	GREEN	
3	YELLOW	
4	BLUE	
5	MAGENTA	
6	CYAN	
7	WHITE	
10	BLACK	colour of a pixel outside the soft clip but inside the hard clip area
11	RED	
12	GREEN	
13	YELLOW	
14	BLUE	
15	MAGENTA	
16	CYAN	
17	WHITE	
255	A pixel was referenced outside the hard clip area.	

The CHECK command responds to the PEN command for plane selection when used in MODE 3, ie. the check will normally operate in the double density plane returning colours green (2 or 12) or black (0 or 10) or 255. However if the red and green PENS are disabled then the check operates on the single density plane and returns blue (4 or 14) or black (0 or 10) or 255.
It should be noticed that the CHECK command only returns with the primary or secondary colour of a pixel, ie the component parts of the extended colours.

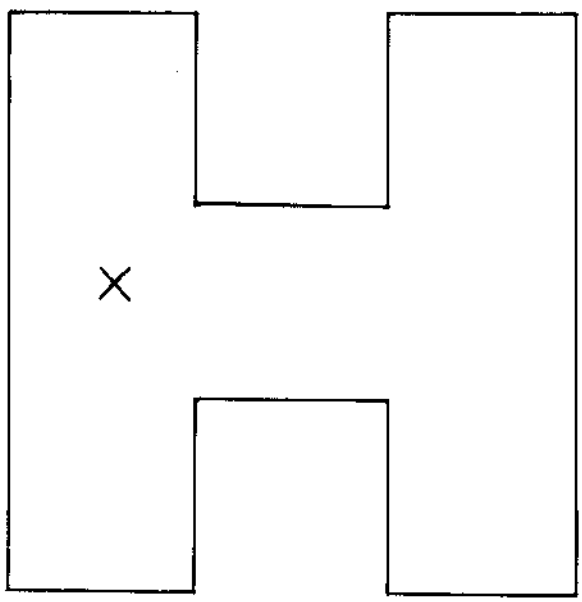
Example

```
10 REM CHECK EXAMPLE
15 MODE:DEFAULT:BACKGND
```

```
 20 REM this program takes ages !
 30 REM set a random pixel with random colour
 40 C = 1+7*RND(1)
 50 X = 392*RND(1) : Y= 256*RND(1)
 51 COLOUR C
 52 PLOT X,Y
 60 PRINT "Plotted a pixel at ";INT(X);",";INT(Y);"of colour
";INT(C)
 70 REM now find that pixel
 80 FOR X = 0 TO 391
 90 FOR Y = 0 TO 255
100 CHECK C,X,Y
110 IF C<>0 THEN GOTO 140
120 NEXT Y,X
130 PRINT "NOT FOUND!" : END
140 PRINT "Pixel found at ";X;",";Y;"with colour ";C
```

TUTORIAL

Using the CHECK and PLOT commands write a basic program to draw a closed shape (eg. a square) and fill it with a given colour.
Use the check command to search for the shape boundaries (place the initial start position within the closed shape). To check your program try filling the shape below, X marks the start point:-



We have seen that the colour of a pixel can be read with it's position either being entered or defaulting to the cursor position (last carriage position). There is a command called CURSOR which allows the user to directly read the cursor position.

Format

```
CURSOR X variable, Y variable
```

The X,Y position of the cursor is returned in the variables (x variable) and (y variable) respectively.

The concept of a HARD CLIP area has already been mentioned, this is in effect the valid plotting area;

```
SINGLE DENSITY      Ø-391,Ø-255
DOUBLE DENSITY      Ø-783,Ø-255
```

Although pixels can be plotted outside this area they will not be visible. These X,Y visible area limits are called the HARD CLIP area, because this area is clipped at the boundary. The SOFT CLIP area is a user defined rectangular area within the HARD CLIP area, this is selected by use of the WINDOW command. The SOFT CLIP area allows user definable windows.

Format

```
WINDOW [Xmin,Xmax,Ymin,Ymax]
```

The default values are to the hard clip limits, ie. Ø,Xmax,Ø,255, where Xmax is set to 391 for single and to 783 for double density. The values for the WINDOW command cannot exceed the hard clip limits. The WINDOW command has overall control over ALL plotting commands including BACKGND. When in MODE 3 the WINDOW command only affects the double density plane, the single density plane is left with hard clip limits only.
The WINDOW command can be used to provide a "split" screen facility for combining graphics with a full scrolling terminal text under CP/M. For further details of windowing the terminal refer to the CP/M AVC TERMINAL MANUAL, also refer to the NASCOM EXTENDED BASIC.AVC appendix at the end of this manual.

Example

```
 1Ø REM WINDOW EXAMPLE
 2Ø REM drawing random rectangles using BACKGND and WINDOW
 3Ø MODE:DEFAULT:BACKGND
 4Ø FOR COUNT = 1 TO 1ØØ
 5Ø X = 391*RND(1)
 6Ø X1 = (391-X)*RND(1)+X
 7Ø Y = 255*RND(1)
 8Ø Y1 = (255-Y)*RND(1)+Y
 9Ø WINDOW X,X1,Y,Y1
1ØØ BACKGND 1+7*RND(1)
```

```
    120 NEXT COUNT
```

Try replacing line 100 with;

```
     100 BACKGND 17*RND(1),17*RND(1),17*RND(1)
```

We have now covered most of the basic plotting commands and are so in a position to move on to more complex plotting commands.

DRAWING RECTANGLES AND TRIANGLES

Format

RECT Xlength,[Ylength,slant angle,rotate angle,fill,carriage]

ARECT X,Y,Xlength,[Ylength,slant angle,rotate angle,fill,carr]

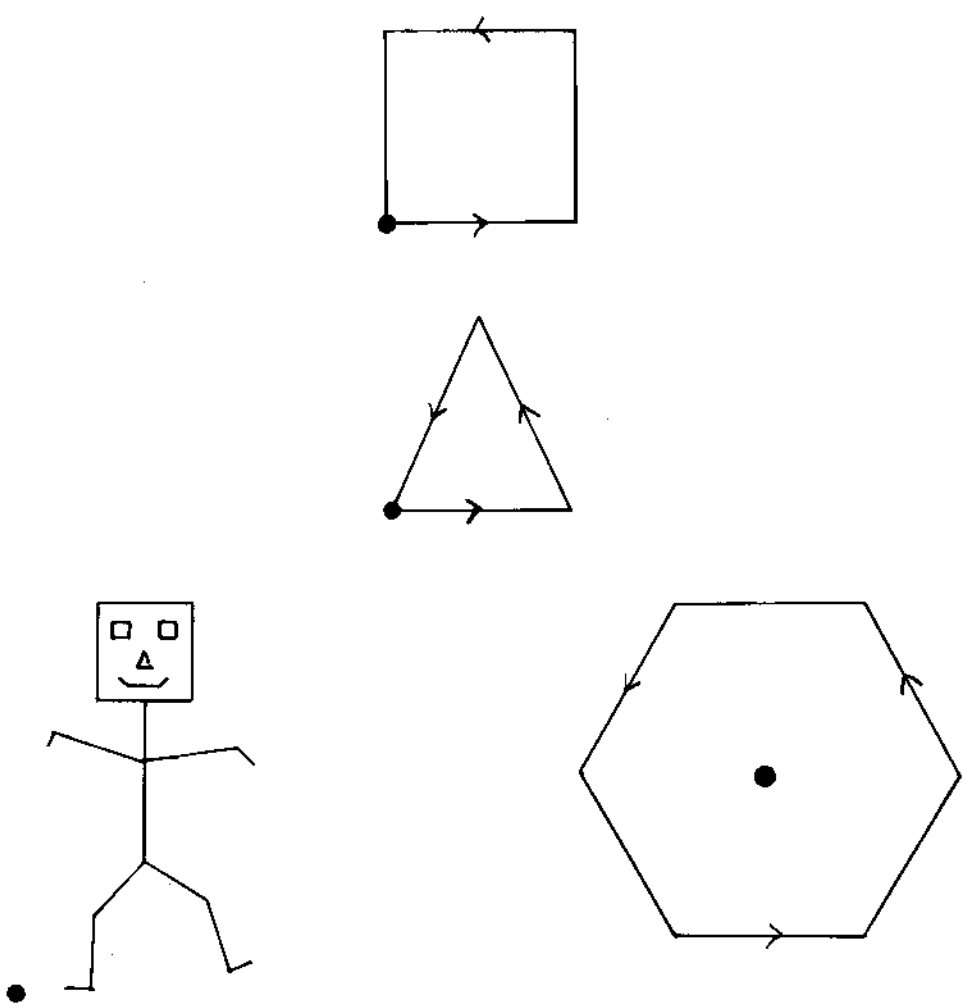
RECT draws a rectangle starting from the cursor position while ARECT draws a rectangle from the entered X,Y parameters. The rectangle is drawn in an anti-clockwise direction from the start point.
The Xlength parameter specifies the horizontal length of the rectangle (before any rotation is performed) and has a valid range of Ø to 255. Note this is an integer and not a floating point number so X lengths of 11.5 and 11.6 will give a true value of 11. The integer truncation performed on floating point numbers is actually a function of the pixel matrix rather than a true integer truncation.
If no Y length is specified then the X length entered is used for both and hence a square is drawn (assuming a slant angle of default or Ø radians).
The next two parameters slant angle and rotate angle are common to many of the graphics commands and require closer examination.

