NICK HAMPSHIRE
WITH RICHARD FRANKLIN AND CARL GRAHAM

ADVANCED COMMODORE 64 GRAPHICS AND SOUND

Advanced Commodore 64 Graphics and Sound

Also by Nick Hampshire

The Commodore 64 ROMs Revealed
0 00 383087 X

Advanced Commodore 64 BASIC Revealed
0 00 383088 8

The Commodore 64 Kernal and Hardware Revealed
0 00 383090 X

The Commodore 64 Disk Drive Revealed
0 00 383091 8

Advanced Commodore 64 Graphics and Sound

Nick Hampshire

with Richard Franklin and Carl Graham

COLLINS
8 Grafton Street, London W1

Collins Professional and Technical Books
William Collins Sons & Co. Ltd
8 Grafton Street, London W1X 3LA

First published in Great Britain by
Collins Professional and Technical Books 1985

British Library Cataloguing in Publication Data
Hampshire, Nick
Advanced Commodore 64 graphics and sound.
1. Commodore 64 (Computer)—Programming
2. Computer sound processing 3. Computer graphics
I. Title II. Franklin, Richard
III. Graham, Carl
001.64'43 QA76.8.C64

ISBN 0-00-383089-6

Typeset by V & M Graphics Ltd, Aylesbury, Bucks
Printed and bound in Great Britain by
Mackays of Chatham, Kent

Contents

Preface	vii
1 The Graphics and Sound Registers	1
2 Extended Graphics Commands	13
1 Wedges To Give The Routines Names	15
2 Screen Management Routines	25
3 Two Dimensional Plotting Routines	35
4 Three Dimensional Plotting Routines	100
5 Miscellaneous Routines For Use With The Graphics Package	107
3 The Theory of High Resolution Graphics Displays	121
4 Games Graphics: Some Hints and Techniques	148
5 Some Advanced Aspects of Sound on the CBM 64	160
Appendix A: Sprite Editor	177
Appendix B: Character Editor	181
Appendix C: Sound Editor	185
Index	190

Preface

The CBM 64 has one of the finest graphics and sound capabilities of any low price popular computer currently on the market. The problem, though, is that the techniques required to utilise the full potentialities of the machine are neither easy to use nor understand. The principal reason is that no graphics or sound commands are included in the Basic interpreter. Anyone wishing to program the CBM 64 to utilise either its graphics or sound must therefore make extensive use of PEEK and POKE commands or write routines in machine code to provide some higher level graphics and sound functions. To aid programmers of the 64 such routines are included in Chapter 2 of this book; the remainder of the book is devoted to explaining how to use the hardware of the machine plus the routines in this graphics package to their full extent. Since advanced graphics techniques are covered here it is assumed that the reader already has an understanding of all the basic principles. Any reader seeking a more elementary level understanding of the subject should refer to one of numerous introductory titles, such as *Commodore 64 Graphics and Sound* by Steve Money, published by Collins Professional and Technical Books.

Nick Hampshire

Chapter One

The Graphics and Sound Registers

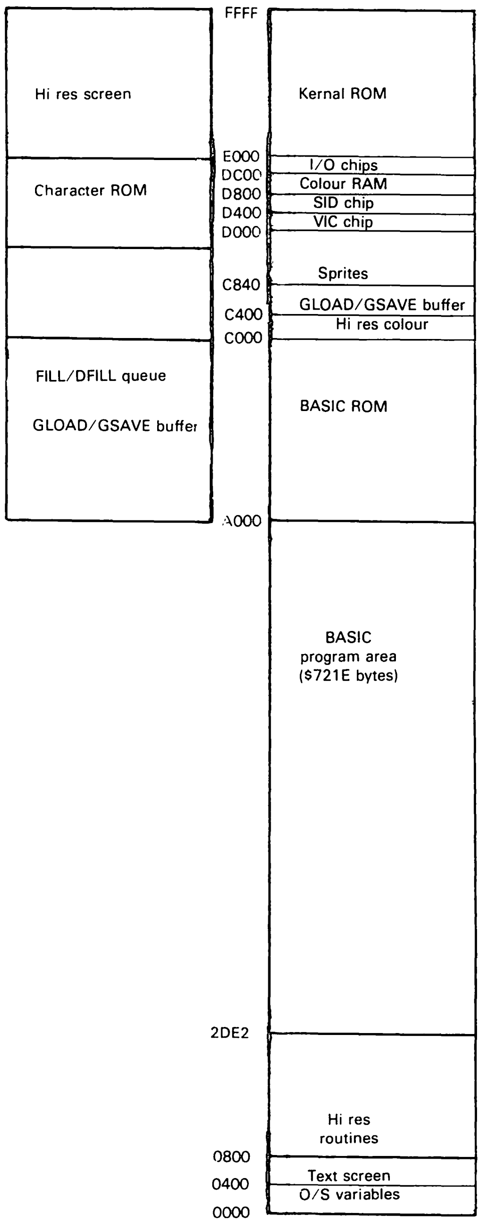


Fig. 1.1. Graphics memory usage for the CBM 64.

All the graphics of the CBM 64 are generated and controlled by a single integrated circuit, the 6567 Video Interface Chip, commonly known as the VIC-11 chip. This circuit not only controls and generates the normal 25 line by 40 character text display but also can display high resolution (320 by 200 pixel) graphics. In addition the VIC-11 chip can display up to 8 different sprites on the screen. All the sound output is generated by another complex chip, the 6581 Sound Interface Device, commonly known as SID. This chip will generate sound of different programmable waveforms and frequencies through one or all of the 3 voices, plus a white noise generator. Both the VIC and SID chips communicate with the processor via a set of memory locations or registers, and it is the values placed in these registers which determine the function and operation of the respective chip.

1.1 Graphics memory usage

Memory space is obviously required for the VIC chip I/O registers. In addition memory is required by the screen, sprites, and character memory. Apart from the VIC I/O registers and the colour memory, none of the areas used is fixed; they can be put in different positions, depending on the display mode and the selected video bank.

1 Display mode The display mode simply means whether the display is a character display or a bit mapped display and whether it is in normal, multicolour or extended colour mode. In all display modes sprites can be enabled in both standard and multicolour modes. The association and derivation of the 8 different attainable display modes are shown in the tree structured diagram in Fig. 1.2.

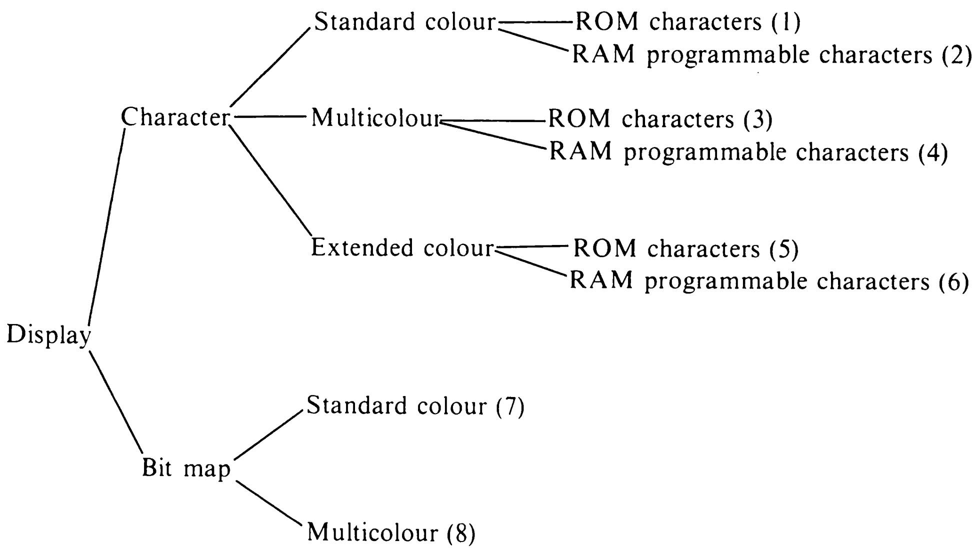


Fig. 1.2.

Note: Mode 3 multicolour ROM characters are most unsatisfactory and are therefore virtually never used.

In this section we are interested in the different memory allocations used in character and bit mapped modes and the memory required for programmable character sets. The various colour modes are dealt with in a later section of this chapter.

2 **Video bank selection** The 'video bank' selected refers to the fact that the VIC chip can access only 16K of memory at a time. However, by using a bank select feature the VIC chip can be moved to any one of the four 16K banks in the CBM 64. There are thus four possible 'banks' of 16K memory, and for the VIC chip to access any one of these it requires the appropriate 'bank select' lines to be set. The bank select lines are derived from lines 0 and 1 of Port A of the 6526 Complex Interface Adapter #2 (commonly known as CIA#2). To enable these bank select lines two memory addresses are required. These are the data direction register of CIA#2 at location $DD02 (decimal 56578) and the I/O register of CIA#2 at $DD00 (decimal 56576). The data direction register is required to set the two lines as outputs. In Basic this can easily be done by the command:

```
POKE 56578, PEEK(56578) OR 3
```

To set the two lines so that VIC is located in the required section of memory the following command is used:

```
POKE 56576, (PEEK(56576) AND 252) OR X
```

where X is a value between 0 and 3 which determines which video bank is to be selected. This is shown in the following table:

Bank	Value in X	Address range for VIC chip
0	3	$0000–$3FFF (normal default position)
1	2	$4000–$7FFF
2	1	$8000–$BFFF
3	0	$C000–$FFFF

3 **Colour memory** The colour memory is at a fixed location and consists of a block of memory situated between locations $D800 (55296) and $DBE7 (56295). These 1000 memory locations are just 4 bits wide rather than the usual 8 bits or byte wide, and they correspond exactly to the 1000 bytes of the screen character memory. The colour memory is used to define the character colour in each of the 1000 screen character locations; the 4 bits are used to define which colour is used. The way the colour memory is used depends on the display mode.

4 **Screen memory** The screen memory is a 1000 byte long block of memory used to store the character codes for each character location on the screen. The

screen memory can be moved by the programmer to start at any one of 16 different memory locations. The starting address of screen memory is controlled by the contents of the upper 4 bits of the control register at location $D018 (53272). Screen memory can be set to start at different addresses using the following command:

POKE 53272, (PEEK (53272) AND 15) OR X

Where X is a value which determines which start address is used, the value of X can be obtained from the following table:

Screen memory block #	Value of X	Memory location of screen memory start in bank Hexadecimal	Decimal
1	0	$0000	0
2	16	$0400	1024 (normal default)
3	32	$0800	2048
4	48	$0C00	3072
5	64	$1000	4096
6	80	$1400	5120
7	96	$1800	6144
8	112	$1C00	7168
9	128	$2000	8192
10	144	$2400	9216
11	160	$2800	10240
12	176	$2C00	11264
13	192	$3000	12288
14	208	$3400	13312
15	224	$3800	14336
16	240	$3C00	15360

It should be noted that the table relates to a single 16K bank and can therefore be applied to any of the 4 memory banks in order to position the screen memory anywhere within addressable memory space. If the kernal is being used then a value equal to the location address divided by 256 must be placed in location 648 (this location therefore normally has a default value of 4).

5 Character memory In the character display mode the screen memory contains a code value in each character location. This code value is used by the VIC chip to get the character display information from an area of memory referred to as the character generator. This display information consists of the pattern of 8 by 8 pixels which are used to define each character. The character generator is normally located in a ROM but the VIC chip can also use character generator data stored in RAM memory. This is particularly useful since it allows user generated characters. The character generator occupies 4K of memory and can store the pattern data on up to 512 different characters. The location of the character generator is determined by bits 1, 2 and 3 of the VIC chip control register at location $DØ18 (53272). Within any one of the four 16K banks of memory accessed by the VIC chip the character generator can start at one of 8 different memory locations. The following Basic line can be used to set the location of character memory:

POKE 53272, (PEEK (53272) AND 24Ø) OR X

where X determines the start position of the character memory within the selected bank according to the following table:

Character memory block #	Value for X	Start location of character memory in bank Hexadecimal	Decimal
1	Ø	$ØØØØ	Ø
2	2	$Ø8ØØ	2Ø48
3	4	$1ØØØ	4Ø96 *
4	6	$18ØØ	6144 **
5	8	$2ØØØ	8192
6	1Ø	$28ØØ	1Ø24Ø
7	12	$3ØØØ	12228
8	14	$38ØØ	14336

* - default ROM image for upper case characters starts at this location when banks Ø and 2 are selected.
** - default ROM image for lower case characters starts at this location when banks Ø and 2 are selected.

In normal default operation bank Ø is selected and the character generator ROM is located to start at address $1ØØØ (4Ø96), though in reality the character generator ROM is located at $DØØØ-$DFFF (53248-57343). The character generator located at $1ØØØ up is known as a ROM image. This is possible due to the separate address bus of the VIC chip which makes the ROM appear as an image starting at location $1ØØØ whilst not actually using any RAM memory. Thus in the mode where the character generator ROM is enabled the RAM memory accessed by VIC is usable for program storage etc.

Since the character generator ROM occupies the same memory space as the VIC chip control registers, though of course in a separate switched memory bank, this can cause some problems if one wishes to access the character generator ROM directly. This is overcome fairly easily by first disabling the interrupts, then switching the I/O out, copying the required character generator data into RAM, then switching the I/O back in and re-enabling the interrupts.

The character generator ROM is divided into two 2K blocks, upper case characters and lower case characters. Their contents can be classified as follows:

Block	Character generator ROM address Hexadecimal	Decimal	ROM image address Hexadecimal	Decimal	Contents
∅	$D∅∅∅	53248	$1∅∅∅	4∅96	upper case characters
	$D2∅∅	5376∅	$12∅∅	46∅8	graphics characters
	$D4∅∅	54272	$14∅∅	512∅	reverse upper case
	$D6∅∅	54784	$16∅∅	5632	reverse graphics
1	$D8∅∅	55296	$18∅∅	6144	lower case characters
	$DA∅∅	558∅8	$1A∅∅	6656	upper case + graphics
	$DC∅∅	5632∅	$1C∅∅	7168	reverse lower case
	$DE∅∅	56832	$1E∅∅	768∅	reverse upper case + graphics

1.2 VIC chip registers

The VIC chip registers are the programmer's means of directly controlling the mode of operation, kind of display and memory allocation of displays on the CBM 64. There are altogether 47 registers on the VIC chip, and each register occupies a single memory location. They are located between $D∅∅∅ and $D∅2E (53248 to 53294). These registers are of vital importance in any graphics programming application, and they can be summarised as follows:

Reg #	Hex	Dec	Bits	Function
∅	D∅∅∅	53248	7–∅	X position of sprite ∅ low byte
1	D∅∅1	53249	7–∅	Y position of sprite ∅
2	D∅∅2	53250	7–∅	X position of sprite 1 low byte
3	D∅∅3	53251	7–∅	Y position of sprite 1
4	D∅∅4	53252	7–∅	X position of sprite 2 low byte
5	D∅∅5	53253	7–∅	Y position of sprite 2
6	D∅∅6	53254	7–∅	X position of sprite 3 low byte
7	D∅∅7	53255	7–∅	Y position of sprite 3
8	D∅∅8	53256	7–∅	X position of sprite 4 low byte
9	D∅∅9	53257	7–∅	Y position of sprite 4
1∅	D∅∅A	53258	7–∅	X position of sprite 5 low byte
11	D∅∅B	53259	7–∅	Y position of sprite 5
12	D∅∅C	5326∅	7–∅	X position of sprite 6 low byte
13	D∅∅D	53261	7–∅	Y position of sprite 6
14	D∅∅E	53262	7–∅	X position of sprite 7 low byte
15	D∅∅F	53263	7–∅	Y position of sprite 7
16	D∅1∅	53264	7	X position of sprite 7 high bit
			6	X position of sprite 6 high bit
			5	X position of sprite 5 high bit

Reg #	Hex	Dec	Bits	Function
			4	X position of sprite 4 high bit
			3	X position of sprite 3 high bit
			2	X position of sprite 2 high bit
			1	X position of sprite 1 high bit
			Ø	X position of sprite Ø high bit
17	DØ11	53265	7	Raster compare high bit
			6	Enable extended colour text
			5	Enable bit map mode
			4	Enable screen display (Ø=blank screen to border)
			3	Enable 25 row text (Ø=24 row)
			2–Ø	Y smooth scroll pos (1/8 char inc.)
18	DØ12	53266	7–Ø	Read : current raster position Write : raster compare value
19	DØ13	53267	7–Ø	Light pen X pos
2Ø	DØ14	53268	7–Ø	Light pen Y pos
21	DØ15	53269	7	Enable sprite 7 display
			6	Enable sprite 6 display
			5	Enable sprite 5 display
			4	Enable sprite 4 display
			3	Enable sprite 3 display
			2	Enable sprite 2 display
			1	Enable sprite 1 display
			Ø	Enable sprite Ø display
22	DØ16	5327Ø	7–6	Not used
			5	Leave as Ø?
			4	Enable multicolour mode
			3	Enable 4Ø column text (Ø=38 column text)
			2–Ø	Smooth scroll X pos (1/8 char inc.)
23	DØ17	53271	7	Enable vertical expansion on sprite 7
			6	Enable vertical expansion on sprite 6
			5	Enable vertical expansion on sprite 5
			4	Enable vertical expansion on sprite 4
			3	Enable vertical expansion on sprite 3
			2	Enable vertical expansion on sprite 2
			1	Enable vertical expansion on sprite 1
			Ø	Enable vertical expansion on sprite Ø
24	DØ18	53272	7–4	Screen address in current bank
			3–1	Dot data address in bank *Note:* In high resolution, bits 2 & 1 ignored
			Ø	Not used
25	DØ19	53273	7	An enabled VIC IRQ occurred
			6–4	Not used
			3	Light pen IRQ condition occurred
			2	Sprite to sprite collision occurred
			1	Sprite to background collision occurred
			Ø	Raster compare IRQ condition occurred

Reg #	Hex	Dec	Bits	Function
26	DØ1A	53274	7–4	Not used
			3	Enable light pen IRQ
			2	Enable sprite to sprite collision IRQ
			1	Enable sprite to background collision IRQ
			Ø	Enable raster compare IRQ
27	DØ1B	53275	7–Ø	Enable priority of sprites 7–Ø over background
28	DØ1C	53276	7–Ø	Enable multicolour mode of sprites 7–Ø
29	DØ1D	53277	7–Ø	Enable horizontal expansion of sprites 7–Ø
3Ø	DØ1E	53278	7	Sprite collision with sprite 7 occurred
			6	Sprite collision with sprite 6 occurred
			5	Sprite collision with sprite 5 occurred
			4	Sprite collision with sprite 4 occurred
			3	Sprite collision with sprite 3 occurred
			2	Sprite collision with sprite 2 occurred
			1	Sprite collision with sprite 1 occurred
			Ø	Sprite collision with sprite Ø occurred
31	DØ1F	53279	7–Ø	Background collision with sprite(s) 7–Ø
32	DØ2Ø	5328Ø	7–4	Not used
			3–Ø	Border colour
33	DØ21	53281	7–4	Not used
			3–Ø	Screen background colour
34	DØ22	53282	7–4	Not used
			3–Ø	Extended background colour 1
35	DØ23	53283	7–4	Not used
			3–Ø	Extended background colour 2
36	DØ24	53284	7–4	Not used
			3–Ø	Extended background colour 3
37	DØ25	53285	7–4	Not used
			3–Ø	Sprite multicolour Ø
38	DØ26	53286	7–4	Not used
			3–Ø	Sprite multicolour 1
39	DØ27	53287	7–4	Not used
			3–Ø	Sprite Ø colour
4Ø	DØ28	53288	7–4	Not used
			3–Ø	Sprite 1 colour
41	DØ29	53289	7–4	Not used
			3–Ø	Sprite 2 colour
42	DØ2A	5329Ø	7–4	Not used
			3–Ø	Sprite 3 colour
43	DØ2B	53291	7–4	Not used
			3–Ø	Sprite 4 colour
44	DØ2C	53292	7–4	Not used
			3–Ø	Sprite 5 colour
45	DØ2D	53293	7–4	Not used
			3–Ø	Sprite 6 colour
46	DØ2E	53294	7–4	Not used
			3–Ø	Sprite 7 colour

1.3 High resolution display storage

High resolution displays, whether in normal or multicolour mode, require 8ØØØ bytes of memory to store the pixel data of the display. A high resolution bit mapped mode is initialised by setting bit 5 of the VIC chip control register at $DØ11 (53265) to 1, as in the following Basic command:

```
POKE 53265, PEEK (53265) OR 32
```

Normal character mapped mode can be resumed by clearing bit 5 as in the following Basic command:

```
POKE 53265, PEEK (53265) AND 223
```

Before placing the screen in the high resolution mode the area of memory used to store the pixel data must be defined. Within any one of the four 16K banks, which must obviously have been defined previously, there are two possible start addresses of this 8ØØØ byte memory area: at the start of the bank and at the start of the upper 8K of the bank. Which of these two locations is used is determined by the contents of bit 3 of location $DØ18 (53272); a Ø = bottom 8K and a 1 = top 8K as in the following Basic command:

```
POKE 53272,PEEK (53272) OR 8        set to start at top 8K
POKE 53272,PEEK (53272) AND 247     set to start at bottom 8K
```

1.4 Sprite display storage

Within any one of the four 16K banks up to eight sprites can be active at one time. These sprites require two different areas of memory within the block. The first, a block of eight bytes, defines the position of the sprite data for the eight enabled sprites, and the second is divided into eight blocks (which need not be continuous within memory) each of 64 bytes which contain the data on each sprite. The eight bytes used to store the position of the sprite data are located at the top eight bytes of the 1Ø24 byte screen memory. If the screen memory starts at $Ø4ØØ and ends at $Ø7FF then the sprite data position pointers are located at $Ø7F8 to $Ø7FF, with sprite Ø pointer from $Ø7F8 upwards to sprite 7 at $Ø7FF. Each of these sprite pointers is a value between Ø and 255 and refers to the fact that the 16K bank can be divided into 256 blocks of 64 bytes. The pointer therefore refers to the 64 byte block corresponding to that value. It should be noted that other than defining the screen memory location no VIC chip registers are used to define sprite data positions.

1.5 The SID chip

The SID chip is a complex integrated circuit which generates the sound output of the CBM 64; it also controls the analog joystick inputs. The SID chip, like the VIC chip, interfaces with the programmer and the operating system via 29 registers which occupy memory locations between $D4ØØ and $D41C

(54272–543∅∅). Some of these registers are read only, the rest are write only. The SID registers can be divided into six logical groupings; these are voices 1, 2, 3, filters, analog input and voice 3 output. The group of seven registers for voice 1 is identical to the groups for voices 2 and 3. These registers control the frequency, pulse width, attack/decay and sustain plus other features such as ring modulation and sync. The following is a list of all 29 SID registers showing their function, location, read/write status and bits used:

SID registers
$D4∅∅–$D41C (54272–543∅∅)

Reg No.	Hex	Dec	R/W	Bits	Function
Voice 1					
∅	D4∅∅	54272	W	7–∅	Oscillator frequency low byte
1	D4∅1	54273	W	7–∅	Oscillator frequency high byte
2	D4∅2	54274	W	7–∅	Pulse width low byte
3	D4∅3	54275	W	7–4	Not used
				3–∅	Pulse width high nibble
4	D4∅4	54276	W	7	Enable noise
				6	Enable pulse
				5	Enable sawtooth
				4	Enable triangle
				3	Disable oscillator
				2	Enable ring modulation with voice 3 output
				1	Enable synchronisation with voice 3 frequency
				∅	Enable start attack/decay/sustain. On reset start release
5	D4∅5	54277	W	7–4	Attack duration select
				3–∅	Decay duration select
6	D4∅6	54278	W	7–4	Sustain level
				3–∅	Release cycle duration select
Voice 2					
7	D4∅7	54279	W	7–∅	Oscillator frequency low byte
8	D4∅8	5428∅	W	7–∅	Oscillator frequency high byte
9	D4∅9	54281	W	7–∅	Pulse width low byte
1∅	D4∅A	54282	W	7–4	Not used
				3–∅	Pulse width high nibble
11	D4∅B	54283	W	7	Enable noise
				6	Enable pulse
				5	Enable sawtooth
				4	Enable triangle
				3	Disable oscillator
				2	Enable ring modulation with voice 1 output
				1	Enable synchronisation with voice 1 frequency

Reg No.	Hex	Dec	R/W	Bits	Function
				∅	Enable start attack/decay/sustain. On reset start release
12	D4∅C	54284	W	7–4	Attack duration select
				3–∅	Decay duration select
13	D4∅D	54285	W	7–4	Sustain level
				3–∅	Release cycle duration select
Voice 3					
14	D4∅E	54286	W	7–∅	Oscillator frequency low byte
15	D4∅F	54287	W	7–∅	Oscillator frequency high byte
16	D41∅	54288	W	7–∅	Pulse width low byte
17	D411	54289	W	7–4	Not used
				3–∅	Pulse width high nibble
18	D412	5429∅	W	7	Enable noise
				6	Enable pulse
				5	Enable sawtooth
				4	Enable triangle
				3	Disable oscillator
				2	Enable ring modulation with voice 1 output
				1	Enable synchronisation with voice 1 frequency
				∅	Enable start attack/decay/sustain. On reset start release
19	D413	54291	W	7–4	Attack duration select
				3–∅	Decay duration select
2∅	D414	54292	W	7–4	Sustain level
				3–∅	Release cycle duration select
Filters etc.					
21	D415	54293	W	7–3	Not used
				2–∅	Filter cutoff frequency low bits
22	D416	54294	W	7–∅	Filter cutoff frequency high byte
23	D417	54295	W	7–4	Filter resonance
				3	Enable filtering of external input
				2	Enable filtering of voice 3 output
				1	Enable filtering of voice 2 output
				∅	Enable filtering of voice 1 output
24	D418	54296	W	7	Disable voice 3 output
				6	Enable high-pass filtering
				5	Enable band-pass filtering
				4	Enable low-pass filtering
				3–∅	Chip output volume
Analog input registers					
25	D419	54297	R	7–∅	Analog/digital converter 1 output
26	D41A	54298	R	7–∅	Analog/digital converter 2 output

Reg No.	Hex	Dec	R/W	Bits	Function
Voice 3 output					
27	D41B	54299	R	7–Ø	Oscillator #3 output
28	D41C	543ØØ	R	7–Ø	Envelope generator 3 output

Chapter Two
Extended Graphics Commands

2.1 Introduction

This chapter contains a collection of programs which will add 27 extra commands; these extra commands will be of considerable use to any Basic programmer. The commands will require the wedge programs at the start of this chapter to be loaded as part of the assembly. These wedges allow the commands which follow to be used as ordinary Basic commands. The commands and a description of their use are given in the documentation accompanying each of the routines. All these extra commands and their associated wedge, tokenising and parsing routines are designed to be stored in locations $Ø8ØØ up for an area of just over 9.25K of memory. The routines are enabled simply by loading and typing RUN. The listings are all in CBM assembler format. For readers wishing to obtain these programs in machine readable form they are available as both source and object code on 1541 disk from Zifra Software Ltd, 4Ø Bowling Green Lane, London EC1, price £10.00 inclusive (make cheques payable to Zifra Software Ltd).

The extended graphics package can be split into 5 sections:

1. **Wedges to give the routines names.**
   - initialisation wedge
   - crunch to tokens wedge
   - tokens to text wedge
   - execute statement wedge
   - execute arithmetic wedge
   - IRQ wedge
2. **Screen management routines.**
   - window
   - hires
   - graph
   - clg
   - clc
   - norm
   - origin
   - screen and border
3. **Two dimensional plotting routines.**
   - plot
   - draw
   - char

point
fill
dfill
polygon
circle
mat
shape

4. Three dimensional plotting routines.

porigin
pplot
pdraw

5. Miscellaneous routines for use with the graphics package.

off
place
sprite
gload and gsave

Variable declaration for the extended graphics package

```
LOC    CODE          LINE

0000                        .LIB DECLARE
0000                 MODE   =$02                 ;MODE FLAG
0000                 T1     =$57                 ;POINTER TO HIRES SCREEN
0000                 T2     =$59                 ;X COORDINATE
0000                 T3     =$5B                 ;Y COORDINATE
0000                 T4     =$5D                 ;Y AND 7
0000                 T5     =$5E
0000                 T6     =$5F                 ;USED AS SECOND BIT IN MULTI
0000                 POINTR =$61                 ;CHARACTER ROM POINTER
0000                 PBR    =$FC                 ;'PAINTBRUSH'
0000                 COL    =$FD                 ;POINT COLOUR
0000                        * =$033C
033C   00 00         X1     .WOR 0               ;COORDINATES FOR LINE
033E   00 00         Y1     .WOR 0               ;PLOT ROUTINE
0340   00 00         X2     .WOR 0               ;AS X1,Y1
0342   00 00         Y2     .WOR 0               ;AND X2,Y2
0344   00 00         XD     .WOR 0               ;DIFFERENCE OF X1 AND X2
0346   00 00         YD     .WOR 0               ;DIFFERENCE OF Y1 AND Y2
0348   00 00         XE     .WOR 0               ;ABS(XD)
034A   00 00         YE     .WOR 0               ;ABS(YD)
034C   00 00         LG     .WOR 0               ;THE NEXT 11 ARE
034E   00 00         SH     .WOR 0               ;VARIABLES USED IN THE
0350   00 00         TS     .WOR 0               ;LINE PLOT ROUTINE
0352   00 00         TT     .WOR 0
0354   00 00         UD     .WOR 0
0356   00 00         CT     .WOR 0
0358   00 00         D1     .WOR 0
035A   00 00         S0     .WOR 0
035C   00 00         S1     .WOR 0
035E   00 00         A0     .WOR 0
0360   00 00         A1     .WOR 0
0362   00 00         XTL    .WOR 0               ;TOP LEFT X OF CHARACTER
0364   00 00         YTL    .WOR 0               ;TOP LEFT Y OF CHARACTER
0366   00            CHAR   .BYT 0               ;CHARACTER
0367   00            RVORN  .BYT 0               ;REVERSE OR NORMAL
0368   00 00         XTEMP  .WOR 0               ;TEMP X LOCATION
036A   00            CNTR1  .BYT 0               ;COUNTER FOR CHAR PLOT
036B   00            POINT  .BYT 0               ;BIT OF CHAR BEING PLOTTED
036C   00 00         X11    .WOR 0
```

```
LOC   CODE        LINE

036E  00          Y11    .BYT 0
036F  00 00       X22    .WOR 0
0371  00          Y22    .BYT 0
0372  00          PTBR   .BYT 0
0373  00 00       BRCOL  .WOR 0
0375  00          LSW    .BYT 0
0376  00          USW    .BYT 0
0377  00 00       XTLTMP .WOR 0
0379  00 00       YTLTMP .WOR 0
037B  00          PBRTMP .BYT 0
037C  00          COLTMP .BYT 0
037D              XMAX   =320
037D              YMAX   =200
037D              VIC    =$D000
037D              CHKCOM =$AEFD               ;SCAN PAST COMMA
037D              PARAMS =$B7EB               ;2 PARAMETERS 2BYTE,1BYTE
037D              PARAM  =$B7F1               ;JUST 1 BYTE
037D                     .END
```

1 WEDGES TO GIVE THE ROUTINES NAMES

The following routines are the start of the graphics extension commands. These are the main control routines that patch the extra commands into the Commodore 64's Basic. They should be used in the order in which they appear.

2.2 Initialisation

This file contains the initialisation routines, the I/O wedges, and the table of added commands and their vectors. The commands are initialised by typing 'RUN' or 'SYS2Ø64'.

The routine labelled 'COLD' is the actual power-up routine and the routine labelled 'WRST' is the NMI routine which ensures that the kernal vector changes for this package are not reset.

```
LOC   CODE        LINE

037D                     .LIB INITRT
037D                     * =$0801
0801  0B 08              .WOR EOL
0803  0A 00              .WOR 10        ;LINE 10
0805  9E                 .BYT $9E,'2064',0 ;SYS2064
0806  32 30
080A  00
080B  00          EOL    .BYT 0,0
080C  00
080D              ;
080D                     * =$0810
0810              ;
0810  4C B5 08           JMP COLD
0813              ;
0813  8B E3       LINK   .WOR $E38B      ;NORMAL ERROR
0815  83 A4              .WOR $A483      ;NORMAL WARM START
```

```
LOC   CODE        LINE

0817  91 0A                .WOR CRNCHT     ;CRUNCH TO TOKENS
0819  5B 0B                .WOR PRINT      ;PRINT TOKENS
081B  B4 0B                .WOR HANDLE     ;HANDLE COMMAND
081D  D6 0B                .WOR ARITH      ;HANDLE ARITHMETIC
081F                ;
081F  4C 48 B2      VECTOR JMP $B248        ;USR JUMP
0822  00                   .BYT 0
0823  4C 0C                .WOR IRQVEC     ;IRQ
0825  5D 08                .WOR WRST01     ;BREAK
0827  43 08                .WOR WRST       ;NMI
0829  4A F3                .WOR $F34A      ;OPEN
082B  91 F2                .WOR $F291      ;CLOSE
082D  0E F2                .WOR $F20E      ;SET INPUT
082F  50 F2                .WOR $F250      ;SET OUTPUT
0831  33 F3                .WOR $F333      ;RESTORE I/O
0833  57 F1                .WOR $F157      ;INPUT
0835  CA F1                .WOR $F1CA      ;OUTPUT
0837  ED F6                .WOR $F6ED      ;TEST-STOP
0839  3E F1                .WOR $F13E      ;GET
083B  2F F3                .WOR $F32F      ;ABORT I/O
083D  5D 08                .WOR WRST01     ;WARM RESTART
083F  06 0A                .WOR LOAD       ;LOAD
0841  11 0A                .WOR SAVE       ;SAVE
0843                ;
0843  48            WRST   PHA
0844  8A                   TXA
0845  48                   PHA
0846  98                   TYA
0847  48                   PHA
0848  A9 7F                LDA #$7F
084A  8D 0D DD             STA $DD0D
084D  AC 0D DD             LDY $DD0D
0850  10 03                BPL WRST2
0852  4C 72 FE      WRST1  JMP $FE72
0855  20 BC F6      WRST2  JSR $F6BC
0858  20 E1 FF             JSR $FFE1
085B  D0 F5                BNE WRST1
085D  20 A3 FD      WRST01 JSR $FDA3        ;INIT I/O
0860  20 18 E5             JSR $E518       ;INIT VIC CHIP
0863  20 76 08             JSR SETKER      ;INIT KERNAL VECTORS
0866  20 C3 FF             JSR $FFC3       ;RESTORE I/O
0869  A9 00                LDA #$00
086B  85 13                STA $13         ;INPUT PROMPT FLAG
086D  20 7A A6             JSR $A67A       ;INIT BASIC
0870  58                   CLI             ;ENABLE IRQ
0871  A2 80         WRST02 LDX #$80         ;SET FOR READY
0873  4C 88 E3             JMP $E388       ;GO TO READY
0876                ;
0876  A2 1F         SETKER LDX #<VECTOR     ;POINT TO
0878  A0 08                LDY #>VECTOR    ;KERNAL VECTORS
087A  86 C3                STX $C3
087C  84 C4                STY $C4
087E  A0 23                LDY #$23        ;LOOP TO COPY VECTORS
0880  B1 C3         STKER1 LDA ($C3),Y      ;GET BYTE
0882  99 10 03             STA $0310,Y     ;STORE IT
0885  88                   DEY
0886  10 F8                BPL STKER1      ;AND NEXT
0888  A9 00                LDA #$00
088A  8D 23 0C             STA WNTFLG
088D  8D 51 0E             STA TXTFLG
0890  8D EE 0C             STA RASFLG
0893  8D 12 D0             STA $D012
0896  AD 11 D0             LDA $D011
0899  29 7F                AND #$7F
089B  8D 11 D0             STA $D011
089E  AD 1A D0             LDA $D01A
```

```
LOC    CODE          LINE

08A1   09 01                 ORA #$01
08A3   8D 1A D0              STA $D01A
08A6   A9 1F                 LDA #$1F
08A8   8D 0D DD              STA $DD0D
08AB   8D 0D DC              STA $DC0D
08AE   AD 0D DD              LDA $DD0D
08B1   AD 0D DC              LDA $DC0D
08B4   60                    RTS
08B5                 ;
08B5   78            COLD    SEI              ;DISABLE IRQ
08B6   20 76 08              JSR SETKER      ;SET KERNAL VECTORS
08B9   58                    CLI              ;ENABLE IRQ
08BA   20 3F 09              JSR SETBAS      ;SET BASIC VECTORS
08BD   A9 E2                 LDA #<BSSTRT    ;SET TOP OF RAM
08BF   85 2B                 STA $2B
08C1   A9 2D                 LDA #>BSSTRT
08C3   85 2C                 STA $2C
08C5   A9 E4                 LDA #<VRSTRT    ;SET VAR POINTERS
08C7   85 2D                 STA $2D
08C9   85 2F                 STA $2F
08CB   85 31                 STA $31
08CD   A9 2D                 LDA #>VRSTRT
08CF   85 2E                 STA $2E
08D1   85 30                 STA $30
08D3   85 32                 STA $32
08D5   A9 E1                 LDA #<POWER     ;POINT TO POWER
08D7   A0 08                 LDY #>POWER     ;UP MESSAGE
08D9   20 2D E4              JSR $E42D       ;OUTPUT MESSAGE
08DC   A2 FB                 LDX #$FB
08DE   9A                    TXS              ;SET STACK POINTER
08DF   D0 90                 BNE WRST02      ;ALWAYS
08E1                 ;
08E1   93            POWER   .BYT $93,$0D
08E2   0D
08E3   20 20                 .BYT '    **** ZIFRA 64 GRAPHICS'
08FD   20 56                 .BYT ' V01 ****',$0D,$0D
0906   0D
0907   0D
0908   20 20                 .BYT '    (C) ZIFRA SOFTWARE'
091E   20 4C                 .BYT ' LIMITED 1984',$0D,$0D
092B   0D
092C   0D
092D   20 36                 .BYT ' 64K RAM SYSTEM  ',$00
093E   00
093F                 ;
093F   A2 0B         SETBAS LDX #$0B          ;LOOP
0941   BD 13 08      STBAS1 LDA LINK,X        ;GET BYTE
0944   9D 00 03              STA $0300,X     ;STORE IT
0947   CA                    DEX
0948   10 F7                 BPL STBAS1      ;DO NEXT
094A   60                    RTS
094B                 ;
094B   48 49         CLIST   .BYT 'HIRE',$D3 ;TOKEN=$CC
094F   D3
0950   50 4C 4F              .BYT 'PLO',$D4 ;       $CD
0953   D4
0954   4E 4F 52              .BYT 'NOR',$CD ;       $CE
0957   CD
0958   44 52 41              .BYT 'DRA',$D7 ;       $CF
095B   D7
095C   43 48 41              .BYT 'CHA',$D2 ;       $D0
095F   D2
0960   57 49                 .BYT 'WINDO',$D7 ;      $D1
0965   D7
0966   46 49 4C              .BYT 'FIL',$CC ;       $D2
0969   CC
```

```
LOC   CODE          LINE

096A  43 4C                  .BYT 'CL',$C7   ;        $D3
096C  C7
096D  43 4C                  .BYT 'CL',$C3   ;        $D4
096F  C3
0970  47 52                  .BYT 'GRAP',$C8 ;        $D5
0974  C8
0975  53 50                  .BYT 'SPRIT',$C5 ;       $D6
097A  C5
097B  4F 46                  .BYT 'OF',$C6   ;        $D7
097D  C6
097E  50 4C                  .BYT 'PLAC',$C5 ;        $D8
0982  C5
0983  50 4F                  .BYT 'POLYGO',$CE ;      $D9
0989  CE
098A  4F 52                  .BYT 'ORIGI',$CE ;       $DA
098F  CE
0990  44 46                  .BYT 'DFIL',$CC ;        $DB
0994  CC
0995  50 50                  .BYT 'PPLO',$D4 ;        $DC
0999  D4
099A  50 44                  .BYT 'PDRA',$D7 ;        $DD
099E  D7
099F  50 4F                  .BYT 'PORIGI',$CE ;      $DE
09A5  CE
09A6  53 48                  .BYT 'SHAP',$C5 ;        $DF
09AA  C5
09AB  4D 41                  .BYT 'MA',$D4   ;        $E0
09AD  D4
09AE  47 53                  .BYT 'GSAV',$C5 ;        $E1
09B2  C5
09B3  47 4C                  .BYT 'GLOA',$C4 ;        $E2
09B7  C4
09B8  43 49                  .BYT 'CIRCL',$C5 ;       $E3
09BD  C5
09BE  53 43                  .BYT 'SCREE',$CE ;       $E4
09C3  CE
09C4  42 4F                  .BYT 'BORDE',$D2 ;       $E5
09C9  D2
09CA  50 4F                  .BYT 'POIN',$D4 ;        $E6
09CE  D4
09CF  00                     .BYT 0
09D0                ;
09D0  FD 0C         CADDR    .WOR R00001-1
09D2  E7 0F                  .WOR R00002-1
09D4  51 0E                  .WOR NORM-1
09D6  3C 11                  .WOR R00004-1
09D8  52 15                  .WOR R00005-1
09DA  04 0C                  .WOR WINDOW-1
09DC  72 16                  .WOR R00007-1
09DE  9B 0D                  .WOR CLRMEM-1
09E0  39 0E                  .WOR CLRCOL-1
09E2  4B 0D                  .WOR R00008-1
09E4  0A 2C                  .WOR SPRITE-1
09E6  A8 2B                  .WOR OFF-1
09E8  B9 2B                  .WOR PLACE-1
09EA  8B 19                  .WOR POLYGN-1
09EC  83 0E                  .WOR ORIGIN-1
09EE  25 18                  .WOR DFILL-1
09F0  19 2B                  .WOR PPLOT-1
09F2  5E 2B                  .WOR PDRAW-1
09F4  0E 29                  .WOR PVIEW-1
09F6  A6 25                  .WOR SHAPE-1
09F8  E0 1E                  .WOR MAT-1
09FA  D9 2C                  .WOR GSAVE-1
09FC  58 2D                  .WOR GLOAD-1
09FE  A0 1C                  .WOR CIRCLE-1
```

```
LOC   CODE        LINE

0A00  A7 0E               .WOR SCREEN-1
0A02  B1 0E               .WOR BORDER-1
0A04  07 AF               .WOR $AF07
0A06              ;
0A06              OFFT    =$D7
0A06              PNTTK   =$E6
0A06              ;
0A06  48          LOAD    PHA
0A07  20 2C 0A            JSR DISAB        ;DISABLE RASTER
0A0A  68                  PLA
0A0B  20 A5 F4            JSR $F4A5        ;LOAD
0A0E  4C 19 0A            JMP IOEXIT       ;RE-ENABLE RASTER
0A11              ;
0A11  48          SAVE    PHA
0A12  20 2C 0A            JSR DISAB DISABLE RASTER
0A15  68                  PLA
0A16  20 ED F5            JSR $F5ED        ;SAVE
0A19  08          IOEXIT PHP                ;SAVE ALL REGISTERS
0A1A  48                  PHA
0A1B  98                  TYA
0A1C  48                  PHA
0A1D  8A                  TXA
0A1E  48                  PHA
0A1F  20 4D 0A            JSR ENAB         ;ENABLE RASTER
0A22  68                  PLA              ;RESET ALL REGISTERS
0A23  AA                  TAX
0A24  68                  PLA
0A25  A8                  TAY
0A26  68                  PLA
0A27  28                  PLP
0A28  60                  RTS
0A29              ;
0A29  00          WINTMP .BYT 0
0A2A  00          ENTMP   .BYT 0
0A2B  00          TXTTMP .BYT 0
0A2C              ;
0A2C  AD 22 0C    DISAB   LDA WINFLG        ;STORE WINDOW FLAG
0A2F  8D 29 0A            STA WINTMP
0A32  AD CA 2C            LDA ENABLE       ;STORE SPRITE ENABLE
0A35  8D 2A 0A            STA ENTMP
0A38  AD 51 0E            LDA TXTFLG       ;STORE GRAPHICS FLAG
0A3B  8D 2B 0A            STA TXTTMP
0A3E  A9 00               LDA #$00
0A40  8D 23 0C            STA WNTFLG       ;DISABLE WINDOW
0A43  8D CA 2C            STA ENABLE       ;DISABLE SPRITES
0A46  8D 15 D0            STA $D015
0A49  20 6E 0A            JSR DISRAS       ;ENABLE NORMAL IRQ
0A4C  60                  RTS
0A4D              ;
0A4D  78          ENAB    SEI
0A4E  20 76 08            JSR SETKER       ;ENABLE RASTER
0A51  58                  CLI
0A52  AD 29 0A            LDA WINTMP       ;RESET WINDOW FLAG
0A55  8D 23 0C            STA WNTFLG
0A58  AD 2A 0A            LDA ENTMP        ;RESET SPRITES
0A5B  8D CA 2C            STA ENABLE
0A5E  AD 2B 0A            LDA TXTTMP
0A61  8D 51 0E            STA TXTFLG
0A64  D0 01               BNE ENAB1
0A66  60                  RTS
0A67  AD 2A 0A    ENAB1   LDA ENTMP
0A6A  8D 15 D0            STA $D015
0A6D  60                  RTS
0A6E              ;
0A6E  78          DISRAS SEI
0A6F  A9 EA               LDA #$EA         ;IRQ VECTOR TO $EA31
```

```
LOC    CODE         LINE

0A71   8D 15 03              STA $0315
0A74   A9 31                 LDA #$31
0A76   8D 14 03              STA $0314
0A79   AD 1A D0              LDA $D01A      ;DISABLE RASTER IRQ
0A7C   29 FE                 AND #$FE
0A7E   8D 1A D0              STA $D01A
0A81   A9 9F                 LDA #$9F       ;RE-START KEY TIMER
0A83   8D 0D DD              STA $DD0D
0A86   8D 0D DC              STA $DC0D
0A89   AD 0D DD              LDA $DD0D      ;CLEAR KEY IRQ
0A8C   AD 0D DC              LDA $DC0D
0A8F   58                    CLI
0A90   60                    RTS
0A91                         .END
```

2.3 Crunch to tokens

This routine is wedged into the crunch token link at locations $Ø3Ø4–$Ø3Ø5 (772–773). Crunch to tokens will take the input line and convert all command words to a one byte token value. This does exactly the same as the original Basic version except that the extended keyword table is checked before the normal Basic table.