ROTATE ANGLE

This parameter specifies the angle through which the shape is to be rotated about it's centre of rotation and is entered as an angle. This angle is specified in radians to make it compatible with the trigometric functions within basic which use radians, to convert to and from degrees use the following;

1 DEGREE = 0.01745329251994 RADIAN or there abouts!

The centre of rotation is normally the cursor position, this remains constant throughout the rotation the centre of rotation being defined as the initial cursor position just before the shape is drawn ie.



All rotations are referenced to this origin point for the duration of that rotation only, a subsequent shape draw and rotate would rotate about a different point unless the two initial start points were identical. It should be noted that RECT and TRI and their A versions always leave the cursor position unchanged.
The LABEL command moves the centre of rotation to the start of each character but more about that later on.
The centre of rotation is as mentioned placed at the cursor position, however by use of the ORIGIN command this centre of rotation can be changed to be anywhere within the global plotting area. This ORIGINed centre of rotation is not moved by plotting shapes and irrespective of the cursor position remains at the selected position.

Format

```
ORIGIN [X,Y]
```

The required centre of rotation is defined by the X and Y parameters. To restore the centre of rotation back to it's standard mode (cursor following) the entered X,Y parameters should not be entered. So the default value of ORIGIN is to cursor following, the standard default after a cold start or a DEFAULT command.
The rotation angle is specified in radians and has a range of +/- PI radians (+/- 3.141592654 radians), with an accuracy of 0.001 radians. However it should be noted that positional accuracy is also affected by pixel spacing.
A negative angle rotates the shape clockwise and a positive angle anti-clockwise.

Example

```
10 REM ROTATION
20 MODE:DEFAULT:BACKGND
30 FOR ANGLE = -PI TO PI STEP PI/12
40 ARECT 195,127,190,190,0,ANGLE
50 NEXT ANGLE

10 REM ROTATION about a point other than the cursor
20 MODE:DEFAULT:BACKGND
30 ORIGIN 250,100
40 FOR ANGLE = -PI TO PI STEP PI/12
50 ARECT 195,127,40,40,0,ANGLE
60 NEXT ANGLE
```

NOTE When ORIGIN is used with rotated labels remember that the cursor is moved to the end of the character, to remove this feature set the character spacing to 0.

TUTORIAL
Using the RECT command, rotate three rectangles to form a three dimensional cube. Then rotate this cube.

SLANT ANGLE

The slant angle parameter effectively defines the 'slant' or 'shear' of the shape. This angle has the same range and accuracy as the rotate angle except that the zero angle position is at 12 o'clock rather than at 3 o'clock as with rotate. The easiest way to understand the slant function is to try examples.

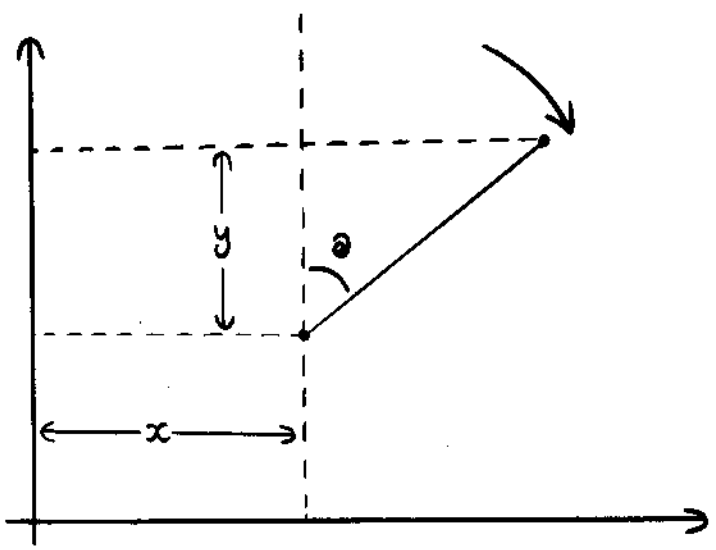
EXAMPLE

```
10 REM SLANT
20 MODE:DEFAULT:BACKGND
30 REM zero slant
40 ARECT 10,100,50,50,0
50 REM +30 degree slant
60 ARECT 120,100,50,50,PI/6
70 REM -30 degree slant
80 ARECT 230,100,50,50,-PI/6
```

The slant angle can be thought of as a hinge at each end of each line and the slant angle defines the hinge angle.
A more accurate definition is;

```
x = x + ySIN(@)

y = yCOS(@)
```



Note the slant function is performed before the rotation.

The remaining parameters are FILL and CARRIAGE.
FILL as might be expected fills the rectangle with colour, using the current COLOUR. The fill observes the WINDOW and PEN commands.

FILL vs. PAINT

There are two ways to colour an area. The first called FILL colours the area irrespective of what might be contained within that area. PAINT on the other hand acts as a 'flood' fill. Imagine that the shape to be filled is formed from small walls instead of lines, the colour is then poured into this 'mould' and floods the area. By it's nature paint requires that the shape be fully contiguous or enclosed otherwise the paint will leak out. The paint command would also not fill any 'island' within the 'mould'.
The AVC software does not include a paint command as the fill command is much more flexible, especially when used with the dominant plotting mode.

Example

```
 10 REM FILL
 20  MODE:DEFAULT:BACKGND
 30 FOR COUNT = 1 TO 100
 40 X = 392*RND(1)
 50 Y = 256*RND(1)
 60 COLOUR 8*RND(1)
 70 ARECT X,Y,50,50,0,0,1
 80 REM now put a black boundary to enhance shape
 90 COLOUR 0
100 ARECT X,Y,50
110 NEXT COUNT
```

It is a convenient point to start 'playing' with the various plotting modes, try the above example using XOR plotting. Remember that when using XOR not plotting the last point of a line improves the result.

TUTORIAL

Using the rectangle and it's fill write a basic program to display a bar chart (histogram), use the random number generator to provide the data.
Modify your program to produce a three dimensional histogram using the slant and rotate facilities.

There is a special case of RECT and ARECT which provides additional facilities;

If no parameters are entered for RECT or ARECT then a rectangle is drawn which conformes to the SOFT CLIP area as specified by the WINDOW or DEFAULT commands. This rectangle is filled with the current COLOUR and obeys the current plot mode.
Thus using just RECT or ARECT a facility similar to BACKGND is obtained except that the foreground COLOUR is used and the plot mode is not temporally changed to dominant. This produces striking effect if XOR plotting is used.

Example

After the last example has been run try typing (in direct mode);

```
PEN 1,1,1,2
COLOUR
RECT
```

Drawing triangles is similar to drawing rectangles;

Format

```
TRI baselength,[height,slant,rotate,carriage]
ATRI X,Y,baselength,[height,slant,rotate,carriage]
```

The parameters are exactly the same as RECT and ARECT except that the baselength specifies the length of the base of the triangle and the height specifies the height of the triangle.

TUTORIAL

Modify the previous examples to plot triangles instead of rectangles.
Fill the whole screen with triangles to form a diamond pattern, hint use rotate.

We have seen how two simple shapes can be drawn in various ways, TRIangles and RECTangles. Both these two shapes are POLYGONS, the AVC software allows for the general case of n sided polygons with the POLY command.

Format

```
POLY Xradius,[Yradius,sides,sides displayed,start angle
     ,rotate angle,fill,type,carriage]

APOLY X,Y,Xradius,[Yradius,sides,sides displayed,start angle
     ,rotate angle,fill,type,carriage]
```

The X and Y radii define the lengths of the polygon's X and Y radii, if no parameter is entered for the Y radius then it defaults to the X value. The radii have a range of Ø to 32767.
The centre of the polygon which is also used for it's centre of rotation (not affected by the ORIGIN command) is taken to be the current cursor position with POLY and the X and Y parameters with APOLY.
The polygon is drawn from the zero radian rotate position (3 o'clock) in an anti-clockwise direction.
The radii affect their respective axes before any other functions are applied such as rotate. The radii have a range of Ø to 32767 and as with the rect and tri commands are limited to integers by the pixel grid.
The sides parameter define the number of sides of the polygon;

```
sides=Ø,1,2 INVALID an OV error results
sides=3   triangle
sides=4   rectangle
sides=5   pentagon
sides=6   hexagon  etc.
```

The maximum number of sides is 255. The default value for sides and hence sides displayed is 6Ø, so the default polygon is a circle: well almost! The value of 6Ø gives the best compromise between speed of plotting and accuracy.

EXAMPLE

Try plotting different polygons varing the number of sides and the size (radii) to verify the above statement

```
1Ø MODE:DEFAULT:BACKGND
2Ø REM 'CIRCLE' COMMAND
3Ø REM plotting from the cursor, set to screen centre
4Ø PLOT 195,127,Ø,Ø
5Ø REM note carriage up so point is not visible
6Ø POLY 5Ø   :REM a circle of size 5Ø
7Ø STOP:BACKGND
8Ø APOLY 195,127,R?,R?,S?
```

To try various polygons remove the STOP command in line 7Ø and replace R? and S? with various sizes(radii) and sides. Note the

sides displayed is not entered as it defaults to sides

Note as the start position for plotting is at 3 o'clock (the X radius distance from the cursor) triangles and rectangles are rotated initially 90 degrees from the shapes produced by the rect and tri commands.
The sides displayed parameter as the name suggests defines the number of sides displayed, it's range is from 0 to the number of sides. Note if the number of sides displayed is not entered then the default value is to the number of sides, ie. a complete polygon is drawn. This of course assumes that a complete polygon as specified can be drawn within the hard/soft clip areas.
Plotting of a polygon unless redefined by the start angle always begins with the initial vertex at the zero radian position. The start angle can be moved over the whole circle by changing the start angle. This has a default value of 0 which is the same as an entry of 0 and produces no change in the start angle.
The start angle has a range of +/- PI radians with the +ve values causing the start point to move anti-clockwise and a -ve value to move clockwise.
The rotation parameter operates in the same way as for the rect and tri commands however it should be noticed that the centre of rotation is always the centre of the polygon which is the entered X,Y cooridinates or the cursor. After the polygon command has finished the cursor is NOT returned to this centre of rotation but at the end of the last side of the polygon which was plotted, if a cord or radii is specified this is performed last.
The order of the functions is

```
FIRST select number of sides
then  scale according to radii
      do start angle
      do rotation
      stop at number of sides displayed
      do cord or radii
      fill shape if enclosed
```

The next parameter is the fill flag, this has a default of 0 which is the normal non fill mode. If fill is set to a 1 then the polygon is filled with the current COLOUR using the current plot mode. The polygon specified must be a totally enclosed shape otherwise no fill will take place, this does not generate an error message. This requirement does not apply to non enclosed polygons formed by the hard/soft clip areas, but fill is not allowed to exceed these areas.
An explanation of the operation of the fill function is given in the section concerning the MAT command.

The penultimate parameter is the type of polygon and refers to the way in which non enclosed polygons are handled. Note this parameter has no effect if the polygon is enclosed ie. the number of sides is equal to the number of sides displayed.

Type values	Ø	normal section of a polygon
	1	polygon enclosed by the cord
	2	polygon enclosed by the radii

The final parameter is the normal carriage command.

EXAMPLE

a] 60 degree arc of a circle

```
APOLY 195,127,100,100,180,30
```

This has the same result

```
APOLY 195,127,100,100,6,1
```

except that only the cord is drawn

This arc can be rotated around the centre (195,127) by using either the start angle or the rotation angle, they have the same effect for a circle.

b] ellipse

```
APOLY 195,127,100,50
```

c] 'polyfiller'

```
APOLY 195,127,100,100,7,7,0,0,1
```

d] ' MUNCH-MEN ' !!

```
10 MODE:DEFAULT:BACKGND
20 FOR I=1 TO 10
21 COLOUR 1+7*RND(1)
30 PLOT 391*RND(1),255*RND(1),0,0
40 POLY 20,20,60,40,PI*RND(1),0,1,2
50 NEXT I
```

You will have noticed by now that the DEFAULT command has crept into many of the example programs, this is included to select the default options which may have been overwritten since the last cold start, but more of that later, for the time being include but ignore it !

TUTORIAL

Using the polygon and it's fill parameter write a basic program which asks for monthly expenditure and then draws a PIE chart showing the relative expenditures as the size of the portion of PIE. Use colours to high light areas or to give a sliding scale of costs.

Now modify the program to high light portions of the PIE by moving that portion slightly out of the PIE. Having written this program why not take a sneak preview of the label command and add labeling and percentage values to your 'graph plotting package'. When finished compare the result with a list of the input values, 'a picture is worth a thousand words'

You may well have thought that line plotting had been covered however there are two other line commands that should now be covered:-

VECTORS and SLINES

First the VECTOR command
Format

```
VECTOR length,[angle,mode,carriage]
AVECTOR X,Y,length,[angle,mode,carriage]
```

A VECTOR is a line with a difference, it has length and direction. The VECTOR command draws a line from the cursor position for a distance specified by the length parameter and at an angle from the cursor specified by the angle parameter.
The angle specified is again in radians and has a range of +/- PI, the zero radian position being at 3 o'clock. Anti-clockwise direction is selected by a +ve angle.
The AVECTOR command operates from the entered X,Y coordinates rather than from the cursor.

EXAMPLE

```
10 MODE:DEFAULT:BACKGND
20 FOR I=-PI TO PI STEP (2*PI)/12
30 AVECTOR 195,128,100,I
40 NEXT I
```

Try adding the following;

```
35 POLY 30
```

To achieve a reasonable balance between angle generation and accuracy a special alorithm was developed to generate the SINE and COSINE functions required by the angle, slant and rotate operations. These subroutines which are located within the AVC software package operate on 16 bit values, wheras the SINE and COSINE functions available from basic use 32 bit values. This reduced resolution does not adversely affect the slant, angle and rotate functions but if used to build up a complex shape the cumalative errors may combine to form visible errors. If this occurs then try building the shape so as to cancel errors (+ve and -ve rotations) or use the basic trig functions.

The internal cursor is stored as 16 bit values, with each bit corresponding to a pixel location, this representation always truncates any rotation function to the nearest pixel position. This would cause severe positional problems if say a circle is drawn consisting of many short lines. To overcome this problem the software saves the cursor position when doing rotations as 24 bit values. Normally this would not concern the user as the 16 bit values are restored after the command is finished.
The vector command is slightly different, as it will often be used as a 'TURTLE' command and hence requires to keep positional

accuracy from command to command. In essence the vector command saves separate 24 bit cursor values and compares the 16 bit values with the normal 16 bit cursor. If the two most significant bytes compare the vector command uses it's more accurate values, if not it uses the normal cursor value.

This feature may not always be required and so there is a requirement to be able to clear the extra accuracy, to do this either use the DEFAULT command or the vector command with no parameters.

The above is perhaps a bit heavy but will save headaches later on.

TURTLE was not a typing error, but is an 'americanism' for relative vectors.
The mode parameter of vector is different to that of line, a Ø produces a line from the cursor or the entered point at the specified angle but a 1 draws a line at an angle relative to the last vector angle. The analogy is that the programmer has control over a special turtle, which moves around the screen leaving a trail behind it (if the pen is selected). This turtle responds to directions in the form of relative vectors. You tell the turtle to move forward a certain distance (length), he will move in the same direction as he was previously going unless you tell him to deviate (angle).
After a cold start, default or a vector with no parameters the direction is taken to be zero radians, ie. to the right.
The remaining parameter for vector is the standard carriage control.

As stated before there are certain accuracy considerations to be made when using vector angles, if a polygon is required then greater positional accuracy will result from using the poly command rather than the vector command.

TUTORIAL

Using the relative vector command draw a series of polygons.

SCALED LINES: SLINES

The SLINE command is a similar command to the line command except that the lines produced can be magnified in the X and Y directions and rotated and slanted, hence SLINE, Scaled LINEs.

FORMAT

```
SLINE X,Y,[mode,Xmag,Ymag,slant,rotate,carriage]
ASLINE X,Y,X1,Y1,[mode,Xmag,Ymag,slant,rotate,carriage]
```

Again the A version draws from the entered point (X,Y) rather than from the cursor.
With SLINE, X and Y define the end point of the line (the start point being the cursor) which is absolute if the mode is Ø and

relative if the mode is 1. With ASLINE X and Y define the line start while X1,Y1 define the line end. The end of the line as so defined only applies BEFORE ANY OTHER FUNCTION IS APPLIED.

The next two parameters specify the line magnifactions (or reduction), the range being from Ø to 255. Xmag specifies the horizontal while Ymag specifies the vertical scaling.

The SLANT and ROTATE parameters operate in exactly the same way as previously defined (see RECT and TRI), note the centre of rotation is the start of the line unless redefined by an origin command.

The remaining parameter is the standard carriage control.

The next command covered allows text (characters) to be displayed.

The LABEL command

FORMAT

```
LABEL string,[Xmag,Ymag,slant,rotate,char spacing,carriage]
ALABEL X,Y,string,[Xmag,Ymag,slant,rotate,char spacing,carr]
```

Guess what the difference between ALABEL and LABEL is, correct!

The LABEL command is similar to the basic's print command in that it prints text strings. However the LABEL command used by the ROM BASIC version has certain limitations (the EXTENDED BASIC VERSION is much more powerful). In both versions the print formating controls do not work, eg. , ; TAB etc.
The label command only prints a string literal ("literal") or a string variable (MESS$), and does not allow string arithmetic. The EXTENDED version does allow more flexibility, see the appropriate appendix.
The label command unlike the print command does not respond to control codes, for instance it will not respond to a carriage return or perform any scrolling.
It would seem that the label command is very restricted however it is much more flexible than the print command as will be shown. For those users who require a print type facility be be mixed with graphics the split screen and graphics overlay facilities available under NASCOM'S CP/M provide a highly advanced capability.

The label command is not restricted to printing characters on row and column boundaries, it can start printing a text string at ANY pixel position. This is the cursor position with the LABEL command or the entered position (X,Y) with the ALABEL command. It should be noted that the start position is defined as the bottom left hand position of the first character of the string, and that this applies BEFORE any other function is applied.

As stated before the text string can be a lateral or a variable;

EXAMPLE

```
10 REM Text printing
20 MODE:DEFAULT:BACKGND
30 REM String literal
40 ALABEL 30,100,"HELLO EVERYONE"
50 REM String variable
60 LET A$=" I'M "+"A LABEL"
70 LABEL A$
```

The AVC software comes with a precoded character table which supports ASCII characters from 020H to 05FH inclusive, other user defined character tables can be added, see the assembler interface section.

Standard AVC character set (ASCII values in HEX)

ASCII	character	ASCII	character
20	"space"	40	@
21	!	41	A
22	"	42	B
23	#	43	C
24	$	44	D
25	%	45	E
26	&	46	F
27	'	47	G
28	(	48	H
29	)	49	I
2A	*	4A	J
2B	+	4B	K
2C	,	4C	L
2D	-	4D	M
2E	.	4E	N
2F	/	4F	O
30	0	50	P
31	1	51	Q
32	2	52	R
33	3	53	S
34	4	54	T
35	5	55	U
36	6	56	V
37	7	57	W
38	8	58	X
39	9	59	Y
3A	:	5A	Z
3B	;	5B	[
3C	<	5C	\
3D	=	5D	]
3E	>	5E	up arrow
3F	?	5F	left arrow

Two character sets are allowed for; standard and graphics, both can be redefined by the user, the graphics set is defined as a null table in the standard AVC software. Any character which is defined as a null (ie. not in the table) will NOT cause an error but will neither cause the cursor to move, ie. they are ignored.