Crunch to tokens is performed directly after the warm start routine encounters a carriage return, no matter whether the command is in direct mode or for entering or deleting a line in memory.

```
LOC    CODE         LINE

0A91                         .LIB CRUNCH-TOKEN
0A91                ; CRUNCH KEYWORD LINK
0A91                ;
0A91                ;
0A91                ;
0A91                ;
0A91   A6 7A        CRNCHT LDX $7A
0A93   A0 04               LDY #$04
0A95   84 0F               STY $0F
0A97   BD 00 02     CRNC01 LDA $0200,X     ;GET CHAR
0A9A   10 07               BPL CRNC02     ;CHAR IS OK
0A9C   C9 FF               CMP #$FF       ;PI?
0A9E   F0 2B               BEQ CRNC08     ;YES,SEND IT
0AA0   E8                  INX            ;NO, ILLEGAL CHAR
0AA1   D0 F4               BNE CRNC01     ; SO DO NEXT
0AA3                ;
0AA3   C9 20        CRNC02 CMP #$20        ;SPACE?
0AA5   F0 24               BEQ CRNC08     ;YES, SEND IT
0AA7   85 08               STA $08
0AA9   C9 22               CMP #$22       ;QUOTES?
0AAB   F0 45               BEQ CRNC12     ;YES, SCAN QUOTE END
0AAD   24 0F               BIT $0F
0AAF   70 1A               BVS CRNC08     ;SEND CHAR
0AB1   C9 3F               CMP #$3F       ;'?' ?
0AB3   D0 04               BNE CRNC03     ;NO
0AB5   A9 99               LDA #$99       ;SET TO PRINT TOKEN
0AB7   D0 12               BNE CRNC08     ;SEND IT
0AB9                ;
```

```
LOC    CODE        LINE

0AB9   C9 30       CRNC03 CMP #$30        ;<0 ?
0ABB   90 04              BCC CRNC04     ;YES, HUNT FOR KEYWORD
0ABD   C9 3C              CMP #$3C       ;< '<' ?
0ABF   90 0A              BCC CRNC08     ;YES, SEND CHAR
0AC1   4C 03 0B    CRNC04 JMP CRNC15      ;HUNT FOR KEYWORD
0AC4               ;
0AC4   05 0B       CRNC05 ORA $0B         ;ONE OF BASIC'S
0AC6   2C                 .BYT $2C
0AC7   A5 0B       CRNC06 LDA $0B
0AC9   A4 71       CRNC07 LDY $71         ;RESTORE Y
0ACB   E8          CRNC08 INX             ;NEXT POSITION
0ACC   C8                 INY
0ACD   99 FB 01           STA $01FB,Y    ;STORE IT
0AD0   C9 EE              CMP #$EE       ;MINE?
0AD2   B9 FB 01           LDA $01FB,Y    ;NO, END OF INPUT?
0AD5   F0 22              BEQ CRNC13     ;YES
0AD7   38                 SEC
0AD8   E9 3A              SBC #$3A       ;':' ?
0ADA   F0 04              BEQ CRNC09     ;YES
0ADC   C9 49              CMP #$49       ;DATA?
0ADE   D0 02              BNE CRNC10     ;NO
0AE0   85 0F       CRNC09 STA $0F
0AE2   38          CRNC10 SEC
0AE3   E9 55              SBC #$55       ;REM?
0AE5   D0 B0              BNE CRNC01     ;NO DO NEXT CHAR
0AE7   85 08              STA $08        ;SET QUOTE FLAG
0AE9   BD 00 02    CRNC11 LDA $0200,X     ;GET BYTE
0AEC   F0 DD              BEQ CRNC08     ;END OF INPUT, SEND
0AEE   C5 08              CMP $08        ;QUOTE FLAG?
0AF0   F0 D9              BEQ CRNC08     ;YES, SEND
0AF2   C8          CRNC12 INY             ;STORE CHAR
0AF3   99 FB 01           STA $01FB,Y
0AF6   E8                 INX
0AF7   D0 F0              BNE CRNC11     ;DO NEXT
0AF9               ;
0AF9   99 FD 01    CRNC13 STA $01FD,Y     ;STORE ZERO
0AFC   C6 7B              DEC $7B
0AFE   A9 FF              LDA #$FF
0B00   85 7A              STA $7A
0B02   60                 RTS            ;EXIT CRUNCH
0B03               ;
0B03   84 71       CRNC15 STY $71         ;SAVE OFF Y
0B05   A0 FF              LDY #$FF
0B07   86 7A              STX $7A        ; AND X POINTERS
0B09   CA                 DEX
0B0A   A9 CC              LDA #$CC       ;START TOKEN VAL=$CC
0B0C   85 0B              STA $0B
0B0E   C8          CRNC16 INY
0B0F   E8                 INX
0B10   BD 00 02    CRNC17 LDA $0200,X     ;GET BYTE
0B13   38                 SEC
0B14   F9 4B 09           SBC CLIST,Y    ;AS KEYWORD TABLE?
0B17   F0 F5              BEQ CRNC16     ;YES, CHECK NEXT
0B19   C9 80              CMP #$80       ;SHIFT OUT?
0B1B   F0 AA              BEQ CRNC06     ;YES, FOUND
0B1D   A6 7A              LDX $7A        ;RESTORE BUFFER POINTER
0B1F   E6 0B              INC $0B        ;NEXT TOKEN
0B21   C8          CRNC18 INY
0B22   B9 4A 09           LDA CLIST-1,Y  ;END OF KEYWORD?
0B25   10 FA              BPL CRNC18     ;NO
0B27   B9 4B 09           LDA CLIST,Y    ;END OF TABLE?
0B2A   D0 E4              BNE CRNC17     ;NO, CHECK NEXT
0B2C   A0 00              LDY #$00       ;START TOKEN AT 0
0B2E   84 0B              STY $0B        ;FOR BASIC
0B30   88                 DEY
0B31   A6 7A              LDX $7A        ;GET INPUT POINTER
```

```
LINE# LOC   CODE          LINE

0B33  CA                  DEX
0B34  C8           CRNC19 INY
0B35  E8                  INX
0B36  BD 00 02     CRNC20 LDA $0200,X     ;GET BYTE
0B39  38                  SEC
0B3A  F9 9E A0            SBC $A09E,Y     ;AS IN TABLE?
0B3D  F0 F5               BEQ CRNC19      ;YES, CHECK NEXT
0B3F  C9 80               CMP #$80        ;SHIFT OUT?
0B41  D0 03               BNE CRNC21      ;NO, TRY NEXT WORD
0B43  4C C4 0A            JMP CRNC05      ;YES, SEND BASIC TOKEN
0B46  A6 7A        CRNC21 LDX $7A         ;RESTORE INPUT POINTER
0B48  E6 0B               INC $0B         ;NEXT TOKEN
0B4A  C8           CRNC22 INY
0B4B  B9 9D A0            LDA $A09D,Y     ;END OF WORD?
0B4E  10 FA               BPL CRNC22      ;NO
0B50  B9 9E A0            LDA $A09E,Y     ;END OF TABLE?
0B53  D0 E1               BNE CRNC20      ;NO, TRY NEXT WORD
0B55  BD 00 02            LDA $0200,X     ;ELSE SEND BYTE
0B58  4C C9 0A            JMP CRNC07
0B5B                      .END
```

2.4 Tokens to text

This routine is wedged into the print token link at locations $Ø3Ø6–$Ø3Ø7 (774–775). Tokens to text is used in the list command only to convert any token value (greater than 127) back into the command word and print it to the output device.

```
LOC   CODE          LINE

0B5B                       .LIB PRINT-TOKEN
0B5B                ; PRINT TOKENS LINK
0B5B                ;
0B5B                ;
0B5B                ;
0B5B                ;
0B5B  30 03         PRINT  BMI PRIN02     ;A TOKEN
0B5D  4C F3 A6      PRIN01 JMP $A6F3      ;PRINT IT
0B60  C9 FF         PRIN02 CMP #$FF       ;IS IT PI?
0B62  F0 F9                BEQ PRIN01     ;YES
0B64  24 0F                BIT $0F        ;QUOTES?
0B66  30 F5                BMI PRIN01     ;YES
0B68  C9 CC                CMP #$CC       ;ONE OF MINE?
0B6A  B0 05                BCS PRIN08     ;DO MINE
0B6C  20 96 0B             JSR PRIN09     ;DO BASIC
0B6F  30 03                BMI PRIN13     ;ALWAYS
0B71  20 77 0B      PRIN08 JSR PRIN03     ;DO MINE
0B74  4C EF A6      PRIN13 JMP $A6EF      ;AND NEXT
0B77                ;
0B77  38            PRIN03 SEC
0B78  E9 CB                SBC #$CB
0B7A  AA                   TAX
0B7B  84 49                STY $49        ;SAVE Y
0B7D  A0 FF                LDY #$FF
0B7F  CA            PRIN04 DEX
0B80  F0 08                BEQ PRIN06     ;FOUND IT
0B82  C8            PRIN05 INY
0B83  B9 4B 09             LDA CLIST,Y    ;GET CHAR FROM TABLE
```

```
LOC   CODE       LINE

0B86  10 FA            BPL PRIN05     ;UNTIL END OF WORD
0B88  30 F5            BMI PRIN04     ;FOUND END OF WORD
0B8A  C8         PRIN06 INY
0B8B  B9 4B 09         LDA CLIST,Y    ;GET CHAR FROM TABLE
0B8E  30 05            BMI PRIN07     ;LAST CHAR OF WORD
0B90  20 D2 FF         JSR $FFD2      ;PRINT IT
0B93  D0 F5            BNE PRIN06     ;NEXT CHAR
0B95  60         PRIN07 RTS            ;DO LAST
0B96             ;
0B96  38         PRIN09 SEC
0B97  E9 7F            SBC #$7F       ;REMOVE SHIFT
0B99  AA               TAX
0B9A  84 49            STY $49        ;SAVE .Y
0B9C  A0 FF            LDY #$FF
0B9E  CA         PRIN10 DEX
0B9F  F0 08            BEQ PRIN12     ;FOUND IT
0BA1  C8         PRIN11 INY
0BA2  B9 9E A0         LDA $A09E,Y    ;GET CHAR FROM TABLE
0BA5  10 FA            BPL PRIN11     ;UNTIL END OF WORD
0BA7  30 F5            BMI PRIN10     ;FOUND END OF WORD
0BA9  C8         PRIN12 INY
0BAA  B9 9E A0         LDA $A09E,Y    ;GET CHAR FROM TABLE
0BAD  30 E6            BMI PRIN07     ;LAST CHAR OF WORD
0BAF  20 D2 FF         JSR $FFD2      ;PRINT CHAR
0BB2  D0 F5            BNE PRIN12     ;ALWAYS
0BB4                   .END
```

2.5 Execute statement

This routine is wedged into the start new Basic code link at locations \$Ø3Ø8–\$Ø3Ø9 (776–777). This is the control part of the main Basic interpreter loop. It takes a token value and executes the routine via the vector table in the initialisation file.

```
LOC   CODE       LINE

0BB4                   .LIB HANDLE-TOKEN
0BB4             ; EXECUTE STATEMENT LINK
0BB4             ;
0BB4             ;
0BB4             ;
0BB4             ;
0BB4  20 73 00   HANDLE JSR $0073       ;GET CODE
0BB7  C9 CC            CMP #$CC        ;IS IT MY TOKEN?
0BB9  B0 06            BCS HAND01      ;YES, DO IT
0BBB  20 79 00         JSR $0079       ;GET CURRENT CHAR
0BBE  4C E7 A7         JMP $A7E7       ;DO BASIC CODE
0BC1             ;
0BC1  20 C7 0B   HAND01 JSR HAND02      ;EXECUTE THE CODE
0BC4  4C AE A7         JMP $A7AE       ;AND NEXT
0BC7             ;
0BC7  E9 CC      HAND02 SBC #$CC
0BC9  0A               ASL A           ;TIMES 2
0BCA  A8               TAY
0BCB  B9 D1 09         LDA CADDR+1,Y   ;GET HI BYTE
0BCE  48               PHA             ;TO STACK
0BCF  B9 D0 09         LDA CADDR,Y     ;GET LO BYTE
0BD2  48               PHA             ;TO STACK
0BD3  4C 73 00         JMP $0073       ;EXECUTE IT
0BD6                   .END
```

2.6 Execute arithmetic

This routine is wedged into the arithmetic link at locations $Ø3ØA–$Ø3ØB (778–779). This routine is called by the evaluate expression and transfers control to the arithmetic routine included in this package. If the extended Basic command is not 'POINT', Syntax error is output.

```
LOC   CODE        LINE

0BD6                      .LIB ARITH-TOKEN
0BD6              ; ARITHMETIC LINK
0BD6              ;
0BD6              ;
0BD6              ;
0BD6              ;
0BD6  A9 00       ARITH  LDA #$00        ;TYPE FLAG TO NUMERIC
0BD8  85 0D              STA $0D
0BDA  20 73 00           JSR $0073      ;GET BYTE
0BDD  C9 FF              CMP #$FF       ;IS IT PI?
0BDF  F0 04              BEQ ARITH3
0BE1  C9 CC              CMP #$CC       ;ONE OF MINE?
0BE3  B0 06              BCS ARITH1     ;YES
0BE5  20 79 00    ARITH3 JSR $0079       ;GET CURRENT CHAR
0BE8  4C 8D AE           JMP $AE8D      ;OPERATE
0BEB              ;
0BEB  C9 E6       ARITH1 CMP #PNTTK      ;POINT?
0BED  F0 03              BEQ ARITH2     ;YES
0BEF  4C 08 AF           JMP $AF08      ;'SYNTAX ERROR'
0BF2              ;
0BF2  A9 AD       ARITH2 LDA #$AD        ;SETUP RETURN ADDRESS
0BF4  48                 PHA
0BF5  A9 8C              LDA #$8C
0BF7  48                 PHA
0BF8  AD 04 0C           LDA FNADDR+1   ;GET HI BYTE
0BFB  48                 PHA
0BFC  AD 03 0C           LDA FNADDR     ;GET LO BYTE
0BFF  48                 PHA
0C00  4C 73 00           JMP $0073      ;EXECUTE FUNCTION
0C03  F5 15       FNADDR .WOR POINTC-1
0C05                     .END
```

2.7 IRQ wedge

This wedge replaces the normal IRQ wedge thereby allowing the use of raster interrupts and kernal switching as well as all the normal IRQ functions e.g. cursor flashing etc. It should be noted that the interrupt frequency has been changed from 6Ø Hz to 5Ø Hz so that it matches raster interrupts. During LOAD and SAVE the normal IRQ vectors are restored. (The routine is included in the WINDOW listing in the next section.)

2 SCREEN MANAGEMENT ROUTINES

WINDOW

Abbreviated entry: W(shift)I

Affected Basic abbreviations: None

Token: Hex $D1 Decimal 2Ø9

Purpose: To enable or disable a text window at the bottom of the graphics screen.

Syntax: WINDOW ON or
WINDOW OFF

Errors: Syntax error – if the command following WINDOW is neither ON nor OFF

Use: WINDOW allows the concurrent display of high resolution graphics and text. Its principal use is when the user is required to input data from the keyboard whilst a high resolution screen is displayed, since the INPUT command will only generate a display onto a text screen. The window is four lines high and is only produced when the computer is in graphics mode and the window is enabled using the command WINDOW ON. The cursor is forced to lie in the four lines of the screen window area which thus behaves as a normal text screen.

Routine entry point: $ØCØ5

Routine operation: The command WINDOW simply sets a flag to indicate whether the window is active or not. The token following the WINDOW command is checked for 'ON' or 'OFF'. If neither of these is found, the command NORM is done followed by the output of Syntax error.

The routine to create the window runs on a raster interrupt which has two splits. The first split is at the top of the screen, and checks if both the graphics flag and the window flag are enabled. If both are enabled, the command GRAPH is done followed by the normal keyboard scanning. The second raster interrupt occurs at the top of the text window. If the flags are enabled the command NORM is done.

```
LOC   CODE        LINE

0C05                      .LIB WINDOW
0C05  C9 91       WINDOW CMP #$91       ;'ON'?
0C07  F0 0A              BEQ WIND1
0C09  C9 D7              CMP #OFFT
0C0B  F0 0C              BEQ WIND2
0C0D  20 52 0E           JSR NORM
0C10  4C 08 AF           JMP $AF08      ;SYNTAX ERROR
0C13  20 EF 0C    WIND1  JSR TSTCUR     ;FLAG 'ON'
0C16  A9 FF              LDA #$FF       ;FLAG 'ON'
0C18  2C                 .BYT $2C
0C19  A9 00       WIND2  LDA #$00       ;FLAG 'OFF'
```

```
0C1B  8D 23 0C          STA WNTFLG
0C1E  20 73 00          JSR $0073
0C21  60                RTS
0C22             ;
0C22  00         WINFLG .BYT 0
0C23  00         WNTFLG .BYT 0
0C24  00         BKFLG  .BYT 0
0C25             ;
0C25  48         IRQINT PHA
0C26  8A                TXA
0C27  48                PHA
0C28  98                TYA
0C29  48                PHA
0C2A  A5 01             LDA $01
0C2C  8D 24 0C          STA BKFLG       ;BASIC/KERNAL FLAG
0C2F  A9 37             LDA #$37
0C31  85 01             STA $01
0C33  BA                TSX
0C34  BD 04 01          LDA $0104,X     ;BRK?
0C37  29 10             AND #$10
0C39  F0 03             BEQ IRQ1        ;NO
0C3B  6C 16 03          JMP ($0316)     ;YES
0C3E             ;
0C3E  20 52 0C   IRQ1   JSR DORAST
0C41  AD 24 0C          LDA BKFLG
0C44  85 01             STA $01
0C46  68         IRQEXT PLA
0C47  A8                TAY
0C48  68                PLA
0C49  AA                TAX
0C4A  68                PLA
0C4B  40                RTI
0C4C             ;
0C4C  20 52 0C   IRQVEC JSR DORAST
0C4F  4C 46 0C          JMP IRQEXT
0C52             ;
0C52  AD EE 0C   DORAST LDA RASFLG       ;WHICH RASTER?
0C55  F0 03             BEQ RAS217      ;TOP OF WINDOW
0C57  4C 7C 0C          JMP RAS000      ;TOP OF SCREEN
0C5A             ;
0C5A  AD 22 0C   RAS217 LDA WINFLG       ;WINDOW ENABLED?
0C5D  D0 02             BNE R217ON      ;YES
0C5F  F0 0B             BEQ R217DN
0C61  AD 51 0E   R217ON LDA TXTFLG       ;TEXT MODE?
0C64  F0 06             BEQ R217DN
0C66             ;
0C66  20 57 0E   R217GR JSR NORM1        ;SET TO TEXT
0C69  20 EF 0C          JSR TSTCUR
0C6C  A9 01      R217DN LDA #$01         ;SAY 'DONE'
0C6E  8D 19 D0          STA $D019
0C71  A9 00             LDA #$00        ;SET NEXT VAL
0C73  8D 12 D0          STA $D012
0C76  A9 01             LDA #$01
0C78  8D EE 0C          STA RASFLG
0C7B  60                RTS
0C7C             ;
0C7C  AD 22 0C   RAS000 LDA WINFLG       ;WINDOW ENABLED?
0C7F  D0 02             BNE R000ON      ;YES
0C81  F0 08             BEQ R000DN
0C83  AD 51 0E   R000ON LDA TXTFLG       ;TEXT MODE?
0C86  F0 03             BEQ R000DN
0C88             ;
0C88  20 5D 0D   R000GR JSR GRAPH        ;SET TO GRAPHICS
0C8B  A9 01      R000DN LDA #$01         ;SAY 'DONE'
0C8D  8D 19 D0          STA $D019
0C90  A9 D9             LDA #217        ;SET NEXT VAL
0C92  8D 12 D0          STA $D012
0C95  A9 00             LDA #$00
0C97  8D EE 0C          STA RASFLG
```

```
LOC   CODE       LINE

0C9A  AD 23 0C          LDA WNTFLG
0C9D  8D 22 0C          STA WINFLG
0CA0             ;
0CA0  20 EA FF          JSR $FFEA       ;UPDATE CLOCK
0CA3  A5 CC             LDA $CC         ;BLINKING CURSOR?
0CA5  D0 29             BNE L810        ;NO
0CA7  C6 CD             DEC $CD         ;TIME TO BLINK?
0CA9  D0 25             BNE L810        ;NO
0CAB  A9 14             LDA #$14        ;RESET BLINK COUNTER
0CAD  85 CD             STA $CD
0CAF  A4 D3             LDY $D3         ;CURSOR POSITION
0CB1  46 CF             LSR $CF         ;CARRY SET IF ORIG CHAR
0CB3  AE 87 02          LDX $0287       ;GET CHARS ORIG COLOUR
0CB6  B1 D1             LDA ($D1),Y     ;GET CHAR
0CB8  B0 11             BCS L745        ;NOT NEEDED
0CBA  E6 CF             INC $CF         ;SET TO 1
0CBC  85 CE             STA $CE         ;SAVE ORIG CHAR
0CBE  20 24 EA          JSR $EA24
0CC1  B1 F3             LDA ($F3),Y     ;GET ORIG COLOUR
0CC3  8D 87 02          STA $0287       ;SAVE IT
0CC6  AE 86 02          LDX $0286       ;BLINK IN THIS COLOUR
0CC9  A5 CE             LDA $CE         ;WITH ORIG CHAR
0CCB  49 80      L745   EOR #$80         ;BLINK IT
0CCD  20 1C EA          JSR $EA1C       ;DISPLAY IT
0CD0  A5 01      L810   LDA $01          ;GET CASSETTE SWITCH
0CD2  29 10             AND #$10        ;SWITCH DOWN?
0CD4  F0 0A             BEQ L809        ;YES
0CD6  A0 00             LDY #$00
0CD8  84 C0             STY $C0         ;CASSETTE OFF SWITCH
0CDA  A5 01             LDA $01
0CDC  09 20             ORA #$20
0CDE  D0 08             BNE L812
0CE0  A5 C0      L809   LDA $C0
0CE2  D0 06             BNE L813
0CE4  A5 01             LDA $01
0CE6  29 1F             AND #$1F        ;TURN MOTOR OFF
0CE8  85 01      L812   STA $01
0CEA  20 87 EA   L813   JSR $EA87        ;SCAN KEYBOARD
0CED  60                RTS
0CEE             ;
0CEE  00         RASFLG .BYT 0
0CEF             ;
0CEF  38         TSTCUR SEC
0CF0  20 F0 FF          JSR $FFF0       ;READ CURSOR POS
0CF3  E0 15             CPX #21         ;ABOVE WINDOW?
0CF5  B0 06             BCS TSTCR1      ;NO
0CF7  A2 15             LDX #21         ;SET TO TOP OF WINDOW
0CF9  18                CLC
0CFA  20 F0 FF          JSR $FFF0       ;SET CURSOR POS
0CFD  60         TSTCR1 RTS
0CFE                    .END
```

HIRES

Abbreviated entry: H(shift)I

Affected Basic abbreviations: None

Token: Hex $CC   Decimal 2Ø4

Purpose: To select which graphics plotting mode, screen and border colours.

Syntax: HIRES mode, screen [,border]

Errors: Syntax error – if the syntax is not as above
Illegal quantity – if any of the values are <Ø or >255

Use: The command HIRES sets the CBM 64 into graphics screen mode for the first time. The parameter 'mode' is a value to specify standard high resolution (Ø) or multicolour (non zero). The values screen and border (optional) are the equivalent values which would be POKEd into 53281 and 53280 to set the screen and border colours.

Routine entry point: $ØCFE

Routine operation: The values are read in and stored away. The mode flag is stored in location $Ø2 and tells the plotting routine in which mode to plot the points.

This routine also calls three other commands:

GRAPH to go to graphics mode.
CLG to clear the graphics screen.
CLC to clear the colour memory to black.

```
LOC   CODE        LINE


0CFE                     .LIB MODE
0CFE              ;
0CFE              ; ROUTINE TO SET UP HIRES SCREEN
0CFE              ;
0CFE  20 EB B7    R00001 JSR PARAMS      ; READ OFF PARAMETERS
0D01  8E 82 0E           STX SCTMP1     ; FOR MODE AND COLOUR
0D04  8E 83 0E           STX BDTMP1
0D07  20 79 00           JSR $0079
0D0A  F0 09              BEQ R00011
0D0C  20 F1 B7           JSR PARAM
0D0F  8E 83 0E           STX BDTMP1
0D12  AE 82 0E           LDX SCTMP1
0D15  A5 14       R00011 LDA $14         ; MODE IN LOC $14
0D17  85 02              STA MODE
0D19  F0 02              BEQ MULTI
0D1B  A2 00              LDX #$00
0D1D  20 9C 0D    MULTI  JSR CLRMEM      ; CLEAR HIRES SCREEN
0D20  20 27 0E           JSR CLRSCN     ; CLEAR VIDEO SCREEN
0D23  20 3A 0E           JSR CLRCOL     ; CLEAR COLOUR MEMORY
0D26              ;
0D26              ;SET DEFAULT ORIGINS
0D26              ;
0D26  A9 00              LDA #$00       ;Y TO ZERO
0D28  8D 4F 0E           STA YORIG
0D2B  8D 50 0E           STA YORIG+1
0D2E  8D FB 28           STA VX+1
0D31  8D FD 28           STA VY+1
0D34  8D FF 28           STA VZ+1
0D37  8D 4D 0E           STA XORIG
0D3A  8D 4E 0E           STA XORIG+1
0D3D  A9 A0              LDA #160       ;3D X TO 160
0D3F  8D FA 28           STA VX
0D42  A9 64              LDA #100       ;3D Y TO 100
0D44  8D FC 28           STA VY
0D47  A9 C8              LDA #200       ;3D Z TO 200
0D49  8D FE 28           STA VZ
```

GRAPH

Abbreviated entry: G(shift)R

Affected Basic abbreviations: None

Purpose: To switch from the text screen to the graphics screen.

Syntax: GRAPH

Errors: None

Use: GRAPH is used to switch from the text display screen to the graphics display screen. When called, the screen and border values specified in HIRES are stored in the correct locations in the VIC chip and the graphics flag is set.

Routine entry point: $ØD4C

Routine operation: Upon entry to this routine, the graph flag is set and the current screen and border settings are stored away for NORM. The graphics screen and border values are then set and the graphics mode is selected.

```
LOC   CODE       LINE

0D4C             ;GRAPH COMMAND ENTRY
0D4C             ;
0D4C  A9 FF      R00008 LDA #$FF        ;FLAG GRAPH MODE
0D4E  8D 51 0E          STA TXTFLG
0D51  AD 20 D0          LDA $D020
0D54  8D 81 0E          STA BDTMP
0D57  AD 21 D0          LDA $D021
0D5A  8D 80 0E          STA SCTMP
0D5D  AD CA 2C   GRAPH  LDA ENABLE
0D60  8D 15 D0          STA VIC+21
0D63  A9 3B             LDA #$3B
0D65  8D 11 D0          STA $D011       ; SELECT BIT MAP MODE
0D68  A9 0D             LDA #$0D
0D6A  8D 18 D0          STA $D018       ; CHOOSE HIRES SCREEN
0D6D  A5 02             LDA MODE
0D6F  D0 07             BNE SETMUL      ; IF MODE=0
0D71  A9 C8             LDA #$C8
0D73  8D 16 D0          STA $D016
0D76  D0 05             BNE DONE
0D78  A9 D8      SETMUL LDA #$D8
0D7A  8D 16 D0          STA $D016       ; ELSE SET MULTI MODE
0D7D  AD 02 DD   DONE   LDA $DD02        ; SELECT BANK 2 FOR HIRES
0D80  09 03             ORA #$03        ; SCREEN
0D82  8D 02 DD          STA $DD02
0D85  AD 00 DD          LDA $DD00
0D88  29 FC             AND #$FC
0D8A  09 00             ORA #$00
0D8C  8D 00 DD          STA $DD00
0D8F  AD 82 0E          LDA SCTMP1      ;SCREEN & BORDER
0D92  8D 21 D0          STA $D021       ; COLOURS
0D95  AD 83 0E          LDA BDTMP1
0D98  8D 20 D0          STA $D020
0D9B  60                RTS
```

CLG

Abbreviated entry: C(shift)L

Affected Basic abbreviations: CLR - CLR

Token: Hex $D3 Decimal 211

Purpose: To clear the graphics screen.

Syntax: CLG

Errors: None

Use: CLG is used to clear the 8K of RAM behind the kernal ROM thus wiping the graphics screen clean. This routine also sets up the IRQ and NMI vectors for use when the kernal is switched out.

Routine entry point: $ØD9C

Routine operation: The routine simply stores zero in every location of the RAM from $EØØØ to $FFFF and then stores the vectors to point to IRQ and NMI.

```
LOC   CODE       LINE

0D9C             ;CLG COMMAND ENTRY
0D9C             ;
0D9C  A0 00      CLRMEM LDY #$00          ; LOOP TO CLEAR HIRES
0D9E  98                TYA
0D9F  99 00 E0   LOOP   STA $E000,Y
0DA2  99 00 E1          STA $E100,Y
0DA5  99 00 E2          STA $E200,Y
0DA8  99 00 E3          STA $E300,Y
0DAB  99 00 E4          STA $E400,Y
0DAE  99 00 E5          STA $E500,Y
0DB1  99 00 E6          STA $E600,Y
0DB4  99 00 E7          STA $E700,Y
0DB7  99 00 E8          STA $E800,Y
0DBA  99 00 E9          STA $E900,Y
0DBD  99 00 EA          STA $EA00,Y
0DC0  99 00 EB          STA $EB00,Y
0DC3  99 00 EC          STA $EC00,Y
0DC6  99 00 ED          STA $ED00,Y
0DC9  99 00 EE          STA $EE00,Y
0DCC  99 00 EF          STA $EF00,Y
0DCF  99 00 F0          STA $F000,Y
0DD2  99 00 F1          STA $F100,Y
0DD5  99 00 F2          STA $F200,Y
0DD8  99 00 F3          STA $F300,Y
0DDB  99 00 F4          STA $F400,Y
0DDE  99 00 F5          STA $F500,Y
0DE1  99 00 F6          STA $F600,Y
0DE4  99 00 F7          STA $F700,Y
0DE7  99 00 F8          STA $F800,Y
0DEA  99 00 F9          STA $F900,Y
0DED  99 00 FA          STA $FA00,Y
0DF0  99 00 FB          STA $FB00,Y
0DF3  99 00 FC          STA $FC00,Y
0DF6  99 00 FD          STA $FD00,Y
0DF9  99 00 FE          STA $FE00,Y
0DFC  99 00 FF          STA $FF00,Y
```

```
LOC    CODE          LINE

0DFF   88                     DEY
0E00   D0 9D                  BNE LOOP
0E02   A9 40                  LDA #$40
0E04   8D F9 FF               STA $FFF9
0E07   A9 60                  LDA #$60
0E09   8D F8 FF               STA $FFF8
0E0C   A9 F9                  LDA #$F9
0E0E   8D FA FF               STA $FFFA
0E11   8D FD FF               STA $FFFD
0E14   8D FB FF               STA $FFFB
0E17   A9 F8                  LDA #$F8
0E19   8D FC FF               STA $FFFC
0E1C   A9 25                  LDA #<IRQINT
0E1E   8D FE FF               STA $FFFE
0E21   A9 0C                  LDA #>IRQINT
0E23   8D FF FF               STA $FFFF
0E26   60                     RTS
0E27   A0 00         CLRSCN   LDY #$00          ; LOOP TO CLEAR
0E29   8A                     TXA               ;VIDEO SCREEN
0E2A   99 00 C0      LOOP1    STA $C000,Y
0E2D   99 00 C1               STA $C100,Y
0E30   99 00 C2               STA $C200,Y
0E33   99 00 C3               STA $C300,Y
0E36   88                     DEY
0E37   D0 F1                  BNE LOOP1
0E39   60                     RTS
```

CLC

Abbreviated entry: CLC

Affected Basic abbreviations: None

Token: Hex $D3 Decimal 211

Purpose: To set all locations of the colour memory to black.

Syntax: CLC

Errors: None

Use: CLC is used by the HIRES command to set all locations of the colour memory to zero. It can be used to reset text colours after using multicolour mode.

Routine entry point: $ØE3A

Routine operation: The routine simply stores zero in every location of RAM from $D8ØØ to $DBFF.

```
LOC    CODE          LINE

0E3A                 ;CLC COMMAND ENTRY
0E3A                 ;
0E3A   A0 00         CLRCOL LDY #$00          ; LOOP TO CLEAR
0E3C   98                   TYA               ; COLOUR MEMORY.
```

```
LOC    CODE        LINE

0E3D   99 00 D8    LOOP2  STA $D800,Y
0E40   99 00 D9           STA $D900,Y
0E43   99 00 DA           STA $DA00,Y
0E46   99 00 DB           STA $DB00,Y
0E49   88                 DEY
0E4A   D0 F1              BNE LOOP2
0E4C   60                 RTS
0E4D   00 00       XORIG  .WOR 0
0E4F   00 00       YORIG  .WOR 0
0E51   00          TXTFLG .BYT 0
0E52                      .END
```

NORM

Abbreviated entry: N(shift)O

Affected Basic abbreviations: NOT – NOT

Token: Hex $CE   Decimal 2Ø6

Purpose: To switch from the graphics screen to the text screen.

Syntax: NORM

Errors: None

Use: NORM is used to restore output to the text screen. This is useful if an error has occurred in a Basic program.

Routine entry point: $ØE52

Routine operation: NORM is one of the simplest commands. It simply resets the screen to normal display mode, flags the text mode, disables sprites, and restores the text screen and border colour values.

```
LOC    CODE        LINE

0E52                      .LIB NORM
0E52               ;
0E52               ; ROUTINE TO RETURN TO NORMAL SCREEN
0E52               ;
0E52   A9 00       NORM   LDA #$00        ;FLAG TEXT MODE
0E54   8D 51 0E           STA TXTFLG
0E57   A9 00       NORM1  LDA #$00
0E59   8D 15 D0           STA VIC+21
0E5C   AD 02 DD           LDA $DD02
0E5F   29 FC              AND #$FC
0E61   8D 02 DD           STA $DD02       ; BACK TO BANK 0
0E64   A9 1B              LDA #$1B
0E66   8D 11 D0           STA $D011       ; BIT MAP MODE OFF
0E69   A9 C8              LDA #$C8
0E6B   8D 16 D0           STA $D016       ; MULTCOLOUR OFF
0E6E   A9 15              LDA #$15
0E70   8D 18 D0           STA $D018       ; NORMAL SCREEN
0E73   AD 80 0E           LDA SCTMP       ;SET SCREEN COLOUR
0E76   8D 21 D0           STA $D021
0E79   AD 81 0E           LDA BDTMP       ;SET BORDER COLOUR
```

```
LOC   CODE        LINE

0E7C  8D 20 D0           STA $D020
0E7F  60                 RTS
0E80  00          SCTMP  .BYT 0
0E81  00          BDTMP  .BYT 0
0E82  00          SCTMP1 .BYT 0
0E83  00          BDTMP1 .BYT 0
0E84                     .END
```

ORIGIN

Abbreviated entry: O(shift)R

Affected Basic abbreviations: None

Token: Hex $DA Decimal 218

Purpose: To set the origin position for 2 dimensional plotting.

Syntax: ORIGIN XO, YO

Errors: Illegal quantity - if either of the values XO, YO are >32767 or <−32768

Use: ORIGIN is used to set the screen coordinates relative to which all points will be plotted. The coordinates specified are with respect to Ø,Ø (the bottom left of the screen). This position will then be taken in all 2D plotting commands as the new origin axis, position Ø,Ø. The origin is set to Ø,Ø at the bottom left of the screen when the command HIRES is performed.

Routine entry point: $ØE84

Routine operation: The integer values XO and YO are read in and stored away to be added to the X and Y coordinates of all 2D plotting routines.

```
LOC   CODE        LINE

0E84                     .LIB ORIGIN
0E84  20 8A AD    ORIGIN JSR $AD8A      ;GET X
0E87  20 BF B1           JSR $B1BF      ;FIX IT
0E8A  A5 65              LDA $65
0E8C  8D 4D 0E           STA XORIG
0E8F  A5 64              LDA $64
0E91  8D 4E 0E           STA XORIG+1
0E94  20 FD AE           JSR $AEFD      ;CHECK ','
0E97  20 8A AD           JSR $AD8A      ;GET Y
0E9A  20 BF B1           JSR $B1BF      ;FIX IT
0E9D  A5 65              LDA $65
0E9F  8D 4F 0E           STA YORIG
0EA2  A5 64              LDA $64
0EA4  8D 50 0E           STA YORIG+1
0EA7  60                 RTS
0EA8                     .END
```

SCREEN and BORDER

Abbreviated entry: SCREEN: S(shift)C
BORDER: B(shift)O

Affected Basic abbreviations: None

Token: SCREEN: Hex $E4 Decimal 228
BORDER: Hex $E5 Decimal 229

Purpose: To change the screen colour or border colour.

Syntax: SCREEN=col
BORDER=col

Errors: Illegal quantity - if col <Ø or >15

Use: SCREEN and BORDER replace the POKEs 5328Ø and 53281 to change colours. The values are also stored for use in the NORM command. When used with the graphics screen enabled and the window off, the commands will affect the background and border colours of the graphics screen, but GRAPH will reset the original values.

Routine entry point: SCREEN: $ØEA8
BORDER: $ØEB2

Routine operation: The value is read in and checked for range. If it is in range, the value is stored to both the VIC chip register and the temporary store for text screen.

```
LOC     CODE         LINE

0EA8                        .LIB SCREEN/BORDER
0EA8    20 BC 0E     SCREEN JSR GSBCOL       ;GET SCREEN COLOUR
0EAB    8E 21 D0            STX $D021       ;STORE IT
0EAE    8E 80 0E            STX SCTMP
0EB1    60                  RTS
0EB2                 ;
0EB2    20 BC 0E     BORDER JSR GSBCOL       ;GET BORDER COLOUR
0EB5    8E 20 D0            STX $D020       ;STORE IT
0EB8    8E 81 0E            STX BDTMP
0EBB    60                  RTS
0EBC                 ;
0EBC    A9 B2        GSBCOL LDA #$B2         ;'='
0EBE    20 FF AE            JSR $AEFF       ;SCAN IT
0EC1    20 9E B7            JSR $B79E       ;GET VALUE
0EC4    E0 10               CPX #$10        ;>16?
0EC6    B0 01               BCS GSBCL1      ;YES
0EC8    60                  RTS
0EC9    A2 0E        GSBCL1 LDX #$0E         ;'ILLEGAL QUANTITY'
0ECB    4C 37 A4            JMP $A437       ;SEND IT
0ECE                        .END
```

3 TWO DIMENSIONAL PLOTTING ROUTINES

In this section are the main plotting routines of the package. The routines in PLOT and DRAW are also used by the 3 dimensional routines after the perspective correction has been calculated.

The first routine is possibly the most important in the package (though it does not have a command name). This routine calculates the position in the bit map memory of a given X,Y screen coordinate.

Entry conditions:
Locations $59,$5A hold the X coordinate in signed integer format.
Locations $5B,$5C hold the Y coordinate in signed integer format.
Both of these values are scaled to the origin position before entry.

Exit conditions:
If the coordinate is plottable:
Location $57,$58 holds a two byte pointer to the byte in the bit map.
Location $5B,$5C holds the offset for the colour to be stored.
Location $5E holds the mask for the bit in the byte.
If the plotting mode is multicolour, then $5E and $5F hold the masks for the 2 bits in the byte.
Return with carry flag clear.
If the coordinate is not plottable, the routine exits with the carry flag set and none of the calculations are made.

Routine entry point: $ØECE

Routine operation: When called, this routine first checks to see if the coordinates specified are on the screen. If they are not, the carry flag is set and then exits.

If the coordinates are on the screen, these calculations are made (with the help of tables for speed):

$57,$58=$EØØØ+INT(Y/8)*32Ø+INT(X/8)*8+(Y AND 7)
$5B,$5C=INT(Y/8)*4Ø+INT(X/8)

In standard high resolution mode:

$5E=2↑(7−(X AND 7))

or in multicolour mode:

$5E=2↑(INT(7−(X AND 7)/2)*2)
$5F=2*$5E

The carry flag is cleared and the routine exits.

```
LOC    CODE          LINE

0ECE                          .LIB DOT
0ECE                 ;
0ECE                 ; ROUTINE TO CALCULATE LOCATION
```

```
LOC   CODE        LINE

0ECE              ;  AND BIT(S) FROM THE X AND Y
0ECE              ;  COORDINATES.
0ECE              ;
0ECE  A5 5A       DOT     LDA T2+1        ; CHECK THAT X AND Y
0ED0  C9 00               CMP #$00       ; ARE WITHIN BOUNDS
0ED2  F0 0C               BEQ XOK
0ED4  C9 01               CMP #$01
0ED6  D0 06               BNE XER
0ED8  A5 59               LDA T2
0EDA  C9 40               CMP #$40
0EDC  90 02               BCC XOK
0EDE  38          XER     SEC             ; TOO LARGE EXIT
0EDF  60                  RTS
0EE0  A5 5C       XOK     LDA T3+1
0EE2  D0 FA               BNE XER
0EE4  A5 5B               LDA T3
0EE6  C9 C8               CMP #$C8
0EE8  B0 F4               BCS XER
0EEA  A9 C7               LDA #199
0EEC  38                  SEC
0EED  E5 5B               SBC T3
0EEF  85 5B               STA T3
0EF1  A5 59               LDA T2         ;CALCULATE THE BIT TO
0EF3  29 07               AND #$07       ; BE PLOTTED AS
0EF5  85 5E               STA T5         ; 7-(X AND Y)
0EF7  A9 07               LDA #$07
0EF9  38                  SEC
0EFA  E5 5E               SBC T5
0EFC  85 5E               STA T5
0EFE  85 5F               STA T6
0F00  A5 02               LDA MODE       ;MULTI?
0F02  F0 0A               BEQ BITOK      ;NO
0F04  46 5E               LSR T5         ;T5=INT(T5/2)*2
0F06  06 5E               ASL T5
0F08  A5 5E               LDA T5
0F0A  85 5F               STA T6
0F0C  E6 5F               INC T6
0F0E  A6 5E       BITOK   LDX T5          ;CALCULATE 2↑T5
0F10  BD E0 0F            LDA TOP2X,X    ; AND 2↑T6
0F13  85 5E               STA T5
0F15  A6 5F               LDX T6
0F17  BD E0 0F            LDA TOP2X,X
0F1A  85 5F               STA T6
0F1C              ;
0F1C              ;CALCULATE INT(Y/8)*320
0F1C              ;AND STORE IN T1
0F1C              ;
0F1C  A5 5B               LDA T3
0F1E  4A                  LSR A
0F1F  4A                  LSR A
0F20  4A                  LSR A
0F21  0A                  ASL A
0F22  AA                  TAX
0F23  BD 7C 0F            LDA MUL320,X
0F26  85 57               STA T1
0F28  BD 7D 0F            LDA MUL320+1,X
0F2B  85 58               STA T1+1
0F2D              ;
0F2D              ;ADD Y AND 7 TO T1
0F2D              ;
0F2D  A5 5B               LDA T3
0F2F  29 07               AND #$07
0F31  18                  CLC
0F32  65 57               ADC T1
0F34  85 57               STA T1
0F36  90 02               BCC YND7OK
```

```
LOC    CODE          LINE

0F38   E6 58               INC T1+1
0F3A                 ;
0F3A                 ;CALCULATE INT(Y/8)*40
0F3A                 ;
0F3A   A5 5B         YND7OK LDA T3
0F3C   4A                  LSR A
0F3D   4A                  LSR A
0F3E   4A                  LSR A
0F3F   0A                  ASL A
0F40   AA                  TAX
0F41   BD AE 0F            LDA MUL40,X
0F44   85 5B               STA T3
0F46   BD AF 0F            LDA MUL40+1,X
0F49   85 5C               STA T3+1
0F4B                 ;
0F4B                 ;CALCULATE INT(X/8)
0F4B                 ;
0F4B   A0 03               LDY #$03
0F4D   46 5A         DIV8  LSR T2+1
0F4F   66 59               ROR T2
0F51   88                  DEY
0F52   D0 F9               BNE DIV8
0F54                 ;
0F54                 ;ADD INT(X/8) INTO T3
0F54                 ;
0F54   A5 5B               LDA T3
0F56   18                  CLC
0F57   65 59               ADC T2
0F59   85 5B               STA T3
0F5B   A5 5C               LDA T3+1
0F5D   65 5A               ADC T2+1
0F5F   85 5C               STA T3+1
0F61                 ;
0F61                 ;CALCULATE INT(X/8)*8
0F61                 ;
0F61   A0 03               LDY #$03
0F63   06 59         MUL8  ASL T2
0F65   26 5A               ROL T2+1
0F67   88                  DEY
0F68   D0 F9               BNE MUL8
0F6A                 ;
0F6A                 ;ADD INT(X/8)*8 INTO T1
0F6A                 ;
0F6A   A5 57               LDA T1
0F6C   18                  CLC
0F6D   65 59               ADC T2
0F6F   85 57               STA T1
0F71   A5 58               LDA T1+1
0F73   65 5A               ADC T2+1
0F75                 ;
0F75                 ;ADD $E000 INTO T1
0F75                 ;
0F75   18                  CLC
0F76   69 E0               ADC #$E0
0F78   85 58               STA T1+1
0F7A   18                  CLC
0F7B   60                  RTS
0F7C   00 00         MUL320 .WOR 0,320,640,960,1280
0F7E   40 01
0F80   80 02
0F82   C0 03
0F84   00 05
0F86   40 06               .WOR 1600,1920,2240,2560,2880
0F88   80 07
0F8A   C0 08
0F8C   00 0A
0F8E   40 0B
```

```
LOC   CODE         LINE


0F90  80 0C                .WOR 3200,3520,3840,4160,4480
0F92  C0 0D
0F94  00 0F
0F96  40 10
0F98  80 11
0F9A  C0 12                .WOR 4800,5120,5440,5760,6080
0F9C  00 14
0F9E  40 15
0FA0  80 16
0FA2  C0 17
0FA4  00 19                .WOR 6400,6720,7040,7360,7680
0FA6  40 1A
0FA8  80 1B
0FAA  C0 1C
0FAC  00 1E
0FAE  00 00        MUL40   .WOR 0,40,80,120,160
0FB0  28 00
0FB2  50 00
0FB4  78 00
0FB6  A0 00
0FB8  C8 00                .WOR 200,240,280,320,360
0FBA  F0 00
0FBC  18 01
0FBE  40 01
0FC0  68 01
0FC2  90 01                .WOR 400,440,480,520,560
0FC4  B8 01
0FC6  E0 01
0FC8  08 02
0FCA  30 02
0FCC  58 02                .WOR 600,640,680,720,760
0FCE  80 02
0FD0  A8 02
0FD2  D0 02
0FD4  F8 02
0FD6  20 03                .WOR 800,840,880,920,960
0FD8  48 03
0FDA  70 03
0FDC  98 03
0FDE  C0 03
0FE0  01           TOP2X   .BYT 1,2,4,8,16,32,64,128
0FE1  02
0FE2  04
0FE3  08
0FE4  10
0FE5  20
0FE6  40
0FE7  80
0FE8                       .END
```

PLOT

Abbreviated entry: P(shift)L

Affected Basic abbreviations: None

Token: Hex $CD   Decimal 205

Purpose: To plot a single point on the high resolution screen.

Syntax: PLOT X,Y,col,br

Errors: Illegal quantity – if the values X or Y are <−32768 or >32767
if the values col or br are <Ø or >255

Use: PLOT is the main point plotting routine called by all other line or shape plotting routines. This routine places a point at a given X,Y coordinate on the high resolution screen. The value 'col' is the colour of the point to be plotted. The parameter 'br' is the brush to be plotted with (Ø or 1) in standard high resolution or (Ø,1,2, or 3) in multicolour. 'br' is ANDed with either 1 or 3 to keep it in the range. In both plotting modes, the colour is ignored if the brush value is zero (unplot a point).

Routine entry point: $ØFE8

Routine operation: The routine reads in the coordinates, the colour and brush values. The dot calculate routine is then called and returns if carry is set. If carry is clear, the plotting mode is found and the choice of brush made. The point is stored in the bit map along with the associated colour.

This routine exits with carry set (point unplottable) or carry clear (point plotted).

```
LOC   CODE        LINE


0FE8                     .LIB PLOT
0FE8              ;
0FE8              ; ROUTINE TO PLOT A POINT
0FE8              ;
0FE8  20 E3 10    R00002 JSR GXY          ; GET X AND Y
0FEB  20 42 2B           JSR GCB         ; GET COLOUR AND BRUSH
0FEE  AD 09 29           LDA TX
0FF1  85 59              STA T2          ; X IS STORED AS A DOUBLE
0FF3  AD 0A 29           LDA TX+1        ; BYTE SIGNED INT
0FF6  85 5A              STA T2+1        ; SO IS Y
0FF8  AD 0C 29           LDA TY
0FFB  85 5B              STA T3
0FFD  AD 0D 29           LDA TY+1
1000  85 5C              STA T3+1
1002  20 CE 0E    PLOT   JSR DOT
1005  90 01              BCC PLOT1
1007  60                 RTS
1008  20 2D 11    PLOT1  JSR KEROUT       ;DISABLE IRQ
100B  A5 02              LDA MODE
100D  D0 03              BNE MULTII
100F  4C AB 10           JMP HIRES
1012  A5 FC       MULTII LDA FBR
1014  C9 00              CMP #$00        ; CHOOSE BRUSH 0
1016  F0 10              BEQ BRUSH0
1018  C9 01              CMP #$01        ; CHOOSE BRUSH 1
101A  F0 26              BEQ BRUSH1
101C  C9 02              CMP #$02        ; CHOOSE BRUSH 2
101E  F0 4E              BEQ BRUSH2
1020  C9 03              CMP #$03        ; CHOOSE BRUSH 3
1022  F0 6E              BEQ BRUSH3
1024  38                 SEC             ; ERROR IN BRUSH NUMBER
1025  4C 34 11           JMP KERIN       ; RESTORE BASIC ROM AND
1028  A0 00       BRUSH0 LDY #$00
102A  A5 5E              LDA T5          ; UNPLOT BOTH POINTS IN
102C  49 FF              EOR #$FF        ; MULTICOLOUR MODE
102E  85 5E              STA T5
1030  A5 5F              LDA T6
1032  49 FF              EOR #$FF
1034  85 5F              STA T6
```

```
LOC   CODE         LINE

1036  B1 57               LDA (T1),Y     ; STORE ON HIRES SCREEN
1038  25 5E               AND T5
103A  25 5F               AND T6
103C  91 57               STA (T1),Y
103E  18                  CLC            ; ENABLE BASIC ROM
103F  4C 34 11            JMP KERIN
1042  A0 00        BRUSH1 LDY #$00
1044  A5 5F               LDA T6         ; STORE COMBINATION 01 IN
1046  49 FF               EOR #$FF       ; THE BYTES FOR MULTI
1048  85 5F               STA T6         ; POINT.
104A  B1 57               LDA (T1),Y
104C  05 5E               ORA T5
104E  25 5F               AND T6
1050  91 57               STA (T1),Y     ; STORE ON HIRES SCREEN
1052  A9 C0               LDA #$C0       ; ADD START OF VIDEO RAM
1054  18                  CLC            ; TO T3
1055  65 5C               ADC T3+1
1057  85 5C               STA T3+1
1059  06 FD               ASL COL
105B  06 FD               ASL COL
105D  06 FD               ASL COL
105F  06 FD               ASL COL        ; COLOUR TIMES 16
1061  B1 5B               LDA (T3),Y
1063  29 0F               AND #$0F       ; MASK OFF TOP 4 BITS
1065  18                  CLC
1066  65 FD               ADC COL        ; ADD COLOUR TIMES 16
1068  91 5B               STA (T3),Y     ; STORE IN VIDEO RAM
106A  18                  CLC            ; ENABLE BASIC ROM
106B  4C 34 11            JMP KERIN
106E  A0 00        BRUSH2 LDY #$00
1070  A5 5E               LDA T5         ; PLOT POINTS FOR BRUSH 2
1072  49 FF               EOR #$FF
1074  85 5E               STA T5
1076  B1 57               LDA (T1),Y
1078  25 5E               AND T5
107A  05 5F               ORA T6
107C  91 57               STA (T1),Y     ; STORE ON HIRES SCREEN
107E  18                  CLC
107F  A9 C0               LDA #$C0       ; ADD START OF VIDEO RAM
1081  65 5C               ADC T3+1       ; TO T3
1083  85 5C               STA T3+1
1085  B1 5B               LDA (T3),Y
1087  29 F0               AND #$F0       ; MASK OFF BOTTOM 4 BITS
1089  18                  CLC
108A  65 FD               ADC COL        ; ADD IN THE COLOUR
108C  91 5B               STA (T3),Y     ; STORE ON VIDEO RAM
108E  18                  CLC            ; ENABLE BASIC ROM
108F  4C 34 11            JMP KERIN
1092  A0 00        BRUSH3 LDY #$00
1094  B1 57               LDA (T1),Y     ; PLOT POINTS FOR BRUSH 3
1096  05 5E               ORA T5
1098  05 5F               ORA T6
109A  91 57               STA (T1),Y     ; STORE ON HIRES SCREEN
109C  A9 D8               LDA #$D8       ; ADD START OF COLOUR RAM
109E  18                  CLC            ; TO T3
109F  65 5C               ADC T3+1
10A1  85 5C               STA T3+1
10A3  A5 FD               LDA COL
10A5  91 5B               STA (T3),Y     ; STORE COLOUR IN COLOUR RAM
10A7  18                  CLC            ; ENABLE BASIC ROM
10A8  4C 34 11            JMP KERIN
10AB  A0 00        HIRES  LDY #$00
10AD  A5 FC               LDA PBR        ; IF BRUSH=0 THEN UNPLOT
10AF  F0 22               BEQ UNPLOT     ; IN STANDARD MODE
10B1  B1 57               LDA (T1),Y     ; OTHERWISE PLOT POINT
10B3  05 5E               ORA T5
```

```
LOC   CODE        LINE

10B5  91 57              STA (T1),Y
10B7  A9 C0              LDA #$C0       ;ADD START OF VIDEO RAM
10B9  18                 CLC            ;TO T3
10BA  65 5C              ADC T3+1
10BC  85 5C              STA T3+1
10BE  A5 FD              LDA COL
10C0  0A                 ASL A
10C1  0A                 ASL A
10C2  0A                 ASL A
10C3  0A                 ASL A
10C4  85 5F              STA T6         ;COLOUR TIMES 16
10C6  B1 5B              LDA (T3),Y
10C8  29 0F              AND #$0F       ;MASK OFF TOP 4 BITS
10CA  18                 CLC
10CB  65 5F              ADC T6         ;ADD COLOUR
10CD  91 5B              STA (T3),Y     ;STORE IN VIDEO RAM
10CF  18                 CLC            ;ENABLE KERNAL ROM
10D0  4C 34 11           JMP KERIN
10D3  A5 5E       UNPLOT LDA T5          ;UNPLOT IN STANDARD MODE
10D5  49 FF              EOR #$FF
10D7  85 5E              STA T5
10D9  B1 57              LDA (T1),Y
10DB  25 5E              AND T5
10DD  91 57              STA (T1),Y
10DF  18                 CLC            ;ENABLE KERNAL ROM
10E0  4C 34 11           JMP KERIN
10E3              ;
10E3              ;GET X AND Y VALUE
10E3              ;INTO TX AND TY
10E3              ;
10E3  20 8A AD    GXY    JSR $AD8A       ;GET X
10E6  20 BF B1           JSR $B1BF      ;FIX IT
10E9  A6 65              LDX $65
10EB  A4 64              LDY $64
10ED  8E 09 29           STX TX
10F0  8C 0A 29           STY TX+1
10F3  20 FD AE           JSR $AEFD      ;CHECK ','
10F6  20 8A AD           JSR $AD8A      ;GET Y
10F9  20 BF B1           JSR $B1BF      ;FIX IT
10FC  A6 65              LDX $65
10FE  A4 64              LDY $64
1100  8E 0C 29           STX TY
1103  8C 0D 29           STY TY+1
1106              ;
1106              ;ADD ORIGIN VALUES
1106              ;
1106  AD 09 29           LDA TX
1109  18                 CLC
110A  6D 4D 0E           ADC XORIG
110D  8D 09 29           STA TX
1110  AD 0A 29           LDA TX+1
1113  6D 4E 0E           ADC XORIG+1
1116  8D 0A 29           STA TX+1
1119  AD 0C 29           LDA TY
111C  18                 CLC
111D  6D 4F 0E           ADC YORIG
1120  8D 0C 29           STA TY
1123  AD 0D 29           LDA TY+1
1126  6D 50 0E           ADC YORIG+1
1129  8D 0D 29           STA TY+1
112C  60                 RTS
112D              ;
112D              ;DISABLE KERNAL AND IRQ
112D              ;
112D  A5 01       KEROUT LDA $01
112F  29 FD              AND #$FD       ;SWITCH OUT
```

```
LOC   CODE        LINE

1131  85 01              STA $01
1133  60                 RTS
1134              ;
1134              ;ENABLE KERNAL AND IRQ
1134              ;
1134  48          KERIN  PHA
1135  A5 01              LDA $01
1137  09 02              ORA #$02      ;SWITCH IN
1139  85 01              STA $01
113B  68                 PLA
113C  60                 RTS
113D                     .END
```

DRAW

Abbreviated entry: D(shift)R

Affected Basic abbreviations: None

Token: Hex $CF   Decimal 2Ø7

Purpose: To draw a straight line between two points.