The next two parameters define the horizontal and vertical magnifications, with a range of 0 to 255, 1 being the default values providing true scaling. The parameters are Xmag and Ymag, if only Xmag is entered then Ymag defaults to the Xmag value. It should be noted that unless redefined that when operating in AVC MODE 1 there are 56 characters per row.

MODE 1: 392 pixels / pixel spacing (7) = 56

MODE 2/3 : 784 / 7 = 112 characters per row.

These two default modes are the equivalent to the CP/M 40 and 80 column screens but CP/M uses character spacing of 8.

Try the previous example with mode 2/3 selected.

The next parameter defines the slant angle of the characters. This should now be a familar function, in fact it originated in the label command. A zero (or default) slant angle produces upright text, +ve angles and the text slopes to the right, -ve and the text slopes to the left. Note the line of writing is still horizontal unless redefined by the next parameter.

The rotate parameter causes the line of text to be printed along a vector. In effect each character is rotated around it's initial cursor position (bottom left of character) unless redefined by the origin command.

EXAMPLE

```
  5 MODE:DEFAULT:BACKGND
 10 REM big text
 20  A$="HELLO"
 30  ALABEL 40,70,A$,4    :GOSUB 500
 40 REM flat text
 50  ALABEL 40,70,A$,4,2 :GOSUB 500
 60 REM tall text
 70  ALABEL 40,70,A$,2,4 :GOSUB 500
 80 REM script
 90  ALABEL 40,70,A$,4,4,-.6   :GOSUB 500
100 REM script at 45 degrees
110  ALABEL 40,70,A$,4,4,.6,PI/6
120  END
500  FOR I=1 TO 1000
510  NEXT I
520  BACKGND
530  RETURN
```

Try replacing A$ with "HELLO i am lower case OK"

TUTORIAL

Write a program to emulate a terminal, but convert scrolling to page mode operation. A rather nice feature for those using NASCOM EXTENDED basic is to use the SCRN$ command to list the program it'self onto the AVC.
Now convert the program to provide 98 lines of 36 characters, hint: turn the display screen on it's side.
Using the information given in the assembler section concerning adding character sets to add lower case to the standard set. A program allowing the user to define character sets using the cursor keys is a possible extension.

The next parameter allows the character spacing to be changed for the duration of that command only. The character spacing has a default of 7 using the standard AVC character set, the default

value is stored as the first character in the table. The character spacing is actually defined as the pixel spacing (BEFORE magnifaction) between the start positions of adjacent characters. The character spacing has a range of Ø to 255.
The remaining parameter is the carriage command.

The shapes so far used are all variations of polygons, the next command allows other shapes to be moved, scaled and coloured as required.

The MATrix command.

FORMAT

```
MAT address,[ Xmag,Ymag,slant,rotate,fill,carriage]
AMAT X,Y,address,[ Xmag,Ymag,slant,rotate,fill,carriage]
```

Again the A version allows plotting from an X,Y position rather than from the cursor.

The MAT command allows shapes to be defined as a series of lines which are coded into a matrix for use by MAT.

Before the explanation of the MAT command an example.

EXAMPLE

```
  5 REM Transistor
 10 MODE:DEFAULT:BACKGND
 20 DATA 0,40,0
 30 DATA 30,40,7
 40 DATA 30,70,23
 50 DATA 40,70,23
 60 DATA 40,10,23
 70 DATA 30,10,23
 80 DATA 30,40,7
 90 DATA 40,50,0
100 DATA 70,60,7
110 DATA 70,80,7
120 DATA 70,0,0
130 DATA 70,20,7
140 DATA 40,30,7
150 DATA 56,34,0
160 DATA 62,23,7
170 DATA 51,19,7
180 M=9*4096 :REM for rom basic use 2'S complement
190 DOKE M,16
200 M=M+2
210 FOR I= 1 TO 16
220 READ X,Y,P
230 DOKE M,X
240 M=M+2
250 DOKE M,Y
260 M=M+2
270 POKE M,P
280 M=M+1
290 NEXT I
300 AMAT 60,0,9*4096,2
```

The MAT command requires that the lines forming the required shape be entered in the following format;

Starting at the beginning of the array.

BYTE number Description

1 ..... MSB of number of lines
2 ..... LSB of number of lines
3 ..... MSB of X coor.
4 ..... LSB of X coor.
5 ..... MSB of Y coor.
6 ..... LSB of Y coor.
7 ..... Control byte

The sequence of bytes 3-7 is then repeated for the number of lines as defined by bytes 1-2.
The X and Y coordinates are entered in the same way as for normal lines, except that lines are always drawn from the cursor, relative line plotting is not allowed.
The first two bytes in the array as stated before define the number of lines to be used starting from the first, this would usually be the total number of lines.

The control byte is used to provide extra facilities for each line, it's control is via individual bits;

CONTROL BYTE BIT ALLOCATION

BIT 0 (LSB) RED INK
BIT 1 GREEN INK
BIT 2 BLUE INK
BIT 3 Next three bytes form a colour change, R,G,B ints
BIT 4 FILL FLAG
BIT 5 - BIT 7 Reserved for future use.

The first three bits define the three primary colour 'INKS', these are in effect secondary pen commands. If on entry to the MAT command the colour is white (7 or 16,16,16) and the pens are all down then the inks will act as pen commands allowing coloured lines to be drawn. For instance if the red and blue ink bits are set to 0 and the green bit set to 1 then the colour of that line would be green. However it should be noted that with a colour of white setting all the inks to 0 will NOT cause a black line, to achieve this an entry colour of black should be used. The PEN command overrides the ink controls. Using the ink control with an entry colour of white only the primary and secondary colours can be controlled, or the primary components of extended colours.

The next bit of the control word allows the current colour to be redefined and causes this change to be permanent. If this bit is set to 1 than the software expects the next three bytes to be the new values of the colour. No defaults are allowed and the format is;

First byte RED intensity (0-16)

next byte GREEN intensity (Ø-16)
third byte BLUE intensity (Ø-16)

After the three bytes the normal sequence of X,Y,control must continue, unless at the end of the array.
The next and final bit used in this version of the software is the fill flag. When set to a 1 the lines are stored (as well as displayed) and when a control byte is next encountered that has this bit set to Ø the lines drawn while the bit was 1 are filled in the current colour.
The use of the fill flag outside of the rect,tri or poly commands requires some explanation this is dealt with under the FILL command which is covered next.

Before the actual MAT command is discussed the method of generating an array for use by MAT is required.

The array as specified before is a sequence of bytes, these bytes are NOT a basic array but a table formed in memory. Note using NASCOM EXTENDED basic the MAT array could be an .OBJ FILE.
The previous example shows one method of generating an array using the poke and doke functions from basic. It is a simple shape which shows the diagramatic representation of a transistor. Lines 2Ø to 17Ø are the data statements which hold the lines.

[2Ø] Moves the cursor to the start position of X=Ø, Y=4Ø with the carriage up, the inks all off.
[3Ø] Draws a line from Ø,4Ø to 3Ø,4Ø with all inks on, colour white as a default has been performed.
[4Ø-7Ø] Draws the main block of the transistor and when drawn fills it.

Note when entered from basic the values for the various bits of the control word are in DECIMAL.

[8Ø] Causes the fill to be performed, the fill flag set to Ø.

[1ØØ-17Ø] Draws the remainder of the transistor.

The program lines from 18Ø-29Ø take the data statements and put them into memory using the correct MAT format.

[18Ø] Set up where the MAT array will reside, in this case at Ø9ØØØH (hex).

Note the array can be located anywhere there is valid user ram including the ram which is paged by the AVC, however don't overwrite the AVC software.
[19Ø] The size of the array, or the initial number of lines to use. Note a 16 bit number (DOKE) allowing an array of 64,536 lines !
[21Ø] A loop which writes the data into the memory array. In this case 16 lines.

[2ØØ,24Ø,26Ø,28Ø] Memory pointer incrementer

[230] Put in X values

[250] Put in Y values

[270] Put in control words

[300] display MAT array

The first parameter of MAT (third of AMAT) is the address of the array. This is a 16 bit value internally but may be a floating point basic number. REMEMBER that NASCOM ROM basic requires that this be a DECIMAL number and that if an address is required to be within the range 08000H to 0FFFFH then the 2's complement form is required (number-65536). Because the start address of the array can be changed there can be more than one array present in memory at one time.
The next three parameters slant, rotate and carriage are exactly as previously defined.

TUTORIAL

Modify the example to perform a colour change during the array, note the program area which converts the data statements into an array will have to be modified to account for the extra three colour change bytes.
Having played about with the example a rather ambitious project would be to write a program which would allow a designer to lay out a circuit diagram using MAT arrays for the components.
Write a program to allow creation of MAT arrays using the cursor keys to define lines, and other keys to define colour etc. A useful method is to allow 'rubber band' lines, this is where the cursor keys are used to define the start position of the line (hit a key to freeze) as the cursor is moved a line is drawn and un-drawn (XOR) from the frozen point when can then be frozen and the sequence again.

HARD COPY PRINTOUT

The AVC does not have as standard a HARD COPY facility however it does allow the user to add a hard copy function that is configured for a graphics printer.

Format

DUMP [planes]

The dump command accepts a variable (planes) in the range Ø to 7, the default value being 7. This parameter would normally be the primary colour planes that require printing (assuming a monochrome printer). A value of 1 would only print the red plane, while 7 prints all planes, it is suggested that when printing more than one plane the bits of the planes are ORed together.
The planes parameter is returned in the AVC software register called POLTYP, for details of where this is refer to the AVC software register appendix.
In the standard AVC software the dump command is routed via a RAM vector back to the valid AVC return, the user's program start address (execution) should be placed in this RAM vector (see AVC software register appendix).
Reference should be made to your printer manual and to the AVC hardware manual but the following may help;

Most printers require a control code to be sent to turn on and off the graphics facility. Then a string of bytes to be sent that define the bit patterns of the next 7 rows (assuming a 7 wire head). Therefore as the AVC memory is organised as rows of 64 bytes defining a graphics row and the printer requires a byte to be a vertical column of 7 (not 8) pixels, a conversion program will have to be written. It is suggested that assembler is used to speed up the process. The aspect ratio of most printers is about Ø.7 so that when using MODE 1 many of the horizontal pixels will have to be repeated to give a correct aspect ratio. Note when using MODEs 2 or 3 the horizontal bytes are in a red, green, red, green etc. sequence.
The remaining problem is that due to heat problems many printers (NASCOM IMP) cannot print adjacent pixels, the control program will have to take this into account.

For details of the memory map of the colours planes refer to the end of APPENDIX D.

As described previously the fill function is available for many commands, there might however be a need to fill a shape drawn from standard lines, this requirement is fulfilled by the fill command!

Format

FILL [mode]

All lines drawn by the LINE,VECTOR and SLINE commands can be filled. When the mode is set to a 1 this sets the fill flag and the positions of the lines drawn from then on are stored. Now when the mode is changed to Ø (default value) the shape previously drawn while the fill mode was 1 is filled. Things are not quite that simple as an explanation of the fill operation will show.

The fill command orginated in the POLY command and was designed to fill all the possible shapes, however it is limited to only FOUR direction changes per scan row;

The fill flag indicates to the line plotting program to save the individual coordinates of each line. In essence only the coordinates of the line which CROSS a scan line are saved, thus a horizontal line would only have one cross over point saved while a vertical line would have all. The coordinates are saved in the right hand margin of the blue plane, with the X value stored at the current Y address. This method saves storing both X and Y. The space available in the margin allows for four X values which is why only four crossovers per scan line are allowed. This allows for the worst case of poly, a part poly with radii.
This limitation must be bourne in mind when using FILL and the fill function with MAT. If complex fills are required then the best approach is to break up the shape into shapes which obey the four crossover rule.
Remember that the fill function will obey the colour and plot mode commands but dot/dash commands are ignored.

The next three commands covered are for disk users only, however the latter two may be of interest to cassette users and would form a good software project.

The DOS command

This command allows NAS-DOS users to exit gracefully from the graphics basic. The NAS-DOS reset vector is directed to the AVC software WARM start address, this has the effect of restoring the AVC ram vectors and performing a default. It also sets the AVC to external video mode but unlike a cold start does NOT clear the screen. For details of the options for COLD and WARM starts see the DEFAULT command.
The DOS command restores the original NAS-DOS reset vector and returns control to NAS-DOS via it's warm start. It is suggested that this command is always used when finialy leaving the graphics basic. Note NASCOM EXTENDED basic does not require this command. If for any reason the reset vector is required to be returned to the default value and the AVC software is not available then using the memory command change location ØDØØH to ØØ and then hit reset.

STORING GRAPHICS PICTURES

The AVC software allows disk users to store complete pictures on disk. There is no reason why using the ØB2H port and the NAS-DOS]O command that the three primary colour planes cannot be stored on disk. However as each plane is 16K bytes a total of 192 sectors are required per picture !
The GLOAD and GSAVE commands use a special compression algorithm to reduce this storage, massive disk storage space saving can be made allowing many more pictures to be stored on a disk.

Format
Loading pictures from disk

```
GLOAD string,[ drive ]
```

Saving pictures on disk

```
GSAVE string,[ drive ]
```

The string defines the disk file name and has the following restrictions;

A] The file name may be from 1 to 8 ASCII characters
B] The following ASCII characters are NOT allowed;
ASCII codes > 127
. , < > ; : = ? *
C] GLOAD files MUST exist
D] GSAVE files MUST NOT exist
E] The file name must be a string literal or variable only

The optional drive parameter (default of drive Ø) has a range of Ø to 7 allowing 8 logical (4 physical) drives.

EXAMPLE

Using the example picture stored on the 'demo' disk

```
10 MODE:BACKGND:DEFAULT
20 REM assume demo disk in drive 0
30 GLOAD "RIPPLE2",0
```

The format for NASCOM EXTENDED basic users is similar except that there is no drive parameter, the drive being taken as the currently logged drive (DRIVE command). A file can be saved or loaded from a drive other then the logged drive by inserting the drive name at the start of the file name;

```
GSAVE "D:TEST"
```

This saves the current AVC picture in a file called 'TEST' on drive D.

TUTORIAL

Write a program which emulates the GLOAD/GSAVE commands but "packs" and "unpacks" the pictures into memory before loading or saving on disk. The standard GLOAD/GSAVE commands use the same 256K byte buffer to act both as the pack/unpack buffer and the NAS-DOS I/O buffer. This approach uses the minimum memory but is slow due to the constant disk access.
This program could then be extended to provide a 'slide show' facility. (Cassette users should also try this tutorial).
HINT: The GSAVE program scans the three AVC primary colour planes in columns, 256 bytes at a time. If a byte is not repeated then it is saved in the buffer, if repeated then an 0FEH byte is included, then the byte, then the number of repeats. Note single 0FEH bytes are 'compressed' as 0FEH,byte,01. Beware NAS-DOS takes 0FFH and 0DH as special characters! Loading from disk is the reverse. One possible modification is to XOR the saved bytes with the existing screen bytes.

THE DEFAULT COMMAND

This command defaults various AVC system variables, the following lists the variables and their default values. For details of the variables concerned refer to the AVC software register section.

All decimal values.
CRTC REGISTERS

HORIZONTAL TOTAL (-1)	=	129
HORIZONTAL DISPLAYED	=	128
HORIZONTAL SYNC POSITION	=	110
HORIZONTAL SYNC WIDTH	=	9
VERTICAL TOTAL (-1)	=	76
VERTICAL TOTAL ADJUST	=	3
VERTICAL DISPLAYED	=	64
VERTICAL SYNC POSITION	=	68
INTERLACE MODE	=	0
MAX SCAN LINE ADDRESS	=	3
CURSOR START	=	0
CURSOR END	=	0
START ADDRESS HIGH	=	0
START ADDRESS LOW	=	0

Other variables

WIBFL = AVC stack position -582. NAS-DOS input buffer
WOBFL = WIBFL+256. NAS-DOS output buffer
BASTOP = start address of AVC software-1. This is the basic's top of memory and is only changed if a size clash occurs. Note BASTOP is not required in the extended basic.
COLUMN = 0 ; Fill column counter
FILL = 0 ; Fill off
RBACK = 0 ; BACKGND set to black
GBACK = 0
BBACK = 0
ORGFLG = 0 ; set origin to cursor following
X = 0 ; set cursor to 0,0. Bottom left hand of screen.
Y = 0
VECTOR ANGLE & POSITION = 0
DOT/DASH = SOLID
PLOT MODE = REPLACE (0)

RINT = GINT = BINT = 16 ; set current plotting colour to white.

LAST PIXEL = to be plotted
PENFLAG = all pens down

ASCPTR = standard character table
ASCCHR = standard character spacing from table (7)
WXMIN = 0 ;window X min to 0
WXMAX = 391 ; window X max set to 391 for single density and to 783 for double density.

```
WYMIN     =   Ø ; window Y min
WYMAX     =   255 ; window Y max
```

Apart from the appendices this is the end of the manual.

If you have any comments (good or bad) about this manual please contact NASCOM. Also we are always interested in modifications to the software and application/games programs.

Now the commerical break;

AVC HARDWARE DESIGN : ANDREW BROWN
PETER HORTON

AVC COLOUR SOFTWARE : ANDREW BROWN
CHRIS HARVEY
PETER HORTON

CP/M SOFTWARE : IAN CLOUGH (aided by DIGITAL RESEARCH)

AVC DOCUMENTATION : PETER HORTON

CP/M DOCUMENTATION : IAN CLOUGH

PROJECT MANAGEMENT : MIKE HESSEY

CONTACTS

PRODUCTION, DISTRIBUTION and SERVICE

LUCAS LOGIC at WARWICK

SALES and TECHNICAL

LUCAS LOGIC at KENILWORTH

LUCAS LOGIC
3 WARWICK HOUSE
STATION ROAD
KENILWORTH
WARCKS.

APPENDIX A

ADVANCED VIDEO CONTROLLER REGISTERS

The registers are described as in the standard configuration.