Syntax: DRAW X1,Y1,X2,Y2,col,br

Errors: Illegal quantity – if the coordinates X1,Y1 or X2,Y2 are out of integer range
– if col or br are <Ø or >255.

Use: DRAW will plot a line between the coordinates X1,Y1 and X2,Y2 in the colour 'col' using the plotting brush 'br'. DRAW uses the plot routine to plot each single point.

Routine entry point: $113D

Routine operation: DRAW uses a simple algorithm that calculates a step value for X and Y directions and adds them each time through until complete.

```
LOC   CODE        LINE

113D                      .LIB LINE
113D              ;
113D              ; ROUTINE TO PLOT A LINE BETWEEN
113D              ; TWO COORDINATES
113D              ;
113D  20 61 13    R00004 JSR GLPARS     ;GET PARAMETERS
1140              ;
1140  AD 40 03    BOX    LDA X2         ; XD=X2-X1
1143  38                 SEC
1144  ED 3C 03           SBC X1
1147  8D 44 03           STA XD
114A  AD 41 03           LDA X2+1
114D  ED 3D 03           SBC X1+1
1150  8D 45 03           STA XD+1
1153  AD 42 03           LDA Y2         ; YD=Y2-Y1
1156  38                 SEC
```

```
LOC   CODE        LINE

1157  ED 3E 03           SBC Y1
115A  8D 46 03           STA YD
115D  AD 43 03           LDA Y2+1
1160  ED 3F 03           SBC Y1+1
1163  8D 47 03           STA YD+1
1166              ;
1166              ;NEAREST DIAGONAL
1166              ;
1166  A9 01              LDA #$01
1168  8D 5E 03           STA A0          ; A0=1
116B  8D 60 03           STA A1          ; A1=1
116E  A9 00              LDA #$00
1170  8D 5F 03           STA A0+1
1173  8D 61 03           STA A1+1
1176  AD 47 03           LDA YD+1        ; IF YD<0 THEN CHECKX
1179  10 08              BPL CHECKX
117B  A9 FF              LDA #$FF
117D  8D 5E 03           STA A0          ; A0=-1
1180  8D 5F 03           STA A0+1
1183  AD 45 03    CHECKX LDA XD+1         ; IF XD<0 THEN POS1
1186  10 26              BPL POS1
1188  A9 FF              LDA #$FF
118A  8D 60 03           STA A1          ; A1=-1
118D  8D 61 03           STA A1+1
1190  AD 45 03    NRHOR  LDA XD+1
1193  49 FF              EOR #$FF
1195  8D 49 03           STA XE+1
1198  18                 CLC
1199  AD 44 03           LDA XD
119C  49 FF              EOR #$FF
119E  69 01              ADC #$01
11A0  8D 48 03           STA XE
11A3  AD 49 03           LDA XE+1
11A6  69 00              ADC #$00
11A8  8D 49 03           STA XE+1        ; XE=ABS(XD)
11AB  4C BA 11           JMP CHECKY
11AE  AD 44 03    POS1   LDA XD
11B1  8D 48 03           STA XE
11B4  AD 45 03           LDA XD+1
11B7  8D 49 03           STA XE+1
11BA  AD 47 03    CHECKY LDA YD+1
11BD  10 1E              BPL POS2
11BF  AD 47 03           LDA YD+1
11C2  49 FF              EOR #$FF
11C4  8D 4B 03           STA YE+1
11C7  18                 CLC
11C8  AD 46 03           LDA YD
11CB  49 FF              EOR #$FF
11CD  69 01              ADC #$01
11CF  8D 4A 03           STA YE
11D2  AD 4B 03           LDA YE+1
11D5  69 00              ADC #$00
11D7  8D 4B 03           STA YE+1        ; YE=ABS(YD)
11DA  4C E9 11           JMP CALCD1
11DD  AD 46 03    POS2   LDA YD
11E0  8D 4A 03           STA YE
11E3  AD 47 03           LDA YD+1
11E6  8D 4B 03           STA YE+1
11E9              ;
11E9              ; NEAREST HORIZONTAL/VERTICAL
11E9              ;
11E9  AD 48 03    CALCD1 LDA XE
11EC  38                 SEC
11ED  ED 4A 03           SBC YE
11F0  8D 58 03           STA D1
11F3  AD 49 03           LDA XE+1
```

```
LOC   CODE        LINE

11F6  ED 4B 03          SBC YE+1
11F9  8D 59 03          STA D1+1      ; D1=XE-YE
11FC  10 3A             BPL L360      ; IF D1>=0 THEN L360
11FE  A9 FF             LDA #$FF
1200  8D 5A 03          STA S0
1203  8D 5B 03          STA S0+1      ; S0=-1
1206  A9 00             LDA #$00
1208  8D 5C 03          STA S1
120B  8D 5D 03          STA S1+1      ; S1=0
120E  AD 4A 03          LDA YE
1211  8D 4C 03          STA LG        ; LG=YE
1214  AD 4B 03          LDA YE+1
1217  8D 4D 03          STA LG+1
121A  AD 48 03          LDA XE
121D  8D 4E 03          STA SH        ; SH=XE
1220  AD 49 03          LDA XE+1
1223  8D 4F 03          STA SH+1
1226  AD 47 03          LDA YD+1
1229  30 44             BMI L380      ; IF YD<0 THEN L380
122B  A9 01             LDA #$01
122D  8D 5A 03          STA S0        ; S0=1
1230  A9 00             LDA #$00
1232  8D 5B 03          STA S0+1
1235  4C 6F 12          JMP L380
1238  A9 00       L360  LDA #$00
123A  8D 5A 03          STA S0
123D  8D 5B 03          STA S0+1      ;S0=0
1240  A9 FF             LDA #$FF
1242  8D 5C 03          STA S1
1245  8D 5D 03          STA S1+1      ; S1=-1
1248  AD 48 03          LDA XE
124B  8D 4C 03          STA LG        ; LG=XE
124E  AD 49 03          LDA XE+1
1251  8D 4D 03          STA LG+1
1254  AD 4A 03          LDA YE
1257  8D 4E 03          STA SH        ; SH=YE
125A  AD 4B 03          LDA YE+1
125D  8D 4F 03          STA SH+1
1260  AD 45 03          LDA XD+1
1263  30 0A             BMI L380      ; IF XD<0 THEN L380
1265  A9 01             LDA #$01
1267  8D 5C 03          STA S1
126A  A9 00             LDA #$00
126C  8D 5D 03          STA S1+1      ; S1=1
126F              ;
126F              ; SET UP
126F              ;
126F  AD 4C 03    L380  LDA LG
1272  8D 52 03          STA TT        ; TT=LG
1275  AD 4D 03          LDA LG+1
1278  8D 53 03          STA TT+1
127B  AD 4E 03          LDA SH
127E  8D 50 03          STA TS        ; TS=SH
1281  AD 4F 03          LDA SH+1
1284  8D 51 03          STA TS+1
1287  AD 4C 03          LDA LG
128A  38                SEC           ; UD=LG-SH
128B  ED 4E 03          SBC SH
128E  8D 54 03          STA UD
1291  AD 4D 03          LDA LG+1
1294  ED 4F 03          SBC SH+1
1297  8D 55 03          STA UD+1
129A  4E 4D 03          LSR LG+1      ; LG/2
129D  6E 4C 03          ROR LG
12A0  AD 4E 03          LDA SH        ; CT=SH-LG(/2)
12A3  38                SEC
```

```
LOC   CODE        LINE

12A4  ED 4C 03          SBC LG
12A7  8D 56 03          STA CT
12AA  AD 4F 03          LDA SH+1
12AD  ED 4D 03          SBC LG+1
12B0  8D 57 03          STA CT+1
12B3              ;
12B3              ; WHILE MORE POINTS DO
12B3              ;
12B3  AD 3C 03    L420  LDA X1
12B6  85 59             STA T2
12B8  AD 3D 03          LDA X1+1
12BB  85 5A             STA T2+1
12BD  AD 3E 03          LDA Y1
12C0  85 5B             STA T3
12C2  A5 FE             LDA COL+1
12C4  85 FD             STA COL
12C6  AD 3F 03          LDA Y1+1
12C9  85 5C             STA T3+1
12CB  20 02 10          JSR PLOT        ; CALL PLOT POINT ROUTINE
12CE  AD 57 03    NOPLOT LDA CT+1
12D1  10 3C             BPL L460        ; IF CT>=0 THEN L460
12D3  AD 56 03          LDA CT
12D6  18                CLC             ; CT=CT+TS
12D7  6D 50 03          ADC TS
12DA  8D 56 03          STA CT
12DD  AD 57 03          LDA CT+1
12E0  6D 51 03          ADC TS+1
12E3  8D 57 03          STA CT+1
12E6  AD 3C 03          LDA X1
12E9  18                CLC
12EA  6D 5C 03          ADC S1
12ED  8D 3C 03          STA X1          ; X1=X1+S1
12F0  AD 3D 03          LDA X1+1
12F3  6D 5D 03          ADC S1+1
12F6  8D 3D 03          STA X1+1
12F9  AD 3E 03          LDA Y1
12FC  18                CLC
12FD  6D 5A 03          ADC S0
1300  8D 3E 03          STA Y1          ; Y1=Y1+S0
1303  AD 3F 03          LDA Y1+1
1306  6D 5B 03          ADC S0+1
1309  8D 3F 03          STA Y1+1
130C  4C 48 13          JMP L470
130F  AD 56 03    L460  LDA CT
1312  38                SEC
1313  ED 54 03          SBC UD
1316  8D 56 03          STA CT          ; CT=CT-UD
1319  AD 57 03          LDA CT+1
131C  ED 55 03          SBC UD+1
131F  8D 57 03          STA CT+1
1322  AD 3C 03          LDA X1
1325  18                CLC
1326  6D 60 03          ADC A1
1329  8D 3C 03          STA X1          ; X1=X1+A1
132C  AD 3D 03          LDA X1+1
132F  6D 61 03          ADC A1+1
1332  8D 3D 03          STA X1+1
1335  AD 3E 03          LDA Y1
1338  18                CLC
1339  6D 5E 03          ADC A0
133C  8D 3E 03          STA Y1          ; Y1=Y1+A0
133F  AD 3F 03          LDA Y1+1
1342  6D 5F 03          ADC A0+1
1345  8D 3F 03          STA Y1+1
1348  AD 52 03    L470  LDA TT
134B  38                SEC
```

```
LOC    CODE         LINE

134C   E9 01                SBC #$01
134E   8D 52 03             STA TT          ; TT=TT-1
1351   AD 53 03             LDA TT+1
1354   E9 00                SBC #$00
1356   8D 53 03             STA TT+1
1359   C9 FF                CMP #$FF        ; IF TT<0 THEN RETURN
135B   F0 03                BEQ RTN
135D   4C B3 12             JMP L420        ; GOTO L420
1360   60           RTN     RTS
1361                ;
1361                ;GET PARAMTERS FOR LINE
1361                ;
1361   20 E3 10     GLPARS JSR GXY           ;X1 AND Y1
1364   AD 0C 29             LDA TY
1367   8D 3E 03             STA Y1
136A   AD 0D 29             LDA TY+1
136D   8D 3F 03             STA Y1+1
1370   AD 09 29             LDA TX
1373   8D 3C 03             STA X1
1376   AD 0A 29             LDA TX+1
1379   8D 3D 03             STA X1+1
137C   20 FD AE             JSR CHKCOM      ; X2 AND Y2
137F   20 E3 10             JSR GXY
1382   AD 0C 29             LDA TY
1385   8D 42 03             STA Y2
1388   AD 0D 29             LDA TY+1
138B   8D 43 03             STA Y2+1
138E   AD 09 29             LDA TX
1391   8D 40 03             STA X2
1394   AD 0A 29             LDA TX+1
1397   8D 41 03             STA X2+1
139A   4C 42 2B             JMP GCB         ; COLOUR AND BRUSH
139D                        .END
```

Character plot

This routine does not have a command name and is only called by the routine CHAR. Its purpose is to plot a single character (ASCII value in $0366) onto the screen, where the locations $0362,$0363 hold the top left position in the X direction, and $0364,$0365 hold the top left position in the Y direction.

The character value is converted to its screen POKE value using a look up table. If the value from the table is $FF, then the character is not a plottable one and the routine exits. The character ROM is then switched in and each byte of the character is displayed bit by bit onto the screen (reversing it if specified). In standard high resolution mode the character is plotted in the normal character size, but in multicolour mode the character is plotted in double width.

```
LOC    CODE         LINE

139D                        .LIB CHAR-PLOT
139D                ;
139D                ; ROUTINE TO PLOT A CHARACTER
139D                ;
139D   AE 66 03     MAIN    LDX CHAR
13A0   BD 53 14             LDA CONV,X
13A3   85 61                STA POINTR
13A5   10 01                BPL CHAROK
```

```
LOC   CODE        LINE

13A7  60                 RTS
13A8  A9 00       CHAROK LDA #$00
13AA  85 62              STA POINTR+1
13AC  A0 03              LDY #$03
13AE  06 61       LOOP0  ASL POINTR        ;CHAR*8
13B0  26 62              ROL POINTR+1
13B2  88                 DEY
13B3  D0 F9              BNE LOOP0
13B5  A5 62              LDA POINTR+1
13B7  18                 CLC
13B8  69 D8              ADC #$D8
13BA  85 62              STA POINTR+1     ;START OF CHAR ROM
13BC  A9 08              LDA #$08
13BE  8D 6A 03           STA CNTR1
13C1  A5 01       LOOP01 LDA $01
13C3  29 F9              AND #$F9
13C5  85 01              STA $01
13C7  A0 00              LDY #$00
13C9  B1 61              LDA (POINTR),Y ; GET BYTE
13CB  8D 66 03           STA CHAR
13CE  A5 01              LDA $01         ; SWITCH OUT CHARACTER ROM
13D0  09 06              ORA #$06
13D2  85 01              STA $01
13D4  AD 67 03           LDA RVORN
13D7  F0 06              BEQ NORMAL
13D9  4D 66 03           EOR CHAR        ; REVERSE BYTE
13DC  8D 66 03           STA CHAR
13DF  AD 63 03    NORMAL LDA XTL+1
13E2  8D 69 03           STA XTEMP+1     ;X COORDINATE OF BIT
13E5  AD 62 03           LDA XTL
13E8  8D 68 03           STA XTEMP
13EB  A9 80              LDA #$80
13ED  8D 6B 03           STA POINT
13F0  AD 66 03    LOOP02 LDA CHAR         ; LOOP FOR 8
13F3  2D 6B 03           AND POINT
13F6  F0 1D              BEQ NXTPNT      ; IF NOT SET, NEXT POINT
13F8  AD 68 03           LDA XTEMP
13FB  85 59              STA T2
13FD  AD 69 03           LDA XTEMP+1
1400  85 5A              STA T2+1
1402  AD 64 03           LDA YTL
1405  85 5B              STA T3
1407  AD 65 03           LDA YTL+1
140A  85 5C              STA T3+1
140C  A5 FE              LDA COL+1
140E  85 FD              STA COL
1410  20 02 10           JSR PLOT        ; PLOT POINT
1413  B0 3D              BCS FIN         ;POINT OUT OF BOUNDS
1415  4E 6B 03    NXTPNT LSR POINT
1418  F0 17              BEQ NXTLNE      ; IF BYTE FINISHED, NEXT LINE
141A  A5 02              LDA MODE
141C  18                 CLC
141D  69 01              ADC #$01
141F  18                 CLC
1420  6D 68 03           ADC XTEMP
1423  8D 68 03           STA XTEMP
1426  AD 69 03           LDA XTEMP+1
1429  69 00              ADC #$00
142B  8D 69 03           STA XTEMP+1
142E  4C F0 13           JMP LOOP02
1431  A5 61       NXTLNE LDA POINTR
1433  18                 CLC
1434  69 01              ADC #$01
1436  85 61              STA POINTR      ; INCREASE POINTER BY 1
1438  A5 62              LDA POINTR+1
143A  69 00              ADC #$00
```

```
LOC   CODE        LINE

143C  85 62                STA POINTR+1
143E  18                   CLC
143F  AD 64 03             LDA YTL
1442  D0 03                BNE DERE1
1444  CE 65 03             DEC YTL+1
1447  CE 64 03    DERE1    DEC YTL          ; DECREASE Y COORD
144A  CE 6A 03             DEC CNTR1
144D  F0 03                BEQ FIN         ; IF ALL 8 BYTES PLOTTED, FINISH
144F  4C C1 13             JMP LOOP01
1452  60          FIN      RTS
1453  FF          CONV     .BYT $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
1454  FF
1455  FF
1456  FF
1457  FF
1458  FF
1459  FF
145A  FF
145B  FF                   .BYT $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
145C  FF
145D  FF
145E  FF
145F  FF
1460  FF
1461  FF
1462  FF
1463  FF                   .BYT $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
1464  FF
1465  FF
1466  FF
1467  FF
1468  FF
1469  FF
146A  FF
146B  FF                   .BYT $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
146C  FF
146D  FF
146E  FF
146F  FF
1470  FF
1471  FF
1472  FF
1473  20                   .BYT 32,33,34,35,36,37,38,39
1474  21
1475  22
1476  23
1477  24
1478  25
1479  26
147A  27
147B  28                   .BYT 40,41,42,43,44,45,46,47
147C  29
147D  2A
147E  2B
147F  2C
1480  2D
1481  2E
1482  2F
1483  30                   .BYT 48,49,50,51,52,53,54,55
1484  31
1485  32
1486  33
1487  34
1488  35
1489  36
148A  37
```

```
LOC   CODE      LINE

148B  38              .BYT 56,57,58,59,60,61,62,63
148C  39
148D  3A
148E  3B
148F  3C
1490  3D
1491  3E
1492  3F
1493  00              .BYT 0,1,2,3,4,5,6,7
1494  01
1495  02
1496  03
1497  04
1498  05
1499  06
149A  07
149B  08              .BYT 8,9,10,11,12,13,14,15
149C  09
149D  0A
149E  0B
149F  0C
14A0  0D
14A1  0E
14A2  0F
14A3  10              .BYT 16,17,18,19,20,21,22,23
14A4  11
14A5  12
14A6  13
14A7  14
14A8  15
14A9  16
14AA  17
14AB  18              .BYT 24,25,26,27,28,29,30,31
14AC  19
14AD  1A
14AE  1B
14AF  1C
14B0  1D
14B1  1E
14B2  1F
14B3  40              .BYT 64,65,66,67,68,69,70,71
14B4  41
14B5  42
14B6  43
14B7  44
14B8  45
14B9  46
14BA  47
14BB  48              .BYT 72,73,74,75,76,77,78,79
14BC  49
14BD  4A
14BE  4B
14BF  4C
14C0  4D
14C1  4E
14C2  4F
14C3  50              .BYT 80,81,82,83,84,85,86,87
14C4  51
14C5  52
14C6  53
14C7  54
14C8  55
14C9  56
14CA  57
14CB  58              .BYT 88,89,90,91,92,93,94,95
14CC  59
```

```
LOC   CODE        LINE

14CD  5A
14CE  5B
14CF  5C
14D0  5D
14D1  5E
14D2  5F
14D3  FF                 .BYT $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
14D4  FF
14D5  FF
14D6  FF
14D7  FF
14D8  FF
14D9  FF
14DA  FF
14DB  FF                 .BYT $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
14DC  FF
14DD  FF
14DE  FF
14DF  FF
14E0  FF
14E1  FF
14E2  FF
14E3  FF                 .BYT $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
14E4  FF
14E5  FF
14E6  FF
14E7  FF
14E8  FF
14E9  FF
14EA  FF
14EB  FF                 .BYT $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
14EC  FF
14ED  FF
14EE  FF
14EF  FF
14F0  FF
14F1  FF
14F2  FF
14F3  60                 .BYT 96,97,98,99,100,101,102,103
14F4  61
14F5  62
14F6  63
14F7  64
14F8  65
14F9  66
14FA  67
14FB  68                 .BYT 104,105,106,107,108,109,110,111
14FC  69
14FD  6A
14FE  6B
14FF  6C
1500  6D
1501  6E
1502  6F
1503  70                 .BYT 112,113,114,115,116,117,118,119
1504  71
1505  72
1506  73
1507  74
1508  75
1509  76
150A  77
150B  78                 .BYT 120,121,122,123,124,125,126,127
150C  79
150D  7A
150E  7B
150F  7C
```

```
LOC   CODE        LINE

1510  7D
1511  7E
1512  7F
1513  40              .BYT 64,65,66,67,68,69,70,71
1514  41
1515  42
1516  43
1517  44
1518  45
1519  46
151A  47
151B  48              .BYT 72,73,74,75,76,77,78,79
151C  49
151D  4A
151E  4B
151F  4C
1520  4D
1521  4E
1522  4F
1523  50              .BYT 80,81,82,83,84,85,86,87
1524  51
1525  52
1526  53
1527  54
1528  55
1529  56
152A  57
152B  58              .BYT 88,89,90,91,92,93,94,95
152C  59
152D  5A
152E  5B
152F  5C
1530  5D
1531  5E
1532  5F
1533  60              .BYT 96,97,98,99,100,101,102,103
1534  61
1535  62
1536  63
1537  64
1538  65
1539  66
153A  67
153B  68              .BYT 104,105,106,107,108,109,110,111
153C  69
153D  6A
153E  6B
153F  6C
1540  6D
1541  6E
1542  6F
1543  70              .BYT 112,113,114,115,116,117,118,119
1544  71
1545  72
1546  73
1547  74
1548  75
1549  76
154A  77
154B  78              .BYT 120,121,122,123,124,125,126,94
154C  79
154D  7A
154E  7B
154F  7C
1550  7D
1551  7E
1552  5E
1553                  .END
```

CHAR

Abbreviated entry: C(shift)H

Affected Basic abbreviations: CHR$ – CH(shift)R

Token: Hex $DØ   Decimal 2Ø8

Purpose: To plot a string of characters to the graphics screen.

Syntax: CHAR X,Y,col,br,rv,string

Errors: Illegal quantity – if X or Y are <−32768 or >32767
– if col, br, or rv are <Ø or >255
String too long – if the string's length exceeds 255 characters
Type mismatch – if the parameter 'string' is not a string

Use: CHAR is used to put a string of characters onto the screen. The coordinates X,Y are the top left of the first character, 'col' and 'br' are as in PLOT, and 'rv' is a flag to say whether the characters in the string are reversed or not (Ø=normal, non zero=reversed). Cursor control characters, colour characters etc. are plotted as a space. The character set used is the upper/lower case set.

Routine entry point: $1553

Routine operation: The parameters are read in and each character is plotted using the plot character routine, until all characters are displayed. After each character, the X coordinate is increased by 8 in standard high resolution mode or 16 in multicolour mode. There is no wrap around at the end of a line.

```
LOC   CODE        LINE

1553                     .LIB CHAR
1553              ;
1553              ; ROUTINE TO PLOT A STRING
1553              ;
1553  20 E3 10    R00005 JSR GXY          ; X AND Y
1556  AD 0C 29           LDA TY
1559  8D 79 03           STA YTLTMP
155C  AD 0D 29           LDA TY+1
155F  8D 7A 03           STA YTLTMP+1
1562  AD 09 29           LDA TX
1565  8D 77 03           STA XTLTMP
1568  AD 0A 29           LDA TX+1
156B  8D 78 03           STA XTLTMP+1
156E  20 42 2B           JSR GCB          ;COLOUR AND BRUSH
1571  A5 FC              LDA PBR
1573  8D 7B 03           STA PBRTMP
1576  A5 FD              LDA COL
1578  8D 7C 03           STA COLTMP
157B  20 F1 B7           JSR PARAM        ; RVORN
157E  8E 67 03           STX RVORN
1581  E0 00              CPX #$00
1583  F0 05              BEQ CHAR01
1585  A9 FF              LDA #$FF
1587  8D 67 03           STA RVORN
```

```
158A   20 FD AE    CHAR01 JSR CHKCOM
158D   20 9E AD           JSR $AD9E       ; GET STRING
1590   20 A3 B6           JSR $B6A3
1593   85 24              STA $24
1595   A0 00              LDY #$00
1597   84 25              STY $25
1599   AD 79 03    LOOPDI LDA YTLTMP       ;GET COORDINATE
159C   8D 64 03           STA YTL         ; AND STORE FOR
159F   AD 7A 03           LDA YTLTMP+1    ; CHAR-PLOT ROUTINE
15A2   8D 65 03           STA YTL+1
15A5   AD 77 03           LDA XTLTMP
15A8   8D 62 03           STA XTL
15AB   AD 78 03           LDA XTLTMP+1
15AE   8D 63 03           STA XTL+1
15B1   AD 7B 03           LDA PBRTMP      ;SET BRUSH
15B4   85 FC              STA PBR
15B6   AD 7C 03           LDA COLTMP      ;SET COLOUR
15B9   85 FE              STA COL+1
15BB   B1 22              LDA ($22),Y     ;GET A CHARACTER
15BD   8D 66 03           STA CHAR        ;STORE IT
15C0   20 9D 13           JSR MAIN        ;PLOT IT
15C3   A4 25              LDY $25         ;DO NEXT?
15C5   C8                 INY
15C6   C4 24              CPY $24
15C8   F0 2B              BEQ CHAREX      ;NO, END OF STRING
15CA   84 25              STY $25
15CC   AD 77 03           LDA XTLTMP      ;INCREASE X COORDINATE
15CF   18                 CLC             ; BY 8
15D0   69 08              ADC #$08
15D2   8D 77 03           STA XTLTMP
15D5   AD 78 03           LDA XTLTMP+1
15D8   69 00              ADC #$00
15DA   8D 78 03           STA XTLTMP+1
15DD   A5 02              LDA MODE        ;IIN MULTICOLOUR?
15DF   F0 B8              BEQ LOOPDI      ;NO
15E1   AD 77 03           LDA XTLTMP      ;INCREASE X BY A
15E4   18                 CLC             ; FURTHER 8
15E5   69 08              ADC #$08
15E7   8D 77 03           STA XTLTMP
15EA   AD 78 03           LDA XTLTMP+1
15ED   69 00              ADC #$00
15EF   8D 78 03           STA XTLTMP+1
15F2   4C 99 15           JMP LOOPDI      ;DO NEXT CHARACTER
15F5   60          CHAREX RTS              ;STRING DONE
15F6                      .END
```

POINT

Abbreviated entry: PO(shift)I

Affected Basic abbreviation: None

Token: Hex $E6   Decimal 23Ø

Purpose: To test a certain pixel on the graphics screen.

Syntax: POINT (X,Y)

Errors: Syntax error – if used on the wrong side of an expression
Illegal quantity – if either X or Y is <−32768 or >32767

Use: POINT is a function and should thus be used on the right hand side of an expression:

A=POINT(16Ø,1ØØ)

The command cannot be used alone:

POINT(16Ø,1ØØ)

POINT returns the brush value of the point plotted at X,Y. The value is either Ø (not plotted) or 1, 2, 3 (plotted using brush 1, 2 or 3) or −1 (off the screen).

Routine entry point: $15F6

Routine operation: The X and Y values are read in and the dot routine called. If the carry flag is set a value of −1 is returned, otherwise the bit combination is tested and that value is returned.

```
LOC   CODE          LINE


15F6                      .LIB POINT
15F6                ;
15F6                ;ROUTINE TO TEST A PIXEL FOR A CERTAIN
15F6                ; BRUSH. THIS IS A USR ROUTINE.
15F6                ;
15F6  20 FA AE      POINTC JSR $AEFA        ;PICK OFF X AND Y COORDINATES
15F9  20 E3 10             JSR GXY
15FC  AD 0C 29             LDA TY
15FF  85 5B                STA T3
1601  AD 0D 29             LDA TY+1
1604  85 5C                STA T3+1
1606  AD 09 29             LDA TX
1609  85 59                STA T2
160B  AD 0A 29             LDA TX+1
160E  85 5A                STA T2+1
1610  20 F7 AE             JSR $AEF7
1613  20 19 16             JSR POINTT      ;TEST POINT
1616  4C 91 B3             JMP $B391       ;FLOAT VALUE
1619                ;
1619  20 CE 0E      POINTT JSR DOT          ;CALCULATE BYTE AND BITS
161C  90 06                BCC POINTK      ;IN BOUNDS
161E  A9 FF                LDA #$FF        ;NO, RESULT=-1
1620  AA                   TAX
1621  38                   SEC
1622  B0 4A                BCS SEND1       ;ALWAYS
1624  A5 02         POINTK LDA MODE
1626  F0 07                BEQ STANDD
1628  A5 5F                LDA T6
162A  18                   CLC
162B  65 5E                ADC T5
162D  85 5E                STA T5
162F  20 2D 11      STANDD JSR KEROUT       ;DISABLE IRQ
1632  A0 00                LDY #$00
1634  B1 57                LDA (T1),Y
1636  25 5E                AND T5
1638  F0 2C                BEQ SEND0       ;NO POINT
163A  A5 02                LDA MODE
163C  F0 1A                BEQ BITIS1      ;BRUSH 1
163E  A5 5E                LDA T5
1640  38                   SEC
1641  E5 5F                SBC T6
1643  85 5E                STA T5
1645  B1 57                LDA (T1),Y
1647  25 5F                AND T6
```

```
LOC   CODE        LINE

1649  F0 0D              BEQ BITIS1      ;BRUSH 1
164B  B1 57              LDA (T1),Y
164D  25 5E              AND T5
164F  F0 0E              BEQ BITIS2      ;BRUSH 2
1651  A9 03              LDA #$03        ;MUST BE BRUSH 3
1653  8D 72 03           STA PTBR
1656  D0 13              BNE SEND        ;PUT 3 IN FAC
1658  A9 01       BITIS1 LDA #$01
165A  8D 72 03           STA PTBR
165D  D0 0C              BNE SEND        ;PUT 1 IN FAC
165F  A9 02       BITIS2 LDA #$02
1661  8D 72 03           STA PTBR
1664  D0 05              BNE SEND
1666  A9 00       SEND0  LDA #$00
1668  8D 72 03           STA PTBR
166B  A2 00       SEND   LDX #$00
166D  18                 CLC
166E  A8          SEND1  TAY
166F  8A                 TXA
1670  4C 34 11           JMP KERIN
1673                     .END
```

FILL

Abbreviated entry: F(shift)I

Affected Basic abbreviations: None

Token: Hex $D2   Decimal 21Ø

Purpose: To fill an enclosed area.

Syntax: Either FILL X,Y,col,br1,br2
Or FILL [X1,Y1,...Xn,Yn],col,br1,br2

Errors: Illegal quantity – if any parameters are out of their range

Use: This command basically fills an enclosed area on the screen with a given brush value. The area to be filled is enclosed by brush 'br2' (i.e. draw a box using br2). The brush to fill with is specified as 'br1' and the start coordinate(s) are X,Y (or as in the second type, X1,Y1...Xn,Yn). The second form of the command will cause the routine to fill from more than one start point.

FILL is illustrated in Program 1.

Routine entry point: $1673

Routine operation: The parameters are read in and, if multi start, all coordinates except the last are pushed to the fill queue. The only (or last) start position is left in a location for the main fill routine to use as its first point.

The main FILL routine is most easily explained by the use of a flow chart (see Fig. 2.1).

```
1 REM BASIC EQUIVALENT OF THE FILL COMMAND
5 HIRES1,1
10 DIMA%(999,1)
20 QO=0:QI=0:M=1:IFPEEK(2)<>0THENM=2
30 DRAW140,90,180,90,0,1
31 DRAW180,90,180,110,0,1
32 DRAW140,90,140,110,0,1
33 DRAW140,110,155,110,0,1
34 DRAW165,110,180,110,0,1
35 POLYGON6,160,100,60,0,1,0
40 X=160:Y=100:B1=2:B2=1:C=5
50 GOSUB10000
60 NORM:END
10000 LS=0:US=0
10010 GOSUB20000
10020 B=POINT(X,Y)
10030 IF(B1=B)OR(B2=B)OR(B=-1)THEN15000
10040 B=POINT(X,Y-1)
10050 IF(B1<>B)AND(B2<>B)THENGOSUB11000:GOTO10060
10055 LS=0
10060 B=POINT(X,Y+1)
10070 IF(B1<>B)AND(B2<>B)THENGOSUB11020:GOTO10090
10080 US=0
10090 PLOTX,Y,C,B1
10100 X=X-M:GOTO10020
11000 IFLS=1THENRETURN
11010 LS=1:Y1=Y-1:GOTO11040
11020 IFUS=1THENRETURN
11030 US=1:Y1=Y+1
11040 A%(QO,0)=X:A%(QO,1)=Y1
11050 QO=QO+1:IFQO=1000THENQO=0
11060 RETURN
15000 IFQI=QOTHENRETURN
15010 X=A%(QI,0):Y=A%(QI,1)
15020 QI=QI+1:IFQI=1000THENQI=0
15030 GOTO10000
20000 B=POINT(X,Y)
20010 IF(B1=B)OR(B2=B)OR(B=-1)THEN21000
20020 X=X+M:GOTO20000
21000 X=X-M:RETURN
```

```
1 REM*****************************
2 REM FILL DEMO
3 REM -BAR CHART-
4 REM*****************************
5 :
10 DIMA(11,2)
11 A=0
12 B=6
15 C=1
20 FORI=0TO11
30 FORJ=0TO2
40 A(I,J)=RND(1)*160+1
50 NEXTJ,I
60 HIRES1,14,2
70 ORIGIN38,10
80 DRAW0,0,300,0,C,1
90 DRAW0,0,0,190,C,1
100 FORJ=0TO2
110 FORI=0TO11
120 X=I*24+J*4
130 SY=0
140 IFJ=0THEN170
150 IFA(I,J-1)>=A(I,J)THEN:DRAWX+4,A(I,J),X+8,A(I,J),C,1:GOTO190
160 SY=A(I,J-1)
170 DRAWX,SY,X,A(I,J),C,1
180 DRAWX,A(I,J),X+8,A(I,J),C,1
190 DRAWX+8,A(I,J),X+8,0,C,1
200 NEXTI,J
```

```
210 FILL[2,1,26,1,50,1,74,1,98,1,122,1,146,1,170,1,194,1,218,1,242,1,266,1],A,2,
1
220 FILL[10,1,34,1,58,1,82,1,106,1,130,1,154,1,178,1,202,1,226,1,250,1,274,1],B,
3,1
230 FILL[14,1,38,1,62,1,86,1,110,1,134,1,158,1,182,1,206,1,230,1,254,1,278,1],C,
1,1
240 FORI=0TO11
250 CHARI*24,-2,C,1,0,MID$("╰─\♠\`♣♥┌╱‾",I+1,1)
260 NEXT
270 FORI=0TO180STEP10
280 CHAR-38,I+4,C,1,0,RIGHT$("  "+STR$(I/10),2)
290 NEXT
1000 GETA$:IFA$<>"←"THEN1000
1010 NORM
```

Program 1.

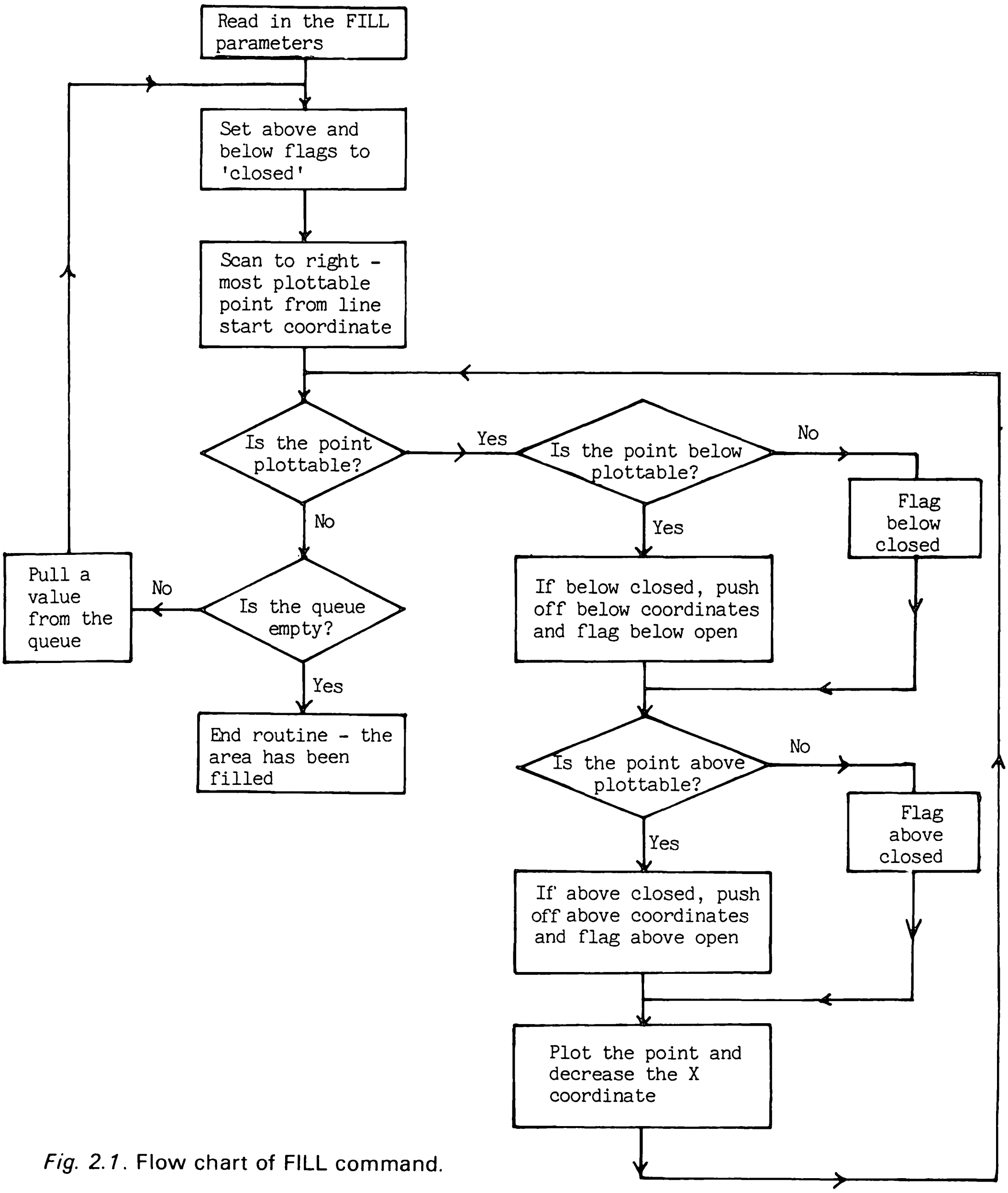


Fig. 2.1. Flow chart of FILL command.

```
LOC   CODE        LINE

1673                     .LIB FILL1
1673              ;ROUTINE TO FILL AN ENCLOSED COMPLEX
1673              ;AREA
1673              ;
1673  20 0A 19    R00007 JSR GFPARS      ;GET X AND Y COORDINATES
1676  20 F2 18           JSR FSTUP      ;SET UP FOR FILL
1679  A9 00       BEGIN  LDA #$00        ;START OF MAIN FILL
167B  8D 76 03           STA USW        ;LOOP. STARTS HERE FOR EACH
167E  8D 75 03           STA LSW        ;LINE
1681  AD 3C 03           LDA X1
1684  8D 40 03           STA X2
1687  85 59              STA T2
1689  AD 3D 03           LDA X1+1
168C  8D 41 03           STA X2+1
168F  85 5A              STA T2+1
1691  AD 3E 03           LDA Y1
1694  85 5B              STA T3
1696  A9 00              LDA #$00
1698  85 5C              STA T3+1
169A  20 F9 17           JSR FILRT2
169D  20 2F 17    M      JSR SETUP       ;START OF FILL LOOP FOR
16A0  20 19 16           JSR POINTT     ;EACH SEPARATE LINE
16A3  AD 72 03           LDA PTBR
16A6  CD 73 03           CMP BRCOL      ;TEST POINT TO BE PLOTTED
16A9  F0 05              BEQ M1
16AB  CD 74 03           CMP BRCOL+1
16AE  D0 06              BNE M2
16B0  20 43 17    M1     JSR PULL
16B3  4C 79 16           JMP BEGIN
16B6  AD 3E 03    M2     LDA Y1          ;TEST BELOW POINT
16B9  C9 C7              CMP #YMAX-1
16BB  F0 20              BEQ M4
16BD  20 2F 17           JSR SETUP
16C0  E6 5B              INC T3
16C2  20 19 16           JSR POINTT
16C5  AD 72 03           LDA PTBR
16C8  CD 73 03           CMP BRCOL
16CB  F0 0B              BEQ M3
16CD  CD 74 03           CMP BRCOL+1
16D0  F0 06              BEQ M3
16D2  20 83 17           JSR PUSHU
16D5  4C DD 16           JMP M4
16D8  A9 00       M3     LDA #$00
16DA  8D 76 03           STA USW
16DD  A9 00       M4     LDA #$00        ;TEST ABOVE POINT
16DF  CD 3E 03           CMP Y1
16E2  F0 20              BEQ M6
16E4  20 2F 17           JSR SETUP
16E7  C6 5B              DEC T3
16E9  20 19 16           JSR POINTT
16EC  AD 72 03           LDA PTBR
16EF  CD 73 03           CMP BRCOL
16F2  F0 0B              BEQ M5
16F4  CD 74 03           CMP BRCOL+1
16F7  F0 06              BEQ M5
16F9  20 96 17           JSR PUSHL
16FC  4C 04 17           JMP M6
16FF  A9 00       M5     LDA #$00
1701  8D 75 03           STA LSW
1704  20 2F 17    M6     JSR SETUP       ;PLOT POINT
1707  A5 FE              LDA COL+1
1709  85 FD              STA COL
170B  AD 73 03           LDA BRCOL
170E  85 FC              STA PBR
1710  20 02 10           JSR PLOT
1713  38                 SEC
```

```
LOC     CODE        LINE

1714    AD 3C 03            LDA X1
1717    ED 6A 03            SBC CNTR1       ;DECREASE X AND TEST FOR
171A    8D 3C 03            STA X1          ;OUT OF BOUNDS. IF NOT THEN
171D    AD 3D 03            LDA X1+1        ;RETURN TO INNER LOOP
1720    E9 00               SBC #$00
1722    8D 3D 03            STA X1+1
1725    C9 02               CMP #$02
1727    90 03               BCC M7
1729    4C B0 16            JMP M1
172C    4C 9D 16    M7      JMP M
172F    AD 3C 03    SETUP   LDA X1           ;SET UP PARAMETERS FOR
1732    85 59               STA T2          ;DOT ROUTINE
1734    AD 3D 03            LDA X1+1
1737    85 5A               STA T2+1
1739    AD 3E 03            LDA Y1
173C    85 5B               STA T3
173E    A9 00               LDA #$00
1740    85 5C               STA T3+1
1742    60                  RTS
1743    A5 AE       PULL    LDA $AE          ;PULL VALUES OFF USER
1745    C5 C1               CMP $C1         ;STACK PROVIDED THAT
1747    D0 09               BNE PULL1       ;THERE IS STILL SOME VALUE
1749    A5 AF               LDA $AF
174B    C5 C2               CMP $C2
174D    D0 03               BNE PULL1
174F    68                  PLA
1750    68                  PLA
1751    60                  RTS
1752    A5 01       PULL1   LDA $01
1754    29 FE               AND #$FE        ;BASIC ROM OUT
1756    85 01               STA $01
1758    A0 02               LDY #$02        ;GET NEXT STACK
175A    B1 AE               LDA ($AE),Y     ; VALUE
175C    8D 3E 03            STA Y1
175F    88                  DEY
1760    B1 AE               LDA ($AE),Y
1762    8D 3D 03            STA X1+1
1765    88                  DEY
1766    B1 AE               LDA ($AE),Y
1768    8D 3C 03            STA X1
176B    A5 AE               LDA $AE
176D    18                  CLC
176E    69 04               ADC #$04
1770    85 AE               STA $AE
1772    A5 AF               LDA $AF
1774    69 00               ADC #$00
1776    29 3F               AND #$3F
1778    09 A0               ORA #$A0
177A    85 AF               STA $AF
177C    A5 01               LDA $01
177E    09 01               ORA #$01        ;BASIC ROM IN
1780    85 01               STA $01
1782    60                  RTS
1783    AD 76 03    PUSHU   LDA USW          ;PUSH COORDINATES OF
1786    F0 01               BEQ PUSHUC      ;POINT BELOW IF NOT ALREADY
1788    60                  RTS             ;DONE SO
1789    A9 01       PUSHUC  LDA #$01
178B    8D 76 03            STA USW
178E    20 2F 17            JSR SETUP
1791    E6 5B               INC T3
1793    4C A6 17            JMP PUSH
1796    AD 75 03    PUSHL   LDA LSW          ;PUSH COORDINATES OF
1799    F0 01               BEQ PUSHLC      ;POINT ABOVE IF NOT ALREADY
179B    60                  RTS             ;DONE SO
179C    A9 01       PUSHLC  LDA #$01
179E    8D 75 03            STA LSW
```

```
LOC   CODE        LINE

17A1  20 2F 17           JSR SETUP
17A4  C6 5B              DEC T3
17A6  A5 59       PUSH   LDA T2          ;MAIN PART OF PUSH
17A8  A0 00              LDY #$00
17AA  91 C1              STA ($C1),Y
17AC  C8                 INY
17AD  A5 5A              LDA T2+1
17AF  91 C1              STA ($C1),Y
17B1  C8                 INY
17B2  A5 5B              LDA T3
17B4  91 C1              STA ($C1),Y
17B6  A5 C1              LDA $C1
17B8  18                 CLC
17B9  69 04              ADC #$04
17BB  85 C1              STA $C1
17BD  A5 C2              LDA $C2
17BF  69 00              ADC #$00
17C1  29 3F              AND #$3F
17C3  09 A0              ORA #$A0
17C5  85 C2              STA $C2
17C7  60          PUSEXT RTS
17C8  18          FILRT  CLC             ;LINE SCAN FOR EACH NEW LINE
17C9  AD 3C 03           LDA X1         ;TO FIND BORDER
17CC  6D 6A 03           ADC CNTR1
17CF  8D 40 03           STA X2
17D2  85 59              STA T2
17D4  AD 3D 03           LDA X1+1
17D7  69 00              ADC #$00
17D9  8D 41 03           STA X2+1
17DC  85 5A              STA T2+1
17DE  AD 3E 03           LDA Y1
17E1  85 5B              STA T3
17E3  A9 00              LDA #$00
17E5  85 5C              STA T3+1
17E7  AD 41 03           LDA X2+1
17EA  F0 0D              BEQ FILRT2
17EC  C9 01              CMP #$01
17EE  F0 01              BEQ FILRT1
17F0  60                 RTS
17F1  AD 40 03    FILRT1 LDA X2
17F4  C9 40              CMP #$40
17F6  90 01              BCC FILRT2
17F8  60          FILRT4 RTS
17F9  20 19 16    FILRT2 JSR POINTT
17FC  AD 72 03           LDA PTBR
17FF  CD 74 03           CMP BRCOL+1
1802  F0 F4              BEQ FILRT4
1804  CD 73 03           CMP BRCOL
1807  F0 EF              BEQ FILRT4
1809  AD 40 03           LDA X2         ;REPEAT UNTIL FOUND
180C  8D 3C 03           STA X1         ;EDGE OF SCREEN OR A BORDER
180F  AD 41 03           LDA X2+1
1812  8D 3D 03           STA X1+1
1815  4C C8 17           JMP FILRT
1818                     .END
```

DFILL

Abbreviated entry: D(shift)F

Affected Basic abbreviations: None

Token: Hex $DB   Decimal 219

Purpose: To Diamond FILL an enclosed area.