PORT ØBØH : CRTC address register
PORT ØB1H : CRTC data register
PORT ØB2H : AVC control register

The register of the CRTC that is required is obtained by placing the register code in the CRTC address register, the register is then available from the CRTC data register.

address register	data register
Ø	HORIZONTAL TOTAL -1
1	HORIZONTAL DISPLAYED
2	HORIZONTAL SYNC POSITION
3	HORIZONTAL SYNC WIDTH
4	VERTICAL TOTAL -1
5	VERTICAL TOTAL ADJUST
6	VERTICAL DISPLAYED
7	VERTICAL SYNC POSITION
8	INTERLACE MODE
9	MAX SCAN LINE ADDRESS
1Ø	CURSOR START
11	CURSOR END
12	START ADDRESS HIGH
13	START ADDRESS LOW
14	CURSOR HIGH
15	CURSOR LOW
16	LIGHT PEN HIGH
17	LIGHT PEN LOW

AVC CONTROL PORT pin allocations

PORT ØB2H: High values shown, low values in brackets

BIT Ø : red plane page select into memory [paged out]
BIT 1 : green plane page select into memory [paged out]
BIT 2 : blue plane page select into memory [paged out]
BIT 3 : double density [single density]
BIT 4 : red plane output to video [no red output]
BIT 5 : green plane output to video [no green output]
BIT 6 : blue plane output to video [no blue output]
BIT 7 : external video selected [AVC video selected]

NOTE. On power up the AVC control port can be configured to any state, therefore ALWAYS after switch on type, O B2 8Ø , or alternatively use NAS-SYS:AVC.

APPENDIX B

COLD - WARM STARTING

There are two versions of the AVC colour graphics software, G48 and G32. They relate to 48K byte and 32K byte NASCOM systems. For the G32 package change all ? to C and for the G32 package change all ? to 8.

At the top of the software there are a few jumps which the user should know about;

COLD START JUMPS

To cold start the AVC software first of all enter basic using the J command. The software will reduce the size of basic to make room for it'self, but if a smaller size of basic is required then enter this when asked for by the basic. The software will not increase the size of the basic's program area.
The AVC colour graphics software should then be executed at one of it's cold start addresses, control will then be passed either to NAS-DOS, NAS-SYS or the the calling program. The basic and the colour graphics software are now linked, to re-enter the basic perform a basic WARM start using the Z command.
Programs which have been created using the linked basic on disk systems need not be cold started, however the AVC software must be present.
If using a different version (or size) of the colour graphics software with disks then load down the basic program and then cold start the AVC software. After this has been performed resave the basic program according to the instructions given in the NAS-DOS manual for resaving basic programs created under different system sizes.
Cassette user's should note that a 256 byte disk I/O buffer is included within the AVC software and is available for use as general program area or for MAT arrays. The address of the start of this buffer is given the AVC software register appendix.

Ø?FFDH cold NAS-DOS. This is the normal cold start address for NAS-DOS users, it performs correct initialisation of the software and links it to NASCOM ROM basic, control is then passed to NAS-DOS.

Ø?FFAH cold NAS-SYS. This is the normal cold start for NAS-SYS (only) user's, control is returned to NAS-SYS.

Ø?FF7H cold RET. This performes a cold start but returns via a ØC9H, ie. it is a subroutine.

Ø?FF4H Assembler calling point, see ASSEMBLER INTERFACE appendix.

Ø?FF2H This and address Ø?FF3H contain the start address of the AVC colour graphics variables. NOTE this is the highest address.

As stated before the cold start does an automatic default, it also does the following;

NAS-SYS cold start :Sets up the ram jump vectors for;

FILL, DUMP, PLOT and LINE

Checks that the AVC is present in the system. A 'ERROR: no AVC in system' message results if the test failed.

Performs a default

The AVC is selected for external video with no planes paged in.

The vector at location ØFF40H and ØFF41H is loaded into the variable USRVCT. This is the normal ROM basic SET command ram vector.

The screen is cleared.

NAS-DOS : Does the same as NAS-SYS except that it also sets up the RESET vector (ØDØØH to ØDØ2H) to point to the warm start address. And sets up the NAS-DOS disk I/O buffers.

The cold start then 'falls' into the warm start routine which performs the following;

Reselects external video

Re-defaults

Gets the SET vector from USRVCT and places it in SETWDG. Then puts the AVC SET interface vector in SETREF.

The logo is printed

The top of basic is checked and then the amount of ram available to basic printed.

Control is then passed to the calling program, NAS-SYS or NAS-DOS.

WARM STARTS

The warm start is exactly the same as the cold start except that the screen is NOT cleared.
NAS-DOS user's warm start is the RESET KEY.
For NAS-SYS users's the address of the warm start may be found by the following method;

Execute the NAS-DOS cold start (Ø?FFDH), the warm start address is then placed in memory locations ØDØ1H and ØDØ2H.

MEMORY MAP

The AVC software is approximately 7.6K bytes long with a Ø.7K bytes workspace, and resides at the top of user ram downwards.

APPENDIX C

ASSEMBLER INTERFACE

The assembler interface is the main control program of the AVC software. Programs can be written which control the AVC directly, ie. not using the NASCOM ROM BASIC interface. In fact the NASCOM ROM basic interface does just this. As the ROM basic interface will probably no longer be required this can be deleted allowing extra space for programs. This deletion facility can only be performed with the G48 version as due to the positioning of the AVC memory planes the module order with the G32 version forced the basic interface to be included midway in the package. However the space can still be used.

ROM basic interface deletion
(There is of course no need to delete this if the extra space is not required)

First the AVC software module map.

For G48 versions;

```
AVCBAS.OBJ : Rom basic interface
AVCPOL.OBJ : Polygon,vector and text generation
AVCCHR.OBJ : Character table
AVCWRK.OBJ : Workspace and disk buffers
AVCDSC.OBJ : Disk picture load,save
AVCASM.OBJ : Lines,points,colour etc.
AVCJMP.OBJ : Jump table. Cold starts
```

For G32 versions;

```
AVCASM.OBJ
AVCDSC.OBJ
AVCWRK.OBJ
AVCBAS.OBJ
AVCPOL.OBJ
AVCCHR.OBJ
AVCJMP.OBJ
```

The modules AVCWRK.OBJ, AVCASM.OBJ and AVCDSC.OBJ cannot be resident underneath the AVC memory planes, this explains the difference. Neither can any other software which DIRECTLY accesses the AVC planes, it's also best to move the stack out of the way too, unless pushes, calls etc. are not used! The AVC software maintains it's own stack (70 bytes) in AVCWRK.OBJ.

START ADDRESSES

The start address of the first module is the start address of the AVC software.
The start address of AVCBAS.OBJ is contained in SETREF
The start address of AVCASM.OBJ is contained in the jump table
The start address of AVCPOL.OBJ is contained in the workspace

The END address of AVCWRK.OBJ is contained in the jump table

To access a graphics command from assembler the appropriate AVC software registers should be set up and processor register A loaded with the command number. A CALL is then made to the AVCALL address in the jump table.

BEWARE. The module AVCBAS.OBJ is reasonably intelligent and performs limit checking on the variables, this is not available when using AVCALL. The only major difference is that relative plotting with the PLOT command must be handled separately, ie. add the values together before calling.

EVERY REGISTER (except the interrupt & refresh) IS CORRUPTED

The command codes for register A are;

register	command
Ø	BACKGND
1	PLOT
2	LINE
3	CHECK
4	LABEL
5	POLY
6	VECTOR
7	DEFAULT
8	ORIGIN
9	MAT
A	RECT
B	TRI
C	SLINE
D	DUMP
E	GLOAD
F	GSAVE
1Ø	FILL
11	DEFRECT Special case of rect with no parameters

Before starting to use the assembler interface refer to the AVC software register appendix. It is advisable to make a copy of the AVC software before modifying it.

APPENDIX D

AVC SOFTWARE REGISTERS

The registers described all start at AR and proceed DOWNWARDS in memory, the address of AR is in the jump table (top of workspace).

All the intensities have a range of Ø to 16 where Ø is zero intensity and 16 is full intensity.

AR	RINT	Red intensity foreground colour
AR-1	GINT	Green intensity foreground colour
AR-2	BINT	Blue intensity foreground colour

The LINON and LINOFF variables has a range of Ø to 255.

AR-3	LINON	Line on number of pixels
AR-4	LINOFF	Line off number of pixels

Note if the LINOFF is set to Ø then the LINON is ignored and a solid line is drawn.

The PENSTR has the same format as PENFLG except that AVCBAS.OBJ uses the PENSTR as a temporary storage location for PENFLG when the carriage is up. PENFLG is the variable which controls the pen.

AR-5	PENSTR	Pen temp store

The X, Y, X1, and Y1 variables are used for line and point plotting and have a range of Ø to 65536 (or -32768 to 32767)

AR-7	X	Initial X value (cursor)
AR-9	Y	Initial Y value (cursor)
AR-11	X1	Final X value
AR-13	Y1	Final Y value

SIDES contains the number of sides to the polygon and has a range of 3 to 255. SDIS contains the number of sides to display of the polygon and has a range of Ø to SIDES.

AR-14	SIDES	Number of polygon sides
AR-15	SDIS	Number of polygon sides displayed

The X and Y radii are used for POLY, RECT, TRI and MAT. They have a range of Ø to 65536 (or -32768 to 32767), the value is effectively multiplied by 256 so the number is actually 255.255. A radius (or magnification of one would take the value of Ø100H).

AR-17	RX	X radius / magnification
AR-19	RY	Y radius / magnification

The start and rotate angles have a 16 bit range (-32768 to 32767) but are multiplied by 10,ØØØ ie. a value of PI would be taken as 3.141592654 * 10,ØØØ = 31,416. The negative angles are the two's

complement of the positive angles.

AR-21	STANG	Start angle or slant angle
AR-23	ROTANG	Rotate angle

The fill flag is set to 1 to save the line coordinates. A value of Ø does not save the coordinates.

AR-24	FILL	Fill flag

The next two variables are used for the label and Gload/save commands. The start address of the character string is held in STRPTR and it's length in STRLEN.

AR-26	STRPTR	String pointer
AR-27	STRLEN	String length

The pen flag hold the status of the three primary colour pens. Each pen is controlled by a bit, the pen is down if that bit is set to a one. The red, green and blue pens are controlled by bits Ø, 1 and 2 respectively.