Syntax: Either DFILL X,Y,col,br1,br2
Or   DFILL [X1,Y1,...Xn,Yn],col,br1,br2

Errors: As in FILL

Use: DFILL performs exactly the same function as FILL except that the method used is different. Whereas FILL fills in horizontal lines, DFILL fills in a diamond shape from the start point(s) (see Program 2).

```
5 REM BASIC EQUIVALENT OF THE DFILL COMMAND
10 DIMQ(1000,1)
15 HIRES0,1
17 X=160:Y=100
20 DRAW130;95,140,95,0,1
30 DRAW150,95,170,95,0,1
40 DRAW130,105,170,105,0,1
50 GOSUB1000:NORM
60 END
1000 BB=0
1020 QI=0:QO=0
1030 XP=X:YP=Y
1040 X=XP:Y=YP
1050 IF0<>POINT(X,Y)THEN1140
1055 PLOTXP,YP,0,1
1060 X=XP+1
1070 GOSUB1200
1080 X=XP-1
1090 GOSUB1200
1100 X=XP:Y=YP+1
1110 GOSUB1200
1120 Y=YP-1
1130 GOSUB1200
1140 IFQI=QOTHENRETURN
1150 XP=Q(QO,0):YP=Q(QO,1)
1160 QO=QO+1:IFQO>1000THENQO=0
1170 GOTO1040
1200 IF0<>POINT(X,Y)THENRETURN
1210 Q(QI,0)=X:Q(QI,1)=Y
1220 QI=QI+1
1225 IFQI>1000THENQI=0
1230 RETURN
```

Program 2.

Routine entry point: $1826

Routine operation: DFILL uses the same routine as FILL to read and check the starting coordinates. As in FILL, the main operation of the routine is best described in the form of a flowchart (see Fig. 2.2).

```
LOC    CODE         LINE


1818                      .LIB FILL2
1818   A9 A0        BROK  LDA #$A0
181A   85 C2              STA $C2
181C   85 AF              STA $AF
181E   A9 00              LDA #$00
1820   85 C1              STA $C1
```

```
LOC   CODE       LINE

1822  85 AE            STA $AE
1824  F0 03            BEQ BROK1
1826                ;
1826  20 0A 19      DFILL  JSR GFPARS     ;GET FILL PARAMETERS
1829  20 F2 18      BROK1  JSR FSTUP
182C  20 2F 17      DMAIN  JSR SETUP
182F  20 19 16             JSR POINTT    ;TEST POINT
1832  AD 72 03             LDA PTBR
1835  CD 73 03             CMP BRCOL
1838  F0 0E                BEQ DM4
183A  CD 74 03             CMP BRCOL+1
183D  F0 09                BEQ DM4
183F  20 2F 17             JSR SETUP
1842  20 4D 19             JSR PLOTPT
1845  4C 4B 18             JMP DM1
1848                ;
1848  20 43 17      DM4    JSR PULL       ;GET OFF STACK
184B  20 70 18      DM1    JSR TSTYP      ;TEST ABOVE
184E  20 8C 18             JSR TSTYM     ;TEST BELOW
1851  20 AA 18             JSR TSTXP     ;TEST TO RIGHT
1854  20 CE 18             JSR TSTXM     ;TEST TO LEFT
1857  4C 48 18             JMP DM4       ;AND REPEAT
185A                ;
185A                ;TEST POINT ROUTINE
185A                ; IF THIS ROUTINE RETURNS TO THE
185A                ; ONE THAT CALLED IT, THE POINT
185A                ; IS PLOTABLE
185A                ;
185A  20 19 16      TESTPT JSR POINTT     ;TEST POINT
185D  90 03                BCC TEST02    ;O.K.
185F  68            TEST01 PLA            ;NOT O.K.
1860  68                   PLA
1861  60                   RTS
1862  AD 72 03      TEST02 LDA PTBR       ;PLOTABLE POINT?
1865  CD 73 03             CMP BRCOL
1868  F0 F5                BEQ TEST01    ;NO
186A  CD 74 03             CMP BRCOL+1
186D  F0 F0                BEQ TEST01    ;NO
186F  60                   RTS           ;YES
1870                ;
1870                ;TEST Y PLUS 1
1870                ;
1870  20 82 18      TSTYP  JSR TSTYP1     ;SET COORDINATES
1873  20 5A 18             JSR TESTPT    ;TEST IT
1876  20 82 18             JSR TSTYP1    ;SET COORDINATES
1879  20 4D 19             JSR PLOTPT    ;PLOT IT
187C  20 82 18             JSR TSTYP1    ;SET COORDINATES
187F  4C A6 17             JMP PUSH      ;PUSH TO QUEUE
1882  20 2F 17      TSTYP1 JSR SETUP      ;TRANSFER COORDINATES
1885  E6 5B                INC T3        ; INTO LOCATIONS FOR
1887  D0 02                BNE TSTYP2    ; DOT ROUTINE AND
1889  E6 5C                INC T3+1      ; INCREASE Y BY 1
188B  60            TSTYP2 RTS
188C                ;
188C                ;TEST Y MINUS 1
188C                ;
188C  20 9E 18      TSTYM  JSR TSTYM1     ;SET COORDINATES
188F  20 5A 18             JSR TESTPT    ;TEST THE POINT
1892  20 9E 18             JSR TSTYM1    ;SET COORDINATES
1895  20 4D 19             JSR PLOTPT    ;PLOT THE POINT
1898  20 9E 18             JSR TSTYM1    ;SET COORDINATES
189B  4C A6 17             JMP PUSH      ;PUSH TO QUEUE
189E  20 2F 17      TSTYM1 JSR SETUP      ;TRANSFER COORDINATES
18A1  A5 5B                LDA T3        ; INTO LOCATIONS FOR
18A3  D0 02                BNE TSTYM2    ; DOT ROUTINE AND
18A5  C6 5C                DEC T3+1      ; DECREASE Y BY 1
```

```
LOC   CODE        LINE

18A7  C6 5B       TSTYM2 DEC T3
18A9  60                 RTS
18AA              ;
18AA              ;TEST X PLUS 1 OR 2
18AA              ;
18AA  20 BC 18    TSTXP  JSR TSTXP1     ;SET COORDINATES
18AD  20 5A 18           JSR TESTPT     ;TEST THE POINT
18B0  20 BC 18           JSR TSTXP1     ;SET COORDINATES
18B3  20 4D 19           JSR PLOTPT     ;PLOT THE POINT
18B6  20 BC 18           JSR TSTXP1     ;SET COORDINATES
18B9  4C A6 17           JMP PUSH       ;PUSH TO QUEUE
18BC  20 2F 17    TSTXP1 JSR SETUP      ;TRANSFER COORDINATES
18BF  A5 59              LDA T2         ; INTO LOCATIONS FOR
18C1  18                 CLC            ; DOT ROUTINE AND
18C2  6D 6A 03           ADC CNTR1      ; INCREASE X BY
18C5  85 59              STA T2         ; EITHER 1 OR 2
18C7  A5 5A              LDA T2+1       ; DEPENDING ON THE
18C9  69 00              ADC #$00       ; PLOTTING MODE
18CB  85 5A              STA T2+1
18CD  60                 RTS
18CE              ;
18CE              ;TEST X MINUS 1 OR 2
18CE              ;
18CE  20 E0 18    TSTXM  JSR TSTXM1     ;SET COORDINATES
18D1  30 5A 18           JSR TESTPT     ;TEST THE POINT
18D4  20 E0 18           JSR TSTXM1     ;SET COORDINATES
18D7  20 4D 19           JSR PLOTPT     ;TEST THE POINT
18DA  20 E0 18           JSR TSTXM1     ;SET COORDINATES
18DD  4C A6 17           JMP PUSH       ;PUSH TO QUEUE
18E0  20 2F 17    TSTXM1 JSR SETUP      ;TRANSFER COORDINATES
18E3  A5 59              LDA T2         ; INTO LOCATIONS FOR
18E5  38                 SEC            ; DOT ROUTINE AND
18E6  ED 6A 03           SBC CNTR1      ; DECREASE X BY 1
18E9  85 59              STA T2         ; OR 2 DEPENDING
18EB  A5 5A              LDA T2+1       ;ON WHICH PLOTTING MODE
18ED  E9 00              SBC #$00
18EF  85 5A              STA T2+1
18F1  60                 RTS
18F2              ;
18F2              ;SETUP FOR FILL
18F2              ;
18F2  20 2F 17    FSTUP  JSR SETUP      ;SET IN COORDINATES
18F5  20 CE 0E           JSR DOT        ;IS POINT ON SCREEN?
18F8  90 03              BCC FSTUP1     ;YES
18FA  68                 PLA
18FB  68                 PLA
18FC  60                 RTS
18FD  A5 02       FSTUP1 LDA MODE       ;MULTICOLOUR?
18FF  D0 03              BNE FSTUP2     ;YES
1901  A9 01              LDA #$01       ;NO, X INCREASE OF 1
1903  2C                 .BYT $2C       ;SKIP NEXT INSTRUCTION
1904  A9 02       FSTUP2 LDA #$02       ;X INCREASE OF 2
1906  8D 6A 03           STA CNTR1      ;STORE IT
1909  60                 RTS
190A              ;
190A              ;GET FILL PARAMETERS
190A              ;
190A  A9 A0       GFPARS LDA #$A0       ;INITIAL STACK
190C  85 C2              STA $C2        ; PUSH
190E  85 AF              STA $AF        ; PULL
1910  A9 00              LDA #$00
1912  85 C1              STA $C1        ; PUSH
1914  85 AE              STA $AE        ; PULL
1916  20 5C 19           JSR GXTRA      ;GET X AND Y
1919  20 42 2B           JSR GCB        ;GET COLOUR AND BRUSH
191C  8D 73 03           STA BRCOL
```

```
LOC   CODE         LINE

191F  20 F1 B7            JSR PARAM       ;GET BOUNDARY BRUSH
1922  8E 74 03            STX BRCOL+1     ;STORE IT
1925  A5 02               LDA MODE        ;MULTICOLOUR?
1927  F0 03               BEQ GFPAR1      ;NO
1929  A9 03               LDA #$03        ;AND BRUSH WITH 3
192B  2C                  .BYT $2C        ;SKIP NEXT COMMAND
192C  A9 01        GFPAR1 LDA #$01         ;AND BRUSH WITH 1
192E  2D 74 03            AND BRCOL+1
1931  8D 74 03            STA BRCOL+1     ;STORE IT
1934  AD 0C 29            LDA TY          ;LAST COORDINATE IS
1937  8D 3E 03            STA Y1          ; STARTING COORDINATE
193A  AD 0D 29            LDA TY+1
193D  8D 3F 03            STA Y1+1
1940  AD 09 29            LDA TX
1943  8D 3C 03            STA X1
1946  AD 0A 29            LDA TX+1
1949  8D 3D 03            STA X1+1
194C  60                  RTS
194D               ;
194D               ;PLOT THE POINT
194D               ;
194D  A5 FE        PLOTPT LDA COL+1        ;SET COLOUR
194F  85 FD               STA COL
1951  AD 73 03            LDA BRCOL       ;SET BRUSH
1954  85 FC               STA PBR
1956  4C 02 10            JMP PLOT        ;PLOT THE POINT
1959               ;
1959               ;GET EXTRA STARTS (IF ANY)
1959               ;
1959  4C E3 10     GXTRA1 JMP GXY          ;ONLY 1 START
195C               ;
195C  20 79 00     GXTRA  JSR $0079        ;GET CHAR
195F  C9 5B               CMP #91         ;IS IT '['?
1961  D0 F6               BNE GXTRA1      ;NO
1963  20 73 00            JSR $0073       ;GET CHAR
1966  20 E3 10     GLOOP  JSR GXY          ;GET X AND Y
1969  20 79 00            JSR $0079       ;GET BYTE
196C  C9 5D               CMP #93         ;IS IT ']'
196E  F0 18               BEQ GXTRA3      ;YES
1970  AD 09 29            LDA TX
1973  85 59               STA T2
1975  AD 0A 29            LDA TX+1
1978  85 5A               STA T2+1
197A  AD 0C 29            LDA TY
197D  85 5B               STA T3
197F  20 A6 17            JSR PUSH        ;PUSH TO QUEUE
1982  20 FD AE            JSR $AEFD       ;SCAN PAST ','
1985  4C 66 19            JMP GLOOP       ;GET NEXT
1988               ;
1988  20 73 00     GXTRA3 JSR $0073        ;GET CHARACTER
198B  60                  RTS             ;ALL DONE
198C                      .END
```

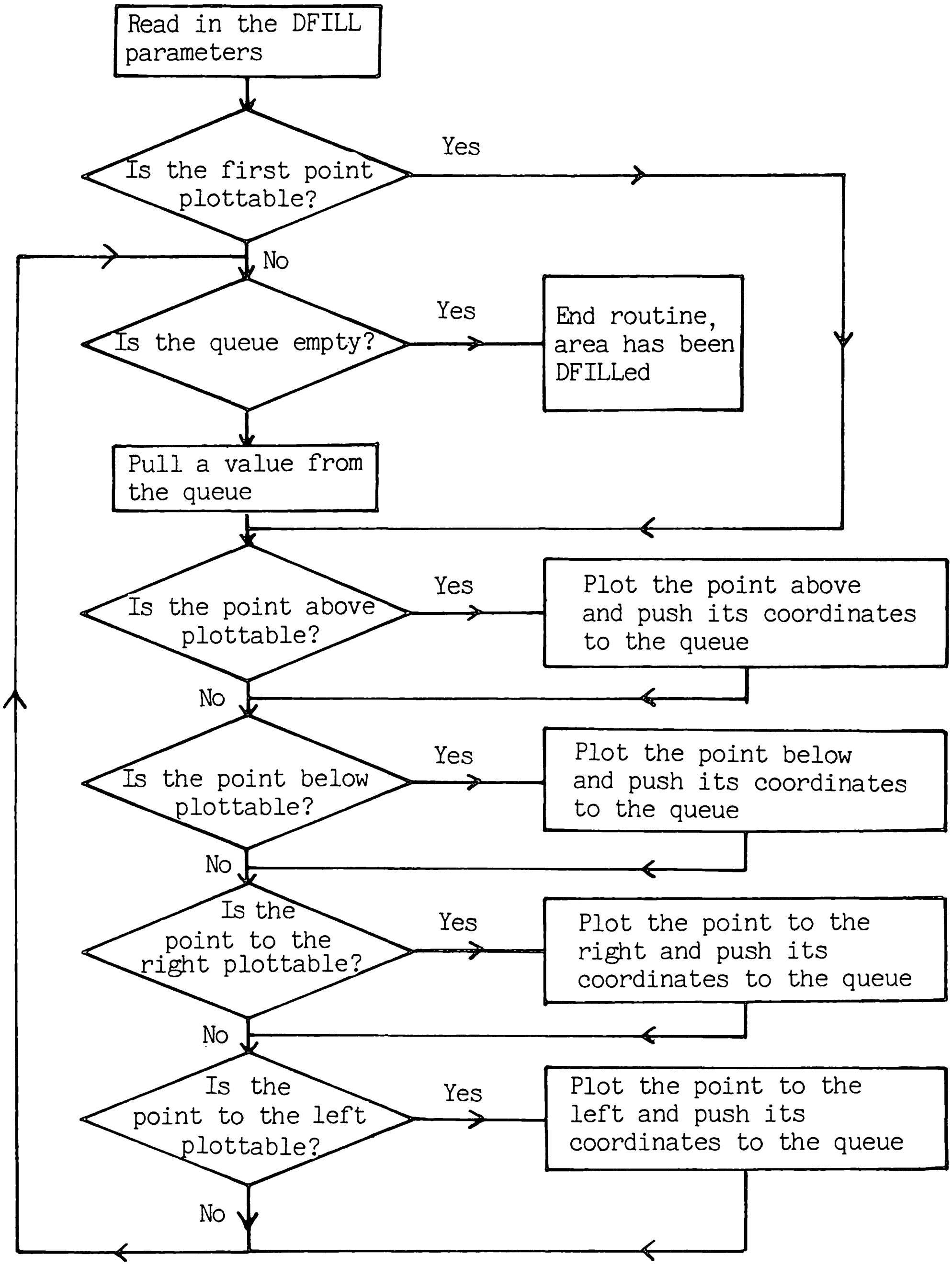


Fig. 2.2. Flow chart of DFILL command.

POLYGON

Abbreviated entry: P(shift)O

Affected Basic abbreviations: POKE – PO(shift)K

Token: Hex $D9 Decimal 217

Syntax: POLYGON sides,XC,YC,rad,col,br,offset

Errors: Illegal quantity – if any parameters are out of their range

Use: POLYGON will plot a regular polygon on the screen. Any polygon from 3 to 2Ø sides can be plotted; the radius value 'rad' must lie between Ø and 255. The centre coordinates 'XC' and 'YC' must lie in the range −32768 to 32767 and the offset value (optional) is the angle in degrees from the normal start point at the top of the polygon. Any offset value is in a clockwise direction from that point.

Routine entry point: $198C

Routine operation: The POLYGON routine uses several Basic arithmetic routines. These are:

Multiply
Subtract
Add
Sin
Cos
Fix to float
Float to fix

The polygon is drawn using lines. Each corner coordinate is calculated, and a line drawn to this point from the previous corner until the polygon has been completed. The calculations for the next X and Y value are:

$$X(n+1)=XC+X(n)*\cos(\theta)+Y(n)*\sin(\theta)$$
$$Y(n+1)=YC+Y(n)*\cos(\theta)-X(n)*\sin(\theta)$$

$$X(1)=XC$$
$$Y(1)=YC+rad$$

The value θ is the angle between one line and the next:

$$\theta=(2*\text{'pi'})/\text{sides\#}$$

```
LOC   CODE        LINE

198C                     .LIB POLYGON
198C  20 9E B7    POLYGN JSR $B79E         ;GET # OF SIDES
198F  8E BA 1B           STX SIDES
1992  E0 03              CPX #3            ;<3?
1994  B0 06              BCS SIDEOK        ;NO
1996  20 52 0E    POLERR JSR NORM
1999  4C 48 B2           JMP $B248
```

```
LOC     CODE            LINE

199C    E0 15           SIDEOK CPX #21          ;>20?
199E    B0 F6                  BCS POLERR       ;YES
19A0    20 FD AE               JSR CHKCOM
19A3    20 E3 10               JSR GXY          ;XC AND YC
19A6    AD 0C 29               LDA TY
19A9    8D BB 1B               STA POLYYC
19AC    AD 0D 29               LDA TY+1
19AF    8D BC 1B               STA POLYYC+1
19B2    AD 09 29               LDA TX
19B5    8D BD 1B               STA POLYXC
19B8    AD 0A 29               LDA TX+1
19BB    8D BE 1B               STA POLYXC+1
19BE    20 F1 B7               JSR PARAM        ;RADIUS
19C1    8E BF 1B               STX RADIUS
19C4    20 42 2B               JSR GCB          ;COLOUR AND BRUSH
19C7                    ;
19C7                    ;CALCULATE START X AND Y
19C7                    ;
19C7    AC BF 1B               LDY RADIUS
19CA    A9 00                  LDA #$00
19CC    20 91 B3               JSR $B391        ;CONVERT RADIUS TO FLOAT
19CF    A2 3D                  LDX #<POLYY1
19D1    A0 1C                  LDY #>POLYY1
19D3    20 D7 BB               JSR $BBD7        ;FAC1 TO Y1
19D6    A2 38                  LDX #<INITY
19D8    A0 1C                  LDY #>INITY
19DA    20 D7 BB               JSR $BBD7        ;FAC1 TO INITIAL Y
19DD    A9 00                  LDA #$00
19DF    A0 00                  LDY #$00
19E1    20 91 B3               JSR $B391        ;FLOAT ZERO
19E4    A2 D4                  LDX #<POLYX1
19E6    A0 1B                  LDY #>POLYX1
19E8    20 D7 BB               JSR $BBD7        ;FAC1 TO X1
19EB    A2 CF                  LDX #<INITX
19ED    A0 1B                  LDY #>INITX
19EF    20 D7 BB               JSR $BBD7        ;FAC1 TO INITIAL X
19F2    20 79 00               JSR $0079
19F5    F0 03                  BEQ POLYMN       ;END OF INPUT
19F7    20 6D 1B               JSR SETOFF       ;CALCULATE OFFSET VAL
19FA                    ;
19FA                    ;MAIN PLOTTING ROUTINE
19FA                    ;
19FA                    ;CALCULATE POSITION IN SIN AND
19FA                    ;COS TABLES
19FA                    ;
19FA    AD BA 1B        POLYMN LDA SIDES
19FD    0A                     ASL A            ;SIDES * 4
19FE    0A                     ASL A
19FF    18                     CLC
1A00    6D BA 1B               ADC SIDES        ;+ SIDES =
1A03    A8                     TAY              ;SIDES * 5
1A04    B9 CF 1B               LDA SINT,Y       ;TRANSFER SIN VAL
1A07    8D C5 1B               STA SINVAL       ;INTO TEMP SIN
1A0A    B9 D0 1B               LDA SINT+1,Y
1A0D    8D C6 1B               STA SINVAL+1
1A10    B9 D1 1B               LDA SINT+2,Y
1A13    8D C7 1B               STA SINVAL+2
1A16    B9 D2 1B               LDA SINT+3,Y
1A19    8D C8 1B               STA SINVAL+3
1A1C    B9 D3 1B               LDA SINT+4,Y
1A1F    8D C9 1B               STA SINVAL+4
1A22    B9 38 1C               LDA COST,Y       ;TRANSFER COS VAL
1A25    8D CA 1B               STA COSVAL       ;INTO TEMP COS
1A28    B9 39 1C               LDA COST+1,Y
1A2B    8D CB 1B               STA COSVAL+1
```

```
LOC   CODE        LINE


1A2E  B9 3A 1C           LDA COST+2,Y
1A31  8D CC 1B           STA COSVAL+2
1A34  B9 3B 1C           LDA COST+3,Y
1A37  8D CD 1B           STA COSVAL+3
1A3A  B9 3C 1C           LDA COST+4,Y
1A3D  8D CE 1B           STA COSVAL+4
1A40  CE BA 1B           DEC SIDES
1A43              ;
1A43  20 F9 1A    POLYLP JSR CALCXY
1A46  20 71 1A           JSR FIXALL
1A49  20 E1 1A           JSR X2TOX1     ;TRANSFER X2 TO X1
1A4C  20 ED 1A           JSR Y2TOY1     ;TRANSFER Y2 TO Y1
1A4F  20 40 11           JSR BOX        ;PLOT LINE
1A52  CE BA 1B           DEC SIDES      ;NEXT COORDINATES
1A55  AD BA 1B           LDA SIDES
1A58  D0 E9              BNE POLYLP
1A5A  A2 04              LDX #$04
1A5C  BD CF 1B    X2RAD  LDA INITX,X
1A5F  9D D9 1B           STA POLYX2,X
1A62  BD 38 1C           LDA INITY,X
1A65  9D 42 1C           STA POLYY2,X
1A68  CA                 DEX
1A69  10 F1              BPL X2RAD
1A6B  20 71 1A           JSR FIXALL
1A6E  4C 40 11           JMP BOX        ;END OF POLYGON
1A71              ;
1A71  20 C2 1A    FIXALL JSR FIXX1       ;FIX X1
1A74  A5 14              LDA $14        ;ADD XC
1A76  18                 CLC
1A77  6D BD 1B           ADC POLYXC
1A7A  8D 3C 03           STA X1
1A7D  A5 15              LDA $15
1A7F  6D BE 1B           ADC POLYXC+1
1A82  8D 3D 03           STA X1+1
1A85  20 C9 1A           JSR FIXY1      ;FIX Y1
1A88  A5 14              LDA $14        ;ADD YC
1A8A  18                 CLC
1A8B  6D BB 1B           ADC POLYYC
1A8E  8D 3E 03           STA Y1
1A91  A5 15              LDA $15
1A93  6D BC 1B           ADC POLYYC+1
1A96  8D 3F 03           STA Y1+1
1A99  20 D0 1A           JSR FIXX2      ;FIX X2
1A9C  A5 14              LDA $14        ;ADD XC
1A9E  18                 CLC
1A9F  6D BD 1B           ADC POLYXC
1AA2  8D 40 03           STA X2
1AA5  A5 15              LDA $15
1AA7  6D BE 1B           ADC POLYXC+1
1AAA  8D 41 03           STA X2+1
1AAD  20 D7 1A           JSR FIXY2      ;FIX Y2
1AB0  A5 14              LDA $14        ;ADD YC
1AB2  18                 CLC
1AB3  6D BB 1B           ADC POLYYC
1AB6  8D 42 03           STA Y2
1AB9  A5 15              LDA $15
1ABB  6D BC 1B           ADC POLYYC+1
1ABE  8D 43 03           STA Y2+1
1AC1  60                 RTS
1AC2              ;
1AC2  A9 D4       FIXX1  LDA #<POLYX1
1AC4  A0 1B              LDY #>POLYX1
1AC6  4C DB 1A           JMP FIXIT
1AC9  A9 3D       FIXY1  LDA #<POLYY1
1ACB  A0 1C              LDY #>POLYY1
```

```
LOC   CODE        LINE

1ACD  4C DB 1A           JMP FIXIT
1AD0  A9 D9       FIXX2  LDA #<POLYX2
1AD2  A0 1B              LDY #>POLYX2
1AD4  4C DB 1A           JMP FIXIT
1AD7  A9 42       FIXY2  LDA #<POLYY2
1AD9  A0 1C              LDY #>POLYY2
1ADB  20 A2 BB    FIXIT  JSR $BBA2       ;MEM TO FAC1
1ADE  4C FB B7           JMP $B7FB       ;FLOAT TO FIX
1AE1              ;
1AE1  A2 04       X2TOX1 LDX #$04
1AE3  BD D9 1B    X2X1LP LDA POLYX2,X
1AE6  9D D4 1B           STA POLYX1,X
1AE9  CA                 DEX
1AEA  10 F7              BPL X2X1LP
1AEC  60                 RTS
1AED              ;
1AED  A2 04       Y2TOY1 LDX #$04
1AEF  BD 42 1C    Y2Y1LP LDA POLYY2,X
1AF2  9D 3D 1C           STA POLYY1,X
1AF5  CA                 DEX
1AF6  10 F7              BPL Y2Y1LP
1AF8  60                 RTS
1AF9              ;
1AF9  A9 D4       CALCXY LDA #<POLYX1
1AFB  A0 1B              LDY #>POLYX1
1AFD  20 A2 BB           JSR $BBA2       ;X1 TO FAC1
1B00  A9 CA              LDA #<COSVAL
1B02  A0 1B              LDY #>COSVAL
1B04  20 8C BA           JSR $BA8C       ;SIN TO FAC2
1B07  20 30 BA           JSR $BA30       ;MULTIPLY
1B0A  A2 D9              LDX #<POLYX2
1B0C  A0 1B              LDY #>POLYX2
1B0E  20 D7 BB           JSR $BBD7       ;RESULT TO X2
1B11  A9 3D              LDA #<POLYY1
1B13  A0 1C              LDY #>POLYY1
1B15  20 A2 BB           JSR $BBA2       ;Y1 TO FAC1
1B18  A9 C5              LDA #<SINVAL
1B1A  A0 1B              LDY #>SINVAL
1B1C  20 8C BA           JSR $BA8C       ;COS TO FAC2
1B1F  20 30 BA           JSR $BA30       ;MULTIPLY
1B22  A9 D9              LDA #<POLYX2
1B24  A0 1B              LDY #>POLYX2
1B26  20 8C BA           JSR $BA8C       ;X2 TO FAC2
1B29  20 6F B8           JSR $B86F       ;ADD FAC2 TO FAC1
1B2C  A2 D9              LDX #<POLYX2
1B2E  A0 1B              LDY #>POLYX2
1B30  20 D7 BB           JSR $BBD7       ;FAC1 TO X2
1B33              ;
1B33              ;NOW CALCULATE Y2
1B33              ;
1B33  A9 3D              LDA #<POLYY1
1B35  A0 1C              LDY #>POLYY1
1B37  20 A2 BB           JSR $BBA2       ;Y1 TO FAC1
1B3A  A9 CA              LDA #<COSVAL
1B3C  A0 1B              LDY #>COSVAL
1B3E  20 8C BA           JSR $BA8C       ;SIN TO FAC2
1B41  20 30 BA           JSR $BA30       ;MULTIPLY
1B44  A2 42              LDX #<POLYY2
1B46  A0 1C              LDY #>POLYY2
1B48  20 D7 BB           JSR $BBD7       ;RESULT IN Y2
1B4B  A9 D4              LDA #<POLYX1
1B4D  A0 1B              LDY #>POLYX1
1B4F  20 A2 BB           JSR $BBA2       ;X1 TO FAC1
1B52  A9 C5              LDA #<SINVAL
1B54  A0 1B              LDY #>SINVAL
```

```
LOC   CODE       LINE

1B56  20 8C BA          JSR $BA8C      ;COS TO FAC2
1B59  20 30 BA          JSR $BA30      ;MULTIPLY
1B5C  A9 42             LDA #<POLYY2
1B5E  A0 1C             LDY #>POLYY2
1B60  20 8C BA          JSR $BA8C      ;RESULT OF 1ST TO FAC2
1B63  20 53 B8          JSR $B853      ;FAC1=FAC2-FAC1
1B66  A2 42             LDX #<POLYY2
1B68  A0 1C             LDY #>POLYY2
1B6A  4C D7 BB          JMP $BBD7      ;FAC1 TO Y2 AND EXIT
1B6D                ;
1B6D                ;GET OFFSET IN DEGREES AND
1B6D                ;SET START COORDINATES
1B6D                ;
1B6D  20 FD AE      SETOFF JSR $AEFD    ;MAKE SURE IS ','
1B70  20 9E AD          JSR $AD9E      ;GET DEGREE OFFSET INTO FAC1
1B73  A9 C0             LDA #<PID180   ;SET FAC1 TO PI/180
1B75  A0 1B             LDY #>PID180
1B77  20 8C BA          JSR $BA8C      ;PUT INTO FAC2
1B7A  20 30 BA          JSR $BA30      ;FAC1=FAC1*FAC2
1B7D  A2 CA             LDX #<COSVAL   ;STORE RADIANS IN TEMP
1B7F  A0 1B             LDY #>COSVAL
1B81  20 D7 BB          JSR $BBD7      ;MOVE FAC1 TO TEMP
1B84  20 6B E2          JSR $E26B      ;CALCULATE SIN(FAC1)
1B87  A2 C5             LDX #<SINVAL   ;STORE SIN AWAY
1B89  A0 1B             LDY #>SINVAL
1B8B  20 D7 BB          JSR $BBD7      ;FAC1 TO SINVAL
1B8E  A9 CA             LDA #<COSVAL   ;MOVE RADIANS TO FAC1
1B90  A0 1B             LDY #>COSVAL
1B92  20 A2 BB          JSR $BBA2      ;TO FAC1
1B95  20 64 E2          JSR $E264      ;CALCULATE COS(FAC1)
1B98  A2 CA             LDX #<COSVAL   ;STORE COS AWAY
1B9A  A0 1B             LDY #>COSVAL
1B9C  20 D7 BB          JSR $BBD7      ;FAC1 TO COSVAL
1B9F  20 F9 1A          JSR CALCXY     ;CALCULATE START
1BA2  20 E1 1A          JSR X2TOX1     ;MOVE TO X1
1BA5  20 ED 1A          JSR Y2TOY1     ;MOVE TO Y1
1BA8                ;
1BA8                ;SET INIT VALS TO X1 AND Y1
1BA8                ;
1BA8  A2 04             LDX #$04
1BAA  BD D4 1B      INITUP LDA POLYX1,X
1BAD  9D CF 1B          STA INITX,X
1BB0  BD 3D 1C          LDA POLYY1,X
1BB3  9D 38 1C          STA INITY,X
1BB6  CA                DEX
1BB7  10 F1             BPL INITUP
1BB9  60                RTS            ;ALL DONE
1BBA                ;
1BBA                ;VARIABLE STORAGE AREA
1BBA                ;
1BBA  00            SIDES  .BYT 0       ;# OF SIDES
1BBB  00 00         POLYYC .WOR 0       ;Y CENTRE CO
1BBD  00 00         POLYXC .WOR 0       ;X CENTRE CO
1BBF  00            RADIUS .BYT 0       ;POLYGON RADIUS
1BC0  7B            PID180 .BYT $7B,$0E,$FA,$35,$12 ;CONSTANT PI/180
1BC1  0E
1BC2  FA
1BC3  35
1BC4  12
1BC5                SINVAL *=*+5        ;WORK SIN
1BCA                COSVAL *=*+5        ;WORK COS
1BCF                SINT                ;SIN TABLE
1BCF                INITX  *=*+5
1BD4                POLYX1 *=*+5
1BD9                POLYX2 *=*+5
```

```
LOC   CODE        LINE

1BDE  80               .BYT 128,93,179,215,68 ;3 SIDES
1BDF  5D
1BE0  B3
1BE1  D7
1BE2  44
1BE3  80               .BYT 128,127,255,255,255 ;4
1BE4  7F
1BE5  FF
1BE6  FF
1BE7  FF
1BE8  80               .BYT 128,115,120,112,153 ;5
1BE9  73
1BEA  78
1BEB  70
1BEC  99
1BED  80               .BYT 128,93,179,215,66 ;6
1BEE  5D
1BEF  B3
1BF0  D7
1BF1  42
1BF2  80               .BYT 128,72,38,27,167 ;7
1BF3  48
1BF4  26
1BF5  1B
1BF6  A7
1BF7  80               .BYT 128,53,4,243,51 ;8
1BF8  35
1BF9  04
1BFA  F3
1BFB  33
1BFC  80               .BYT 128,36,141,186,145 ;9
1BFD  24
1BFE  8D
1BFF  BA
1C00  91
1C01  80               .BYT 128,22,121,24,35 ;10
1C02  16
1C03  79
1C04  18
1C05  23
1C06  80               .BYT 128,10,103,111,198 ;11
1C07  0A
1C08  67
1C09  6F
1C0A  C6
1C0B  7F               .BYT 127,127,255,255,255 ;12
1C0C  7F
1C0D  FF
1C0E  FF
1C0F  FF
1C10  7F               .BYT 127,109,240,50,18 ;13
1C11  6D
1C12  F0
1C13  32
1C14  12
1C15  7F               .BYT 127,94,38,2,106 ;14
1C16  5E
1C17  26
1C18  02
1C19  6A
1C1A  7F               .BYT 127,80,63,201,7 ;15
1C1B  50
1C1C  3F
1C1D  C9
1C1E  07
1C1F  7F               .BYT 127,67,239,21,53 ;16
```

```
LOC   CODE        LINE

1C20  43
1C21  EF
1C22  15
1C23  35
1C24  7F                .BYT 127,56,244,170,234 ;17
1C25  38
1C26  F4
1C27  AA
1C28  EA
1C29  7F                .BYT 127,47,29,67,164 ;18
1C2A  2F
1C2B  1D
1C2C  43
1C2D  A4
1C2E  7F                .BYT 127;38;63,2,66 ;19
1C2F  26
1C30  3F
1C31  02
1C32  42
1C33  7F                .BYT 127,30,55,121,185 ;20
1C34  1E
1C35  37
1C36  79
1C37  B9
1C38              COST                    ;COS TABLE
1C38              INITY  *=*+5
1C3D              POLYY1 *=*+5
1C42              POLYY2 *=*+5
1C47  7F                .BYT 127,255,255,255,252 ;3
1C48  FF
1C49  FF
1C4A  FF
1C4B  FC
1C4C  00                .BYT 0,73,15,218,162 ;4
1C4D  49
1C4E  0F
1C4F  DA
1C50  A2
1C51  7F                .BYT 127,30,55,121,188 ;5
1C52  1E
1C53  37
1C54  79
1C55  BC
1C56  7F                .BYT 127,127,255,255,252 ;6
1C57  7F
1C58  FF
1C59  FF
1C5A  FC
1C5B  80                .BYT 128,31,157,7,21 ;7
1C5C  1F
1C5D  9D
1C5E  07
1C5F  15
1C60  80                .BYT 128,53,4,243,54 ;8
1C61  35
1C62  04
1C63  F3
1C64  36
1C65  80                .BYT 128,68,27,125,22 ;9
1C66  44
1C67  1B
1C68  7D
1C69  16
1C6A  80                .BYT 128,79,27,188,221 ;10
1C6B  4F
```

```
LOC   CODE      LINE

1C6C  1B
1C6D  BC
1C6E  DD
1C6F  80              .BYT 128,87,92,100,59 ;11
1C70  57
1C71  5C
1C72  64
1C73  3B
1C74  80              .BYT 128,93,179,215,68 ;12
1C75  5D
1C76  B3
1C77  D7
1C78  44
1C79  80              .BYT 128,98,173,62,255 ;13
1C7A  62
1C7B  AD
1C7C  3E
1C7D  FF
1C7E  80              .BYT 128,102,165,229,77 ;14
1C7F  66
1C80  A5
1C81  E5
1C82  4D
1C83  80              .BYT 128,105,222,29,120 ;15
1C84  69
1C85  DE
1C86  1D
1C87  78
1C88  80              .BYT 128,108,131,94,121 ;16
1C89  6C
1C8A  83
1C8B  5E
1C8C  79
1C8D  80              .BYT 128,110,182,128,2 ;17
1C8E  6E
1C8F  B6
1C90  80
1C91  02
1C92  80              .BYT 128,112,143,178,18 ;18
1C93  70
1C94  8F
1C95  B2
1C96  12
1C97  80              .BYT 128,114,33,20,40 ;19
1C98  72
1C99  21
1C9A  14
1C9B  28
1C9C  80              .BYT 128,115,120,112,155 ;20
1C9D  73
1C9E  78
1C9F  70
1CA0  9B
1CA1                  .END
```

CIRCLE

Abbreviated entry: C(shift)I

Affected Basic abbreviations: None

Token: Hex $E3   Decimal 227

Purpose: To display circles, ellipses and arcs.

Syntax: CIRCLE CX,CY,XR,YR,sa,ea,col,br

Errors: Illegal quantity – if any parameters are out of their range

Use: CIRCLE is a multipurpose routine that allows the drawing of circles, ellipses and arcs of circles or ellipses. The parameters are as follows:

CX	X coordinate of the centre (−32768 to 32767)
CY	Y coordinate of the centre (−32768 to 32767)
XR	the radius in the X direction (Ø to 255)
YR	the radius in the Y direction (Ø to 255)
sa	the angle in degrees to start plotting (anti-clockwise from top)
ea	the angle in degrees to stop plotting (anti-clockwise from top)
col	the colour (Ø to 15)
br	the brush (Ø to 3)

An example of the use is:

CIRCLE 16Ø,1ØØ,9Ø,9Ø,Ø,36Ø,Ø,1

This will draw a full circle around the centre of the screen with a radius of 9Ø pixels in black using brush 1. Or:

CIRCLE 16Ø,1ØØ,8Ø,8Ø,9Ø,27Ø,Ø,1

will draw a semi-circle (bottom half).

Routine entry point: $1CA1

Routine operation: The circle routine uses 16 bit separate sign integer arithmetic only. Whilst plotting the circle, no trigonometric routines are used at all. The easiest way to demonstrate it is in the Basic program shown (Program 3).

```
5 REM THE BASIC EQUIVALENT OF THE CIRCLE COMMAND
10 HIRES1,1
20 ORIGIN160,100
25 WINDOWON:INPUT"R.X ,R.Y ,S.A, E.A ";RX,RY,SA,EA
26 WINDOWOFF
30 R=65535
31 SA=256*SA*π/180
32 EA=256*EA*π/180
33 A=0
40 FORI=0TOSA
60 A=INT(A-R/256)
70 R=INT(R+A/256)
80 NEXT
100 FORI=SATOEA
150 PLOTINT(RX*A/256)/256,INT(RY*R/256)/256,0,1
160 A=INT(A-R/256)
170 R=INT(R+A/256)
180 NEXT
190 NORM
```

Program 3.

```
LOC   CODE       LINE

1CA1                   .LIB CIRCLE
1CA1             ;***************************
1CA1             ; CIRCLE & ARC ROUTINE
1CA1             ;
1CA1             ; CIRCLE CX,CY,XR,YR,SA,EA,C,B
1CA1             ;***************************
1CA1  20 E3 10   CIRCLE JSR GXY          ;GET CX,CY TO TX,TY
1CA4  20 F1 B7          JSR PARAM       ;GET XR
1CA7  8E 3C 03          STX X1          ;STORE IN X1
1CAA  A9 00             LDA #0
1CAC  8D 3D 03          STA X1+1
1CAF  8D 3F 03          STA Y1+1
1CB2  20 F1 B7          JSR PARAM       ;GET YR
1CB5  8E 3E 03          STX Y1          ;PUT IN Y1
1CB8  20 4F 1E          JSR CIRANG      ;GET SA*PI/180*256
1CBB  8D 5B 03          STA S0+1
1CBE  A5 65             LDA $65
1CC0  8D 5A 03          STA S0
1CC3  20 4F 1E          JSR CIRANG      ;GET EA*PI/180*256
1CC6  8D 5D 03          STA S1+1
1CC9  A5 65             LDA $65
1CCB  8D 5C 03          STA S1
1CCE  20 42 2B          JSR GCB         ;GET COLOUR & BRUSH
1CD1  AD 5A 03          LDA S0          ;CHECK END > START
1CD4  CD 5C 03          CMP S1
1CD7  AD 5B 03          LDA S0+1
1CDA  ED 5D 03          SBC S1+1
1CDD  30 01             BMI CIROK1
1CDF  60                RTS
1CE0  A9 00      CIROK1 LDA #0           ;SET FACM TO 0
1CE2  8D C6 1E          STA FACM
1CE5  8D C7 1E          STA FACM+1
1CE8  8D C8 1E          STA FACM+2
1CEB  8D CD 1E          STA FACT+2
1CEE  8D 52 03          STA TT          ;ZERO COUNTER
1CF1  8D 53 03          STA TT+1
1CF4  A9 FF             LDA #$FF        ;SET FACT TO 65535
1CF6  8D CB 1E          STA FACT
1CF9  8D CC 1E          STA FACT+1
1CFC  AD 52 03   CIRLOO LDA TT
1CFF  CD 5A 03          CMP S0
1D02  D0 08             BNE CIR01
1D04  AD 53 03          LDA TT+1
1D07  CD 5B 03          CMP S0+1
1D0A  F0 06             BEQ CIRLO2
1D0C  20 92 1D   CIR01  JSR CIRBMP       ;CALC NEXT POINT
1D0F  4C FC 1C          JMP CIRLOO
1D12  A5 FE      CIRLO2 LDA COL+1        ;PLOT COLOUR
1D14  85 FD             STA COL
1D16  AD C7 1E          LDA FACM+1      ;DIV BY 256
1D19  8D 00 29          STA IAC1
1D1C  AD C8 1E          LDA FACM+2
1D1F  8D 04 29          STA SIAC
1D22  A9 00             LDA #0
1D24  8D 01 29          STA IAC1+1
1D27  A0 00             LDY #0
1D29  AE 3C 03          LDX X1
1D2C  20 68 2A          JSR MULT        ;MULT BY X RADIUS
1D2F  AD 04 29          LDA SIAC        ;DIV BY 256
1D32  AE 06 29          LDX IACR+1
1D35  A0 00             LDY #0
1D37  20 5B 29          JSR MERSGN      ;MERGE SGN
1D3A  18                CLC
1D3B  8A                TXA
1D3C  6D 09 29          ADC TX
1D3F  85 59             STA T2
```

```
LOC   CODE        LINE

1D41  98                 TYA
1D42  6D 0A 29           ADC TX+1
1D45  85 5A              STA T2+1
1D47  AD CC 1E           LDA FACT+1      ;DIV BY 256
1D4A  8D 00 29           STA IAC1
1D4D  AD CD 1E           LDA FACT+2
1D50  8D 04 29           STA SIAC
1D53  A9 00              LDA #0
1D55  8D 01 29           STA IAC1+1
1D58  A0 00              LDY #0
1D5A  AE 3E 03           LDX Y1
1D5D  20 68 2A           JSR MULT        ;MULT BY Y RADIUS
1D60  AD 04 29           LDA SIAC        ;DIV BY 256
1D63  AE 06 29           LDX IACR+1
1D66  A0 00              LDY #0
1D68  20 5B 29           JSR MERSGN      ;MERGE SGN
1D6B  18                 CLC
1D6C  8A                 TXA
1D6D  6D 0C 29           ADC TY
1D70  85 5B              STA T3
1D72  98                 TYA
1D73  6D 0D 29           ADC TY+1
1D76  85 5C              STA T3+1
1D78  20 02 10           JSR PLOT        ;PLOT POINT
1D7B  AD 52 03           LDA TT
1D7E  CD 5C 03           CMP S1          ;CHECK FINISHED
1D81  D0 09              BNE CIR03
1D83  AD 53 03           LDA TT+1
1D86  CD 5D 03           CMP S1+1
1D89  D0 01              BNE CIR03
1D8B  60                 RTS
1D8C  20 92 1D    CIR03  JSR CIRBMP       ;CALC NEXT POINT
1D8F  4C 12 1D           JMP CIRLO2
1D92              ;
1D92  EE 52 03    CIRBMP INC TT
1D95  D0 03              BNE CIRTNC
1D97  EE 53 03           INC TT+1
1D9A  AD C6 1E    CIRTNC LDA FACM         ;FACM=FACM-FACT/256
1D9D  8D 00 29           STA IAC1
1DA0  AD C7 1E           LDA FACM+1
1DA3  8D 01 29           STA IAC1+1
1DA6  AD C8 1E           LDA FACM+2
1DA9  8D 04 29           STA SIAC
1DAC  AD CD 1E           LDA FACT+2
1DAF  AE CC 1E           LDX FACT+1
1DB2  A0 00              LDY #0
1DB4  20 F9 1D           JSR SUB         ;IACR=IAC1-A.X.Y
1DB7  AD 05 29           LDA IACR
1DBA  8D C6 1E           STA FACM
1DBD  AD 06 29           LDA IACR+1
1DC0  8D C7 1E           STA FACM+1
1DC3  AD 04 29           LDA SIAC
1DC6  8D C8 1E           STA FACM+2
1DC9  AD CB 1E           LDA FACT        ;FACT=FACT+FACM/256
1DCC  8D 00 29           STA IAC1
1DCF  AD CC 1E           LDA FACT+1
1DD2  8D 01 29           STA IAC1+1
1DD5  AD CD 1E           LDA FACT+2
1DD8  8D 04 29           STA SIAC
1DDB  AD C8 1E           LDA FACM+2
1DDE  AE C7 1E           LDX FACM+1
1DE1  A0 00              LDY #0
1DE3  20 FB 1D           JSR ADD         ;IACR=IAC1+A.X.Y
1DE6  AD 05 29           LDA IACR
1DE9  8D CB 1E           STA FACT
```

```
LOC   CODE        LINE

1DEC  AD 06 29          LDA IACR+1
1DEF  8D CC 1E          STA FACT+1
1DF2  AD 04 29          LDA SIAC
1DF5  8D CD 1E          STA FACT+2
1DF8  60                RTS
1DF9              ;
1DF9  49 FF       SUB   EOR #$FF
1DFB  4D 04 29    ADD   EOR SIAC
1DFE  D0 1B             BNE ADDS        ;HANDLE DIF SIGNS
1E00  18                CLC
1E01  8A                TXA             ;DIRECT ADD
1E02  6D 00 29          ADC IAC1
1E05  8D 05 29          STA IACR
1E08  98                TYA
1E09  6D 01 29          ADC IAC1+1
1E0C  8D 06 29          STA IACR+1
1E0F  B0 01             BCS ADDSCO
1E11  60                RTS
1E12  A9 FF       ADDSCO LDA #$FF        ;APPROX CORRECT OVERFLOW
1E14  8D 05 29          STA IACR
1E17  8D 06 29          STA IACR+1
1E1A  60                RTS
1E1B  38          ADDS  SEC              ;CALC POS DIF
1E1C  8A                TXA
1E1D  ED 00 29          SBC IAC1
1E20  8D 05 29          STA IACR
1E23  98                TYA
1E24  ED 01 29          SBC IAC1+1
1E27  8D 06 29          STA IACR+1
1E2A  90 09             BCC ADDSF
1E2C  A9 FF             LDA #$FF
1E2E  4D 04 29          EOR SIAC
1E31  8D 04 29          STA SIAC
1E34  60                RTS
1E35  8E 02 29    ADDSF STX IAC2
1E38  8C 03 29          STY IAC2+1
1E3B  38                SEC
1E3C  AD 00 29          LDA IAC1
1E3F  ED 02 29          SBC IAC2
1E42  8D 05 29          STA IACR
1E45  AD 01 29          LDA IAC1+1
1E48  ED 03 29          SBC IAC2+1
1E4B  8D 06 29          STA IACR+1
1E4E  60                RTS
1E4F              ;
1E4F  20 FD AE    CIRANG JSR $AEFD       ;CHECK ','
1E52  20 8A AD          JSR $AD8A       ;GET SA
1E55  A9 C0             LDA #<PID180
1E57  A0 1B             LDY #>PID180
1E59  20 28 BA          JSR $BA28       ;MULT BY PI/180
1E5C  A5 61             LDA $61
1E5E  18                CLC
1E5F  69 08             ADC #8
1E61  B0 0A             BCS SAERR       ;ERROR
1E63  85 61             STA $61
1E65  20 BF B1          JSR $B1BF       ;FIX IT
1E68  A5 64             LDA $64
1E6A  30 01             BMI SAERR       ;ERROR
1E6C  60                RTS
1E6D  20 52 0E    SAERR JSR NORM
1E70  A2 0E             LDX #$0E        ;ILL QUANTITY
1E72  4C 37 A4          JMP $A437       ;ERROR
1E75                    .END
```

MAT

Abbreviated entry: M(shift)A

Token: Hex $EØ   Decimal 224

Purpose: To perform arithmetic operations on entire arrays, assuming their contents to be matrices.

Syntax: MAT array name = (arithmetic expression). Assign scalar value to all elements of the matrix in the array. Brackets are required around the expression.

MAT array name = array name. Assign all corresponding elements from one array to another. Both arrays must be numeric and of the same dimensions.

MAT array name = array name operator (arithmetic expression) or

MAT array name = (arithmetic expression) operator array name. The operator may be + or * to add or multiply a matrix with a scalar value.

MAT array name = array name + array name. All three arrays must be of the same dimensions and numeric.

MAT array name = array name * array name. Array sizes must follow the convention for matrix multiplication i.e. (a × c) = (a × b)*(b × c), where a,b,c are the array sizes in the DIM statement plus 1 (element Ø is used).

The MAT command will only accept arrays of 1 or 2 dimensions, of only numeric type and with not more than 255 elements in either dimension.

Errors: Syntax error – when the expression is not in brackets or an illegal operator is used
Type mismatch – for string arrays
Bad subscript – for arrays of incorrect size etc.

Use: High speed matrix arithmetic is approximately eight times faster than an equivalent basic subroutine. Using this command also saves the use of nested FOR...NEXT loops, thereby reducing the chances of an Out of memory error due to the stack being full. Since most versions of Basic on mainframe computers have full matrix arithmetic, this subset of the full MAT command will be useful in converting programs to run on the CBM 64. Matrix arithmetic is often used in programs handling large amounts of numbers in linear equations.

The routine uses the simple convention that a matrix of size a × b will be stored in an array dimensioned by DIM A(a−1,b−1). This means that a routine to read a 5 × 2 matrix from data statements would be:

```
DIM A(4,1)

FOR I = Ø TO 4
FOR J = Ø TO 1
READ A(I,J)
NEXT J,I
```

```
DATA ∅, 4
DATA 3 , 5
DATA −5,3.45
DATA 1 , 1
DATA .4,−4
```

To print an array use a routine like:

```
FOR I = ∅ TO 4
FOR J = ∅ TO 1
PRINT A(I,J),
NEXT J
PRINT
NEXT I
```

The matrix multiplication is equivalent to: (a × c) = (a × b) * (b × c).

```
DIM A(a−1,c−1),B(a−1,b−1)c,(b−1,c−1)
MAT A = B * C
```

is the same as but faster than:

```
FOR I = ∅ TO a−1
FOR J = ∅ TO c−1
T = ∅
FOR K = ∅ TO b−1
T = T + B(J,K) * C(K,I)
NEXT K
A(J,I) = T
NEXT J
NEXT I
```

Routine entry: $1EE1

Routine operation: The MAT routine uses the following BASIC ROM calls.

$AEF1 - Evaluate expression in brackets
$BBD4 - FAC#1 to memory (X.Y)
$BBA2 - Memory (X.Y) to FAC#1
$B1BF - Float to fixed
$B391 - Fixed to float
$B867 - Memory (A.Y) + FAC#1 to FAC #1
$B85∅ - Memory (A.Y) − FAC#1 to FAC#1
$BA28 - Memory (A.Y) * FAC#1 to FAC#1

The routine for assignment will, for speed, perform just a block memory move if the two arrays are both of the same type e.g. both integer. The multiply routine works in the same way as the Basic version above. It calculates the address of the next element required just by adding a pre-calculated offset for speed.

Readers are advised to consult a standard mathematics textbook for details of matrix arithmetic.

```
LOC    CODE        LINE

1E75                     .LIB MAT
1E75               ;************************
1E75               ; 16 BIT UNSIGNED MULTIPLY
1E75               ;************************
1E75               ; WAREA = N1 * N2
1E75               ;
1E75   00 00       N1      .WOR 0
1E77   00 00       N2      .WOR 0
1E79   00 00       RESULT  .WOR 0
1E7B               ;
1E7B   A9 00       MMULT   LDA #0          ;ZERO RESULT
1E7D   8D 79 1E            STA RESULT
1E80   8D 7A 1E            STA RESULT+1
1E83   AD 77 1E            LDA N2         ; END IF N2=0
1E86   0D 78 1E            ORA N2+1
1E89   F0 08               BEQ MMULT2
1E8B   AD 75 1E    MMULT1  LDA N1          ;N1 = 0 ?
1E8E   0D 76 1E            ORA N1+1
1E91   D0 01               BNE MMULT3
1E93   60          MMULT2  RTS
1E94   A9 01       MMULT3  LDA #1          ;IF BIT 0 OF N1
1E96   2D 75 1E            AND N1         ;THEN ADD N2 TO RESULT
1E99   F0 13               BEQ MMULT4
1E9B   18                  CLC            ;ADD N2 TO RESULT
1E9C   AD 77 1E            LDA N2
1E9F   6D 79 1E            ADC RESULT
1EA2   8D 79 1E            STA RESULT
1EA5   AD 78 1E            LDA N2+1
1EA8   6D 7A 1E            ADC RESULT+1
1EAB   8D 7A 1E            STA RESULT+1
1EAE   0E 77 1E    MMULT4  ASL N2          ;N2 = N2 * 2
1EB1   2E 78 1E            ROL N2+1
1EB4   4E 76 1E            LSR N1+1       ;N1 = N1 / 2
1EB7   6E 75 1E            ROR N1
1EBA   4C 8B 1E            JMP MMULT1
1EBD               ;
1EBD               ;
1EBD               ;
1EBD               ;************************
1EBD               ; MATRIX ARITHMETIC
1EBD               ;************************
1EBD               ;
1EBD               ISNALF = $B113
1EBD               CHRGOT = $79
1EBD               CHRGET = $73
1EBD   00 00       VNAME1 .WOR 0           ;VARIABLE NAMES
1EBF   00          VTYPE1 .BYT 0
1EC0   00 00       VNAME2 .WOR 0
1EC2   00          VTYPE2 .BYT 0
1EC3   00 00       VNAME3 .WOR 0
1EC5   00          VTYPE3 .BYT 0
1EC6               FACM    * = *+5         ;TEMP FLOATING STORE
1ECB               FACT    * = *+5         ;TEMP FLOATING STORE
1ED0   00 00       VSIZE1 .WOR 0           ;ARRAY SIZES
1ED2   00 00       VSIZE2 .WOR 0
1ED4   00 00       VSIZE3 .WOR 0
1ED6   00          OPTYPE .BYT 0           ;OPERAND TYPE
1ED7               VPTR1  = $FB
1ED7               VPTR2  = $FD
1ED7               VPTR3  = $9E
1ED7   00 00       VSTT1  .WOR 0
1ED9   00 00       VSTT2  .WOR 0
1EDB   00 00       VSTT3  .WOR 0
1EDD   00 00       TE1    .WOR 0
1EDF   00 00       TE2    .WOR 0
1EE1               ;
```

```
LOC   CODE        LINE

1EE1              ;
1EE1              ;
1EE1  8D BD 1E    MAT     STA VNAME1      ;GET FIRST ARRAY
1EE4  20 13 B1            JSR ISNALF      ;NAME AND CHECK
1EE7  B0 03               BCS CHOK        ; LEGAL
1EE9  4C 08 AF            JMP $AF08       ; SYNTAX
1EEC  A9 00       CHOK    LDA #0
1EEE  8D BE 1E            STA VNAME1+1
1EF1  8D C1 1E            STA VNAME2+1
1EF4  8D C4 1E            STA VNAME3+1
1EF7  8D BF 1E            STA VTYPE1
1EFA  8D C2 1E            STA VTYPE2
1EFD  8D C5 1E            STA VTYPE3
1F00  20 73 00            JSR CHRGET
1F03  90 05               BCC CHOK1
1F05  20 13 B1            JSR ISNALF
1F08  90 0D               BCC EDVNA1      ;GO CHECK FOR % $ =
1F0A  8D BE 1E    CHOK1   STA VNAME1+1
1F0D  20 73 00    LNE     JSR CHRGET      ;SCAN PAST REST
1F10  90 FB               BCC LNE         ;OF VAR NAME
1F12  20 13 B1            JSR ISNALF
1F15  B0 F6               BCS LNE
1F17  C9 24       EDVNA1  CMP #'$         ;CHECK FOR STRING
1F19  D0 05               BNE NSTR1
1F1B  A2 16       TYMISE  LDX #22
1F1D  4C 37 A4            JMP $A437       ;TYPE MISMATCH
1F20  C9 25       NSTR1   CMP #'%
1F22  D0 06               BNE NTINT1      ;NOT INTEGER ARRAY
1F24  CE BF 1E            DEC VTYPE1      ;SET TYPE FLAG TO $FF
1F27  20 73 00            JSR CHRGET      ;GET NEXT CHAR
1F2A  C9 B2       NTINT1  CMP #$B2        ; TOKEN FOR =
1F2C  F0 03               BEQ FOEQ
1F2E  4C 08 AF            JMP $AF08       ;SYNTAX NOT =
1F31  20 73 00    FOEQ    JSR CHRGET
1F34  C9 28               CMP #'(         ;CHECK FOR ( EXP. )
1F36  D0 16               BNE NTEXP2
1F38  20 F1 AE            JSR $AEF1       ;EVAL. EXP. IN ( )
1F3B  A5 0D               LDA $0D         ;CHECK NUMERIC
1F3D  D0 DC               BNE TYMISE
1F3F  A2 C6               LDX #<FACM      ;FAC#1 TO FACM
1F41  A0 1E               LDY #>FACM
1F43  20 D4 BB            JSR $BBD4
1F46  A2 01               LDX #1          ; SET TYPE FLAG TO CONST
1F48  8E C2 1E            STX VTYPE2
1F4B  4C 83 1F            JMP CHKOP
1F4E  20 13 B1    NTEXP2  JSR ISNALF      ;GET NAME
1F51  B0 03               BCS CHOK2       ;CHECK LEGAL
1F53  4C 08 AF            JMP $AF08       ;SYNTAX
1F56  8D C0 1E    CHOK2   STA VNAME2
1F59  20 73 00            JSR CHRGET      ;GET SECOND CHAR
1F5C  90 05               BCC CHOK2A      ;NUMBER?
1F5E  20 13 B1            JSR ISNALF
1F61  90 0D               BCC EDVNA2      ;CHECK FOR $ %
1F63  8D C1 1E    CHOK2A  STA VNAME2+1
1F66  20 73 00    LNE2    JSR CHRGET      ;SCAN TO END
1F69  90 FB               BCC LNE2        ;OF VARIABLE NAME
1F6B  20 13 B1            JSR ISNALF
1F6E  B0 F6               BCS LNE2
1F70  C9 24       EDVNA2  CMP #'$         ;CHECK FOR '$'
1F72  D0 05               BNE NSTR2
1F74  A2 16               LDX #22         ; TYPE MISMATCH
1F76  4C 37 A4            JMP $A437
1F79  C9 25       NSTR2   CMP #'%         ;CHECK IF INTEGER
1F7B  D0 06               BNE CHKOP
1F7D  20 73 00            JSR CHRGET
1F80  CE C2 1E            DEC VTYPE2      ;SET INTEGER FLAG
```

```
LOC   CODE       LINE

1F83  A2 00      CHKOP  LDX #0        ;CHECK OPERAND TYPE
1F85  8E D6 1E          STX OPTYPE
1F88  20 79 00          JSR CHRGOT   ;END STATEMENT?
1F8B  D0 03             BNE NASSIG
1F8D  4C 12 20          JMP DOMAT
1F90  EE D6 1E   NASSIG INC OPTYPE
1F93  C9 AA             CMP #$AA     ;CHECK FOR ADD +
1F95  F0 11             BEQ GETV3
1F97  EE D6 1E          INC OPTYPE
1F9A  C9 AB             CMP #$AB     ;CHECK FOR SUB -
1F9C  F0 0A             BEQ GETV3
1F9E  EE D6 1E          INC OPTYPE
1FA1  C9 AC             CMP #$AC     ;CHECK FOR MULT *
1FA3  F0 03             BEQ GETV3
1FA5  4C 08 AF          JMP $AF08    ;SYNTAX
1FA8  20 73 00   GETV3  JSR CHRGET
1FAB  C9 28             CMP #'(      ;CHECK FOR ( EXP )
1FAD  D0 28             BNE NTEXP3
1FAF  AD C2 1E          LDA VTYPE2   ;CHECK TYPE2 FOR
1FB2  C9 01             CMP #1       ;BEING  CONSTANT
1FB4  D0 03             BNE BEXPOK
1FB6  4C 08 AF          JMP $AF08    ;SYNTAX
1FB9  20 F1 AE   BEXPOK JSR $AEF1     ;EVAL EXP
1FBC  A5 0D             LDA $0D
1FBE  F0 03             BEQ NUMOK
1FC0  4C 1B 1F          JMP TYMISE   ;TYPE MISMATCH
1FC3  A2 C6      NUMOK  LDX #<FACM    ;FAC#1 TO FACM
1FC5  A0 1E             LDY #>FACM
1FC7  20 D4 BB          JSR $BBD4
1FCA  A9 01             LDA #1       ;SET TYPE FLAG TO CONST
1FCC  8D C5 1E          STA VTYPE3
1FCF  20 79 00          JSR CHRGOT   ;END OF STATEMENT?
1FD2  F0 3E             BEQ DOMAT
1FD4  4C 08 AF   SYNTE  JMP $AF08     ;SYNTAX
1FD7  20 13 B1   NTEXP3 JSR ISNALF    ;GET ARRAY NAME
1FDA  90 F8             BCC SYNTE    ;SYNTAX ERROR
1FDC  8D C3 1E          STA VNAME3
1FDF  20 73 00          JSR CHRGET
1FE2  F0 2E             BEQ DOMAT    ; : OR END OF LINE
1FE4  90 05             BCC CHOK3
1FE6  20 13 B1          JSR ISNALF
1FE9  90 0F             BCC EDVNA3
1FEB  8D C4 1E   CHOK3  STA VNAME3+1
1FEE  20 73 00   LNE3   JSR CHRGET
1FF1  F0 1F             BEQ DOMAT
1FF3  90 F9             BCC LNE3
1FF5  20 13 B1          JSR ISNALF
1FF8  B0 F4             BCS LNE3
1FFA  C9 24      EDVNA3 CMP #'$       ;IS IT A STRING?
1FFC  D0 05             BNE NSTR3
1FFE  A2 16             LDX #22
2000  4C 37 A4          JMP $A437
2003  C9 25      NSTR3  CMP #'%       ;IS IT INTEGER?
2005  D0 08             BNE NTINT3
2007  CE C5 1E          DEC VTYPE3
200A  20 73 00          JSR CHRGET   ;NEXT CHAR
200D  F0 03             BEQ DOMAT
200F  4C 08 AF   NTINT3 JMP $AF08     ;SYNTAX
2012  AD BF 1E   DOMAT  LDA VTYPE1    ;FIND ARRAY 1
2015  F0 10             BEQ V1REAL
2017  A9 80             LDA #128     ;SET HI BITS ARRAY NAME
2019  0D BD 1E          ORA VNAME1
201C  8D BD 1E          STA VNAME1
201F  A9 80             LDA #128
2021  0D BE 1E          ORA VNAME1+1
2024  8D BE 1E          STA VNAME1+1
```

```
LOC    CODE          LINE

2027   20 41 22      V1REAL JSR FINDAR      ;FIND ARRAY ADDR
202A   8E D0 1E             STX VSIZE1
202D   8C D1 1E             STY VSIZE1+1
2030   A5 FB                LDA VPTR1       ;STORE IT
2032   8D D7 1E             STA VSTT1
2035   A5 FC                LDA VPTR1+1
2037   8D D8 1E             STA VSTT1+1
203A   AD C2 1E             LDA VTYPE2
203D   C9 01                CMP #1
203F   F0 2A                BEQ GAR3        ;EXPRESSION
2041   AD C2 1E             LDA VTYPE2      ;SET UP ARRAY NAME 2
2044   29 80                AND #$80        ;FOR SEARCH ROUTINE
2046   8D DD 1E             STA TE1
2049   0D C0 1E             ORA VNAME2
204C   8D BD 1E             STA VNAME1
204F   AD C1 1E             LDA VNAME2+1
2052   0D DD 1E             ORA TE1
2055   8D BE 1E             STA VNAME1+1
2058   20 41 22             JSR FINDAR      ;FIND ADDRESS ARRAY 2
205B   8E D2 1E             STX VSIZE2
205E   8C D3 1E             STY VSIZE2+1
2061   A5 FB                LDA VPTR1
2063   8D D9 1E             STA VSTT2
2066   A5 FC                LDA VPTR1+1
2068   8D DA 1E             STA VSTT2+1
206B   AD D6 1E      GAR3   LDA OPTYPE      ;ARRAY 3?
206E   F0 31                BEQ DOMATA      ;NO ARRAY 3
2070   AD C5 1E             LDA VTYPE3
2073   C9 01                CMP #1          ;IS IT A CONSTANT?
2075   F0 2A                BEQ DOMATA      ;YES
2077   29 80                AND #$80        ;IS ARRAY 3 INTEGER?
2079   8D DD 1E             STA TE1
207C   AD C3 1E             LDA VNAME3
207F   0D DD 1E             ORA TE1
2082   8D BD 1E             STA VNAME1
2085   AD C4 1E             LDA VNAME3+1
2088   0D DD 1E             ORA TE1
208B   8D BE 1E             STA VNAME1+1
208E   20 41 22             JSR FINDAR      ;FIND ARRAY 3
2091   8E D4 1E             STX VSIZE3
2094   8C D5 1E             STY VSIZE3+1
2097   A5 FB                LDA VPTR1
2099   8D DB 1E             STA VSTT3
209C   A5 FC                LDA VPTR1+1
209E   8D DC 1E             STA VSTT3+1
20A1   AD D6 1E      DOMATA LDA OPTYPE      ;SET A JUMP VECTOR
20A4   0A                   ASL A           ;FOR OPERATION
20A5   AA                   TAX
20A6   BD B7 20             LDA OPJTAB,X
20A9   8D B5 20             STA OPJMP
20AC   BD B8 20             LDA OPJTAB+1,X
20AF   8D B6 20             STA OPJMP+1
20B2   6C B5 20             JMP (OPJMP)
20B5                 ;
20B5   00 00         OPJMP  .WOR 0          ;JUMP VECTOR
20B7   BF 20         OPJTAB .WOR ASSGN      ;JUMP TABLE
20B9   CA 22                .WOR ADDSUB
20BB   CA 22                .WOR ADDSUB
20BD   33 24                .WOR AMULT
20BF                 ; ***  MAT AA = C
20BF   A9 01         ASSGN  LDA #1
20C1   CD C2 1E             CMP VTYPE2
20C4   F0 03                BEQ ASSIC
20C6   4C 2C 21             JMP ASARAR
20C9   A2 05         ASSIC  LDX #5          ;ARRAY =CONSTANT
20CB   AD BF 1E             LDA VTYPE1
```

```
LOC   CODE          LINE

20CE  F0 16                 BEQ ASSR1
20D0  A9 C6                 LDA #<FACM       ;FACM TO FAC#1
20D2  A0 1E                 LDY #>FACM
20D4  20 A2 BB              JSR $BBA2
20D7  20 BF B1              JSR $B1BF        ;FLOAT TO FIXED
20DA  A5 64                 LDA $64          ;STORE INT IN FACM
20DC  8D C6 1E              STA FACM
20DF  A5 65                 LDA $65
20E1  8D C7 1E              STA FACM+1
20E4  A2 02                 LDX #2
20E6  8E C2 1E     ASSR1    STX VTYPE2        ;STORE ELEMENT LENGTH
20E9  A9 00                 LDA #0           ;CALC NUMBER OF ELEMENTS
20EB  8D 76 1E              STA N1+1
20EE  8D 78 1E              STA N2+1
20F1  AD D0 1E              LDA VSIZE1
20F4  8D 75 1E              STA N1
20F7  AD D1 1E              LDA VSIZE1+1
20FA  8D 77 1E              STA N2
20FD  20 7B 1E              JSR MMULT        ;RESULT =N1 * N2
2100  20 89 23              JSR TRPT1        ;COPY POINTER TO ZERO PAGE
2103  A0 00                 LDY #0
2105  A2 00        ASLOOP   LDX #0            ;FACM TO ARRAY
2107  BD C6 1E     ASLOP    LDA FACM,X
210A  91 FB                 STA (VPTR1),Y
210C  E8                    INX
210D  E6 FB                 INC VPTR1
210F  D0 02                 BNE ASNC
2111  E6 FC                 INC VPTR1+1
2113  EC C2 1E     ASNC     CPX VTYPE2
2116  D0 EF                 BNE ASLOP
2118  AD 79 1E              LDA RESULT
211B  D0 03                 BNE ASNC9
211D  CE 7A 1E              DEC RESULT+1
2120  CE 79 1E     ASNC9    DEC RESULT        ;ARRAY FILLED?
2123  AD 79 1E              LDA RESULT
2126  0D 7A 1E              ORA RESULT+1
2129  D0 DA                 BNE ASLOOP
212B  60                    RTS
212C               ;
212C  A2 05        ASARAR   LDX #5            ;SET VAR LENGTH
212E  AD BF 1E              LDA VTYPE1
2131  F0 02                 BEQ ASR1R
2133  A2 02                 LDX #2
2135  8E BF 1E     ASR1R    STX VTYPE1
2138  A2 05                 LDX #5
213A  AD C2 1E              LDA VTYPE2
213D  F0 02                 BEQ ASR2R
213F  A2 02                 LDX #2
2141  8E C2 1E     ASR2R    STX VTYPE2
2144  AD D0 1E              LDA VSIZE1       ;COMPARE ARRAY SIZES
2147  CD D2 1E              CMP VSIZE2
214A  F0 05                 BEQ ASRSOK
214C  A2 12        ASRSUB   LDX #$12          ;? BAD SUBSCRIPT ERROR
214E  4C 37 A4              JMP $A437
2151  AD D1 1E     ASRSOK   LDA VSIZE1+1
2154  CD D3 1E              CMP VSIZE2+1
2157  D0 F3                 BNE ASRSUB       ; ERROR
2159  AD BF 1E              LDA VTYPE1       ;ARRAYS SAME TYPE?
215C  CD C2 1E              CMP VTYPE2
215F  D0 5A                 BNE ASRIR        ;NO
2161  A9 00                 LDA #0           ;CALC SIZE OF ARRAYS
2163  8D 76 1E              STA N1+1
2166  8D 78 1E              STA N2+1
2169  AD D0 1E              LDA VSIZE1
216C  8D 75 1E              STA N1
216F  AD D1 1E              LDA VSIZE1+1
```

```
LOC   CODE         LINE

2172  8D 77 1E            STA N2
2175  20 7B 1E            JSR MMULT
2178  AD 79 1E            LDA RESULT
217B  8D 75 1E            STA N1
217E  AD 7A 1E            LDA RESULT+1
2181  8D 76 1E            STA N1+1
2184  AD BF 1E            LDA VTYPE1
2187  8D 77 1E            STA N2
218A  A9 00               LDA #0
218C  8D 78 1E            STA N2+1
218F  20 7B 1E            JSR MMULT
2192  20 7F 23            JSR TRPT2        ;SET POINTERS TO ARRAYS
2195  A0 00               LDY #0
2197  B1 FD        ASSTLO LDA (VPTR2),Y     ;BLOCK MOVE OF
2199  91 FB               STA (VPTR1),Y    ;LENGTH IN RESULT
219B  E6 FB               INC VPTR1
219D  D0 02               BNE ASSTN1
219F  E6 FC               INC VPTR1+1
21A1  E6 FD        ASSTN1 INC VPTR2
21A3  D0 02               BNE ASSTN2
21A5  E6 FE               INC VPTR2+1
21A7  AD 79 1E     ASSTN2 LDA RESULT
21AA  D0 03               BNE ASSTN3
21AC  CE 7A 1E            DEC RESULT+1
21AF  CE 79 1E     ASSTN3 DEC RESULT
21B2  AD 79 1E            LDA RESULT
21B5  0D 7A 1E            ORA RESULT+1
21B8  D0 DD               BNE ASSTLO
21BA  60                  RTS
21BB  A9 00        ASRIR  LDA #0
21BD  8D 76 1E            STA N1+1
21C0  8D 78 1E            STA N2+1
21C3  AD D0 1E            LDA VSIZE1
21C6  8D 75 1E            STA N1           ;CALC NUMBER OF ELEMENTS
21C9  AD D1 1E            LDA VSIZE1+1
21CC  8D 77 1E            STA N2
21CF  20 7B 1E            JSR MMULT
21D2  20 7F 23            JSR TRPT2
21D5  A0 00        ASRLOP LDY #0
21D7  A2 00               LDX #0           ;ARRAY ELEMENT TO FACM
21D9  B1 FD        ASRLP1 LDA (VPTR2),Y
21DB  9D C6 1E            STA-FACM,X
21DE  E6 FD               INC VPTR2
21E0  D0 02               BNE ASRNC2
21E2  E6 FE               INC VPTR2+1
21E4  E8           ASRNC2 INX
21E5  EC C2 1E            CPX VTYPE2
21E8  D0 EF               BNE ASRLP1
21EA  E0 05               CPX #5
21EC  D0 17               BNE ASRITR
21EE  A9 C6               LDA #<FACM       ;FACM TO FAC#1
21F0  A0 1E               LDY #>FACM
21F2  20 A2 BB            JSR $BBA2
21F5  20 BF B1            JSR $B1BF        ;FLOAT TO FIXED
21F8  A5 64               LDA $64
21FA  8D C6 1E            STA FACM
21FD  A5 65               LDA $65
21FF  8D C7 1E            STA FACM+1
2202  4C 15 22            JMP ASRTM        ;FACM TO ARRAY
2205  AD C6 1E     ASRITR LDA FACM
2208  AC C7 1E            LDY FACM+1
220B  20 91 B3            JSR $B391        ;FIXED TO FLOAT
220E  A2 C6               LDX #<FACM       ;FAC#1 TO FACM
2210  A0 1E               LDY #>FACM
2212  20 D4 BB            JSR $BBD4
2215  A0 00        ASRTM  LDY #0
```

```
LOC   CODE          LINE


2217  A2 00                LDX #0
2219  BD C6 1E      ASRTM1 LDA FACM,X
221C  91 FB                STA (VPTR1),Y
221E  E8                   INX
221F  E6 FB                INC VPTR1
2221  D0 02                BNE ASRNC1
2223  E6 FC                INC VPTR1+1
2225  EC BF 1E      ASRNC1 CPX VTYPE1
2228  D0 EF                BNE ASRTM1
222A  AD 79 1E             LDA RESULT
222D  D0 03                BNE ASRTM3
222F  CE 7A 1E             DEC RESULT+1
2232  CE 79 1E      ASRTM3 DEC RESULT
2235  AD 79 1E             LDA RESULT
2238  0D 7A 1E             ORA RESULT+1
223B  F0 03                BEQ ASREXT
223D  4C D5 21             JMP ASRLOP
2240  60            ASREXT RTS
2241                ;
2241                ; FIND ARRAY
2241  A5 2F         FINDAR LDA $2F          ;START OF ARRAYS
2243  85 FB                STA VPTR1
2245  A5 30                LDA $30
2247  85 FC                STA VPTR1+1
2249  A5 FB         FALOOP LDA VPTR1        ;CMP. END OF ARRAYS
224B  C5 31                CMP $31
224D  D0 0B                BNE FACONT
224F  A5 FC                LDA VPTR1+1
2251  C5 32                CMP $32
2253  D0 05                BNE FACONT
2255  A2 12                LDX #$12        ;? BAD SUBSCRIPT ERROR
2257  20 37 A4             JSR $A437
225A  A0 00         FACONT LDY #0
225C  B1 FB                LDA (VPTR1),Y   ;FIRST CHAR OF NAME
225E  C8                   INY
225F  CD BD 1E             CMP VNAME1
2262  D0 07                BNE FANAR       ;TRY NEXT ARRAY
2264  B1 FB                LDA (VPTR1),Y
2266  CD BE 1E             CMP VNAME1+1
2269  F0 1D                BEQ FAGETS      ; GET ARRAY DATA
226B  C8            FANAR  INY              ;FIND NEXT ARRAY
226C  B1 FB                LDA (VPTR1),Y
226E  8D DD 1E             STA TE1
2271  C8                   INY
2272  B1 FB                LDA (VPTR1),Y
2274  18                   CLC
2275  65 FC                ADC VPTR1+1
2277  85 FC                STA VPTR1+1
2279  AD DD 1E             LDA TE1
227C  18                   CLC
227D  65 FB                ADC VPTR1
227F  85 FB                STA VPTR1
2281  90 02                BCC FANC
2283  E6 FC                INC VPTR1+1
2285  4C 49 22      FANC   JMP FALOOP
2288  A9 01         FAGETS LDA #1           ;GET ARRAY DATA
228A  8D DE 1E             STA TE1+1
228D  C8                   INY
228E  C8                   INY
228F  C8                   INY
2290  B1 FB                LDA (VPTR1),Y
2292  C9 03                CMP #3
2294  30 05                BMI FANDOK
2296  A2 12         FAE1   LDX #$12         ;ERROR MORE THAN 2 DIM
2298  4C 37 A4             JMP $A437
229B  AA            FANDOK TAX
```

```
LOC   CODE       LINE

229C  C8               INY
229D  B1 FB            LDA (VPTR1),Y
229F  D0 F5            BNE FAE1       ;FIRST DIM TOO BIG
22A1  C8               INY
22A2  B1 FB            LDA (VPTR1),Y
22A4  8D DD 1E         STA TE1
22A7  8A               TXA
22A8  CA               DEX
22A9  F0 0B            BEQ FAEX       ;ONE DIM ARRAY
22AB  C8               INY
22AC  B1 FB            LDA (VPTR1),Y
22AE  D0 E6            BNE FAE1       ;SECOND DIM TOO BIG
22B0  C8               INY
22B1  B1 FB            LDA (VPTR1),Y
22B3  8D DE 1E         STA TE1+1
22B6  C8         FAEX  INY
22B7  98               TYA
22B8  18               CLC
22B9  65 FB            ADC VPTR1
22BB  85 FB            STA VPTR1
22BD  A5 FC            LDA VPTR1+1
22BF  69 00            ADC #0
22C1  85 FC            STA VPTR1+1
22C3  AE DD 1E         LDX TE1
22C6  AC DE 1E         LDY TE1+1
22C9  60               RTS
22CA             ;
22CA  20 4A 23   ADDSUB JSR ORDER      ;PUT CONST LAST
22CD  AD D0 1E         LDA VSIZE1     ;CHECK ARRAY SIZES
22D0  8D 75 1E         STA N1
22D3  CD D2 1E         CMP VSIZE2
22D6  D0 22            BNE ADBADS
22D8  AD D1 1E         LDA VSIZE1+1
22DB  8D 77 1E         STA N2
22DE  CD D3 1E         CMP VSIZE2+1
22E1  D0 17            BNE ADBADS
22E3  AD C2 1E         LDA VTYPE2     ;V2 CONSTANT?
22E6  C9 01            CMP #1
22E8  F0 15            BEQ ABSC
22EA  AD D2 1E         LDA VSIZE2     ;V3 IS ARRAY
22ED  CD D4 1E         CMP VSIZE3
22F0  D0 08            BNE ADBADS
22F2  AD D3 1E         LDA VSIZE2+1
22F5  CD D5 1E         CMP VSIZE3+1
22F8  F0 05            BEQ ABSC
22FA  A2 12      ADBADS LDX #$12       ;?BAD SUBSCRIPT
22FC  4C 37 A4         JMP $A437
22FF  20 75 23   ABSC  JSR TRPT3      ;COPY POINTER TO Z PAGE
2302  A9 00            LDA #0         ;CALC NO. ELEMENTS
2304  8D 76 1E         STA N1+1
2307  8D 78 1E         STA N2+1
230A  20 7B 1E         JSR MMULT
230D  20 94 23   ABSLOP JSR V2TOT2     ;V2 TO (T2)
2310  20 DC 23         JSR V3TOF1     ;V2 TO FAC#1
2313  AD DF 1E         LDA TE2
2316  AC E0 1E         LDY TE2+1
2319  AE D6 1E         LDX OPTYPE
231C  E0 01            CPX #1
231E  D0 06            BNE DOSUB
2320  20 67 B8         JSR $B867      ; (A.Y) + FAC#1
2323  4C 33 23         JMP ABFA
2326  E0 02      DOSUB CPX #2
2328  D0 06            BNE DOMULT
232A  20 50 B8         JSR $B850      ;(A.Y)-FAC#1
232D  4C 33 23         JMP ABFA
2330  20 28 BA   DOMULT JSR $BA28      ;(A.Y) * FAC#1
```

```
LOC   CODE        LINE

2333  20 07 24    ABFA   JSR F1TOV1      ;FAC#1 TO V1
2336  AD 79 1E           LDA RESULT     ;CHECK ALL DONE
2339  D0 03              BNE ABNC
233B  CE 7A 1E           DEC RESULT+1
233E  CE 79 1E    ABNC   DEC RESULT
2341  AD 79 1E           LDA RESULT
2344  0D 7A 1E           ORA RESULT+1
2347  D0 C4              BNE ABSLOP
2349  60                 RTS
234A                     ;
234A  AD C5 1E    ORDER  LDA VTYPE3      ;V2 CONST
234D  C9 01              CMP #1
234F  D0 23              BNE ADV2NC
2351  AD C2 1E           LDA VTYPE2     ;SWOT V2 & V3
2354  8D C5 1E           STA VTYPE3
2357  AD D2 1E           LDA VSIZE2
235A  8D D4 1E           STA VSIZE3
235D  AD D3 1E           LDA VSIZE2+1
2360  8D D5 1E           STA VSIZE3+1
2363  AD D9 1E           LDA VSTT2
2366  8D DB 1E           STA VSTT3
2369  AD DA 1E           LDA VSTT2+1
236C  8D DC 1E           STA VSTT3+1
236F  A9 01              LDA #1
2371  8D C2 1E           STA VTYPE2
2374  60          ADV2NC RTS
2375                     ;
2375  AD DB 1E    TRPT3  LDA VSTT3       ; COPY POINTERS TO
2378  85 9E              STA VPTR3      ;ZERO PAGE
237A  AD DC 1E           LDA VSTT3+1
237D  85 9F              STA VPTR3+1
237F  AD D9 1E    TRPT2  LDA VSTT2
2382  85 FD              STA VPTR2
2384  AD DA 1E           LDA VSTT2+1
2387  85 FE              STA VPTR2+1
2389  AD D7 1E    TRPT1  LDA VSTT1
238C  85 FB              STA VPTR1
238E  AD D8 1E           LDA VSTT1+1
2391  85 FC              STA VPTR1+1
2393  60                 RTS
2394                     ;
2394  AD C2 1E    V2TOT2 LDA VTYPE2      ;V2 TO FAC#2
2397  F0 0D              BEQ V2RA
2399  30 23              BMI V2INT
239B  A9 C6              LDA #<FACM     ;FACM TO FAC#2
239D  A0 1E              LDY #>FACM
239F  8D DF 1E           STA TE2
23A2  8C E0 1E           STY TE2+1
23A5  60                 RTS
23A6  A5 FD       V2RA   LDA VPTR2       ;V2 TO FAC#2
23A8  A4 FE              LDY VPTR2+1
23AA  8D DF 1E           STA TE2
23AD  8C E0 1E           STY TE2+1
23B0  A9 05              LDA #5
23B2  18          V2BPT  CLC             ;BUMP VPTR2
23B3  65 FD              ADC VPTR2
23B5  85 FD              STA VPTR2
23B7  A5 FE              LDA VPTR2+1
23B9  69 00              ADC #0
23BB  85 FE              STA VPTR2+1
23BD  60                 RTS
23BE  A0 00       V2INT  LDY #0          ;FIXED TO FLOAT
23C0  B1 FD              LDA (VPTR2),Y  ;THEN FAC#1 TO FAC#2
23C2  AA                 TAX
23C3  C8                 INY
23C4  B1 FD              LDA (VPTR2),Y
```

```
LOC   CODE       LINE

23C6  A8                TAY
23C7  8A                TXA
23C8  20 91 B3          JSR $B391       ;FIXED TO FLOAT
23CB  A2 CB             LDX #<FACT      ;FAC#1 TO FACT
23CD  8E DF 1E          STX TE2
23D0  A0 1E             LDY #>FACT
23D2  8C E0 1E          STY TE2+1
23D5  20 D4 BB          JSR $BBD4
23D8  A9 02             LDA #2
23DA  D0 D6             BNE V2BPT       ;GO BUMP VPTR2
23DC  AD C5 1E   V3TOF1 LDA VTYPE3
23DF  D0 15             BNE V3INT
23E1  A5 9E             LDA VPTR3       ;V3 TO FAC#1
23E3  A4 9F             LDY VPTR3+1
23E5  20 A2 BB          JSR $BBA2
23E8  A9 05             LDA #5
23EA  18         V3BPT  CLC              ;BUMP VPTR3
23EB  65 9E             ADC VPTR3
23ED  85 9E             STA VPTR3
23EF  A5 9F             LDA VPTR3+1
23F1  69 00             ADC #0
23F3  85 9F             STA VPTR3+1
23F5  60                RTS
23F6  A0 00      V3INT  LDY #0           ;GET V3
23F8  B1 9E             LDA (VPTR3),Y
23FA  AA                TAX
23FB  C8                INY
23FC  B1 9E             LDA (VPTR3),Y
23FE  A8                TAY
23FF  8A                TXA
2400  20 91 B3          JSR $B391       ;FIXED TO FLOAT
2403  A9 02             LDA #2
2405  D0 E3             BNE V3BPT       ;GO BUMP VPTR3
2407  AD BF 1E   F1TOV1 LDA VTYPE1       ;FAC#1 TO V1
240A  D0 15             BNE V1INT
240C  A6 FB             LDX VPTR1
240E  A4 FC             LDY VPTR1+1
2410  20 D4 BB          JSR $BBD4
2413  A9 05             LDA #5
2415  18         V1BPT  CLC              ;BUMP VPTR1
2416  65 FB             ADC VPTR1
2418  85 FB             STA VPTR1
241A  A5 FC             LDA VPTR1+1
241C  69 00             ADC #0
241E  85 FC             STA VPTR1+1
2420  60                RTS
2421  20 BF B1   V1INT  JSR $B1BF        ;FLOAT TO INT
2424  A0 00             LDY #0
2426  A5 64             LDA $64
2428  91 FB             STA (VPTR1),Y
242A  A5 65             LDA $65
242C  C8                INY
242D  91 FB             STA (VPTR1),Y
242F  A9 02             LDA #2
2431  D0 E2             BNE V1BPT
2433             ;
2433  AD C2 1E   AMULT  LDA VTYPE2       ;CHECK FOR MULT.
2436  C9 01             CMP #1          ;ARRAY BY CONSTANT
2438  D0 03             BNE MERR
243A  4C CA 22   GADS   JMP ADDSUB
243D  AD C5 1E   MERR   LDA VTYPE3
2440  C9 01             CMP #1
2442  F0 F6             BEQ GADS
2444  AD D1 1E          LDA VSIZE1+1    ;CHECK ARRAY DIM.
2447  CD D3 1E          CMP VSIZE2+1
244A  D0 30             BNE AAERR
```

```
LOC   CODE        LINE

244C  AD D0 1E          LDA VSIZE1     ;CHECK NOT SAME ARRAYS
244F  CD D4 1E          CMP VSIZE3
2452  D0 28             BNE AAERR
2454  AD D2 1E          LDA VSIZE2
2457  CD D5 1E          CMP VSIZE3+1
245A  D0 20             BNE AAERR
245C  AD D7 1E          LDA VSTT1
245F  CD D9 1E          CMP VSTT2
2462  D0 08             BNE NSARR0
2464  AD D8 1E          LDA VSTT1+1
2467  CD DA 1E          CMP VSTT2+1
246A  F0 10             BEQ AAERR
246C  AD D7 1E   NSARR0 LDA VSTT1
246F  CD DB 1E          CMP VSTT3
2472  D0 0D             BNE AASOK
2474  AD D8 1E          LDA VSTT1+1
2477  CD DC 1E          CMP VSTT3+1
247A  D0 05             BNE AASOK
247C  A2 12      AAERR  LDX #$12       ;BAD SUBSCRIPT ERROR
247E  4C 37 A4          JMP $A437
2481  20 75 23   AASOK  JSR TRPT3      ;COPY POINTERS TO Z. P.
2484  A9 00             LDA #0
2486  8D 76 1E          STA N1+1
2489  8D 78 1E          STA N2+1
248C  A9 01             LDA #1
248E  8D 73 25          STA ROW
2491  8D 72 25          STA NROW
2494  8D 74 25          STA COLM
2497  A9 05             LDA #5         ;CALC LENGTH OF V2 ROW
2499  AE C2 1E          LDX VTYPE2     ; - 1 ELEMENT
249C  F0 02             BEQ AA2R
249E  A9 02             LDA #2
24A0  8D 75 1E   AA2R   STA N1
24A3  8D DD 1E          STA TE1
24A6  AE D3 1E          LDX VSIZE2+1
24A9  CA                DEX
24AA  8A                TXA
24AB  8D 77 1E          STA N2
24AE  20 7B 1E          JSR MMULT
24B1  AD 79 1E          LDA RESULT     ;STORE IT IN LLV2
24B4  8D 75 25          STA LLV2
24B7  AD 7A 1E          LDA RESULT+1
24BA  8D 76 25          STA LLV2+1
24BD  18         AALOOP CLC            ;MAIN LOOP
24BE  AD D9 1E          LDA VSTT2      ;SET V2 COL. PTR. TO NEXT
24C1  85 FD             STA VPTR2
24C3  6D DD 1E          ADC TE1        ;COL OF V2
24C6  8D 77 25          STA V2COLP
24C9  AD DA 1E          LDA VSTT2+1
24CC  85 FE             STA VPTR2+1
24CE  69 00             ADC #0
24D0  8D 78 25          STA V2COLP+1
24D3  A9 00      AALOP  LDA #0         ;ZERO ROW COL TOTAL
24D5  8D C6 1E          STA FACM
24D8  8D C7 1E          STA FACM+1
24DB  8D C8 1E          STA FACM+2
24DE  8D C9 1E          STA FACM+3
24E1  8D CA 1E          STA FACM+4
24E4  20 94 23   AAMRC  JSR V2TOT2     ;GET V2
24E7  20 DC 23          JSR V3TOF1     ;GET V1
24EA  AD DF 1E          LDA TE2
24ED  AC E0 1E          LDY TE2+1
24F0  20 28 BA          JSR $BA28      ;(A.Y) * FAC#1
24F3  A9 C6             LDA #<FACM
24F5  A0 1E             LDY #>FACM
24F7  20 67 B8          JSR $B867      ;(A.Y) + FAC#1
```

```
LOC   CODE         LINE

24FA  AD 73 25           LDA ROW
24FD  CD D2 1E           CMP VSIZE2
2500  F0 1C              BEQ ENDCOL
2502  EE 73 25           INC ROW
2505  A2 C6              LDX #<FACM
2507  A0 1E              LDY #>FACM
2509  20 D4 BB           JSR $BBD4         ;FAC#1 TO (X.Y)
250C  A5 FD              LDA VPTR2         ;V2 PTR DOWN 1 ROW
250E  18                 CLC
250F  6D 75 25           ADC LLV2
2512  85 FD              STA VPTR2
2514  A5 FE              LDA VPTR2+1
2516  6D 76 25           ADC LLV2+1
2519  85 FE              STA VPTR2+1
251B  4C E4 24           JMP AAMRC         ;GO GET NEXT 2 ELEMENTS
251E  20 07 24     ENDCOL JSR F1TOV1        ;FAC#1 (SUM) TO V1
2521  A9 01              LDA #1            ;FIRST ROW
2523  8D 73 25           STA ROW
2526  AD 74 25           LDA COLM
2529  CD D3 1E           CMP VSIZE2+1
252C  F0 26              BEQ ENDROW
252E  AD DB 1E           LDA VSTT3         ;SET V2 PTR. TO START CURRENT
2531  85 9E              STA VPTR3         ;ROW
2533  AD DC 1E           LDA VSTT3+1
2536  85 9F              STA VPTR3+1
2538  EE 74 25           INC COLM
253B  18                 CLC
253C  AD 77 25           LDA V2COLP
253F  85 FD              STA VPTR2
2541  6D DD 1E           ADC TE1
2544  8D 77 25           STA V2COLP
2547  AD 78 25           LDA V2COLP+1
254A  85 FE              STA VPTR2+1
254C  69 00              ADC #0
254E  8D 78 25           STA V2COLP+1
2551  4C D3 24           JMP AALOP
2554  AD 72 25     ENDROW LDA NROW           ;ALL ROWS DONE?
2557  CD D0 1E           CMP VSIZE1
255A  D0 01              BNE NEAA
255C  60                 RTS               ; ALL DONE
255D  A5 9E        NEAA   LDA VPTR3
255F  8D DB 1E           STA VSTT3
2562  A5 9F              LDA VPTR3+1
2564  8D DC 1E           STA VSTT3+1
2567  EE 72 25           INC NROW
256A  A9 01              LDA #1            ;FIRST COL.
256C  8D 74 25           STA COLM
256F  4C BD 24           JMP AALOOP        ;GO NEXT ROW FIRST COL.
2572  00           NROW   .BYT 0
2573  00           ROW    .BYT 0
2574  00           COLM   .BYT 0
2575  00 00        LLV2   .WOR 0
2577  00 00        V2COLP .WOR 0
2579                      .END
```

SHAPE

Abbreviated entry: S(shift)H

Affected Basic abbreviations: None

Token: Hex $DF Decimal 223

Purpose: To draw any 2 dimensional object on the screen at any scaling or angle.

Syntax: SHAPE <array>,SX,SY,TX,TY,a,col,br
<array> : integer array name including the '%'
SX,SY : X and Y scaling factors (real numbers)
TX,TY : plot position of object integer
a : angle to be plotted in degrees

Errors: Bad subscript – if array not found or error in first element of array (see below)
Illegal quantity – scaling or TX,TY values too large or error in data in array

Use: This routine will quickly draw (and fill if required) an object from a shape table stored in an array. The shape table takes the following simple format if stored in Array%().

Array%(Ø) = N :number of points (commands)
Array%(1) = command code
Array%(2) = X coordinate
Array%(3) = Y coordinate
.
.
.
Array%(N*3+1) = last command code
Array%(N*3+2) = X
Array%(N*3+3) = Y

N is checked for the array being large enough to hold that number of points. The command codes are:

Ø: Move to X,Y
1: Draw from last point to X,Y
2: Plot at X,Y
3: Fill at X,Y

Coordinates are relative to the SHAPE origin TX,TY.
SHAPE is illustrated in Program 4.