AR-28	PENFLG	Pen status

The conprt variable holds the current status of the AVC control register except that all the planes are paged out. This variable is ORed with the required colour plane(s) and then sent to PORT ØB2H.

AR-29	CONPRT	AVC control register image

The LPMODE variable is used to indicate to the LINE, VECTOR, and SLINE commands that either absolute or relative plotting is required. A Ø indicates absolute, a 1 indicates relative. The absolute/relative mode for plot is performed in AVCBAS.OBJ and NOT AVCASM.OBJ. Note this variable is used to indicate to the software which mode of operation the AVC is in.

AR-3Ø	LPMODE	Absolute / relative mode

The next variable is used to store the ROM basic's text pointer which is normally returned to basic via the HL register.

AR-32	TXTPTR	Rom basic's text pointer

The next variable is the ROM basic's stack pointer. Note this is saved on entry to the AVC software and a stack is set up within the AVC workspace, on leaving the AVC software the stack is restored.

AR-34	STKPTR	Rom basic's stack position

The next three variables are the red, green and blue intensities for the background colour, they have a range from Ø to 16.

AR-35	RBACK	Red backgnd intensity
AR-36	GBACK	Green backgnd intensity
AR-37	BBACK	Blue backgnd intensity

The next variable hold the address of the start of the non-graphics character table.

AR-39	ASCVCT	Ascii character table (non-graphics)

The check command uses the next variable to return the colour of a pixel. Note the coordinates of that pixel should be supplied in X1 and Y1 and NOT X and Y as this would corrupt the cursor. The condition of a check on a pixel outside the hard clip area would return a value of 255 to basic, however the value in COLOR would be -1 (255).

AR-40	COLOR	Colour of a pixel

The start of the graphics character table is held in GPHVCT.

AR-42	GPHVCT	Graphics character table

The next variable is used to indicate to the poly command what type of polygon to generate. A 0 gives normal, a 1 gives a cord and a 2 gives the radii. This variable is also used to hold the disk drive number (0 to 7) and also the planes variable used with DUMP (0 to 7).

AR-43	POLTYP	Poly type or drive or planes for dump

The next variable is used to store the old vector angles for use with relative vectors (turtle). Note the vector angle required is placed in STANG.

AR-45	VECTAN	Old vector angle

The next four variables define the soft clip areas. BEWARE they also define the hard clip areas, if the limits are exceeded then CHAOS will ensue.

AR-47	WXMIN	X min limit
AR-49	WXMAX	X max limit
AR-51	WYMIN	Y min limit
AR-53	WYMAX	Y max limit

The next seven variables are used by the line plotting algorithm.

AR-55	YDIFF	Y difference
AR-57	XDIFF	X difference
AR-59	YSGN	Y sign
AR-61	XSGN	X sign
AR-62	DUTY	Duty cycle count for dot/dash
AR-64	COUNT	Dot/dash pixel counter
AR-66	DFLAG	Dot/dash flag

The next four are ram vectors for plot mode, line, plot and the SET command. These ram vectors can be changed to jump to other (faster) modules, an example would be to drive a flat bed plotter.

AR-68 DOM Plotting mode. The address of the required plot mode is held here. Assembler users will have to find these addresses by using the basic interface and recording the addresses held here for the different modes.

AR-69	JP1	This holds the jump instruction ØC3H.
AR-71	VCTLIN	The line vector
AR-72	JP2	The jump instruction ØC3H.
AR-74	VCTPLT	The plot vector
AR-75	JP3	The jump instruction ØC3H.
AR-77	SETWDG	The basic SET vector wedge
AR-78	JPSET	The jump instruction ØC3H.
AR-79	TEMP	A temporary variable

The next variable when set to 1 causes the last pixel of a line to be plotted, when set to Ø the last pixel is not plotted.

AR-8Ø	LLPT	Last point flag

The next variable hold the user basic SET vector, it is set to the normal basic set return address during a cold start. If the user requires to add commands using the SET vector then this variable should hold the jump vector.
There is a small amount of space available in the AVC software allocated to adding extra commands, it is located at the end of the command table, the rest is up to you!

AR-82	USRVCT	AVC return vector from basic
AR-84	MATADD	Address of the MAT array
AR-85	PPENPT	MAT local pen store
AR-86	ASCCHR	Ascii character spacing
AR-87	CHASPC	Temp location for ASCCHR
AR-89	SINRO	SINE of rotate angle
AR-91	COSRO	COSINE of rotate angle
AR-93	SINAN	SINE of unrotated angle
AR-95	COSAN	COSINE of unrotated angle
AR-97	SINST	SINE of start angle
AR-99	COSST	COSINE of start angle

AR+101	ASCPTR	ASCII TABLE pointer
AR+103	GRAPTR	GRAPHICS TABLE pointer

The next block are internal variables used by the poly module.

AR+105	XCENT	X offset from centre
AR+107	YCENT	Y offset from centre
AR+110	XA	Poly X variable
AR+113	YA	Poly Y variable
AR+116	XX1	Poly rotated position
AR+119	YY1	Poly rotated position
AR+122	XROT	Poly scaled X
AR+125	YROT	Poly scaled Y
AR+128	OLDVEX	Vector high res. angle
AR+131	OLDVEY	Vector high res. angle
AR+134	XTEMP	Rot temp
AR+137	YTEMP	Rot temp
AR+138	SCOUNT	Side count
AR+139	FIRST	First time round
AR+141	FIXABS	First abs X
AR+143	FIYABS	First abs Y
AR+144	SFLAG	Neg angle
AR+145	FLAG90	> 90 degrees
AR+147	ANGLE	Angle
AR+149	SPTR	SIN/COS pointer
AR+150	TCOUNT	Temp counter
AR+152	XCORN	MAT corner
AR+154	YCORN	MAT corner
AR+155	ORGFLG	Origin flag

The next is the ram vector for the fill routine.

AR+157	AAFILL	Fill ram vector
AR+158	JPFILL	Jump instruction 0C3H.

The next variables are local to the fill routine.

AR+160	FX	Fill X
AR+162	FX1	Fill X1
AR+164	FY	Fill Y
AR+166	FD	Fill difference
AR+167	J	Fill variable
AR+168	PEE	Fill variable
AR+169	SAVFLG	Fill save first line pixel flag
AR+170	COLUMN	Fill current column
AR+171	OLDSGN	Fill old line direction

The user DUMP vector is next.

AR+172	DVECT	User dump vector
AR+174	DUMP	Dump jump instruction 0C3H.

The next 10 bytes are spare

AR+184 SPARE Spare bytes

AR+186 STACK Top of the AVC stack

Primary colour planes memory map
(assuming a base address of 08000H, all vales in HEX.)
Single density

column		0	391	
row 255	8000	8002	8029	803F
254	8040	8042	8069	807F
1	BF80	BF82	BFA9	BFBF
0	BFC0	BFC2	BFE9	BFFF

The above is for single density, the same map applies to ALL planes even when in double density mode, but note that the double density mode uses alternative bytes from the red and green planes. The first byte being from the red plane.

The order of display of bits within a byte for both densities is the MSB to the left and the LSB to the right.

APPENDIX E

SYNTAX REFERENCE

This appendix provides a complete list of the graphics commands, the values in brackets are the default values, a * indicates that a default is not allowed.

```
LINE        X,Y,MODE,CARRIAGE
          [ *,*, 0  ,   1     ]
```

Draws a line from the cursor to X,Y. If MODE = 0 then absolute if MODE = 1 then relative. Carriage up if CARRIAGE set to 0.

```
ALINE       X,Y,X1,Y1,MODE,CARRIAGE
          [ *,*,* ,* , 0  ,   1     ]
```

Same as LINE except draws from X,Y to X1,Y1.

```
PLOT        X,Y,MODE,CARRIAGE
          [ *,*, 0  ,   1     ]
```

Plots a point at the point X,Y. If MODE = 0 then absolute if MODE = 1 then relative. Carriage up if CARRIAGE set to 0.

```
SLINE       X,Y,MODE,XMAG,YMAG,SLANT,ROTATE,CARRIAGE
          [ *,*, 0  , 1  , 1  , 0   ,  0   ,   1     ]
```

Draws a line from the cursor to X,Y, but scaled by the X and Y magnifications and the slant and rotate angles.

```
ASLINE      X,Y,X1,Y1,MODE,XMAG,YMAG,SLANT,ROTATE,CARRIAGE
          [ *,*,* ,* ,  0 , 1  ,  1 , 0  ,  0   ,   1     ]
```

The same as SLINE except draws from X,Y to X1,Y1.

```
VECTOR      LENGTH,ANGLE,MODE,CARRIAGE
          [   *   , 0  , 0 ,   1     ]
```

```
AVECTOR   X,Y,LENGTH,ANGLE,MODE,CARRIAGE
          [ *,*,  *   , 0  , 0 ,   1     ]
```

Draws a vector for a distance of length and in a direction of angle. When MODE = 0 then an absolute angle, MODE = 1 then relative to the last vector angle.

```
VECTOR
```

Resets the vector angle.

```
TRI         BASELENGTH,HEIGHT,SLANT,ROTATE,FILL,CARRIAGE
          [   *       , p  , 0  , 0   , 0  ,   1     ]
```

```
ATRI      X,Y,BASELENGTH,HEIGHT,SLANT,ROTATE,FILL,CARRIAGE
        [ *,*,    *     , P   , Ø , Ø   , Ø , 1     ]
```