Routine entry point: $25A7

Routine operation: Before the command codes in the SHAPE table are used the coordinates are calculated as follows:

$$Xnew = SX * \cos(r) - SY * \sin(r) + tx$$
$$Ynew = SX * \sin(r) + SY * \cos(r) + ty$$

```
1 REM*************************
2 REM SHAPE DEMO
3 REM  -TANK-
4 REM*************************
5 :
10 HIRES 1,0
20 DIMTA%(100)
30 READ TA%(0)
40 FORI=1TO TA%(0)*3 STEP3
50 READTA%(I),TA%(I+1),TA%(I+2)
60 NEXT
70 SHAPETA%,5,10,10,10,0,8,1
80 SHAPETA%,-5 ,10,320,10,0,8,1
90 SHAPETA%,-5 ,-10,320,190,0,8,1
100 SHAPETA%,5 ,-10,10,190,0,8,1
990 GETA$:IFA$<>"←"THEN990
991 NORM:LIST
999 REM SHAPE TABLE FOR A TANK
1000 DATA 32
1010 DATA 0,0,3,   1,2,0,   1,20,0
1020 DATA 1,22,3,  1,0,3,   1,2,5
1030 DATA 1,20,5,  1,22,3,  0,5,5
1040 DATA 1,7,8,   1,14,8,  1,17,5
1050 DATA 0,16,6,  1,24,6,  1,24,7
1060 DATA 1,15,7
1070 DATA 0,2,3,   1,1,2,   1,1,1,1,2,0
1080 DATA 1,3,0,   1,4,1,   1,4,2
1090 DATA 1,3,3
1100 DATA 0,19,3,  1,18,2,  1,18,1
1110 DATA 1,19,0,  1,20,0,  1,21,1
1120 DATA 1,21,2,  1,20,3
```

```
1 REM*********************
2 REM SHAPE DEMO
3 REM -TEXT-
4 REM*********************
5 HIRES 1,0
10 DIM A%(1000)
20 READ N
30 A%(0)=N
40 FORI=1TO N*3 STEP3
50 READA%(I),A%(I+1),A%(I+2)
60 NEXT
61 SHAPEA%,9,20,0,0,0,10,3
65 FORR=0TO 2
70 SHAPEA%,4+R,5,R*50,R*50,R*7,14,2
80 NEXT
90 SHAPEA%,2,7,10,190,-90,1,1
91 SHAPEA%,1.05,1.5,302,81,90,1,1
100 GETA$:IFA$<>"←"THEN100
110 NORM
120 END
999 REM SHAPE TABLE
1000 DATA 63
1010 DATA 1,8,0
1020 DATA 1,8,1
1030 DATA 1,2,1
1040 DATA 1,8,9
1045 DATA 1,8,10
1050 DATA 1,0,10
1060 DATA 1,0,9
1070 DATA 1,6,9
1080 DATA 1,0,1
1090 DATA 1,0,0
1100 DATA 3,4,5
1110 DATA 0,12,0 ,1,13,0 ,1,13,6
1120 DATA 1,12,6 ,1,12,0 ,0,12,7
1130 DATA 1,13,7 ,1,13,8 ,1,12,8
```

```
1140 DATA 1,12,7
1150 DATA 0,16,0 ,1,17,0 ,1,17,4
1160 DATA 1,20,4 ,1,20,5 ,1,17,5
1170 DATA 1,17,7 ,1,21,7 ,1,21,8
1180 DATA 1,16,8 ,1,16,0
1190 DATA 0,24,0 ,1,25,0 ,1,25,4
1200 DATA 1,27,0 ,1,28,0 ,1,26,4
1210 DATA 1,28,4 ,1,28,8 ,1,24,8
1220 DATA 1,24,0 ,0,25,5 ,1,27,5
1230 DATA 1,27,7 ,1,25,7 ,1,25,5
1240 DATA 0,31,0 ,1,32,0 ,1,32,3
1250 DATA 1,34,3 ,1,34;0 ,1,35,0
1260 DATA 1,35,6 ,1,34,8 ,1,32,8
1270 DATA 1,31,6 ,1,31,0 ,0,32,4
1280 DATA 1,34,4 ,1,34,6 ,1,32,6
1290 DATA 1,32,4
```

```
1 REM************************
2 REM SHAPE DEMO
3 REM CALCULATED SHAPE TABLE
4 REM************************
10 DIMA%(153)
20 A%(0)=51
30 FORI=1TO153STEP 3
40 A%(I)=1
50 A%(I+1)=SIN(I/75*π)*100
60 A%(I+2)=COS(I/75*π)*100
70 NEXT
71 A%(1)=0
72 HIRES0,0
75 FORR=0TO175STEP5
90 SHAPEA%,.3,1,160,100,R,1,1
110 NEXT
120 GETA$:IFA$<>"←"THEN120
130 NORM :LIST
```

Program 4.

```
LOC   CODE       LINE

2579                     .LIB SHAPE
2579             ;****************************
2579             ;SHAPE <ARRAY>,SX,SY,TX,TY,R
2579             ; ARRAY   MUST BE INTEGER
2579             ; SX,SY   SCALING FACTORS
2579             ; TX,TY   TRANSLATION VECTOR
2579             ; R       ROTATION ANGLE (DEG)
2579             ;****************************
2579             ;
2579             ;
2579             EVLEXP = $AD8A          ;EVALUATE EXPRESSION
2579             ;
2579  00 00      VANAME .WOR 0           ;VARIABLE NAME
257B  00 00      VSIZE  .WOR 0           ;ARRAY SIZE
257D             MA     * = *+5          ;2 X 2 TRANSFORMATION
2582             MB     * = *+5          ;MATRIX
2587             MC     * = *+5
258C             MD     * = *+5
2591  00 00      NELEM  .WOR 0           ;NUMBER ELEMENTS
2593             SX     * = *+5          ;START COORDINATES
2598             SY     * = *+5
259D             SINR   * = *+5          ;SIN & COS OF R
25A2             COSR   * = *+5
25A7             ;
25A7             ;
25A7  8D 79 25   SHAPE  STA VANAME       ;GET ARRAY
25AA  20 13 B1          JSR ISNALF       ;NAME AND CHECK
25AD  B0 03             BCS SHCHK        ; LEGAL
```

```
LOC   CODE         LINE

25AF  4C 08 AF            JMP $AF08         ; SYNTAX
25B2  A9 00        SHCHK  LDA #0
25B4  8D 7A 25            STA VANAME+1
25B7  20 73 00            JSR CHRGET
25BA  90 05               BCC SHCHK1
25BC  20 13 B1            JSR ISNALF
25BF  90 0D               BCC SHEVNA        ;GO CHECK FOR % $ =
25C1  8D 7A 25     SHCHK1 STA VANAME+1
25C4  20 73 00     SHLNE  JSR CHRGET         :SCAN PAST REST
25C7  90 FB               BCC SHLNE         ;OF VAR NAME
25C9  20 13 B1            JSR ISNALF
25CC  B0 F6               BCS SHLNE
25CE  C9 24        SHEVNA CMP #'$            ;CHECK FOR STRING
25D0  D0 05               BNE NSTR
25D2  A2 16               LDX #22
25D4  4C 37 A4            JMP $A437         ;TYPE MISMATCH
25D7  C9 25        NSTR   CMP #'%
25D9  F0 03               BEQ ATINT1        ;INTEGER ARRAY
25DB  4C 08 AF            JMP $AF08         ;SYNTAX NOT INTEGER
25DE  20 73 00     ATINT1 JSR CHRGET
25E1  A9 80               LDA #128          ;SET HI BITS ARRAY NAME
25E3  0D 79 25            ORA VANAME
25E6  8D 79 25            STA VANAME
25E9  A9 80               LDA #128
25EB  0D 7A 25            ORA VANAME+1
25EE  8D 7A 25            STA VANAME+1
25F1  20 85 28            JSR FANDAR        ;FIND ARRAY ADDR
25F4  A0 00               LDY #0
25F6  B1 9E               LDA (VPTR3),Y
25F8  8D 01 29            STA IAC1+1
25FB  8D 92 25            STA NELEM+1
25FE  10 05               BPL NELEP         ;NO. ELEMENTS POS
2600  A2 0E               LDX #$0E          ;ILLEGAL QUANTITY
2602  4C 37 A4            JMP $A437
2605  C8           NELEP  INY
2606  B1 9E               LDA (VPTR3),Y
2608  8D 91 25            STA NELEM
260B  8D 00 29            STA IAC1
260E  0E 00 29            ASL IAC1          ;(( *3)+1)CHECK NUMBER OF
2611  2E 01 29            ROL IAC1+1        ;POINTS WILL FIT IN
2614  38                  SEC               ;ARRAY
2615  AD 91 25            LDA NELEM
2618  6D 00 29            ADC IAC1
261B  8D 00 29            STA IAC1
261E  AD 92 25            LDA NELEM+1
2621  6D 01 29            ADC IAC1+1
2624  8D 01 29            STA IAC1+1
2627  38                  SEC
2628  AD 7B 25            LDA VSIZE
262B  ED 00 29            SBC IAC1
262E  AD 7C 25            LDA VSIZE+1
2631  ED 01 29            SBC IAC1+1
2634  B0 05               BCS SIZEOK
2636  A2 12               LDX #$12          ;BAD SUBSCRIPT ERROR
2638  4C 37 A4            JMP $A437
263B  20 FD AE     SIZEOK JSR $AEFD          ;CHECK ','
263E  20 8A AD            JSR EVLEXP        ;GET SX
2641  A2 93               LDX #<SX          ;FAC#1 TO SX
2643  A0 25               LDY #>SX
2645  20 D4 BB            JSR $BBD4
2648  20 FD AE            JSR $AEFD         ;CHECK ','
264B  20 8A AD            JSR EVLEXP        ;GET SY
264E  A2 98               LDX #<SY
2650  A0 25               LDY #>SY
2652  20 D4 BB            JSR $BBD4
2655  20 FD AE            JSR $AEFD         ;CHECK ','
```

```
LOC   CODE          LINE

2658  20 E3 10            JSR GXY         ;GET TX AND TY
265B  20 FD AE            JSR $AEFD
265E  20 8A AD            JSR EVLEXP      ;GET K
2661  A9 C0               LDA #<PID180    ;CONVERT TO RAD.
2663  A0 1B               LDY #>PID180
2665  20 28 BA            JSR $BA28       ;MULTIPLY
2668  A2 CB               LDX #<FACT      ;FAC#1 TO FACT
266A  A0 1E               LDY #>FACT
266C  20 D4 BB            JSR $BBD4
266F  20 42 2B            JSR GCB         ;GET COLOUR & BRUSH
2672  A9 CB               LDA #<FACT      ;FACT TO FAC#1
2674  A0 1E               LDY #>FACT
2676  20 A2 BB            JSR $BBA2
2679  20 6B E2            JSR $E26B       ;SIN(FAC#1)
267C  A2 9D               LDX #<SINR      ;FAC#1 TO SINR
267E  A0 25               LDY #>SINR
2680  20 D4 BB            JSR $BBD4
2683  A9 CB               LDA #<FACT      ;FACT TO FAC#1
2685  A0 1E               LDY #>FACT
2687  20 A2 BB            JSR $BBA2
268A  20 64 E2            JSR $E264       ;COS(FAC#1)
268D  A2 A2               LDX #<COSR      ;FAC#1 TO COSR
268F  A0 25               LDY #>COSR
2691  20 D4 BB            JSR $BBD4
2694  A9 A2               LDA #<COSR      ;RELOAD
2696  A0 25               LDY #>COSR
2698  20 A2 BB            JSR $BBA2
269B  A9 93               LDA #<SX        ;MULT BY SX
269D  A0 25               LDY #>SX
269F  20 28 BA            JSR $BA28
26A2  A2 7D               LDX #<MA        ;STORE IN MA
26A4  A0 25               LDY #>MA
26A6  20 D4 BB            JSR $BBD4
26A9  A9 9D               LDA #<SINR      ;SINR TO FAC#1
26AB  A0 25               LDY #>SINR
26AD  20 A2 BB            JSR $BBA2
26B0  A9 93               LDA #<SX        ;MULT BY SX
26B2  A0 25               LDY #>SX
26B4  20 28 BA            JSR $BA28
26B7  A2 82               LDX #<MB        ;STORE IN MB
26B9  A0 25               LDY #>MB
26BB  20 D4 BB            JSR $BBD4
26BE  A9 9D               LDA.#<SINR      ;SINR TO FAC#1
26C0  A0 25               LDY #>SINR
26C2  20 A2 BB            JSR $BBA2
26C5  A5 66               LDA $66         ;FAC#1 = -FAC#1
26C7  49 80               EOR #$80
26C9  85 66               STA $66
26CB  A9 98               LDA #<SY        ;MULTIPLY BY SY
26CD  A0 25               LDY #>SY
26CF  20 28 BA            JSR $BA28
26D2  A2 87               LDX #<MC        ;STORE IN MC
26D4  A0 25               LDY #>MC
26D6  20 D4 BB            JSR $BBD4
26D9  A9 A2               LDA #<COSR      ;COSR TO FAC#1
26DB  A0 25               LDY #>COSR
26DD  20 A2 BB            JSR $BBA2
26E0  A9 98               LDA #<SY        ;MULT BY SY
26E2  A0 25               LDY #>SY
26E4  20 28 BA            JSR $BA28
26E7  A2 8C               LDX #<MD        ;STORE IN MD
26E9  A0 25               LDY #>MD
26EB  20 D4 BB            JSR $BBD4
26EE  AD 09 29            LDA TX          ;SET START POINT FOR SHAPE
26F1  8D 40 03            STA X2
26F4  AD 0A 29            LDA TX+1
```

```
LOC   CODE          LINE

26F7  8D 41 03             STA X2+1
26FA  AD 0C 29             LDA TY
26FD  8D 42 03             STA Y2
2700  AD 0D 29             LDA TY+1
2703  8D 43 03             STA Y2+1
2706  18                   CLC               ;SET PTR TO START OF SHAPE DATA
2707  A9 02                LDA #2
2709  65 9E                ADC VPTR3
270B  85 9E                STA VPTR3
270D  A9 00                LDA #0
270F  65 9F                ADC VPTR3+1
2711  85 9F                STA VPTR3+1
2713  AD 91 25      SHLOOP LDA NELEM          ;CHECK FOR ALL DONE
2716  0D 91 25             ORA NELEM +1
2719  D0 01                BNE CONTSH
271B  60                   RTS               ;ALL DONE
271C  AD 40 03      CONTSH LDA X2             ;MOVE END POS TO START POS
271F  8D 3C 03             STA X1
2722  AD 41 03             LDA X2+1
2725  8D 3D 03             STA X1+1
2728  AD 42 03             LDA Y2
272B  8D 3E 03             STA Y1
272E  AD 43 03             LDA Y2+1
2731  8D 3F 03             STA Y1+1
2734  20 6D 28             JSR GETEL         ;GET ELEMENT
2737  F0 05                BEQ SHCMOK        ;CHECK BETWEEN (0 - 255)
2739  A2 0E                LDX #$0E          ;ILLEGAL QUANTITY
273B  4C 37 A4             JMP $A437
273E  98            SHCMOK TYA                ;LOW BYTE
273F  29 03                AND #3
2741  0A                   ASL A
2742  AA                   TAX
2743  BD 19 28             LDA SHVT,X
2746  8D 17 28             STA SHV
2749  BD 1A 28             LDA SHVT+1,X
274C  8D 18 28             STA SHV+1
274F  20 6D 28             JSR GETEL         ;GET X VALUE FROM ARRAY
2752  20 91 B3             JSR $B391         ;FLOAT IT
2755  A2 CB                LDX #<FACT        ;STORE IN FACT
2757  A0 1E                LDY #>FACT
2759  20 D4 BB             JSR $BBD4
275C  20 6D 28             JSR GETEL         ;GET Y VALUE
275F  20 91 B3             JSR $B391         ;FLOAT IT
2762  A2 C6                LDX #<FACM        ;STORE IN FACM
2764  A0 1E                LDY #>FACM
2766  20 D4 BB             JSR $BBD4
2769  A9 CB                LDA #<FACT        ;LOAD FAC#1 WITH X
276B  A0 1E                LDY #>FACT
276D  20 A2 BB             JSR $BBA2
2770  A9 7D                LDA #<MA          ;MULT BY MA
2772  A0 25                LDY #>MA
2774  20 28 BA             JSR $BA28
2777  20 BF B1             JSR $B1BF         ;FIX IT
277A  A6 65                LDX $65
277C  A4 64                LDY $64
277E  8E 40 03             STX X2            ;PUT IN X2
2781  8C 41 03             STY X2+1
2784  A9 C6                LDA #<FACM        ;GET Y FROM FACM
2786  A0 1E                LDY #>FACM
2788  20 A2 BB             JSR $BBA2
278B  A9 87                LDA #<MC          ;MULTIPLY BY MC
278D  A0 25                LDY #>MC
278F  20 28 BA             JSR $BA28
2792  20 BF B1             JSR $B1BF         ;FIX IT
2795  A5 65                LDA $65
2797  18                   CLC               ;ADD TO X2
```

```
LOC   CODE         LINE

2798  6D 40 03           ADC X2
279B  8D 40 03           STA X2
279E  A5 64              LDA $64
27A0  6D 41 03           ADC X2+1
27A3  8D 41 03           STA X2+1
27A6  18                 CLC              ; AND ADD TX
27A7  AD 09 29           LDA TX
27AA  6D 40 03           ADC X2
27AD  8D 40 03           STA X2
27B0  AD 0A 29           LDA TX+1
27B3  6D 41 03           ADC X2+1
27B6  8D 41 03           STA X2+1
27B9  A9 CB              LDA #<FACT       ;GET X FROM FACT
27BB  A0 1E              LDY #>FACT
27BD  20 A2 BB           JSR $BBA2
27C0  A9 82              LDA #<MB         ;MULT BY MB
27C2  A0 25              LDY #>MB
27C4  20 28 BA           JSR $BA28
27C7  20 BF B1           JSR $B1BF
27CA  A6 65              LDX $65
27CC  A4 64              LDY $64
27CE  8E 42 03           STX Y2           ;PUT IN Y2
27D1  8C 43 03           STY Y2+1
27D4  A9 C6              LDA #<FACM       ;GET Y FROM FACM
27D6  A0 1E              LDY #>FACM
27D8  20 A2 BB           JSR $BBA2
27DB  A9 8C              LDA #<MD         ;MULT BY MD
27DD  A0 25              LDY #>MD
27DF  20 28 BA           JSR $BA28
27E2  20 BF B1           JSR $B1BF        ;FIX IT
27E5  A5 65              LDA $65
27E7  18                 CLC ADD TO Y2
27E8  6D 42 03           ADC Y2
27EB  8D 42 03           STA Y2
27EE  A5 64              LDA $64
27F0  6D 43 03           ADC Y2+1
27F3  8D 43 03           STA Y2+1
27F6  18                 CLC              ;ADD TY
27F7  AD 0C 29           LDA TY
27FA  6D 42 03           ADC Y2
27FD  8D 42 03           STA Y2
2800  AD 0D 29           LDA TY+1
2803  6D 43 03           ADC Y2+1
2806  8D 43 03           STA Y2+1
2809  AD 91 25           LDA NELEM        ;DECREASE LOOP COUNTER
280C  D0 03              BNE NELNC
280E  CE 92 25           DEC NELEM+1
2811  CE 91 25     NELNC DEC NELEM
2814  6C 17 28           JMP (SHV)        ;DO PLOT OPTION
2817               ;
2817  00 00        SHV   .WOR 0
2819  13 27        SHVT  .WOR SHLOOP       ; 0 - MOVE
281B  21 28              .WOR SHDRAW      ; 1 - DRAW
281D  27 28              .WOR SHPLOT      ; 2 - PLOT
281F  45 28              .WOR SHFILL      ; 3 - FILL
2821               ;
2821               ;
2821  20 40 11     SHDRAW JSR BOX          ;DRAW LINE
2824  4C 13 27           JMP SHLOOP
2827               ;
2827  AD 40 03     SHPLOT LDA X2
282A  85 59              STA T2
282C  AD 41 03           LDA X2+1
282F  85 5A              STA T2+1
2831  AD 42 03           LDA Y2
2834  85 5B              STA T3
```

```
LOC   CODE        LINE

2836  AD 43 03           LDA Y2+1
2839  85 5C              STA T3+1
283B  A5 FE              LDA COL+1
283D  85 FD              STA COL
283F  20 02 10           JSR PLOT        ;PLOT POINT
2842  4C 13 27           JMP SHLOOP
2845              ;
2845  AD 40 03    SHFILL LDA X2
2848  8D 3C 03           STA X1
284B  AD 41 03           LDA X2+1
284E  8D 3D 03           STA X1+1
2851  AD 42 03           LDA Y2
2854  8D 3E 03           STA Y1
2857  AD 43 03           LDA Y2+1
285A  8D 3F 03           STA Y1+1
285D  A5 FC              LDA PBR
285F  29 03              AND #3
2861  8D 73 03           STA BRCOL
2864  8D 74 03           STA BRCOL+1
2867  20 18 18           JSR BROK        ;FILL
286A  4C 13 27           JMP SHLOOP
286D              ;
286D  A0 00       GETEL  LDY #0           ;GET ELEMENT FROM ARRAY
286F  B1 9E              LDA (VPTR3),Y   ;HIGH BYTE
2871  48                 PHA
2872  C8                 INY
2873  B1 9E              LDA (VPTR3),Y   ;LOW BYTE
2875  A8                 TAY             ;LOW IN Y
2876  18                 CLC             ;BUMP PTR BY 2
2877  A9 02              LDA #2
2879  65 9E              ADC VPTR3
287B  85 9E              STA VPTR3
287D  A9 00              LDA #0
287F  65 9F              ADC VPTR3+1
2881  85 9F              STA VPTR3+1
2883  68                 PLA
2884  60                 RTS
2885              ;
2885              ;
2885              ;
2885              ;
2885              ; FIND AND CHECK ARRAY
2885              ;
2885  A5 2F       FANDAR LDA $2F          ;START OF ARRAYS
2887  85 9E              STA VPTR3
2889  A5 30              LDA $30
288B  85 9F              STA VPTR3+1
288D  A5 9E       FILOOP LDA VPTR3        ;CMP. END OF ARRAYS
288F  C5 31              CMP $31
2891  D0 0B              BNE FICONT
2893  A5 9F              LDA VPTR3+1
2895  C5 32              CMP $32
2897  D0 05              BNE FICONT
2899  A2 12              LDX #$12        ;? BAD SUBSCRIPT ERROR
289B  4C 37 A4           JMP $A437
289E  A0 00       FICONT LDY #0
28A0  B1 9E              LDA (VPTR3),Y   ;FIRST CHAR OF NAME
28A2  C8                 INY
28A3  CD 79 25           CMP VANAME
28A6  D0 07              BNE FINAR       ;TRY NEXT ARRAY
28A8  B1 9E              LDA (VPTR3),Y
28AA  CD 7A 25           CMP VANAME+1
28AD  F0 1D              BEQ FIGETS      ; GET ARRAY DATA
28AF  C8          FINAR  INY              ;FIND NEXT ARRAY
28B0  B1 9E              LDA (VPTR3),Y
28B2  8D DD 1E           STA TE1
```

```
LOC    CODE        LINE

28B5   C8                   INY
28B6   B1 9E                LDA (VPTR3),Y
28B8   18                   CLC
28B9   65 9F                ADC VPTR3+1
28BB   85 9F                STA VPTR3+1
28BD   AD DD 1E             LDA TE1
28C0   18                   CLC
28C1   65 9E                ADC VPTR3
28C3   85 9E                STA VPTR3
28C5   90 02                BCC FINC
28C7   E6 9F                INC VPTR3+1
28C9   4C 8D 28    FINC     JMP FILOOP
28CC   A9 01       FIGETS   LDA #1           ;GET ARRAY DATA
28CE   8D DE 1E             STA TE1+1
28D1   C8                   INY
28D2   C8                   INY
28D3   C8                   INY
28D4   B1 9E                LDA (VPTR3),Y
28D6   C9 01                CMP #1
28D8   F0 05                BEQ FINDOK
28DA   A2 12       FIE1     LDX #$12         ;ERROR MORE THAN 1 DIM
28DC   4C 37 A4             JMP $A437
28DF   C8          FINDOK   INY
28E0   B1 9E                LDA (VPTR3),Y
28E2   8D 7C 25             STA VSIZE+1
28E5   C8                   INY
28E6   B1 9E                LDA (VPTR3),Y
28E8   8D 7B 25             STA VSIZE
28EB   C8                   INY
28EC   98                   TYA
28ED   18                   CLC
28EE   65 9E                ADC VPTR3
28F0   85 9E                STA VPTR3
28F2   A5 9F                LDA VPTR3+1
28F4   69 00                ADC #0
28F6   85 9F                STA VPTR3+1
28F8   60                   RTS
28F9                        .END
```

4 THREE DIMENSIONAL PLOTTING ROUTINES

PORIGIN

Abbreviated entry: PO(shift)R

Affected Basic abbreviations: None

Token: Hex $DE Decimal 222

Purpose: Set three dimensional plot origin and distance of perspective view point in front of origin.

Syntax: PORIGIN X,Y,D

Errors: Illegal quantity – if X,Y or D cannot be expressed as integer

Use: Sets X and Y coordinates of three dimensional origin on the screen. The last parameter is a constant used in calculating the perspective projection of a point onto the screen plane. This constant is the distance of the view point in front of the screen plane. The Z coordinate of the three dimensional origin is always zero. Positive values of Z in plot commands are taken as the distance of a point behind the screen plane. Default values are set by the HIRES command (160,100,200).

Routine entry point: $290F

Routine operation: Just sets constants.

PPLOT

Abbreviated entry: P(shift)P

Affected Basic abbreviations: None

Token: Hex $DC   Decimal 220

Purpose: To plot the perspective projection of a three dimensional point on the screen plane.

Syntax: PPLOT X,Y,Z,col,br

Use: Allows plotting of three dimensional objects with perspective (distant objects appear smaller).

Routine entry: $2B1A

Routine operation: This is best shown by the equivalent PLOT commands:

```
PORIGIN A,B,D
PPLOT X,Y,Z,COL,BR
```

These will give the same result as:

```
PLOT X*D/(Z+D)+A,Y*D/(Z+D)+B,COL,BR
```

but PPLOT is much faster as it uses its own integer arithmetic routines.

PDRAW

Abbreviated entry: P(shift)D

Affected Basic abbreviations: None

Token: Hex $DD   Decimal 221

Purpose: To draw the perspective projection of a three dimensional line on the screen plane.

Syntax: PDRAW X1,Y1,Z1,X2,Y2,Z2,col,br

Errors: As DRAW

Use: Allows fast drawing of three dimensional perspective objects.

Routine entry point: $2B5F

Routine operation: It uses the routines in PPLOT to calculate the projections on the screen plane of the two ends of the line and then draw a line to link them.

```
LOC   CODE        LINE


28F9                     .LIB PPLOT
28F9              ;****************************
28F9              ;THREE DIMENSIONAL PLOT
28F9              ;      ROUTINES
28F9              ;****************************
28F9              ;
28F9              ;
28F9  00          EFLAG  .BYT 0          ;ERROR FLAG
28FA              ;
28FA  A0 00       VX     .WOR 160        ;VIEW ORIGIN X
28FC  64 00       VY     .WOR 100        ; ... Y
28FE  C8 00       VZ     .WOR 200        ; ... Z
2900              ;
2900  00 00       IAC1   .WOR 0          ;INTEGER ACCUM.
2902  00 00       IAC2   .WOR 0
2904  00          SIAC   .BYT 0          ;SGN OF RESULT
2905  00 00       IACR   .WOR 0          ;RESULT ACCUM.
2907  00 00       ZPVZ   .WOR 0          ;Z + VZ
2909              ;
2909  00 00       TX     .WOR 0          ;TEMP X STORE
290B  00          SGNTX  .BYT 0          ;SGN OF TX
290C  00 00       TY     .WOR 0
290E  00          SGNTY  .BYT 0
290F              ;
290F              ;**********************
290F              ;PORIGIN X,Y,Z
290F              ; SET VIEW POSITION
290F              ;**********************
290F              ;
290F  20 8A AD    PVIEW  JSR $AD8A       ;GET VX
2912  20 BF B1           JSR $B1BF       ;FLOAT TO FIX
2915  A5 65              LDA $65         ;STORE IN VX (LOW HIGH)
2917  8D FA 28           STA VX
291A  A5 64              LDA $64
291C  8D FB 28           STA VX+1
291F  20 FD AE           JSR $AEFD       ;CHECK ','
2922  20 8A AD           JSR $AD8A       ;GET VY
2925  20 BF B1           JSR $B1BF       ;FLOAT TO FIX
2928  A5 65              LDA $65
292A  8D FC 28           STA VY
292D  A5 64              LDA $64
292F  8D FD 28           STA VY+1
2932  20 FD AE           JSR $AEFD       ;CHECK ','
2935  20 8A AD           JSR $AD8A       ;GET VZ
2938  20 BF B1           JSR $B1BF
293B  A5 65              LDA $65
293D  8D FE 28           STA VZ
2940  A5 64              LDA $64
2942  8D FF 28           STA VZ+1
2945  60                 RTS
2946              ;
2946              ;
2946              ;SEPARATE SIGN
```

```
LOC   CODE       LINE

2946             ;IN     X,Y LOW HIGH (16 BIT SIGNED)
2946             ;EXIT   X,Y LOW HIGH A SGN (0) POS
2946             ;
2946  98         SEPSGN TYA
2947  30 03             BMI SEPN
2949  A9 00             LDA #0
294B  60                RTS
294C  8A         SEPN   TXA              ;X.Y =-X.Y
294D  49 FF             EOR #$FF
294F  AA                TAX
2950  98                TYA
2951  49 FF             EOR #$FF
2953  A8                TAY
2954  E8                INX
2955  D0 01             BNE SEPEXN
2957  C8                INY
2958  A9 FF      SEPEXN LDA #$FF
295A  60                RTS
295B             ;
295B             ;
295B             ;MERGE SIGN
295B             ;
295B             ;IN     X,Y LOW HIGH A SIGN (0) POS
295B             ;EXIT X,Y LOW HIGH (16 BIT SIGNED)
295B             ;
295B  09 00      MERSGN ORA #0
295D  D0 ED             BNE SEPN
295F  60                RTS
2960             ;
2960             ;
2960             ;
2960             ; GET  X,Y,Z
2960             ; AND CALCULATE
2960             ; THREE DIMENSIONAL
2960             ; AND CALCULATE
2960             ; COORDINATES
2960             ;
2960  20 8A AD   GXYZ   JSR $AD8A        ;GET X
2963  20 BF B1          JSR $B1BF       ;FLOAT TO FIX
2966  A6 65             LDX $65
2968  A4 64             LDY $64
296A  20 46 29          JSR SEPSGN      ;SEPARATE SIGN
296D  8D 0B 29          STA SGNTX
2970  8E 09 29          STX TX
2973  8C 0A 29          STY TX+1
2976  20 FD AE          JSR $AEFD       ;CHECK ','
2979  20 8A AD          JSR $AD8A       ;GET Y
297C  20 BF B1          JSR $B1BF
297F  A6 65             LDX $65
2981  A4 64             LDY $64
2983  20 46 29          JSR SEPSGN      ;SEPARATE SIGN
2986  8D 0E 29          STA SGNTY
2989  8E 0C 29          STX TY
298C  8C 0D 29          STY TY+1
298F  20 FD AE          JSR $AEFD       ;CHECK ','
2992  20 8A AD          JSR $AD8A       ;GET Z
2995  20 BF B1          JSR $B1BF
2998  A5 65             LDA $65         ;ADD VZ
299A  18                CLC
299B  6D FE 28          ADC VZ
299E  8D 07 29          STA ZPVZ
29A1  A5 64             LDA $64
29A3  6D FF 28          ADC VZ+1
29A6  8D 08 29          STA ZPVZ+1
29A9  10 05             BPL ZPVZNN      ;CHECK >0
29AB  A2 0E      ILLQ   LDX #$0E         ;ILLEGAL QUANTITY ERROR
```

```
LOC    CODE          LINE

29AD   4C 37 A4               JMP $A437
29B0   0D 07 29      ZPVZNN ORA ZPVZ          ;CHECK <>0
29B3   F0 F6                  BEQ ILLQ
29B5   A9 00                  LDA #0          ;ZERO ACCUM. SGN
29B7   8D 04 29               STA SIAC
29BA   AD FE 28               LDA VZ          ; VZ TO ACCUM.
29BD   8D 00 29               STA IAC1
29C0   AD FF 28               LDA VZ+1
29C3   8D 01 29               STA IAC1+1
29C6   AD 0B 29               LDA SGNTX       ;MULTIPLY BY TX
29C9   AE 09 29               LDX TX
29CC   AC 0A 29               LDY TX+1
29CF   20 68 2A               JSR MULT
29D2   AD 05 29               LDA IACR
29D5   8D 00 29               STA IAC1
29D8   AD 06 29               LDA IACR+1
29DB   8D 01 29               STA IAC1+1
29DE   A9 00                  LDA #0          ;DIVIDE BY Z + VZ
29E0   AE 07 29               LDX ZPVZ
29E3   AC 08 29               LDY ZPVZ+1
29E6   20 BE 2A               JSR DIVIDE
29E9   AD 04 29               LDA SIAC
29EC   AE 05 29               LDX IACR
29EF   AC 06 29               LDY IACR+1
29F2   20 5B 29               JSR MERSGN
29F5   8E 09 29               STX TX
29F8   8C 0A 29               STY TX+1
29FB   18                     CLC             ;ADD VIEW OFF SET
29FC   AD FA 28               LDA VX
29FF   6D 09 29               ADC TX
2A02   8D 09 29               STA TX
2A05   AD FB 28               LDA VX+1
2A08   6D 0A 29               ADC TX+1
2A0B   8D 0A 29               STA TX+1
2A0E   A9 00                  LDA #0          ;ZERO ACCUM. SGN
2A10   8D 04 29               STA SIAC
2A13   AD FE 28               LDA VZ          ; VZ TO ACCUM.
2A16   8D 00 29               STA IAC1
2A19   AD FF 28               LDA VZ+1
2A1C   8D 01 29               STA IAC1+1
2A1F   AD 0E 29               LDA SGNTY       ;MULTIPLY BY TY
2A22   AE 0C 29               LDX TY
2A25   AC 0D 29               LDY TY+1
2A28   20 68 2A               JSR MULT
2A2B   AD 05 29               LDA IACR
2A2E   8D 00 29               STA IAC1
2A31   AD 06 29               LDA IACR+1
2A34   8D 01 29               STA IAC1+1
2A37   A9 00                  LDA #0          ;DIVIDE BY Z + VZ
2A39   AE 07 29               LDX ZPVZ
2A3C   AC 08 29               LDY ZPVZ+1
2A3F   20 BE 2A               JSR DIVIDE
2A42   AD 04 29               LDA SIAC
2A45   AE 05 29               LDX IACR
2A48   AC 06 29               LDY IACR+1
2A4B   20 5B 29               JSR MERSGN      ;CONVERT TO 16 BIT SIGNED
2A4E   8E 0C 29               STX TY
2A51   8C 0D 29               STY TY+1
2A54   18                     CLC             ;ADD VIEW OFF SET
2A55   AD FC 28               LDA VY
2A58   6D 0C 29               ADC TY
2A5B   8D 0C 29               STA TY
2A5E   AD FD 28               LDA VY+1
2A61   6D 0D 29               ADC TY+1
2A64   8D 0D 29               STA TY+1
2A67   60                     RTS
```

```
LOC    CODE         LINE

2A68                ;
2A68                ;
2A68                ;MULTIPLY IAC BY X.Y & (A SIGN)
2A68                ;
2A68                ;IN     IAC1 & SIAC
2A68                ;       X.Y (LOW HIGH) A (SIGN)
2A68                ;EXIT   IACR & SIAC
2A68                ;
2A68   4D 04 29     MULT    EOR SIAC        ;CALC SIGN RESULT
2A6B   8D 04 29             STA SIAC
2A6E   8E 02 29             STX IAC2
2A71   8C 03 29             STY IAC2+1
2A74   A9 00                LDA #0         ;ZERO RESULT
2A76   8D 05 29             STA IACR
2A79   8D 06 29             STA IACR+1
2A7C   AD 00 29             LDA IAC1       ;END IF 0
2A7F   0D 01 29             ORA IAC1+1
2A82   F0 08                BEQ MULTEX
2A84   AD 02 29     MULTLO  LDA IAC2        ;END IF 0
2A87   0D 03 29             ORA IAC2+1
2A8A   D0 01                BNE MULTCN
2A8C   60           MULTEX  RTS
2A8D   4E 03 29     MULTCN  LSR IAC2+1      ;IF BIT 0 OF IAC2 THEN
2A90   6E 02 29             ROR IAC2       ;ADD IAC2 TO IACR
2A93   90 1A                BCC MULTNA
2A95   18                   CLC            ;ADD IAC1 TO IACR
2A96   AD 00 29             LDA IAC1
2A99   6D 05 29             ADC IACR
2A9C   8D 05 29             STA IACR
2A9F   AD 01 29             LDA IAC1+1
2AA2   6D 06 29             ADC IACR+1
2AA5   8D 06 29             STA IACR+1
2AA8   90 05                BCC MULTNA     ;NO OVERFLOW
2AAA   A9 FF                LDA #$FF       ;SET ERROR FLAG
2AAC   8D F9 28             STA EFLAG
2AAF   0E 00 29     MULTNA  ASL IAC1        ;IAC1 =IAC1 * 2
2AB2   2E 01 29             ROL IAC1+1
2AB5   90 CD                BCC MULTLO
2AB7   A9 FF                LDA #$FF       ;SET ERROR FLAG
2AB9   8D F9 28             STA EFLAG
2ABC   D0 C6                BNE MULTLO
2ABE                ;
2ABE                ;
2ABE                ;
2ABE                ;DIVIDE IAC1 BY X.Y  (A & SIAC SIGNS)
2ABE                ;IN     IAC LOW HIGH
2ABE                ;       SIAC SIGN VALUE
2ABE                ;     & X,Y LOW HIGH
2ABE                ;       A SIGN
2ABE                ;
2ABE                ;EXIT   IACR LOW HIGH
2ABE                ;       SIAC SIGN
2ABE                ;
2ABE   4D 04 29     DIVIDE  EOR SIAC        ;CALC SIGN
2AC1   8D 04 29             STA SIAC
2AC4   8E 02 29             STX IAC2
2AC7   8C 03 29             STY IAC2+1
2ACA   8A                   TXA
2ACB   0D 03 29             ORA IAC2+1
2ACE   D0 05                BNE DIVN0
2AD0   A2 14                LDX #$14       ;DIVISION BY ZERO
2AD2   4C 37 A4             JMP $A437
2AD5   A9 00        DIVN0   LDA #0          ;ZERO RESULT
2AD7   8D 05 29             STA IACR
2ADA   8D 06 29             STA IACR+1
2ADD   AA                   TAX
```

```
LOC    CODE         LINE

2ADE   AD 03 29            LDA IAC2+1      ;CHECK HI BIT
2AE1   30 09               BMI DIVSUL
2AE3   E8          DIVRL   INX              ;COUNT SHIFTS
2AE4   0E 02 29            ASL IAC2
2AE7   2E 03 29            ROL IAC2+1
2AEA   10 F7               BPL DIVRL       ;SHIFT UNTIL HI BIT SET
2AEC   0E 05 29    DIVSUL  ASL IACR         ;RESULT =RESULT*2
2AEF   2E 06 29            ROL IACR+1
2AF2   38                  SEC             ;Y.A =IAC1-IAC2
2AF3   AD 00 29            LDA IAC1
2AF6   ED 02 29            SBC IAC2
2AF9   A8                  TAY
2AFA   AD 01 29            LDA IAC1+1
2AFD   ED 03 29            SBC IAC2+1
2B00   90 0E               BCC DIVLST      ;IAC1 < IAC2
2B02   EE 05 29            INC IACR        ;RESULT =RESULT+1
2B05   D0 03               BNE DIVNC
2B07   EE 06 29            INC IACR+1
2B0A   8D 01 29    DIVNC   STA IAC1+1       ;IAC1 =Y.A
2B0D   8C 00 29            STY IAC1
2B10   4E 03 29    DIVLST  LSR IAC2+1       ;IAC2=IAC2 /2
2B13   6E 02 29            ROR IAC2
2B16   CA                  DEX             ;COUNT SHIFTS
2B17   10 D3               BPL DIVSUL
2B19   60                  RTS
2B1A                ;
2B1A                ;*************************
2B1A                ;PPLOT X,Y,Z,COLOUR,BRUSH
2B1A                ;     THREE DIMENSIONAL
2B1A                ;     PERSPECTIVE PLOT
2B1A                ;*************************
2B1A                ;
2B1A   A2 00       PPLOT   LDX #0           ;CLEAR ERROR FLAG
2B1C   8E F9 28            STX EFLAG
2B1F   20 60 29            JSR GXYZ        ;GET X,Y,Z
2B22   20 42 2B            JSR GCB         ;GET COLOUR,BRUSH
2B25   AD 09 29            LDA TX
2B28   85 59               STA T2
2B2A   AD 0A 29            LDA TX+1
2B2D   85 5A               STA T2+1
2B2F   AD 0C 29            LDA TY
2B32   85 5B               STA T3
2B34   AD 0D 29            LDA TY+1
2B37   85 5C               STA T3+1
2B39   AD F9 28            LDA EFLAG
2B3C   D0 03               BNE PPABOR
2B3E   4C 02 10            JMP PLOT
2B41   60          PPABOR  RTS
2B42                ;
2B42                ;
2B42                ;GET COLOUR & BRUSH
2B42                ;
2B42   20 F1 B7    GCB     JSR PARAM        ;GET COLOUR
2B45   8A                  TXA
2B46   29 0F               AND #$0F
2B48   85 FD               STA COL
2B4A   85 FE               STA COL+1
2B4C   20 F1 B7            JSR PARAM       ;GET BRUSH
2B4F   86 FC               STX PBR
2B51   A5 02               LDA MODE
2B53   F0 03               BEQ GCB1
2B55   A9 03               LDA #$03
2B57   2C                  .BYT $2C
2B58   A9 01       GCB1    LDA #$01
2B5A   25 FC               AND PBR
2B5C   85 FC               STA PBR
```

```
LOC   CODE        LINE

2B5E  60                 RTS
2B5F              ;
2B5F              ;**************************
2B5F              ;PDRAW X,Y,Z,X1,Y1,Z1,COLOUR,BRUSH
2B5F              ;      THREE DIMENSIONAL
2B5F              ;      PERSPECTIVE DRAW
2B5F              ;      LINE
2B5F              ;**************************
2B5F              ;
2B5F  A2 00       PDRAW  LDX #0          ;CLEAR ERROR FLAG
2B61  8E F9 28           STX EFLAG
2B64  20 60 29           JSR GXYZ        ;GET X,Y,Z
2B67  AD 09 29           LDA TX
2B6A  8D 3C 03           STA X1
2B6D  AD 0A 29           LDA TX+1
2B70  8D 3D 03           STA X1+1
2B73  AD 0C 29           LDA TY
2B76  8D 3E 03           STA Y1
2B79  AD 0D 29           LDA TY+1
2B7C  8D 3F 03           STA Y1+1
2B7F  20 FD AE           JSR $AEFD       ;CHECK ','
2B82  20 60 29           JSR GXYZ        ;GET X1,Y1,Z1
2B85  AD 09 29           LDA TX
2B88  8D 40 03           STA X2
2B8B  AD 0A 29           LDA TX+1
2B8E  8D 41 03           STA X2+1
2B91  AD 0C 29           LDA TY
2B94  8D 42 03           STA Y2
2B97  AD 0D 29           LDA TY+1
2B9A  8D 43 03           STA Y2+1
2B9D  20 42 2B           JSR GCB         ;GET COLOUR & BRUSH
2BA0  AD F9 28           LDA EFLAG
2BA3  D0 03              BNE PDABOR
2BA5  4C 40 11           JMP BOX         ;DRAW LINE
2BA8  60          PDABOR RTS
2BA9                     .END
```

5 MISCELLANEOUS ROUTINES

The commands in this section are for sprite control, and loading and saving pictures.

OFF

Abbreviated entry: O(shift)F

Affected Basic abbreviations: None

Token: Hex $D7   Decimal 215

Purpose: To disable a sprite.

Syntax: OFF sp#

Errors: Illegal quantity – if the sprite number is <Ø or >7

Use: OFF is used to disable a previously enabled sprite. The value sp# is the sprite number (Ø–7).

Routine entry point: $2BA9

Routine operation: This routine is a very short one. It simply reads in the sprite number and resets the relevant bit in the enable register.

PLACE

Abbreviated entry: PL(shift)A

Affected Basic abbreviations: None

Token: Hex $D8 Decimal 216

Purpose: To put a sprite at a certain position.

Syntax: PLACE sp#,X,Y

Errors: Illegal quantity – if any of the values is out of its range

Use: PLACE is used to position a sprite on the screen. The sprite to be positioned is specified by sp#, and is a value between Ø and 7. The value X is the X coordinate of the sprite. This value lies in the range Ø to 511 and the value is the top left corner of the sprite. The value Y is the sprite Y coordinate and is in the range Ø to 255.

The X and Y coordinates of the sprite are not at the same position as the plotting coordinates. The X value is 24 less than the plotting coordinate. The Y value runs from top to bottom (rather than bottom to top as in plotting) and the top line on the screen is equivalent to the value 51:

X=X(plotting)+24
Y=251−Y(plotting)

Routine entry point: $2BBA

Routine operation: The X and Y values are read in and the Y value and low byte of the X value are stored directly into the VIC chip. If the high byte of the X value is not zero, the correct bit in register 16 is set. Otherwise the bit is reset.

SPRITE

Abbreviated entry: S(shift)P

Affected Basic abbreviations: SPC(– SP(shift)C

Token: Hex $D6 Decimal 214

Purpose: To enable a sprite for the graphics screen.

Syntax: SPRITE sp#,bl#,XX,YX,pr,col,mul[,col1,col2]

Errors: Illegal quantity - if any parameter is out of range

Use: SPRITE will, in one command, set the expansions (in X and Y) and set priority to background, colour and multicolour. If in multicolour the two extra colours are allocated. The parameters are as follows:

sp# - Sprite number (Ø–7)
bl# - Sprite block pointer (33–63)
XX - X expand (Ø = off, 1 = on)
YX - Y expand (Ø = off, 1 = on)
pr - Sprite to background priority (Ø = behind, 1 = in front)
col - Sprite main colour (Ø–15)
mul - Multicolour (Ø = off, 1 = on)
col1 - Sprite multicolour 1 (Ø–15)
col2 - Sprite multicolour 2 (Ø–15)

The block number 'bl#' is the pointer to the memory in which the sprite is stored. This value is determined from the start address, thus:

bl# = (address–49152)/64 or
address = 49152+bl#*64

The value of the address lies between 51264 and 53184 in steps of 64. This value is the address in memory of the first byte of the sprite.

SPRITE is illustrated in Program 5.

```
1 REM ***   SPRITE MAN   ***
2 REM     DEMO OF PLACE
5 GOSUB700
10 A=51264
20 GOSUB520
30 A=51328
40 GOSUB520
41 A=51392
42 GOSUB520
43 A=51456
44 GOSUB520
45 GOSUB400
50 FORI=320TO70STEP-8
60 SPRITE0,33,0,0,0,5,0
70 PLACE0,I+6,100
80 FORJ=0TO100:NEXT
90 SPRITE0,35,0,0,0,5,0
100 PLACE0,I+4,100
110 FORJ=0TO100:NEXT
120 SPRITE0,34,0,0,0,5,0
130 PLACE0,I+2,100
140 FORJ=0TO100:NEXT
150 SPRITE0,35,0,0,0,5,0
160 PLACE0,I,100
170 FORJ=0TO100:NEXT
240 NEXT
250 FORI=58TO80STEP4
260 FORJ=50TO20STEP-.5
```

```
270 PLACE3,47,J
280 NEXT
290 FORJ=20TOI
300 PLACE3,47,J:NEXT
310 PLACE0,74,I+42
320 NEXT
330 GETA$:IFA$<>"←"THEN330
398 NORM
399 STOP
400 OFF0:OFF1:OFF2
410 FORI=100TO190
420 DRAW0,I,23,I,7,1
430 NEXT
450 OFF0:OFF1:OFF2
460 FORI=170TO47STEP-0.5
470 PLACE3,I,50
480 NEXT
485 FORI=100TO128
487 DRAW24,I,320,I,12,1
489 NEXT
490 RETURN
520 FORI=0TO20
530 READ A$:P=0
540 FORJ=0TO2:T=0
550 FORK=0TO7:P=P+1
560 IFMID$(A$,P,1)="*"THENT=T OR2↑(7-K)
570 NEXT
580 POKEA,T:A=A+1
590 NEXT
600 NEXT
610 RETURN
700 FORI=51264TO51264+64*4
710 POKEI,0
720 NEXT
740 SPRITE0,33,1,1,1,5,0
750 PLACE0,50,50
760 SPRITE1,34,1,1,1,6,0
770 PLACE1,90,50
780 SPRITE2,35,1,1,1,7,0
790 PLACE2,130,50
800 SPRITE3,36,1,1,1,9,0
810 PLACE3,170,50
820 RETURN
1000 DATA...........****.........
1001 DATA...........****.........
1002 DATA.........*******........
1003 DATA...........****.........
1004 DATA..........**.***........
1005 DATA.........*******........
1006 DATA...........*****........
1007 DATA..........*****.........
1008 DATA............**..........
1009 DATA...........****.........
1010 DATA..........*****.........
1011 DATA.........*.****.........
1012 DATA.......**..****.........
1013 DATA...........****.........
1014 DATA...........****.........
1015 DATA..........****..........
1016 DATA.........**.**..........
1017 DATA........**..**..........
1018 DATA......****..**..........
1019 DATA.......** ..**..........
1020 DATA...........***..........
1999 REM
2000 DATA...........****.........
2001 DATA...........****.........
2002 DATA.........*******........
2003 DATA...........****.........
```

```
2004 DATA..........**.***........
2005 DATA.........*******........
2006 DATA............****........
2007 DATA..........*****.........
2008 DATA............**..........
2009 DATA.......*...****.........
2010 DATA........*.*****.........
2011 DATA.........*.****.........
2012 DATA...........****.........
2013 DATA...........****.........
2014 DATA...........****.........
2015 DATA............**..........
2016 DATA...........***..........
2017 DATA..........*****.........
2018 DATA..........**..**........
2019 DATA..........**...**.......
2020 DATA.........***..*.........
2999 REM
3000 DATA...........****.........
3001 DATA...........****.........
3002 DATA.........*******........
3003 DATA...........****.........
3004 DATA..........**.***........
3005 DATA.........*******........
3006 DATA............****........
3007 DATA..........*****.........
3008 DATA............**..........
3009 DATA...........****.........
3010 DATA..........*****.........
3011 DATA.......***.****.........
3012 DATA...........****.........
3013 DATA...........****.........
3014 DATA...........****.........
3015 DATA............**..........
3016 DATA............**..........
3017 DATA............**..........
3018 DATA............**..........
3019 DATA............**..........
3020 DATA...........***..........
3999 REM
4000 DATA.................**.....
4001 DATA...................**...
4002 DATA....................**..
4003 DATA....................**..
4004 DATA....................**..
4005 DATA...................***..
4006 DATA..................****..
4007 DATA*******..........*****..
4008 DATA***********************.
4009 DATA***********************.
4010 DATA***********************.
4011 DATA*******..........*****..
4012 DATA.................*****..
4013 DATA.................*****..
4014 DATA..................***...
4015 DATA..................***...
4016 DATA..................***...
4017 DATA..................***...
4018 DATA................*******.
4019 DATA................*******.
4020 DATA................*******.
4999 REM
```

Program 5.

Routine entry point: $2C∅B

Routine operation: The values are read in and stored in their relevant locations in the VIC chip (after being checked for range validity).

```
LOC    CODE         LINE

2BA9                      .LIB SPRITE
2BA9               ; ROUTINE TO DISABLE SPRITE
2BA9               ;
2BA9   20 9E B7    OFF     JSR $B79E       ;GET # TO TURN OFF
2BAC   BD E0 0F            LDA TOP2X,X    ;FIND 2↑X
2BAF   49 FF               EOR #$FF
2BB1   2D CA 2C            AND ENABLE     ;DISABLE SPRITE
2BB4   8D CA 2C            STA ENABLE
2BB7   4C CB 2C            JMP STENAB
2BBA               ;
2BBA               ; ROUTINE TO PLACE SPRITE
2BBA               ;
2BBA   20 9E B7    PLACE   JSR $B79E       ;GET # TO PLACE
2BBD   8E 66 03            STX CHAR       ;STORE IT
2BC0   20 FE 2B            JSR CHKNO8     ;CHECK IN RANGE
2BC3   8A                  TXA
2BC4   0A                  ASL A          ;MULTIPLY BY 2
2BC5   8D 67 03            STA RVORN      ;STORE IT
2BC8   20 FD AE            JSR CHKCOM     ;SCAN PAST ','
2BCB   20 EB B7            JSR PARAMS     ;GET SPRITE POSITION
2BCE   AC 67 03            LDY RVORN      ;GET OFFSET FOR PLACE
2BD1   8A                  TXA            ;GET Y POS
2BD2   99 01 D0            STA VIC+1,Y    ;SET IT
2BD5   A5 14               LDA $14        ;GET X POS
2BD7   99 00 D0            STA VIC,Y      ;SET IT
2BDA   A5 15               LDA $15        ;GET X POS HIGH
2BDC   D0 0F               BNE SETHI      ;NOT ZERO, SET HIGH BIT
2BDE   AE 66 03            LDX CHAR
2BE1   BD E0 0F            LDA TOP2X,X    ;ZERO HIGH BIT
2BE4   49 FF               EOR #$FF
2BE6   2D 10 D0            AND VIC+16
2BE9   8D 10 D0            STA VIC+16
2BEC   60                  RTS
2BED   C9 01       SETHI   CMP #$01        ;IS HIGH BIT 1?
2BEF   D0 11               BNE DISPER     ;NO, ERROR
2BF1   AE 66 03            LDX CHAR
2BF4   BD E0 0F            LDA TOP2X,X
2BF7   0D 10 D0            ORA VIC+16     ;SET HIGH BIT TO 1
2BFA   8D 10 D0            STA VIC+16
2BFD   60                  RTS
2BFE               ;
2BFE               ; ROUTINE TO CHECK X VALUE
2BFE               ;
2BFE   E0 08       CHKNO8 CPX #$08         ;IS SPRITE# <8?
2C00   90 08               BCC DONECC     ;YES, O.K.
2C02   20 52 0E    DISPER JSR NORM         ;SET TEXT SCREEN
2C05   A2 0E               LDX #$0E
2C07   4C 37 A4            JMP $A437      ;'ILLEGAL QUANTITY'
2C0A   60          DONECC RTS
2C0B               ;
2C0B               ; ROUTINE TO ENABLE SPRITE
2C0B               ;
2C0B   20 9E B7    SPRITE JSR $B79E        ;GET SPRITE#
2C0E   8E 66 03            STX CHAR       ;STORE IT
2C11   BD E0 0F            LDA TOP2X,X    ;GET 2↑X
2C14   0D CA 2C            ORA ENABLE     ;TURN SPRITE ON
2C17   8D CA 2C            STA ENABLE
2C1A   20 F1 B7            JSR PARAM      ;GET BLOCK#
```

```
LOC    CODE          LINE

2C1D   8A                   TXA              ;TRANSFER TO .A
2C1E   AE 66 03             LDX CHAR
2C21   9D F8 C3             STA $C3F8,X      ;SET POINTER TO BLOCK
2C24   20 F1 B7             JSR PARAM        ;GET X EXPANSION
2C27   E0 01                CPX #$01
2C29   D0 0E                BNE XXOFF
2C2B   AE 66 03             LDX CHAR
2C2E   BD E0 0F             LDA TOP2X,X
2C31   0D 1D D0             ORA VIC+29       ;SET X EXPAND ON
2C34   8D 1D D0             STA VIC+29
2C37   D0 0E                BNE YEXP
2C39   AE 66 03      XXOFF  LDX CHAR
2C3C   BD E0 0F             LDA TOP2X,X
2C3F   49 FF                EOR #$FF
2C41   2D 1D D0             AND VIC+29       ;SET X EXPAND OFF
2C44   8D 1D D0             STA VIC+29
2C47   20 F1 B7      YEXP   JSR PARAM         ;GET Y EXPANSION
2C4A   E0 01                CPX #$01
2C4C   D0 0E                BNE YXOFF
2C4E   AE 66 03             LDX CHAR
2C51   BD E0 0F             LDA TOP2X,X
2C54   0D 17 D0             ORA VIC+23 SET Y EXPAND ON
2C57   8D 17 D0             STA VIC+23
2C5A   D0 0E                BNE PRIOR
2C5C   AE 66 03      YXOFF  LDX CHAR
2C5F   BD E0 0F             LDA TOP2X,X
2C62   49 FF                EOR #$FF
2C64   2D 17 D0             AND VIC+23       ;SET Y EXPAND OFF
2C67   8D 17 D0             STA VIC+23
2C6A   20 F1 B7      PRIOR  JSR PARAM         ;GET PRIORITY
2C6D   E0 00                CPX #$00
2C6F   D0 0E                BNE PROFF
2C71   AE 66 03             LDX CHAR
2C74   BD E0 0F             LDA TOP2X,X
2C77   0D 1B D0             ORA VIC+27       ;SPRITE BEHIND
2C7A   8D 1B D0             STA VIC+27       ; BACKGROUND
2C7D   D0 0E                BNE COLOUR
2C7F   AE 66 03      PROFF  LDX CHAR
2C82   BD E0 0F             LDA TOP2X,X
2C85   49 FF                EOR #$FF
2C87   2D 1B D0             AND VIC+27       ;SPRITE IN FRONT
2C8A   8D 1B D0             STA VIC+27       ; OF BACKGROUND
2C8D   20 F1 B7      COLOUR JSR PARAM         ;GET SPRITE COLOUR
2C90   8A                   TXA
2C91   AE 66 03             LDX CHAR
2C94   9D 27 D0             STA VIC+39,X     ;STORE THE COLOUR
2C97   20 F1 B7             JSR PARAM        ;GET MULTICOLOUR
2C9A   E0 00                CPX #$00
2C9C   D0 11                BNE MCON
2C9E   AE 66 03             LDX CHAR
2CA1   BD E0 0F             LDA TOP2X,X
2CA4   49 FF                EOR #$FF
2CA6   2D 1C D0             AND VIC+28       ;TURN OFF MULTICOLOUR
2CA9   8D 1C D0             STA VIC+28       ; FOR THAT SPRITE
2CAC   4C CB 2C             JMP STENAB       ;SPRITE COMMAND COMPLETE
2CAF   AE 66 03      MCON   LDX CHAR
2CB2   BD E0 0F             LDA TOP2X,X
2CB5   0D 1C D0             ORA VIC+28       ;TURN ON MULTICOLOUR
2CB8   8D 1C D0             STA VIC+28       ; FOR THAT SPRITE
2CBB   20 F1 B7             JSR PARAM        ;GET MULTICOLOUR#1
2CBE   8E 25 D0             STX VIC+37       ;STORE IT
2CC1   20 F1 B7             JSR PARAM        ;GET MULTICOLOUR#2
2CC4   8E 26 D0             STX VIC+38       ;STORE IT
2CC7   4C CB 2C             JMP STENAB       ;COMPLETE SPRITE
2CCA   00            ENABLE .BYT 0
2CCB   A9 20         STENAB LDA #$20          ;ARE WE IN
```

```
LOC   CODE       LINE

2CCD  2C 11 D0          BIT VIC+17     ;TEXT MODE?
2CD0  D0 01             BNE STENA1     ;NO, ENABLE SPRITES
2CD2  60                RTS
2CD3  AD CA 2C   STENA1 LDA ENABLE      ;SET SPRITE ENABLE
2CD6  8D 15 D0          STA VIC+21
2CD9  60                RTS
2CDA                    .END
```

GLOAD and GSAVE

Abbreviated entry: GLOAD: G(shift)L
GSAVE: G(shift)S

Affected Basic abbreviations: None

Token: GLOAD: Hex $E2 Decimal 226
GSAVE: Hex $E1 Decimal 225

Purpose: To load or save the graphics screen.

Syntax: GLOAD filename[,dev]
GSAVE filename[,dev]

Errors: The error messages produced are as in the normal LOAD and SAVE commands.

Use: GLOAD is used to load a previously GSAVEd screen. GSAVE saves off a graphics screen to disk or tape along with the colour, sprite definitions and settings.

Routine entry point: GLOAD: $2D59
GSAVE: $2CDA

Routine operation: GLOAD reads in the filename and device number, and loads the file from the device. After loading, the graphics screen is copied from the temporary location into the display area. The colour screen is treated likewise as are the VIC chip register settings. The other information saved (the graphics mode, sprite enable register, screen and border colours) are stored to their relevant locations.

GSAVE first gets the filename and device number and then copies the graphics screen from behind the kernal ROM (unsavable RAM) to the RAM behind the Basic ROM. The colour memory is then copied over to the memory starting at location $C4ØØ and the VIC chip is copied to memory starting at $C8ØØ. The mode flag, sprite enable flag, screen colour and border colour are stored directly after the colour memory and then the whole memory from $AØØØ to $DØØØ is saved.

```
LOC   CODE        LINE

2CDA                      .LIB LOAD/SAVE
2CDA  20 D4 E1    GSAVE   JSR $E1D4       ;GET FILE PARAMETERS
2CDD  A0 00               LDY #$00
2CDF  B9 00 D8    COPCOL  LDA $D800,Y     ;COPY THE SCREEN
2CE2  99 00 C4            STA $C400,Y     ; COLOUR MEMORY INTO
2CE5  B9 00 D9            LDA $D900,Y     ; SAVABLE RAM
2CE8  99 00 C5            STA $C500,Y
2CEB  B9 00 DA            LDA $DA00,Y
2CEE  99 00 C6            STA $C600,Y
2CF1  B9 00 DB            LDA $DB00,Y
2CF4  99 00 C7            STA $C700,Y
2CF7  88                  DEY
2CF8  D0 E5               BNE COPCOL      ;UNTIL DONE
2CFA  AD CA 2C            LDA ENABLE      ;GET SPRITE ENABLE FLAG
2CFD  8D FE C7            STA $C7FE       ;STORE IT FOR SAVE
2D00  A5 02               LDA MODE        ;GET MODE FLAG
2D02  8D FF C7            STA $C7FF       ;STORE IT FOR SAVE
2D05  AD 82 0E            LDA SCTMP1      ;GET SCREEN COLOUR
2D08  8D FD C7            STA $C7FD       ;STORE IT FOR SAVE
2D0B  AD 83 0E            LDA BDTMP1      ;GET BORDER COLOUR
2D0E  8D FC C7            STA $C7FC       ;STORE IT FOR SAVE
2D11  A0 2F               LDY #47
2D13  B9 00 D0    VICLOP  LDA $D000,Y     ;COPY THE VIC CHIP
2D16  99 00 C8            STA $C800,Y     ; REGISTERS FOR SAVE
2D19  88                  DEY
2D1A  10 F7               BPL VICLOP      ;UNTIL DONE
2D1C  A9 E0               LDA #$E0        ;SET POINTERS TO COPY
2D1E  85 FC               STA $FC         ; HIRES SCREEN TO
2D20  A9 00               LDA #$00        ; SAVABLE RAM
2D22  85 FB               STA $FB
2D24  85 FD               STA $FD
2D26  A9 A0               LDA #$A0
2D28  85 FE               STA $FE
2D2A  20 2D 11            JSR KEROUT      ;SWITCH OUT KERNAL
2D2D  A0 00               LDY #$00
2D2F  B1 FB       COPSC   LDA ($FB),Y     ;GET BYTE
2D31  91 FD               STA ($FD),Y     ;STORE BYTE
2D33  C8                  INY
2D34  D0 F9               BNE COPSC       ;UNTIL PAGE DONE
2D36  E6 FE               INC $FE
2D38  E6 FC               INC $FC
2D3A  D0 F3               BNE COPSC       ;UNTIL 8K DONE
2D3C  20 34 11            JSR KERIN       ;KERNAL BACK IN
2D3F  A9 A0               LDA #$A0        ;SET SAVE POINTERS
2D41  85 FC               STA $FC
2D43  A5 01               LDA $01         ;REMOVE BASIC ROM
2D45  29 FE               AND #$FE
2D47  85 01               STA $01
2D49  A9 FB               LDA #$FB
2D4B  A2 00               LDX #$00
2D4D  A0 D0               LDY #$D0
2D4F  20 D8 FF            JSR $FFD8       ;SAVE FILE
2D52  A5 01               LDA $01         ;PUT BASIC ROM BACK
2D54  09 01               ORA #$01
2D56  85 01               STA $01
2D58  60                  RTS
2D59              ;
2D59  20 D4 E1    GLOAD   JSR $E1D4       ;GET FILE PARAMETERS
2D5C  A9 00               LDA #$00        ;SET FOR ALT LOAD
2D5E  85 B9               STA $B9
2D60  AA                  TAX             ;SET ALT LOAD AT
2D61  A0 A0               LDY #$A0        ;$A000
2D63  20 D5 FF            JSR $FFD5       ;LOAD FILE
2D66  B0 71               BCS LOADER      ;ERROR
2D68  AD FF C7            LDA $C7FF       ;GET MODE FLAG STORE
2D6B  85 02               STA MODE        ;SET MODE
```

```
LOC   CODE        LINE

2D6D  AD FE C7           LDA $C7FE      ;GET SPRITE ENABLE
2D70  8D CA 2C           STA ENABLE     ;SET ENABLE
2D73  AD FD C7           LDA $C7FD      ;GET SCREEN COLOUR
2D76  8D 82 0E           STA SCTMP1     ;SET COLOUR
2D79  8D 21 D0           STA $D021
2D7C  AD FC C7           LDA $C7FC      ;GET BORDER COLOUR
2D7F  8D 83 0E           STA BDTMP1     ;SET BORDER
2D82  8D 20 D0           STA $D020
2D85  A0 00              LDY #$00
2D87  B9 00 C4    COLLOP LDA $C400,Y     ;LOOP TO COPY
2D8A  99 00 D8           STA $D800,Y    ; COLOUR RAM STORE
2D8D  B9 00 C5           LDA $C500,Y    ; BACK INTO THE COLOUR
2D90  99 00 D9           STA $D900,Y    ; RAM
2D93  B9 00 C6           LDA $C600,Y
2D96  99 00 DA           STA $DA00,Y
2D99  B9 00 C7           LDA $C700,Y
2D9C  99 00 DB           STA $DB00,Y
2D9F  88                 DEY
2DA0  D0 E5              BNE COLLOP
2DA2  A0 2F              LDY #47
2DA4  B9 00 C8    MOVVIC LDA $C800,Y     ;COPY VIC CHIP
2DA7  99 00 D0           STA $D000,Y    ; REGISTER STORE
2DAA  88                 DEY            ;INTO THE VIC CHIP
2DAB  10 F7              BPL MOVVIC
2DAD  A9 00              LDA #$00       ;SET POINTERS TO
2DAF  85 FB              STA $FB        ; COPY THE SCREEN
2DB1  85 FD              STA $FD        ; FROM STORE INTO
2DB3  A9 E0              LDA #$E0       ; CORRECT MEMORY
2DB5  85 FC              STA $FC
2DB7  A9 A0              LDA #$A0
2DB9  85 FE              STA $FE
2DBB  A5 01              LDA $01        ;REMOVE BASIC ROM
2DBD  29 FE              AND #$FE
2DBF  85 01              STA $01
2DC1  A0 00              LDY #$00
2DC3  B1 FD       SCCOP  LDA ($FD),Y     ;GET BYTE
2DC5  91 FB              STA ($FB),Y    ;STORE BYTE
2DC7  C8                 INY
2DC8  D0 F9              BNE SCCOP      ;UNTIL END OF PAGE
2DCA  E6 FE              INC $FE
2DCC  E6 FC              INC $FC
2DCE  D0 F3              BNE SCCOP      ;UNTIL 8K DONE
2DD0  A5 01              LDA $01        ;PUT BASIC ROM BACK
2DD2  09 01              ORA #$01
2DD4  85 01              STA $01
2DD6  4C 4C 0D           JMP R00008     ;CAUSE 'GRAPH' COMMAND
2DD9  48          LOADER PHA             ;STORE ERROR
2DDA  20 52 0E           JSR NORM       ;GO TO TEXT
2DDD  68                 PLA            ;RESTORE ERROR
2DDE  4C F9 E0           JMP $E0F9      ;OUTPUT ERROR
2DE1                     .END
2DE1  00                 .BYT 0
2DE2  00          BSSTRT .BYT 0,0
2DE3  00
2DE4  00          VRSTRT .BYT 0
2DE5                     .END
```

Symbol table

```
SYMBOL VALUE
 A0       035E    A1       0360    AA2R     24A0    AAERR    247C
 AALOOP   24BD    AALOP    24D3    AAMRC    24E4    AASOK    2481
 ABFA     2333    ABNC     233E    ABSC     22FF    ABSLOP   230D
 ADBADS   22FA    ADD      1DFB    ADDS     1E1B    ADDSCO   1E12
 ADDSF    1E35    ADDSUB   22CA    ADV2NC   2374    AMULT    2433
```

SYMBOL VALUE

ARITH	0BD6	ARITH1	0BEB	ARITH2	0BF2	ARITH3	0BE5
ASARAR	212C	ASLOOP	2105	ASLOP	2107	ASNC	2113
ASNC9	2120	ASR1R	2135	ASR2R	2141	ASREXT	2240
ASRIR	21BB	ASRITR	2205	ASRLOP	21D5	ASRLP1	21D9
ASRNC1	2225	ASRNC2	21E4	ASRSOK	2151	ASRSUB	214C
ASRTM	2215	ASRTM1	2219	ASRTM3	2232	ASSGN	20BF
ASSIC	20C9	ASSR1	20E6	ASSTLO	2197	ASSTN1	21A1
ASSTN2	21A7	ASSTN3	21AF	ATINT1	25DE	BDTMP	0E81
BDTMP1	0E83	BEGIN	1679	BEXPOK	1FB9	BITIS1	1658
BITIS2	165F	BITOK	0F0E	BKFLG	0C24	BORDER	0EB2
BOX	1140	BRCOL	0373	BROK	1818	BROK1	1829
BRUSH0	1028	BRUSH1	1042	BRUSH2	106E	BRUSH3	1092
BSSTRT	2DE2	CADDR	09D0	CALCD1	11E9	CALCXY	1AF9
CHAR	0366	CHAR01	158A	CHAREX	15F5	CHAROK	13A8
CHECKX	1183	CHECKY	11BA	CHKCOM	AEFD	CHKNO8	2BFE
CHKOP	1F83	CHOK	1EEC	CHOK1	1F0A	CHOK2	1F56
CHOK2A	1F63	CHOK3	1FEB	CHRGET	0073	CHRGOT	0079
CIR01	1D0C	CIR03	1D8C	CIRANG	1E4F	CIRBMP	1D92
CIRCLE	1CA1	CIRLO2	1D12	CIRLOO	1CFC	CIROK1	1CE0
CIRTNC	1D9A	CLIST	094B	CLRCOL	0E3A	CLRMEM	0D9C
CLRSCN	0E27	CNTR1	036A	COL	00FD	COLD	08B5
COLLOP	2D87	COLM	2574	COLOUR	2C8D	COLTMP	037C
CONTSH	271C	CONV	1453	COPCOL	2CDF	COPSC	2D2F
COSR	25A2	COST	1C38	COSVAL	1BCA	CRNC01	0A97
CRNC02	0AA3	CRNC03	0AB9	CRNC04	0AC1	CRNC05	0AC4
CRNC06	0AC7	CRNC07	0AC9	CRNC08	0ACB	CRNC09	0AE0
CRNC10	0AE2	CRNC11	0AE9	CRNC12	0AF2	CRNC13	0AF9
CRNC15	0B03	CRNC16	0B0E	CRNC17	0B10	CRNC18	0B21
CRNC19	0B34	CRNC20	0B36	CRNC21	0B46	CRNC22	0B4A
CRNCHT	0A91	CT	0356	D1	0358	DERE1	1447
DFILL	1826	DISAB	0A2C	DISPER	2C02	DISRAS	0A6E
DIV8	0F4D	DIVIDE	2ABE	DIVLST	2B10	DIVN0	2AD5
DIVNC	2B0A	DIVRL	2AE3	DIVSUL	2AEC	DM1	184B
DM4	1848	DMAIN	182C	DOMAT	2012	DOMATA	20A1
DOMULT	2330	DONE	0D7D	DONECC	2C0A	DORAST	0C52
DOSUB	2326	DOT	0ECE	EDVNA1	1F17	EDVNA2	1F70
EDVNA3	1FFA	EFLAG	28F9	ENAB	0A4D	ENAB1	0A67
ENABLE	2CCA	ENDCOL	251E	ENDROW	2554	ENTMP	0A2A
EOL	080B	EVLEXP	AD8A	F1TOV1	2407	FACM	1EC6
FACONT	225A	FACT	1ECB	FAE1	2296	FAEX	22B6
FAGETS	2288	FALOOP	2249	FANAR	226B	FANC	2285
FANDAR	2885	FANDOK	229B	FICONT	289E	FIE1	28DA
FIGETS	28CC	FILOOP	288D	FILRT	17C8	FILRT1	17F1
FILRT2	17F9	FILRT4	17F8	FIN	1452	FINAR	28AF
FINC	28C9	FINDAR	2241	FINDOK	28DF	FIXALL	1A71
FIXIT	1ADB	FIXX1	1AC2	FIXX2	1AD0	FIXY1	1AC9
FIXY2	1AD7	FNADDR	0C03	FOEQ	1F31	FSTUP	18F2
FSTUP1	18FD	FSTUP2	1904	GADS	243A	GAR3	206B
GCB	2B42	GCB1	2B58	GETEL	286D	GETV3	1FA8
GFPAR1	192C	GFPARS	190A	GLOAD	2D59	GLOOP	1966
GLPARS	1361	GRAPH	0D5D	GSAVE	2CDA	GSBCL1	0EC9
GSBCOL	0EBC	GXTRA	195C	GXTRA1	1959	GXTRA3	1988
GXY	10E3	GXYZ	2960	HAND01	0BC1	HAND02	0BC7
HANDLE	0BB4	HIRES	10AB	IAC1	2900	IAC2	2902
IACR	2905	ILLQ	29AB	INITUP	1BAA	INITX	1BCF
INITY	1C38	IOEXIT	0A19	IRQ1	0C3E	IRQEXT	0C46
IRQINT	0C25	IRQVEC	0C4C	ISNALF	B113	KERIN	1134
KEROUT	112D	L360	1238	L380	126F	L420	12B3
L460	130F	L470	1348	L745	0CCB	L809	0CE0
L810	0CD0	L812	0CE8	L813	0CEA	LG	034C
LINK	0813	LLV2	2575	LNE	1F0D	LNE2	1F66
LNE3	1FEE	LOAD	0A06	LOADER	2DD9	LOOP	0D9F
LOOP0	13AE	LOOP01	13C1	LOOP02	13F0	LOOP1	0E2A
LOOP2	0E3D	LOOPDI	1599	LSW	0375	M	169D
M1	16B0	M2	16B6	M3	16D8	M4	16DD
M5	16FF	M6	1704	M7	172C	MA	257D
MAIN	139D	MAT	1EE1	MB	2582	MC	2587
MCON	2CAF	MD	258C	MERR	243D	MERSGN	295B

SYMBOL VALUE

MMULT	1E7B	MMULT1	1E8B	MMULT2	1E93	MMULT3	1E94
MMULT4	1EAE	MODE	0002	MOVVIC	2DA4	MUL320	0F7C
MUL40	0FAE	MUL8	0F63	MULT	2A68	MULTCN	2A8D
MULTEX	2A8C	MULTI	0D1D	MULTII	1012	MULTLO	2A84
MULTNA	2AAF	N1	1E75	N2	1E77	NASSIG	1F90
NEAA	255D	NELEM	2591	NELEP	2605	NELNC	2811
NOPLOT	12CE	NORM	0E52	NORM1	0E57	NORMAL	13DF
NRHOR	1190	NROW	2572	NSARR0	246C	NSTR	25D7
NSTR1	1F20	NSTR2	1F79	NSTR3	2003	NTEXP2	1F4E
NTEXP3	1FD7	NTINT1	1F2A	NTINT3	200F	NUMOK	1FC3
NXTLNE	1431	NXTPNT	1415	OFF	2BA9	OFFT	00D7
OPJMP	20B5	OPJTAB	20B7	OPTYPE	1ED6	ORDER	234A
ORIGIN	0E84	PARAM	B7F1	PARAMS	B7EB	PBR	00FC
PBRTMP	037B	PDABOR	2BA8	PDRAW	2B5F	PID180	1BC0
PLACE	2BBA	PLOT	1002	PLOT1	1008	PLOTPT	194D
PNTTK	00E6	POINT	036B	POINTC	15F6	POINTK	1624
POINTR	0061	POINTT	1619	POLERR	1996	POLYGN	198C
POLYLP	1A43	POLYMN	19FA	POLYX1	1BD4	POLYX2	1BD9
POLYXC	1BBD	POLYY1	1C3D	POLYY2	1C42	POLYYC	1BBB
POS1	11AE	POS2	11DD	POWER	08E1	PPABOR	2B41
PPLOT	2B1A	PRIN01	0B5D	PRIN02	0B60	PRIN03	0B77
PRIN04	0B7F	PRIN05	0B82	PRIN06	0B8A	PRIN07	0B95
PRIN08	0B71	PRIN09	0B96	PRIN10	0B9E	PRIN11	0BA1
PRIN12	0BA9	PRIN13	0B74	PRINT	0B5B	PRIOR	2C6A
PROFF	2C7F	PTBR	0372	PULL	1743	PULL1	1752
PUSEXT	17C7	PUSH	17A6	PUSHL	1796	PUSHLC	179C
PUSHU	1783	PUSHUC	1789	PVIEW	290F	R00001	0CFE
R00002	0FE8	R00004	113D	R00005	1553	R00007	1673
R00008	0D4C	R00011	0D15	R000DN	0C8B	R000GR	0C88
R000ON	0C83	R217DN	0C6C	R217GR	0C66	R217ON	0C61
RADIUS	1BBF	RAS000	0C7C	RAS217	0C5A	RASFLG	0CEE
RESULT	1E79	ROW	2573	RTN	1360	RVORN	0367
S0	035A	S1	035C	SAERR	1E6D	SAVE	0A11
SCCOF	2DC3	SCREEN	0EA8	SCTMP	0E80	SCTMP1	0E82
SEND	166B	SEND0	1666	SEND1	166E	SEPEXN	2958
SEPN	294C	SEPSGN	2946	SETBAS	093F	SETHI	2BED
SETKER	0876	SETMUL	0D78	SETOFF	1B6D	SETUP	172F
SGNTX	290B	SGNTY	290E	SH	034E	SHAPE	25A7
SHCHK	25B2	SHCHK1	25C1	SHCMOK	273E	SHDRAW	2821
SHEVNA	25CE	SHFILL	2845	SHLNE	25C4	SHLOOP	2713
SHPLOT	2827	SHV	2817	SHVT	2819	SIAC	2904
SIDEOK	199C	SIDES	1BBA	SINR	259D	SINT	1BCF
SINVAL	1BC5	SIZEOK	263B	SPRITE	2C0B	STANDD	162F
STBAS1	0941	STENA1	2CD3	STENAB	2CCB	STKER1	0880
SUB	1DF9	SX	2593	SY	2598	SYNTE	1FD4
T1	0057	T2	0059	T3	005B	T4	005D
T5	005E	T6	005F	TE1	1EDD	TE2	1EDF
TEST01	185F	TEST02	1862	TESTPT	185A	TOP2X	0FE0
TRPT1	2389	TRPT2	237F	TRPT3	2375	TS	0350
TSTCR1	0CFD	TSTCUR	0CEF	TSTXM	18CE	TSTXM1	18E0
TSTXP	18AA	TSTXP1	18BC	TSTYM	188C	TSTYM1	189E
TSTYM2	18A7	TSTYP	1870	TSTYP1	1882	TSTYP2	188B
TT	0352	TX	2909	TXTFLG	0E51	TXTTMP	0A2B
TY	290C	TYMISE	1F1B	UD	0354	UNPLOT	10D3
USW	0376	V1BPT	2415	V1INT	2421	V1REAL	2027
V2BPT	23B2	V2COLP	2577	V2INT	23BE	V2RA	23A6
V2TOT2	2394	V3BPT	23EA	V3INT	23F6	V3TOF1	23DC
VANAME	2579	VECTOR	081F	VIC	D000	VICLOP	2D13
VNAME1	1EBD	VNAME2	1EC0	VNAME3	1EC3	VPTR1	00FB
VPTR2	00FD	VPTR3	009E	VRSTRT	2DE4	VSIZE	257B
VSIZE1	1ED0	VSIZE2	1ED2	VSIZE3	1ED4	VSTT1	1ED7
VSTT2	1ED9	VSTT3	1EDB	VTYPE1	1EBF	VTYPE2	1EC2
VTYPE3	1EC5	VX	28FA	VY	28FC	VZ	28FE
WIND1	0C13	WIND2	0C19	WINDOW	0C05	WINFLG	0C22
WINTMP	0A29	WNTFLG	0C23	WRST	0843	WRST01	085D
WRST02	0871	WRST1	0852	WRST2	0855	X1	033C
X11	036C	X2	0340	X22	036F	X2RAD	1A5C
X2TOX1	1AE1	X2X1LP	1AE3	XD	0344	XE	0348

SYMBOL	VALUE						
XER	0EDE	XMAX	0140	XOK	0EE0	XORIG	0E4D
XTEMP	0368	XTL	0362	XTLTMP	0377	XXOFF	2C39
Y1	033E	Y11	036E	Y2	0342	Y22	0371
Y2TOY1	1AED	Y2Y1LP	1AEF	YD	0346	YE	034A
YEXP	2C47	YMAX	00C8	YND7OK	0F3A	YORIG	0E4F
YTL	0364	YTLTMP	0379	YXOFF	2C5C	ZPVZ	2907
ZPVZNN	29B0						

```
0 REM
1 REM DEMONSTRATION OF THE EXTENDED
2 REM GRAPHICS PACKAGE.
3 REM
10 HIRES1,12:WINDOW OFF:PRINT"▯"
20 CHAR32,191,0,1,1," ♦╮—♠ |—♠┐├╮—♥ "
30 CHAR48,191,7,2,0,"♦╮—♠ |—♠┐├╮—♥"
40 CHAR32,199,0,1,1,"                      "
50 CHAR32,183,0,1,1,"                      "
60 CHAR144,175,2,1,0."BY"
90 CHAR8,29,6,1,0,"♦IFRA ♥OFTWARE LTD."
110 DRAW90,139,229,139,7,1
120 DRAW229,139,229,119,7,1
130 DRAW229,119,130.79,7,1
140 DRAW130,79,229,79,7,1
150 DRAW229,79,229,59,7,1
160 DRAW229,59,90,59,7,1
170 DRAW90,59,90,79,7,1
180 DRAW90,79,189,119,7,1
190 DRAW189,119,90,119,7,1
200 DRAW90,119,90,139,7,1
210 DFILL[227,60,92,138],14,3,1
220 GOSUB1000
225 SCREEN=0:BORDER=0
230 HIRES1,12
240 FORI=0TO62:POKE51264+I,255:NEXT
250 FORI=0TO7:SPRITEI,33,1,1,IAND1,I,0
260 PLACEI,0,0:NEXT
261 CHAR32,179,6,1,1,"♦IFRA ¬ONTROLLED"
262 CHAR32,179,7,3,0,"♦IFRA ¬ONTROLLED"
263 CHAR104,169,6,1,1,"♥┐└╮|¯♥"
264 CHAR104,169,7,3,0,"♥┐└╮|¯♥"
270 FORI=0TO36
280 FORJ=0TO7
290 PLACEJ,I*J+30,160-J*2:NEXT:NEXT
300 DRAW0,39,319,39,0,1
301 DRAW319,39,319,107,0,1
302 DRAW319,107,0,107,0,1
303 DRAW0,107,0,39,0,1
305 CHAR24,149,6,1,1,"♠ND ╮LL —OUTINES"
306 CHAR24,149,7,3,0,"♠ND ╮LL —OUTINES"
310 FILL160,73,13,3,1
315 DFILL[20,73,60,73,100,73,140,73,180,73,220,73,260,73,300,73],3,0,0
320 FORI=0TO7:FORJ=160-I*2TO0STEP-3
330 PLACEI,I*36+30,J:NEXT:OFFI:NEXT
335 WINDOW ON:PRINT"▒▒ PLUS A 4 LINE TEXT WINDOW TO PRINT TO";
340 GOSUB1000
345 WINDOW OFF
360 HIRES 0,1
370 CHAR 0,198,0,1,0," ÐU CAN PLOT A ┬L ‖┌/ OF / SIDES (3-20)"
380 FORS=3TO20
390 CHAR 150,93,6,1,0,STR$(S)
400 POLYGON S,160,89,89,2,1,0
410 FORI=0TO250:NEXT
420 POLYGON S,160,89,89,2,0,0
425 CHAR 150,93,6,0,0,STR$(S)
430 NEXT
440 GOSUB1040
```

```
450 DEF FNQ(D)=2000/SQR(X*X+(100-Z)↑2+70)-100
460 HIRES0,12
470 PORIGIN160,100,100
480 CHAR0,199,1,1,0,"3 ¬IMENSIONAL COMMANDS ALLOW EASY PLOTT-"
490 CHAR0,191,1,1,0,"ING COMMANDS USING ♠,  L AND ♦ ORDINATES"
500 LX=-160
510 FORZ=0TO200STEP10
520 X=-160:LY=FNQ(0):LX=-160
530 FORX=-160TO160STEP5
540 Y=FNQ(0)
550 PDRAW LX,LY,Z,X,Y,Z,0,1
560 LX=X:LY=Y
570 NEXT X,Z
580 LZ=0
590 FORX=-160TO160STEP10
600 Z=0:LY=FNQ(0):LZ=0
610 FORZ=0TO200STEP5
620 Y=FNQ(0)
630 PDRAW X,LY,LZ,X,Y,Z,0,1
640 LY=Y:LZ=Z
650 NEXT Z,X
660 GOSUB1040
860 HIRES0,12
870 CHAR16,198,0,1,0,"┌R JUST NORMAL ┐┘┌I ING CAN BE DONE."
880 DEF FNA(Z)=38*(SIN(Z/24)+.48*SIN(3*Z/24))+20
895 FORX=-100TO0STEP1
900 K=6:L=0:P=1:Z1=0
910 Y1=K*INT(SQR(10000-X*X)/K)
920 FORY=Y1TO-Y1STEP-K
930 Z=INT(80+FNA(SQR(X*X+Y*Y))-.707106*Y)
940 IFZ<LTHEN980
950 M=1:L=Z
960 PLOT160+M*X,Z,0,1
970 IFM=1THENM=-1:GOTO960
980 NEXTY,X
990 GOSUB1040
999 PRINT"■":CLC:NORM:END
1000 FORI=1TO5
1010 DRAW0,0,319,0,0,3
1015 DRAW319,0,319,199,0,3
1016 DRAW319,199,0,199,0,3
1017 DRAW0,199,0,0,0,3
1020 DRAW0,0,319,0,1,2
1021 DRAW319,0,319,199,1,2
1022 DRAW319,199,0,199,1,2
1023 DRAW0,199,0,0,1,2
1030 NEXT:RETURN
1040 FORI=1TO5
1050 DRAW0,0,319,0,0,1
1055 DRAW319,0,319,199,0,1
1056 DRAW319,199,0,199,0,1
1057 DRAW0,199,0,0,0,1
1060 DRAW0,0,319,0,0,0
1061 DRAW319,0,319,199,0,0
1062 DRAW319,199,0,199,0,0
1063 DRAW0,199,0,0,0,0
1070 NEXT:RETURN
```

Program 6. Demonstration program to show the use of commands within the graphics extension to Basic package.

Chapter Three
The Theory of High Resolution Graphics Displays

3.1 Point plotting and coordinates

Fundamental to any high resolution display is the ability to plot single pixel points at any desired location on the screen. On the CBM 64 the X coordinate has a range from Ø–319 addressable points and in the Y coordinates Ø–199 addressable points. The origin of the X and Y coordinates (X=Ø and Y=Ø) is located in the bottom left corner of the screen shown in Fig. 3.1, in all the examples in this book, and is standard for most graphics display programs. The actual mechanics of plotting a point on the screen are given in detail in many introductory texts, but the short Basic program (Program 7) shows the technique for point plotting. All the programs in this book use the PLOT command from the graphics Basic expansion package included in the previous chapter.

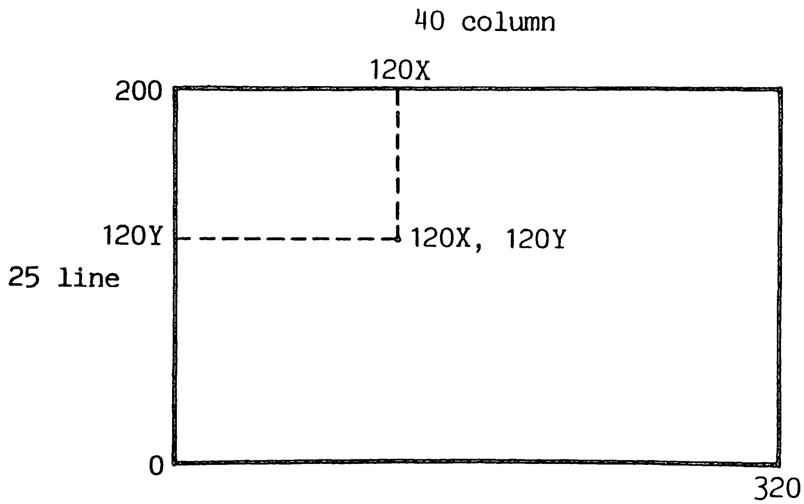


Fig. 3.1. Coordinate plotting on the CBM 64.

```
10 REM************************
20 REM  PLOT POINT
30 REM************************
40 REM SET AND CLEAR HI-RES SCREEN
50 PRINT"";
60 :
70 REM  SET DOT DATA ADDRESS
80 POKE53272,29
90 REM  SET BIT MAP MODE
100 POKE53265,59
110 :
120 REM BACKGROUND COLOUR 12
130 FORI=0TO1000
140 POKE1024+I,12
150 NEXT
160 :
170 DIMP2(7):REM TABLE POWERS OF 2
180 FORI=0TO7
190 P2(7-I)=2↑I
200 NEXT
210 :
220 REM CLEAR HI-RES SCREEN
230 FORI=0TO8000
240 POKEI+8192,0
250 NEXT
260 :
270 :
280 REM PLOT A SIN WAVE IN COLOUR 1 ON 12
290 Y=100:C=12+16*1
292 :
300 FORX=0TO319
310 Y=70*SIN(X/20)+100
320 GOSUB440 PLOT X,Y
330 NEXT
340 :
350 :
360 GETA$:IFA$=""THEN360 WAIT FOR KEY
370 REM RESTORE TEXT SCREEN
380 POKE53272,21
390 POKE53265,27
400 PRINT"";
410 END
420 .
430 .
440 REM **** PLOT ROUTINE  X,Y, COLOUR C
450 REM CALC PIXEL ADDRESS
460 A=8192+(XAND-8)+(YAND-8)*40+(YAND7)
470 :
475 REM PUT IN PIXEL
480 POKEA,PEEK(A)ORP2(XAND7)
481 :
485 REM CALC COLOUR ADDRESS
490 CA=1024+INT(X/8)+(YAND-8)*5
492 :
493 REM PUT IN COLOUR
495 POKECA,C
500 RETURN
510 .
520 NOTE:'X AND -8' IS THE SAME AS
530 :    'INT(X/8)*8'
```

Program 7.

3.2 Line plotting

Of almost equal importance to point plotting in any graphics application is line plotting. Lines obviously have to be built up from dots and there are several different algorithms for determining the position of each dot on the line. These routines have to ensure that the resulting line is straight, terminates accurately, is of a constant density with consistent spacing of dots along the length of the line. The problems involved in line drawing are best shown in Fig. 3.2.

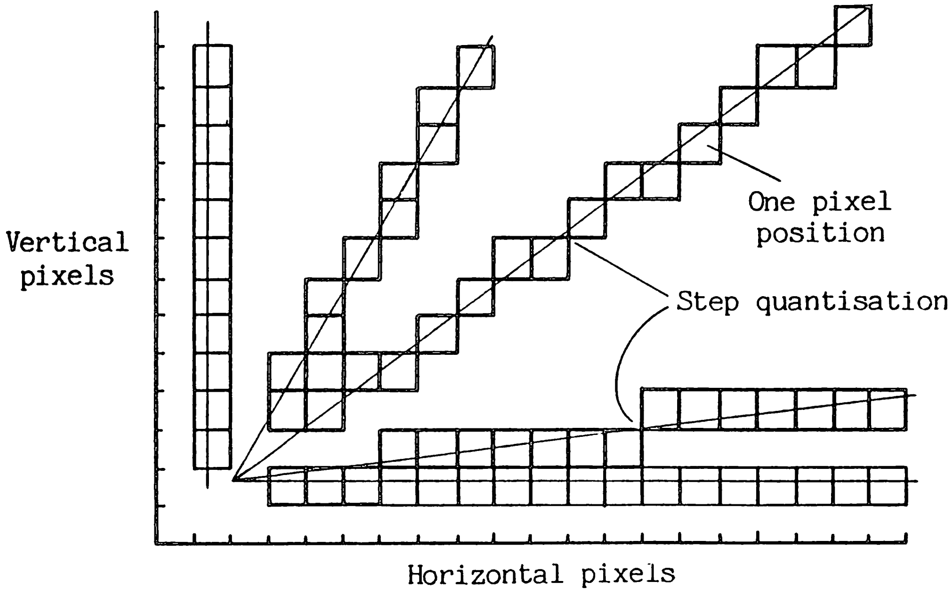


Fig. 3.2. Step quantisation in Basic line drawing.

The equation for a line is fairly straightforward and can be represented in the following form:

Y = M * L + B

where:

L = Y1 − YØ (XØ,YØ are the start coordinates of the line)
M = X1 − XØ (X1,Y1 are the end coordinates of the line)

and

B = YØ − (XØ * M)

One of the best and simplest routines for drawing a line uses what is known as a 'digital differential analyser' or DDA algorithm. This algorithm works on the basis of simultaneously incrementing the X and Y coordinates by small steps which are proportional to the slope of the line. The DDA algorithm is best explained by examining the Basic program, Program 8.

It should be noted that this method is an approximation and because of the fixed dot position and consequential necessity of round results, X and Y

```
10 REM*************************
20 REM  PLOT LINE
30 REM*************************
40 REM SET AND CLEAR HI-RES SCREEN
50 PRINT"";
60 :
70 REM  SET DOT DATA ADDRESS
80 POKE53272,29
90 REM  SET BIT MAP.MODE
100 POKE53265,59
110 :
120 REM BACKGROUND COLOUR 12
130 FORI=0TO1000
140 POKE1024+I,12
150 NEXT
160 :
170 REM CLEAR HI-RES SCREEN
180 REM THE FAST WAY
190 POKE828,PEEK(55):POKE829,PEEK(56)
200 POKE55,64:POKE56,63:REM TOP MEM
210 CLR
220 FORI=1TO250:A$=A$+CHR$(0):NEXT
230 POKE55,PEEK(828):POKE56,PEEK(829)
240 CLR
250 :
260 DIMP2(7):REM TABLE POWERS OF 2
270 FORI=0TO7
280 P2(7-I)=2↑I
290 NEXT
300 :
310 REM DRAW TO A SIN WAVE  COLOUR 1 ON 12
320 C=12+16*1
330 :
335 X1=160:Y1=50
340 FORX2=0TO319STEP 5
350 Y2=70*SIN(X2/20)+100
360 GOSUB1000 DRAW X1,Y1 TO X2,Y2
370 NEXT
380 :
390 :
400 GETA$:IFA$=""THEN400 WAIT FOR KEY
410 REM RESTORE TEXT SCREEN
420 POKE53272,21
430 POKE53265,27
440 PRINT"";
450 END
460 .
470 .
480 REM **** PLOT ROUTINE  X,Y, COLOUR C
490 REM CALC PIXEL ADDRESS
500 A=8192+(XAND-8)+(YAND-8)*40+(YAND7)
510 :
520 REM PUT IN PIXEL
530 POKEA,PEEK(A)ORP2(XAND7)
540 :
550 REM CALC COLOUR ADDRESS
560 CA=1024+INT(X/8)+(YAND-8)*5
570 :
580 REM PUT IN COLOUR
590 POKECA,C
600 RETURN
610 .
620 NOTE:'X AND -8' IS THE SAME AS
630 :     'INT(X/8)*8'
640 .
1000 REM DRAW LINE X1,Y1 TO X2,Y2
1010 REM COLOUR C
1020 L1=ABS(X1-X2)
1030 L2=ABS(Y1-Y2)
```

```
1040 IFL2>L1THEN L1=L2
1050 XI=(X2-X1)/L1
1060 YI=(Y2-Y1)/L1
1070 X=X1+.5
1080 Y=Y1+.5
1090 FORL=1TOL1
1100 GOSUB460  PLOT X,Y
1110 X=X+XI
1120 Y=Y+YI
1130 NEXT
1140 RETURN
```

Program 8.

coordinates can give rise to some inaccuracies on line termination. Otherwise this DDA algorithm is both fast and easy to use. All the programs in this book use the DRAW command from the graphics Basic expansion package included in the previous chapter.

3.3 Circle plotting

In many applications it is advantageous to be able to plot circles, arcs and ellipses. The general equation for generating a circle is quite simple:

$$X = R * SIN(A)$$
$$Y = R * COS(A)$$

where A is the angle around the centre of the circle. The angle increment for each successive plotting of X and Y is determined by the radius of the circle and the desired dot spacing. R is the radius of the circle. Program 9 uses this equation to draw a circle.

```
5 REM BASIC CIRCLE
10 INPUT"RADIUS "; RA
20 HIRES0,1
25 ORIGIN160,100
100 FORI=0TOπ*2 STEP 2/RA/π
150 PLOTSIN(I)*RA,COS(I)*RA,0,1
180 NEXT
185 GETA$:IFA$=""THEN185 PAUSE FOR KEY PRESS
190 NORM
```

Program 9.

The problem with this routine is that it is fairly slow due to the number of trigonometric calculations which must be performed. A much faster method which does not require this is the 'circle digital differential analyser' algorithm. The following Basic program, Program 1Ø, shows how this algorithm works.

```
5 REM BASIC CIRCLE
10 INPUT"RADIUS "; RA
20 HIRES0,1
25 ORIGIN160,100
30 R=RA
33 A=0
100 FORI=0TORA*π*2
150 PLOTA,R,0,1
160 A=A-R/RA
```

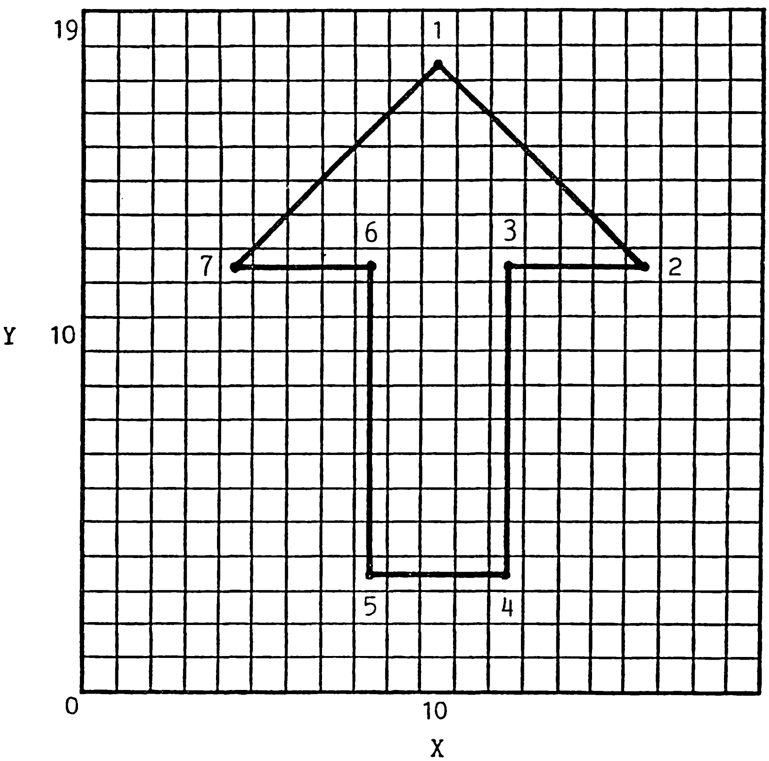
```
170 R=R+A/RA
180 NEXT
185 GETA$:IFA$=""THEN185 PAUSE FOR KEY PRESS
190 NORM
```

Program 10.

All the programs in this book use the CIRCLE command from the graphics Basic expansion package included in the previous chapter.

3.4 Two dimensional shapes

Two dimensional shapes can be divided into two categories: regular polygons whose shape can be calculated from a given formula, and irregular shapes which can be constructed only from a table of data points. Irregular shapes can be further divided into closed fully joined shapes and open disjoined shapes. The calculation of regular polygons can be useful but often has severe limitations when constructing complex displays made up from many separate two dimensional shapes. The best method is to use data tables for all shapes. If regular polygons are to be displayed then it is best first to create the data table using a calculation and then display the figure, thereby allowing greater flexibility in the subsequent manipulation of the figure. The construction of a shape using a data table is best shown in Fig. 3.3.



	X	Y
1	10	18
2	16	12
3	12	12
4	12	3
5	8	3
6	8	12
7	4	12

Fig. 3.3. Two dimensional shape and corresponding data table.

Program 11 is an example of such a program to create regular polygons, store the calculated end of line coordinates in a data table and then display the resulting shape. This program includes the routine to draw a shape from a data table of beginning and end of line coordinates.

```
10 REM ********************
20 REM        POLYGON
30 REM ********************
40 P=0:Q=1000
45 DIM XS(Q-1),YS(Q-1),XE(Q-1),YE(Q-1)
50 PRINT"□POLYGON OF N SIDES"
60 PRINT:PRINT
70 INPUT"N (LESS THAN 3 TO END INPUT)";N
75 IFN<3THEN1000 END INPUT & START PLOT
80 S=π/N
90 INPUT"LENGTH OF SIDE";L
100 INPUT"CENTRE POSITION X,Y";X,Y
110 REM CALC DISTANCE OF CORNERS FROM CENTRE
120 R=L/SIN(S)/2
140 FORI=PTON-1+P
150 XS(I)=SIN(S*2*I)*R+X
160 YS(I)=COS(S*2*I)*R+Y
170 XE(I)=SIN(S*2*(I+1))*R+X
180 YE(I)=COS(S*2*(I+1))*R+Y
190 NEXT
200 P=P+N
210 GOTO 50 RESTART INPUT LOOP
220 :
1000 REM ******************
1010 REM DRAW DATA IN ARRAYS
1020 REM ******************
1030 HIRES 0,1
1040 C=0
1050 FORI=0TOP-1
1070 DRAWXS(I),YS(I),XE(I),YE(I),C,1
1080 NEXT
1090 GETA$:IFA$=""THEN1090
1100 NORM
```

Program 11.

3.5 Translation of a two dimensional shape

Translation of a shape means simply moving the shape. There are two ways in which a two dimensional shape can be moved within the screen plane: translation and rotation. Translation implies that the shape is moved but its angular orientation with respect to the screen coordinates remains constant.

The translation of a shape is very simple. The initial shape coordinate data table is defined at a set position on the screen. To move the shape to the required position on the screen one calculates the X and Y coordinate offsets between the desired position and the position within the data table. These offsets are then added to every point within the data table and the shape is displayed at the new position. Determining the offset coordinate values can cause a problem, namely which coordinate from the data table to measure from. The solution is to calculate the centre of the shape and use this centre point to calculate the offset. To find the coordinates of the centre of the shape add all the X coordinate values together then divide by the number of coordinate values; this gives the centre X coordinate. Repeat this for the Y coordinates to get the centre Y coordinate.

Program 12 takes a shape stored in a data table, finds its centre and then displays the shape at any desired position on the screen.

```
5 REM SHAPE TRANSLATION
10 P=14
20 DIMXS(P),YS(P),XE(P),YE(P)
30 FORI=0TOP
40 READXS(I),YS(I),XE(I),YE(I)
50 NEXT
60 HIRES0,12
70 FORJ=0TO40
80 C=(J AND3)+1
85 B=JAND3
90 XT=RND(1)*310
100 YT=RND(1)*190
110 GOSUB1000 DRAW IT AT XT,YT
120 NEXT
130 GETA$:IFA$=""THEN130
140 NORM
150 END
1000 REM ********************
1010 REM DRAW DATA IN ARRAYS
1015 REM TRANSLATED BY XT & YT
1020 REM ********************
1050 FORI=0TOP
1070 DRAWXS(I)+XT,YS(I)+YT,XE(I)+XT,YE(I)+YT,C,1
1080 NEXT
1090 RETURN
10000 DATA0,11,0,-6,     0,11,5,13
10010 DATA5,13,0,16,     0,16,-5,13
10020 DATA-5,13,0,11,    0,5,-5,0
10030 DATA-5,0,-7,8,     0,5,7,5
10040 DATA7,5,5,-2,      0,-6,5,-10
10050 DATA-5,-11,-1,-12,-1,-12,-3,-14
10060 DATA5,-10,0,-14,   0,-14,3,-15
10070 DATA 0,-6,-5,-11
```

Program 12.

3.6 Rotation of a two dimensional shape

Rotation involves changing the angular orientation of the shape with respect to the screen coordinates and a rotational centre. To understand the rotation of a shape it is first necessary to understand the rotation of a point, as shown in Fig. 3.4.