Note p is set to the BASELENGTH value.
Draws a triangle with the entered BASE LENGTH and HEIGHT. The triangle can be SLANTed, ROTATEd and FILLed.

```
RECT       XLENGTH,YLENGTH,SLANT,ROTATE,FILL,CARRIAGE
         [   *    , p   , Ø , Ø   , Ø , 1     ]

ARECT      X,Y,XLENGTH,YLENGTH,SLANT,ROTATE,FILL,CARRIAGE
         [ *,*,  *    , p   , Ø , Ø   , Ø , 1     ]
```

Note p is set to the X LENGTH VALUE.
Draws a rectangle with X LENGTH and Y LENGTH. The rectangle can be SLANTed, ROTATEd and FILLed.

```
POLY       XRADIUS,YRADIUS,NUMBER OF SIDES,NUMBER OF SIDES
[   *    , P    ,       60       ,       60

DISPLAYED,START  ANGLE,ROTATE  ANGLE,FILL,TYPE,CARRIAGE
         ,    Ø      ,     Ø      ,Ø   , Ø , 1    ]

APOLY      X,Y,XRADIUS,YRADIUS,NUMBER OF SIDES,NUMBER OF SIDES
[ *,*,  *    , P    ,       60       ,       60

DISPLAYED,START  ANGLE,ROTATE  ANGLE,FILL,TYPE,CARRIAGE
         ,    Ø      ,     Ø      , Ø , Ø , 1     ]
```

Draws a polygon of NUMBER OF SIDES displaying NUMBER OF SIDES DISPLAYED. The centre being X,Y (APOLY) or the cursor (POLY) with X RADIUS and Y RADIUS. The polygon can be started at any START ANGLE, ROTATEd and FILLed. TYPE defines either NORMAL (Ø), CORD (1), or RADII (2).

```
LABEL     STRING,XMAG,YMAG,SLANT,ROTATE,CHARACTER SPACING,CARRIAGE
        [ *   , 1 , 1 , Ø , Ø   ,     7            ,  1   ]

ALABEL     X,Y,STRING,XMAG,YMAG,SLANT,ROTATE,CHAR SPACING,CARRIAGE
        [ *,*,  *   , 1 , 1 , Ø , Ø   ,     7      ,  1   ]
```

Prints the STRING with X MAG. and Y MAG. The text can be SLANTed and ROTATEd. The character spacing can also be changed.

```
MAT        ADDRESS,XMAG,YMAG,SLANT,ROTATE,FILL,CARRIAGE
         [   *    , 1 , 1 , Ø , Ø   , Ø , 1       ]

AMAT       X,Y,ADDRESS,XMAG,YMAG,SLANT,ROTATE,FILL,CARRIAGE
         [ *,*,  *    , 1 , 1 , Ø , Ø   , Ø , 1     ]
```

Displays a matrix array at located at ADDRESS with X MAG and Y MAG. It can be SLANTed, ROTATEd and FILLed.

```
BACKGND   RED INTENSITY,GREEN INTENSITY,BLUE INTENSITY,CARRIAGE
        [      Ø       ,        Ø       ,       Ø     ,  1     ]

BACKGND   COLOUR,CARRIAGE
        [   Ø    ,   1     ]
```

Sets the background colour either as R,G,B intensities or primary COLOURs.

```
COLOUR    RED INTENSITY,GREEN INTENSITY,BLUE INTENSITY
        [      16      ,      16              16       ]

COLOUR    COLOUR
        [   7    ]
```

Sets the foreground colour either as R,G,B intensities or primary COLOURs.

```
CURSOR    X VARIABLE,Y VARIABLE
        [     *     ,      *    ]
```

Returns the current cursor position in the X and Y variables.

```
CHECK      VARIABLE,X,Y
         [   *     ,a,b ]
```

Note. a = cursor X and b = cursor Y.
Returns the colour of a pixel in the VARIABLE.

```
DASH       LINE ON,LINE OFF,LAST PIXEL
        [    1    ,    Ø    ,    1      ]
```

Sets the duty cycle of lines ie. number of pixels on (LINE ON) to number of pixels off (LINE OFF). When LAST PIXEL is Ø then the last pixel of a line is not plotted.

```
DEFAULT
```

Defaults many of the system parameters

```
MODE       AVC MODE,AVC OUTPUT
         [    1    ,    p       ]
```

```
Note. when AVC MODE = Ø then p = Ø
            "     "  = 1  "   p = 7
            "     "  = 2  "   p = 2
            "     "  = 3  "   p = 6
```

Sets the avc operating mode and displayed planes.

```
PEN        RED PEN,GREEN PEN,BLUE PEN,PLOTTING MODE
        [    1    ,    1    ,    1    ,     Ø        ]
```

Sets the status of the PENS and the logical PLOTTING MODE.

```
ORIGIN      X,Y
          [ *,* ]
```

Sets the centre of rotation to be X,Y

ORIGIN

Sets the centre of rotation to be cursor following.

DOS

Exits from basic.

```
WINDOW     X MIN,X MAX,Y MIN,Y MAX
         [   Ø   ,   p   ,   Ø   , 255  ]
```

Note. If AVC mode = Ø or 1 then p = 391
" " = 2 " 3 " p = 783
Sets the soft clip window area.

```
FILL       MODE
         [  Ø   ]
```

When mode = 1 all line pixels are stored, when = Ø the area so saved is filled.

```
GSAVE      FILE NAME,DRIVE
         [    *     ,   Ø  ]
```

Saves the current AVC display on disk

```
GLOAD      FILE NAME,DRIVE
         [     *     , Ø   ]
```

Loads the saved AVC display from disk

```
DUMP       PLANES
         [   7   ]
```

Provides a user hard copy facility

APPENDIX F

NASCOM EXTENDED BASIC INTERFACE

The extended basic interface almost the same as the rom basic interface but with the following exceptions:

A] GLOAD and GSAVE file name and drive parameters.

The GLOAD and GSAVE commands will normally use the defaulted drive which is selected by the use of the DRIVE command, note after a basic cold start this is drive A.
The allocation of disk drive numbers using extended basic is;

PHYSICAL DRIVE	LOGICAL DRIVE
Ø side a	A
Ø side b	B
1 side a	C
1 side b	D
2 side a	E
2 side b	F
3 side a	G
3 side b	H
4 side a	I
4 side b	J

Note for a standard dual single sided disk system the two drive are; right hand = A, left hand = C.

To access a drive other than the currently logged drive the drive required can be prefixed to the file name.ie. to access a file called FRED.GPH from drive F then use "F:FRED.GPH".
Note. file names containing the following characters are NOT allowed and will envoke an error;

. , " < > : ; = ? * and ascii codes greater than 127.

B] The COLD and WARM starts described in APPENDIX B are not required as these are performed by the extended basic's cold and warm starts.

C] The label command is not restricted to string literals and variables only but can use string expressions and print the contents of variables.

D] The prefix of the SET command is not required.

E] There is no DOS command as this is not required

F] The address of the top of the AVC variables (AR) is located in two bytes just below the start of basic program area.

G] The assembler entry point (AVCALL) is located in three bytes just below the address of AR.

H] There are approximately 50 bytes of space allocated after the text command extension table.

I] There are approximately 100 bytes of space allocated after the address command extension table.

J] The files G32 or G48 are NOT required as the colour graphics software are an integral part of the EXTENDED COLOUR BASIC.

For details of NASCOM'S CP/M version of the COLOUR EXTENDED BASIC refer to the appropriate manual.

APPENDIX G

DISCLAIMER

It should be noted that this manual and the described software is protected by copyright law.

LUCAS LOGIC LTD. shall incur no liability to any person, company or organisation for any loss or damage caused or alleged to be caused directly or indirectly by equipment or programs or documentation supplied. Such loss or damage includes but is not limited by any interruption of service or loss of business or anticipated profits or consequential damages resulting from the use or operation of such computing equipment, programs or documentation.

The software is NOT guaranteed to be error free but is guaranteed to execute on the destined system.

Copies of part of this manual made be made for use by the purchaser or his/her employees only, complete copies of the manual can be obtained from LUCAS LOGIC. Backup copies of the software may be made for use of the purchaser or the purchasers employees for use on one target system only per purchased copy.

APPENDIX H

AVC SOFTWARE PROBLEM REPORT
(This form may be used for software modifications)

REPORTED BY: DATE:

NAME

ADDRESS

TEL. NUMBER

HARDWARE SYSTEM

SOFTWARE SYSTEM
(and VERSION/SERIAL numbers)

PROBLEM
(Continue on a seperate sheet if necessary, giving listings of programs where possible, and also details of any temporary fixes.)

SEND TO: LUCAS LOGIC LTD.
NASCOM COMPUTERS DIVISION
WELTON ROAD INDUSTRIAL ESTATE
WARWICK CV34 5PZ

AVC CHARACTER TABLE FORMAT

First byte: character spacing from left hand corners of characters.
Second byte: first character in table (in ASCII form).
Third byte: number of characters in table

then characters.

Format of character bytes:-

Bit 7	:	set to 1 for end of character
Bit Ø	:	set to Ø for pen up, to 1 for pen down
Bit 1-3	:	X position
Bit 4-6	:	Y position

Examples

SPACE	8ØH
POUND	Ø7H, 19H, Ø2H, 53H, 65H, 67H, 59H
PLUS	14H, 55H, 3ØH, B9H
M	61H, 45H, 69H, 89H
X	11H, 59H, 69H, 6ØH, 51H, 19H, 89H

To change main character set
either change POLY + 9 to point to table
or ASCPTR to point to table
and ASCCHR to character spacing

To change graphics character set
change GRAPTR to point to table