The centre of rotation is very important in rotation since this is the only fixed reference point for calculating the rotational position of each coordinate. When rotating a point about this fixed rotational centre by an angle T, the following equations give the new coordinate offsets which, when added to the rotational centre coordinates, will give the new coordinates for the rotated point:

$$X2 = X1 * \cos T + Y1 * (-\sin T)$$
$$Y2 = X1 * \sin T + Y1 * \cos T$$

A shape can be rotated by performing this calculation on every corner

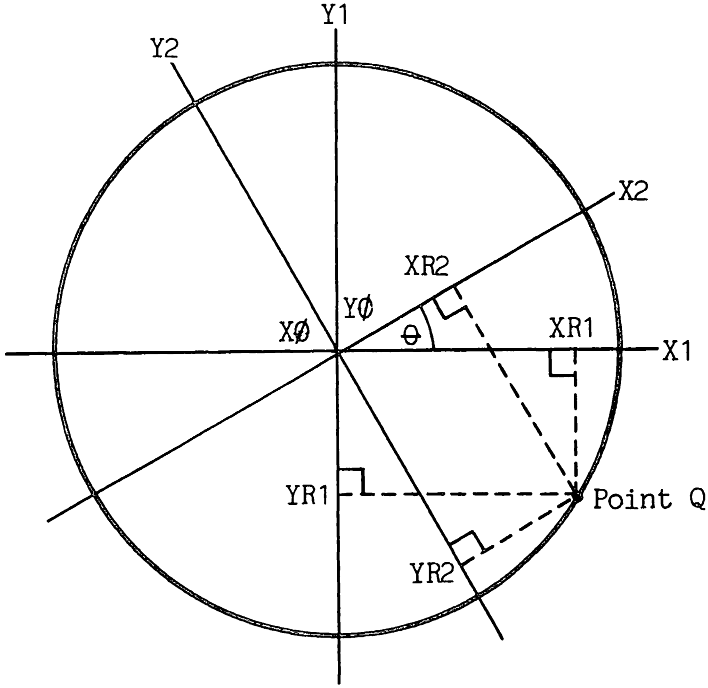


Fig. 3.4. Rotational transformation of a point Q by an angle of θ degrees from the initial coordinates on X1, Y1 to the new coordinates on X2, Y2. Note that the centre of rotation at XΦ, YΦ stays constant.

coordinate in the shape data table prior to drawing the shape. Program 13 shows how a simple shape can be rotated.

```
5 REM SHAPE ROTATION
10 P=28
20 DIMX(P+1),Y(P+1),XP(P+1),YP(P+1)
30 FORI=0TOP+1
40 READX(I),Y(I)
50 NEXT
55 INPUT"ANGLE (DEGREES) ";T
56 T=T*π/180
57 :
60 FORI=0TOP+1
70 XP(I)=X(I)*COS(T)-Y(I)*SIN(T)
80 YP(I)=X(I)*SIN(T)+Y(I)*COS(T)
90 NEXT
95 :
100 HIRES0,1:ORIGIN160,100
110 GOSUB1000 DRAW IT
130 GETA$:IFA$=""THEN130
140 NORM
150 END
900 .
1000 REM ********************
1010 REM DRAW DATA IN ARRAYS
1015 REM ROTATED T
1020 REM *********************
1050 FORI=0TOPSTEP2
1070 DRAWXP(I),YP(I),XP(I+1),YP(I+1),C,1
1080 NEXT
1090 RETURN
10000 DATA0,11,0,-6,      0,11,5,13
```

```
10010 DATA5,13,0,16,     0,16,-5,13
10020 DATA-5,13,0,11,    0,5,-5,0
10030 DATA-5,0,-7,8,     0,5,7,5
10040 DATA7,5,5,-2,      0,-6,5,-10
10050 DATA-5,-11,-1,-12,-1,-12,-3,-14
10060 DATA5,-10,0,-14,   0,-14,3,-15
10070 DATA 0,-6,-5,-11
```

Program 13.

3.7 Scaling and stretching a two dimensional shape

Scaling involves changing the size of a shape with respect to its original size in the data table with all the dimensions being kept proportional to the original shape. Stretching also involves changing the size of the shape but only in one given direction, and is a special form of scaling.

Scaling involves simply multiplying all the line lengths which constitute the shape by a given scaling factor. To make the shape larger the scaling factor is larger than one and to make it smaller the value is less than one. In practice the problem is slightly more complex since the shape data table consists of a table of beginning and end of line coordinates, and simply changing line length will leave a sequence of disjointed lines either shorter or longer than the original but in the same position. Thus in addition to changing line lengths one must also change the position of the line relative to the centre of the shape. The centre of the shape is determined by adding all the X coordinates and then dividing by the number of coordinate points. This gives the X coordinate of the shape centre and works similarly for the Y coordinate. The scaling is then done by multiplying all the coordinates relative to the centre of the shape by the scaling factor as shown in Program 14.

```
5 REM SHAPE SCALING
10 P=28
20 DIMX(P+1),Y(P+1)
30 FORI=0TOP+1
40 READX(I),Y(I)
50 NEXT
55 INPUT"X AND Y SCALING ";X,Y
57 :
60 FORI=0TOP+1
70 X(I)=X(I)*X
80 Y(I)=Y(I)*Y
90 NEXT
95 :
100 HIRES0,1:ORIGIN160,100
110 GOSUB1000 DRAW IT
130 GETA$:IFA$=""THEN130
140 NORM
150 END
900 .
1000 REM *********************
1010 REM DRAW DATA IN ARRAYS
1015 REM SCALED BY X AND Y
1020 REM *********************
1050 FORI=0TOPSTEP2
1070 DRAWX(I),Y(I),X(I+1),Y(I+1),C,1
1080 NEXT
1090 RETURN
```

```
10000 DATA0,11,0,-6,     0,11,5,13
10010 DATA5,13,0,16,     0,16,-5,13
10020 DATA-5,13,0,11,    0,5,-5,0
10030 DATA-5,0,-7,3,     0,5,7,5
10040 DATA7,5,5,-2,      0,-6,5,-10
10050 DATA-5,-11,-1,-12,-1,-12,-3,-14
10060 DATA5,-10,0,-14,   0,-14,3,-15
10070 DATA 0,-6,-5,-11
```

Program 14.

Stretching an object in either the X or Y axis is simple; just use a different scaling factor for X and Y in Program 14. This kind of stretching is not often required, however. It is more common to want to stretch the shape along any

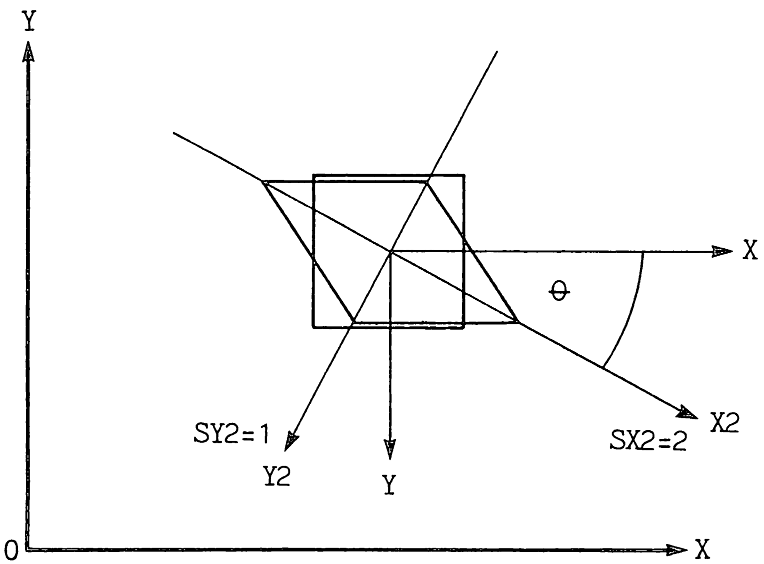


Fig. 3.5. Two dimensional stretching at an angle of θ degrees with no stretching in the Y direction and two times stretching in the X direction.

given angle as shown in Fig. 3.5. For this we require the stretch angle and a scaling factor in order to calculate the relative stretching in the X and Y axes for all coordinates of the shape. This requires some fairly elaborate trigonometric calculations which are best demonstrated in Program 15.

```
5 REM SHAPE STRETCHING
10 P=28
20 DIMX(P+1),Y(P+1),XP(P+1),YP(P+1)
30 FORI=0TOP+1
40 READX(I),Y(I)
50 NEXT
55 INPUT"ANGLE (DEGREES) ";T
56 INPUT"STRETCHING FACTORS ";U,V
58 T=T*π/180
59 :
60 SI=SIN(T):CO=COS(T)
65 FORI=0TOP+1
70 X=(X(I)*CO-Y(I)*SI)*U
75 Y=(X(I)*SI+Y(I)*CO)*V
```

```
80 XP(I)=X*CO+Y*SI
85 YP(I)=-X*SI+Y*CO
90 NEXT
95 :
100 HIRES0,1:ORIGIN160,100
110 GOSUB1000 DRAW IT
130 GETA$:IFA$=""THEN130
140 NORM
150 END
900 .
1000 REM ********************
1010 REM DRAW DATA IN ARRAYS
1015 REM STRETCHED AT ANGLE T
1020 REM ********************
1050 FORI=0TOPSTEP2
1070 DRAWXP(I),YP(I),XP(I+1),YP(I+1),C,1
1080 NEXT
1090 RETURN
10000 DATA0,11,0,-6,     0,11,5,13
10010 DATA5,13,0,16,     0,16,-5,13
10020 DATA-5,13,0,11,    0,5,-5,0
10030 DATA-5,0,-7,8,     0,5,7,5
10040 DATA7,5,5,-2,      0,-6,5,-10
10050 DATA-5,-11,-1,-12,-1,-12,-3,-14
10060 DATA5,-10,0,-14,   0,-14,3,-15
10070 DATA 0,-6,-5,-11
```

Program 15.

3.8 Combining transformations using matrices

The previous sections of this chapter have dealt with the routines which can be used to translate, scale and rotate a two dimensional shape. It is, however, frequently desirable to be able to perform more than one of these functions on a shape at any one time. Successive translations, rotations or scalings would be slow and cumbersome in terms of program length, but this can be overcome by combining the three mathematical operations into a single operation capable of performing all three functions. This can be done using matrix arithmetic. Each of the three calculations can be represented as a 3 by 3 matrix, thus:

Translation: $[X' \; Y' \; 1] = [X \; Y \; 1] \begin{matrix} 1 & \emptyset & \emptyset \\ \emptyset & 1 & \emptyset \\ TX & TY & 1 \end{matrix}$

Rotation: $[X' \; Y' \; 1] = [X \; Y \; 1] \begin{matrix} \cos\theta & -\sin\theta & \emptyset \\ \sin\theta & \cos\theta & \emptyset \\ \emptyset & \emptyset & 1 \end{matrix}$

Scaling: $[X' \; Y' \; 1] = [X \; Y \; 1] \begin{matrix} SX & \emptyset & \emptyset \\ \emptyset & SY & \emptyset \\ \emptyset & \emptyset & 1 \end{matrix}$

These matrices can be combined to give a single 3 by 3 matrix using a process known as matrix concatenation. This gives the following matrix:

$$[X'\ Y'\ 1] = [X\ Y\ 1] \begin{matrix} SX * \cos\theta & SX * \sin\theta & \emptyset \\ SY * -\sin\theta & SY * \cos\theta & \emptyset \\ TX & TY & 1 \end{matrix}$$

An understanding of the maths required is not essential but for those interested the study of a good maths textbook covering matrix arithmetic could be rewarding. To make the use of matrix arithmetic easier the graphics Basic expansion package includes a set of matrix commands which greatly speed up the required calculations. Program 16 uses matrix arithmetic and the MAT command of the graphics Basic expansion package to scale, rotate or translate any two dimensional shape contained within the data table of the program.

```
5 REM COMBINED TRANSFORMATION
10 P=28
20 DIMM1(P+1,2),M2Y(P+1,2),A(2,2)
30 FORI=0TOP+1
40 READM1(I,0),M1(I,1):M1(I,2)=1
50 NEXT
55 T=5
56 SX=1.07:SY=1
57 TX=2:TY=1
58 T=T*π/180
59 :
60 A(0,0)=SX*COS(T)
70 A(0,1)=SX*SIN(T)
80 A(0,2)=0
90 A(1,0)=-SY*SIN(T)
100 A(1,1)=SY*COS(T)
110 A(1,2)=0
120 A(2,0)=TX
130 A(2,1)=TY
140 A(2,2)=1
150 HIRES0,1:C=0:ORIGIN160,100
900 MAT M2=M1*A :MAT M1=M2
910 GOSUB1000 DRAW IT
930 GETA$:IFA$<>" "THEN900
940 NORM
950 END
960 .
1000 REM ********************
1010 REM DRAW DATA IN ARRAYS
1015 REM ROTATED BY ANGLE T
1017 REM MOVED BY TX , TY
1018 REM SCALED BY SX , SY
1020 REM ********************
1050 FORI=0TOPSTEP2
1070 DRAWM2(I,0),M2(I,1),M2(I+1,0),M2(I+1,1),C,1
1080 NEXT
1090 RETURN
10000 DATA0,11,0,-6,     0,11,5,13
10010 DATA5,13,0,16,     0,16,-5,13
10020 DATA-5,13,0,11,    0,5,-5,0
10030 DATA-5,0,-7,8,     0,5,7,5
10040 DATA7,5,5,-2,      0,-6,5,-10
10050 DATA-5,-11,-1,-12,-1,-12,-3,-14
10060 DATA5,-10,0,-14,   0,-14,3,-15
10070 DATA 0,-6,-5,-11
```

Program 16.

3.9 Three dimensional shapes

The displaying of two dimensional shapes on the computer screen is adequate for many applications; however, there is often no substitute for the added realism of a three dimensional display. It is not difficult to display three dimensional graphics representations of objects, the formula required being simply an extension of those already developed for two dimensional displays but with the addition of an extra coordinate.

Of course it is not possible to really display an object in three dimensions on the screen. All that can be done is to use certain rules to display a two dimensional representation of a three dimensional object. To give this two dimensional display the realism which will lead the human eye into believing that it is actually seeing a three dimensional object presents many problems, only some of which are technically within the capabilities of the 64 processor and display hardware. These techniques involve the problems of depth cueing, perspective, surface shading and lighting, hidden surface elimination and many others.

3.10 Simple three dimensional shape display

In the simplest manner a three dimensional shape can be displayed as a wire frame model. This means that the object is defined as a series of corner coordinates joined by straight lines – the wire framework, as shown in Fig. 3.6.

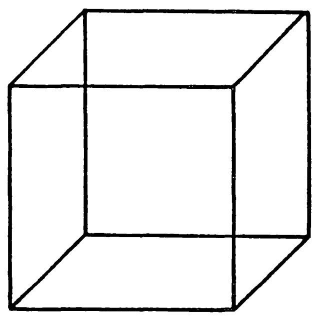


Fig. 3.6. Three dimensional wire frame image of a cube. (*Note:* The image reversal illusion is due to the eye's inability to determine the depth of lines.)

The kind of representation of a shape shown in Fig. 3.7 requires the viewer to imagine the surfaces of the object. This can be difficult; because there are no solid faces the viewer can see both the back and front of the object, and with many shapes this causes the viewer to have problems determining which side of the object he is viewing. This can be overcome by the use of depth cueing, perspective and hidden line removal, all of which will be looked at later.

Fundamental to the display of any three dimensional object is the initial setting up of a system of world coordinates within which the object or objects can be defined, as well as establishing the position of the viewer with respect to the

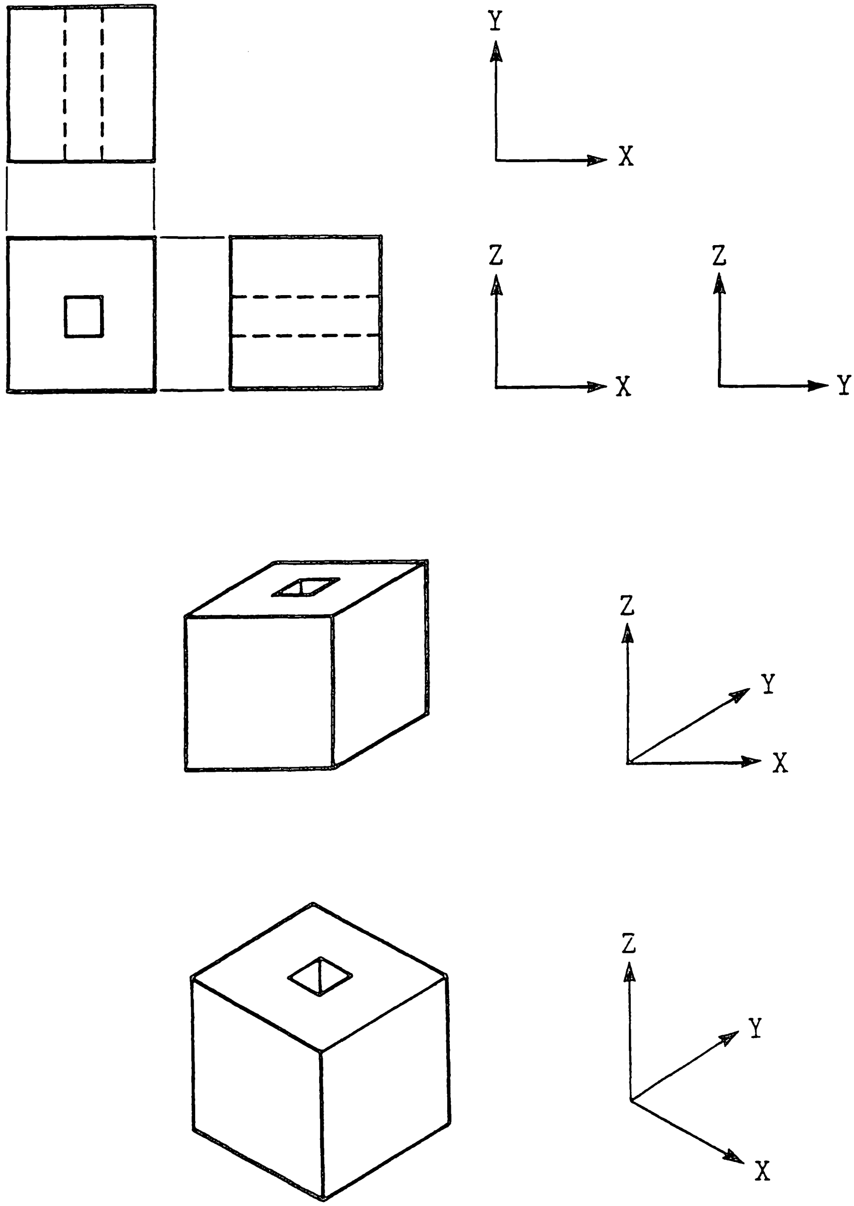


Fig. 3.7. Different methods of three dimensional image representation.

objects. We will use a three dimensional Cartesian coordinate system with the X as the horizontal axis, the Y as the vertical axis and the third Z axis at right angles to the other two axes, as shown in Fig. 3.8.

The individual objects and the topographical relationship between objects can be defined within this coordinate system using predefined units of measurement. The measurement units selected depend on the amount of detail within the image; the greater the detail the smaller the measurement unit and the degree of magnification which may be required.

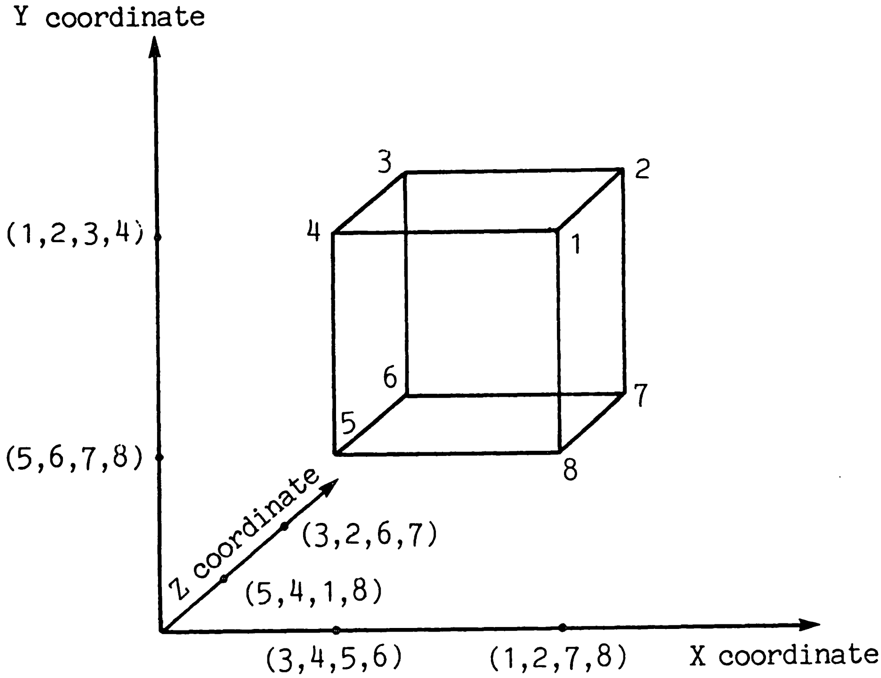


Fig. 3.8. Three dimensional coordinate system.

The point from which the three dimensional image is seen is very important. It is specified as the view point, viewing direction and aperture. The *aperture* would normally be the screen area and the *view point* the distance of the eye from the screen and the distance of the screen from the object. The view point is defined as a point within the three dimensional coordinate system, and its position in front of the screen aperture would normally be regarded as a constant. The *viewing direction* determines what part of the three dimensional world is being viewed with the eye at the defined view point. The *viewing angle* is defined as three sets of angles, one within each coordinate plane. The relationship between aperture, viewing angle and view point is shown in Fig. 3.9. Thus when we rotate or translate a three dimensional image it is the viewing position and angle which change rather than the object.

It is best to consider a three dimensional object as either a simple polyhedron or a collection of polyhedrons. The reason for this is that a simple polyhedron is a complete enclosed solid and therefore easily represented using standard three dimensional display mathematics. Any complex shape can be constructed using several polyhedrons and the repeated use of similar polyhedrons can give considerable savings in the amount of data required to store an object. A wire frame model of a polyhedron can easily be constructed given information on its vertices, edges and faces. For a simple wire frame model only a knowledge of the vertices and edges is necessary, but if hidden line or surface elimination is employed then a knowledge of the faces is essential.

The data for each of the vertices (end of line or corner coordinates) is quite simple and consists of the X,Y and Z coordinates of the point within the

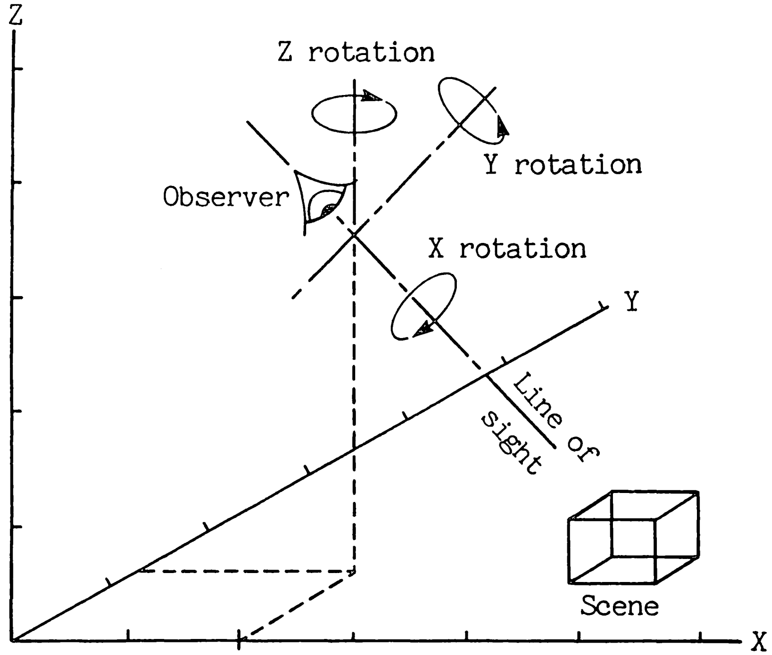


Fig. 3.9. Image viewing coordinates.

coordinate system. A table of these coordinates is the first and most important of the data tables used to define an object. The order in which the vertices are stored in the table is not very important. The second data table contains the edge information and shows which vertices within the first table are joined by lines. The third and last table contains information on the faces of the object. Unlike lines and points, a face has both a front and back and it is essential to be able to determine which side of a face is being viewed, particularly if we are to employ a hidden face removal algorithm. Whereas the order in which data on the vertices and edges is stored in the respective tables is unimportant, the order of the vertices describing a face is important. The standard convention is to list the vertices of a face in an anti-clockwise direction when viewing the face from outside the polyhedron. A face viewed from inside the polyhedron will therefore have a clockwise order of vertices, thereby making the determination of which side of a face is being viewed fairly easy. Figure 3.8 shows a cube and the following data tables describe it:

Table of vertices

	X	Y	Z		X	Y	Z
V1	2	2	1	V5	2	2	2
V2	2	1	1	V6	2	1	2
V3	1	1	1	V7	1	1	2
V4	1	2	1	V8	1	2	2

Table of edges	
E1	V1-V2
E2	V2-V3
E3	V3-V4
E4	V4-V1
E5	V1-V5
E6	V4-V8
E7	V3-V7
E8	V2-V6
E9	V5-V6
E1Ø	V6-V7
E11	V7-V8
E12	V8-V5

Table of faces	
F1	V1,V4,V3,V2
F2	V1,V5,V8,V4
F3	V6,V2,V3,V7
F4	V5,V6,V7,V8
F5	V1,V2,V6,V5
F6	V3,V4,V8,V7

3.11 The mathematics of three dimensional displays

Just as the formulae for two dimensional displays are best performed in matrix format the same applies to three dimensional displays, the only difference being that 4 by 4 matrices are used. The techniques are simply extensions of those applied to two dimensional shapes, and as with the two dimensional formulae the separate function matrices can be concatenated to give a single matrix which combines all functions. We will first look at the basic formulae.

3.11.1 Translation

The transformation matrix to translate a point at coordinate X,Y,Z within a three dimensional image space to a new point X',Y',Z' is as follows:

$$[X',Y',Z',1]=[X,Y,Z,1] \begin{matrix} 1 & Ø & Ø & Ø \\ Ø & 1 & Ø & Ø \\ Ø & Ø & 1 & Ø \\ TX & TY & TZ & 1 \end{matrix}$$

TX,TY, and TZ are the translation components in the X,Y and Z components respectively.

3.11.2 Scaling

The following scaling transformation matrix will scale the dimensions in each coordinate direction separately depending on the respective scaling factors, SX,SY and SZ. The matrix is as follows:

$$[X',Y',Z',1]=[X,Y,Z,1]\begin{matrix} SX & \emptyset & \emptyset & \emptyset \\ \emptyset & SY & \emptyset & \emptyset \\ \emptyset & \emptyset & SZ & \emptyset \\ \emptyset & \emptyset & \emptyset & 1 \end{matrix}$$

3.11.3 Rotation

Whereas the process of either translation or scaling is a simple extension of the two dimensional equivalent, the process of rotation is far more complex. In order to perform a rotation we must first determine a three dimensional axis about which to rotate. To perform the rotation we need three operations; translate the point to the origin, perform the rotation around the appropriate axis, and then translate back. Each axis has its own rotation transformation matrix. The three axes are shown diagramatically in Fig. 3.9.

To rotate a point about the Z coordinate axis by an angle θ requires the following transformation matrix:

$$[X'Y'Z'1]=[XYZ1]\begin{matrix} \cos\theta & -\sin\theta & \emptyset & \emptyset \\ \sin\theta & \cos\theta & \emptyset & \emptyset \\ \emptyset & \emptyset & 1 & \emptyset \\ \emptyset & \emptyset & \emptyset & 1 \end{matrix}$$

It should be noted that the angle θ is measured in a clockwise direction about the origin when viewing the origin from a point on the +Z axis.

To rotate a point about the Y axis use the following matrix:

$$[X'Y'Z'1]=[XYZ1]\begin{matrix} \cos\theta & \emptyset & -\sin\theta & \emptyset \\ \emptyset & 1 & \emptyset & \emptyset \\ \sin\theta & \emptyset & \cos\theta & \emptyset \\ \emptyset & \emptyset & \emptyset & 1 \end{matrix}$$

To rotate a point about the X axis use the following matrix:

$$[X'Y'Z'1]=[XYZ1]\begin{matrix} 1 & \emptyset & \emptyset & \emptyset \\ \emptyset & \cos\theta & \sin\theta & \emptyset \\ \emptyset & -\sin\theta & \cos\theta & \emptyset \\ \emptyset & \emptyset & \emptyset & 1 \end{matrix}$$

3.11.4 Concatenation of three dimensional matrices

In applications where a number of different transformations are to be applied to an image, the individual transformation matrices can be combined to produce a single matrix by a process known as concatenation. An understanding of the maths involved in such a concatenation is not necessary. The following matrix is derived from a concatenation of all the three dimensional transformation matrices and thus enables anyone using it to rotate (in any of the three axes) translate and scale.

[C]=

SX*COS(Y)*COS(Z)	SX*COS(Y)*SIN(Z)	SX*−SIN(Y)	Ø
SY*COS(X)*(−SIN(Z)+SIN(X))*SIN(Y)*COS(Z)	SY*COS(X)*COS(Z)+SIN(X)*SIN(Y)*SIN(Z)	SY*SIN(X)*COS(Y)	Ø
SZ*(−SIN(X))*(−SIN(Z)+COS(X))*SIN(Y)*COS(Z))	SZ*(−SIN(X))*COS(Z)+COS(Z)*SIN(Y)*SIN(Z)	SZ*COS(X)*COS(Y)	Ø
TX	TY	TZ	1

Concatenation of all the three dimensional transformation matrices.

Program 17 illustrates this.

```
1 REM SIMPLE CUBE
5 READ N,LK
10 DIM M(2,2),MS(2,2),A(N,2),B(N,2),LK(LK,1)
20 HIRES1,0
22 ORIGIN160,100
23 SX=10:SY=10:SZ=10
24 AX=0.3:AY=0.39:AZ=0.30
25 GOSUB 300
30 GOSUB 200
35 MAT B=A*MS
36 MAT A=B
40 MAT B=A*M
50 GOSUB 100
98 GETA$:IFA$<>"←"THEN36
99 NORM:LIST
100 REM
103 CLG
107 FORI=0TOLK
108 L1=LK(I,0):L2=LK(I,1)
110 DRAW B(L1,0),B(L1,1),B(L2,0),B(L2,1),3,1
120 NEXT
130 RETURN
200 FORI=0TON
210 READ A(I,0),A(I,1),A(I,2)
220 NEXT
230 FORI=0TOLK
240 READLK(I,0),LK(I,1)
250 NEXT
260 RETURN
300 M(0,0)=COS(AY)*COS(AZ)
310 M(0,1)=COS(AY)*SIN(AZ)
320 M(0,2)=-SIN(AY)
340 M(1,0)=COS(AX)*-SIN(AZ)+SIN(AX)*SIN(AY)*COS(AZ)
350 M(1,1)= COS(AX)*COS(AZ)+SIN(AX)*SIN(AY)*SIN(AZ)
360 M(1,2)=SIN(AX)*COS(AY)
370 M(2,0)=SIN(AX)*SIN(AZ)+COS(AX)*SIN(AY)*COS(AZ)
380 M(2,1)=-SIN(AX)*COS(AZ)+COS(AZ)*SIN(AY)*SIN(AZ)
390 M(2,2)=COS(AX)*COS(AY)
400 MS(0,0)=SX
410 MS(1,1)=SY
420 MS(2,2)=SZ
430 RETURN
1000 DATA 7,11
1010 DATA-5,-5,-5, 5,-5,-5, 5,5,-5, -5,5,-5
1020 DATA-5,-5,5, 5,-5,5, 5,5,5, -5,5,5
1025 REM LINK
1030 DATA 0,1, 1,2, 2,3, 3,0, 0,4
1040 DATA 4,5, 5,6, 6,7, 7,4, 1,5
1050 DATA 2,6, 3,7
9999 REM ROUTINE TO CHECK ACCURACY OF MAT COMMAND
10000 FORI=0TO7:PRINTSQR(B(I,0)↑2+B(I,1)↑2+B(I,2)↑2):NEXT
```

Program 17.

3.12 Techniques to give realism to a three dimensional image

The human eye and mind can have difficulty in understanding a three dimensional wire frame image of any complexity. The basic problem is that projection of a three dimensional object on a two dimensional screen lacks any significant information about the depth of objects and surfaces. To overcome this we need to use techniques which will give visual depth cueing. These techniques work by giving the eye clues that certain lines and surfaces are in front of or behind others; the following is a list of a few of these techniques.

3.12.1 Perspective projection

This takes advantage of a phenomenon familiar to all; that objects of the same size appear to become smaller as they are positioned further and further away from the viewer. Perspective projection is an ideal depth cue when the image has a considerable amount of depth, but it can still give rise to ambiguity in objects which have only limited depth variation. This can be overcome to a degree by exaggerating the perspective effect in a similar manner to that perceived when viewing an object through a wide angle lens. Exaggerated perspective projection, however, can give rise to odd visual effects and should be avoided unless absolutely necessary. This is demonstrated in Program 18 and illustrated in Fig. 3.1Ø.

```
1 REM PERSPECTIVE CUBE
5 READ N,LK
10 DIM M(2,2),MS(2,2),A(N,2),B(N,2),LK(LK,1)
20 HIRES1,0
22 PORIGIN160,100,300
23 SX=10:SY=10:SZ=10
24 AX=0.07:AY=0.05:AZ=0.06
25 GOSUB 300
30 GOSUB 200
35 MAT B=A*MS
36 MAT A=B
40 MAT B=A*M
50 GOSUB 100
98 GETA$:IFA$<>"←"THEN36
99 NORM:LIST
100 CLG
101 FORI=0TOLK
102 L1=LK(I,0):L2=LK(I,1)
110 PDRAW B(L1,0),B(L1,1),B(L1,2),B(L2,0),B(L2,1),B(L2,2),3,1
120 NEXT
130 RETURN
200 FORI=0TON
210 READ A(I,0),A(I,1),A(I,2)
220 NEXT
230 FORI=0TOLK
240 READLK(I,0),LK(I,1)
250 NEXT
260 RETURN
300 M(0,0)=COS(AY)*COS(AZ)
310 M(0,1)=COS(AY)*SIN(AZ)
320 M(0,2)=-SIN(AY)
340 M(1,0)=COS(AX)*-SIN(AZ)+SIN(AX)*SIN(AY)*COS(AZ)
350 M(1,1)= COS(AX)*COS(AZ)+SIN(AX)*SIN(AY)*SIN(AZ)
360 M(1,2)=SIN(AX)*COS(AY)
370 M(2,0)=SIN(AX)*SIN(AZ)+COS(AX)*SIN(AY)*COS(AZ)
380 M(2,1)=-SIN(AX)*COS(AZ)+COS(AZ)*SIN(AY)*SIN(AZ)
390 M(2,2)=COS(AX)*COS(AY)
400 MS(0,0)=SX
410 MS(1,1)=SY
420 MS(2,2)=SZ
430 RETURN
1000 DATA 7,11
1010 DATA-5,-5,-5, 5,-5,-5, 5,5,-5, -5,5,-5
1020 DATA-5,-5,5, 5,-5,5, 5,5,5, -5,5,5
1025 REM LINK
```

```
1030 DATA 0,1, 1,2, 2,3, 3,0, 0,4
1040 DATA 4,5, 5,6, 6,7, 7,4, 1,5
1050 DATA 2,6, 3,7
10000 FORI=0TO7:PRINTSQR(B(I,0)↑2+B(I,1)↑2+B(I,2)↑2):NEXT
```

Program 18.

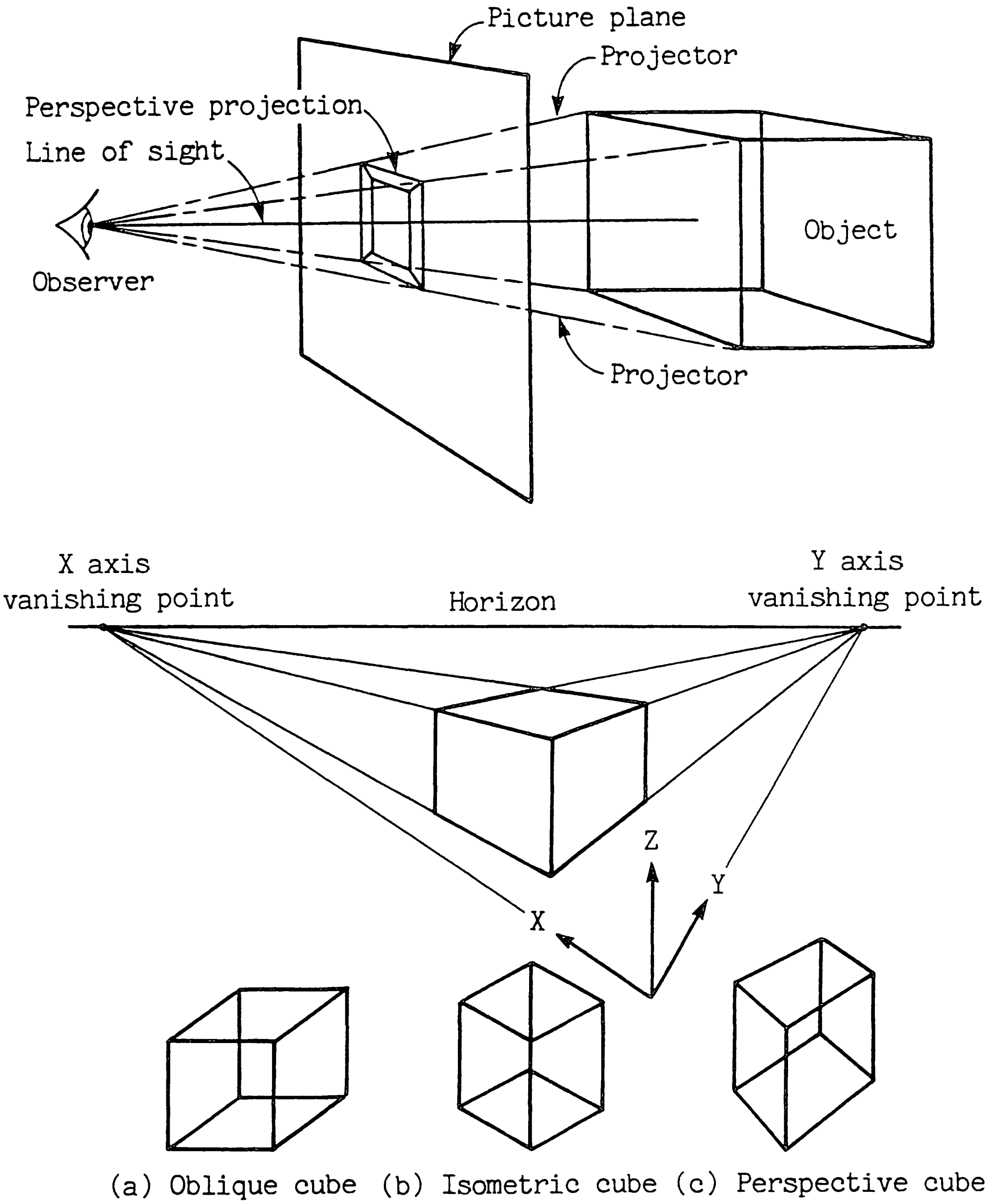


Fig. 3.10. Three dimensional perspective projection.

3.12.2 Line intensity

A variation of the perspective effect is line intensity variation. This simply means that lines which are close to the viewer are drawn thick and bold and lines in the distance are drawn thin and faint. This technique is really only effective on simple objects, and can be very confusing on large objects. It is, however, a useful technique when applied in conjunction with perspective projection. This technique is also constrained by the technical limitations of the CBM 64 raster scan display. This means that the plot resolution does not allow a very wide range of line intensities; 2 or 3 at most. In addition the extra plotting required can slow down the generation of a display considerably.

3.12.3 Kinetic depth effect

One way of giving depth cueing is by moving the viewing angle and perceiving the way in which objects appear to move in relation to each other. The processing power of the CBM 64 is not sufficient to allow the kinetic depth effect to be employed in real time animated displays except where the displays are very simple.

3.12.4 Hidden line elimination

The depth of objects can be determined more easily if lines and surfaces which would normally be hidden from view by an opaque object are eliminated from the display. This is called hidden line elimination and is a very useful way of removing most of the depth of field ambiguities present within a wire frame image. Hidden line elimination can be achieved on a CBM 64 but does add considerably to the amount of computation involved in creating the display, consequently slowing down image generation. The technique of hidden line removal is shown in Program 19.

```
1 REM CUBE HIDDEN SURFACE
5 READ N,LK
10 DIM M(2,2),MS(2,2),A(N,2),B(N,2),LK(LK,1)
20 HIRES1,0
22 ORIGIN160,100
23 SX=10:SY=10:SZ=10
24 AX=0.07:AY=0.05:AZ=0.06
25 GOSUB 300
30 GOSUB 200
35 MAT B=A*MS
36 MAT A=B
40 MAT B=A*M
50 GOSUB 100
98 GETA$:IFA$<>"←"THEN36
99 NORM:LIST
100 MZ=-9999:FORI=0TON
101 IFB(I,2)>MZ THENMZ=B(I,2):ZZ=I
102 NEXT
103 CLG
107 FORI=0TOLK
108 L1=LK(I,0):L2=LK(I,1)
109 IFL1=ZZORL2=ZZTHEN120
110 DRAW B(L1,0),B(L1,1),B(L2,0),B(L2,1),3,1
120 NEXT
130 RETURN
200 FORI=0TON
210 READ A(I,0),A(I,1),A(I,2)
220 NEXT
```

```
230 FORI=0TOLK
240 READLK(I,0),LK(I,1)
250 NEXT
260 RETURN
300 M(0,0)=COS(AY)*COS(AZ)
310 M(0,1)=COS(AY)*SIN(AZ)
320 M(0,2)=-SIN(AY)
340 M(1,0)=COS(AX)*-SIN(AZ)+SIN(AX)*SIN(AY)*COS(AZ)
350 M(1,1)= COS(AX)*COS(AZ)+SIN(AX)*SIN(AY)*SIN(AZ)
360 M(1,2)=SIN(AX)*COS(AY)
370 M(2,0)=SIN(AX)*SIN(AZ)+COS(AX)*SIN(AY)*COS(AZ)
380 M(2,1)=-SIN(AX)*COS(AZ)+COS(AZ)*SIN(AY)*SIN(AZ)
390 M(2,2)=COS(AX)*COS(AY)
400 MS(0,0)=SX
410 MS(1,1)=SY
420 MS(2,2)=SZ
430 RETURN
1000 DATA 7,11
1010 DATA-5,-5,-5, 5,-5,-5, 5,5,-5, -5,5,-5
1020 DATA-5,-5,5, 5,-5,5, 5,5,5, -5,5,5
1025 REM LINK
1030 DATA 0,1, 1,2, 2,3, 3,0, 0,4
1040 DATA 4,5, 5,6, 6,7, 7,4, 1,5
1050 DATA 2,6, 3,7
10000 FORI=0TO7:PRINTSQR(B(I,0)↑2+B(I,1)↑2+B(I,2)↑2):NEXT
```

Program 19.

An enhancement of the hidden line removal technique is shading with hidden surfaces removed. This method, while computationally the most involved and therefore the slowest of the methods so far looked at, gives the most realistic and unambiguous display. The technique of shading an object with hidden surfaces removed is possible on a CBM 64 and is explained later in this chapter.

3.12.5 Stereoscopic views

It is feasible to give a feeling of image depth by creating a stereoscopic image. Such an image consists of two separate but nearly identical images, one constructed using red lines and the other using blue lines. The two images are offset from each other by a small amount according to the rules of stereo image construction. When viewed through glasses which have one eye covered with a red filter and the other with a blue filter the brain will combine the two images to give a perceived image with a convincing appearance of depth. This method has been tried on the CBM 64 but the screen resolution is inadequate to give good quality results; it is also essential that a monitor is used to ensure a good stable coloured image.

3.13 Shading surfaces

Shading surfaces properly is an important part of producing life-like images on a computer screen. Many complex algorithms have been produced to give near perfect results on powerful computers, but the C64 does not have the resolution or the processing power to make good use of them. The short demonstration program, Program 2Ø, uses primitive approximations to give an acceptable result. The program draws a three dimensional cube with shading of visible

```
1 REM *************
2 REM SHADED CUBE
3 REM *************
5 READ N,NF,NS
10 DIM M(2,2),MS(2,2),A(N,2),B(N,2),F(NF,NS)
20 HIRES0,12,11
21 ORIGIN160,100
22 REM SCALING AND ROTATION FACTORS
23 SX=10:SY=10:SZ=10
24 AX=0.3:AY=0.39:AZ=0.30
25 GOSUB 300 :REM GET DATA
30 GOSUB 200 :REM SET UP ARRAYS
35 MAT B=A*MS :REM SCALE BY SX,SY,SZ
36 MAT A=B
40 MAT B=A*M :REM ROTATE BY AX,AY,AZ
50 GOSUB 100 :REM DRAW CUBE
98 GETA$:IFA$<>"←"THEN36
99 NORM:LIST
100 REM CALCULATE SIDE OF FACE BEING VIEWED
101 CLG
105 FORF=0TONF
110 X1=B(F(F,1),0)-B(F(F,0),0)
120 Y1=B(F(F,1),1)-B(F(F,0),1)
130 X2=B(F(F,1),0)-B(F(F,2),0)
140 Y2=B(F(F,1),1)-B(F(F,2),1)
150 P=X1*Y2-Y1*X2
160 IFP > 0.001 THEN GOSUB 500 DRAW FACE
170 NEXT
180 RETURN
199 .
200 REM READ CORNER COORDINATES
205 FORI=0TON
210 READ A(I,0),A(I,1),A(I,2)
220 NEXT
225 REM READ CUBE FACE LINKING
230 FORI=0TONF
240 FOR J=0TONS
250 READF(I,J)
260 NEXTJ,I
270 RETURN
280 .
299 REM ROTATION ARRAY
300 M(0,0)=COS(AY)*COS(AZ)
310 M(0,1)=COS(AY)*SIN(AZ)
320 M(0,2)=-SIN(AY)
340 M(1,0)=COS(AX)*-SIN(AZ)+SIN(AX)*SIN(AY)*COS(AZ)
350 M(1,1)= COS(AX)*COS(AZ)+SIN(AX)*SIN(AY)*SIN(AZ)
360 M(1,2)=SIN(AX)*COS(AY)
370 M(2,0)=SIN(AX)*SIN(AZ)+COS(AX)*SIN(AY)*COS(AZ)
380 M(2,1)=-SIN(AX)*COS(AZ)+COS(AZ)*SIN(AY)*SIN(AZ)
390 M(2,2)=COS(AX)*COS(AY)
399 REM SCALING ARRAY
400 MS(0,0)=SX
410 MS(1,1)=SY
420 MS(2,2)=SZ
430 RETURN
440 .
500 REM DRAW A FACE OF THE CUBE
510 FOR I=1TONS
520 DRAWB(F(F,I-1),0),B(F(F,I-1),1),B(F(F)I),0),B(F(F,I),1),0,1
530 NEXT
600 REM SHADE IT
610 Z1=B(F(F,1),2)-B(F(F,0),2)
620 Z2=B(F(F,1),2)-B(F(F,2),2)
630 W1=SQR(X1*X1+Y1*Y1+Z1*Z1)
640 W2=SQR(X2*X2+Y2*Y2+Z2*Z2)
650 X1=X1/W1
660 X2=X2/W2
670 Y1=Y1/W1
```

```
680 Y2=Y2/W2
690 Z1=Z1/W1
700 Z2=Z2/W2
710 U= X1*Z2-X2*Z1 :REM U=1 FOR SURFACE UP THE RIGHT WAY & HORIZONTAL
720 PN=X1*Y2-Y1*X2 :REM PN=1 FOR SURFACE EXACTLY FACING PLANE OF VIEWING
730 SS=(U+2)*4-PN*2
750 FORR=1TOW1STEP SS
760 FORS=1TOW2STEP SS
770 X=B(F(F,1),0)-R*X1-S*X2
780 Y=B(F(F,1),1)-R*Y1-S*Y2
790 PLOTX,Y,0,1
800 NEXT S,R
810 RETURN
1000 DATA 7,5,4
1005 REM CORNERS
1010 DATA-5,-5,-5, 5,-5,-5, 5,5,-5, -5,5,-5
1020 DATA-5,-5,5, 5,-5,5, 5,5,5, -5,5,5
1021 :
1025 REM CORNERS OF FACES IN ANTI-CLOCKWISE ORDER AS VIEWED FROM OUTSIDE CUBE
1030 DATA 4,5,6,7,4
1040 DATA 2,1,0,3,2
1050 DATA 5,1,2,6,5
1060 DATA 0,4,7,3,0
1070 DATA 7,6,2,3,7
1080 DATA 0,1,5,4,0
1090 .
9999 REM ROUTINE TO CHECK ACCURACY OF MAT COMMAND
10000 FORI=0TO7:PRINTSQR(B(I,0)↑2+B(I,1)↑2+B(I,2)↑2):NEXT
```

Program 20.

surfaces. This program works on hidden surfaces rather than hidden lines. To make this possible the data for the cube is organised as the four edges of each face of the cube, and not as the 12 edges of the cube. The data on the linking of the edges of each face is stored in anti-clockwise order of the corners of the face as seen from outside the cube.

The program uses one main algorithm to calculate the direction in which a face is facing relative to an axis. It takes the form of a calculation to give a result of one number. If the result is zero then the surface is a parallel axis, otherwise the sign of the result gives the way it is facing along the axis. The magnitude is maximum when the plane is perpendicular to the axis. The equation requires the coordinates of 3 points on the surface. The program uses 3 corners of a face of the cube. If the points are numbered 1,2 and 3 then the lines 1 to 2 and 2 to 3 need to be at right angles.

If the coordinates of point 1 are X1,Y1,Z1 then the following formula is required to calculate the orientation of a plane compared to the Z axis:

$$r=(X2-X1)*(Y2-Y3)-(Y2-Y1)*(X2-X1)$$

The program also uses this with the length of the lines prescaled to 1. This is calculated twice; once to approximate the inclination relative to a light source above the cube and again for the inclination to the plane of view. These two values are used to find the number of dots to put on the face.

Chapter Four
Games Graphics: Some Hints and Techniques

4.1 Initial planning of a graphics game

When writing games for the Commodore 64, a very early part of the planning is deciding which type of graphics to use. There is the choice of character or high resolution graphics, either of which can be used with or without sprites. Further considerations are that the game may require multiple screens, smooth scrolling, or split screens to give mixed text and high resolution or to enable more than eight sprites to be displayed simultaneously. There are many other decisions to be made; last in the list is often which colour combinations to use.

When ready to start writing the game it is best to have a good plan for where in memory to put code, data and graphics screens. Start by considering which bank of 16K you will give the VIC chip access to. Your game can use more than one bank switching as desired. The following is a guide to choosing which banks to use.

Bank Ø
(3 in $DDØØ bank select register.) If you use the VIC chip with this bank not all of the 16K can be used for graphics. Zero page is always best reserved for variables and pointers, and most of page one has to be reserved for stack. If 64 ROM routines are to be used they will also require pages Ø,2 and 3 to be intact.

Bank 1
All of bank 1 can be used for graphics data. The only disadvantage is that this bank is near the centre of the memory map and may cause problems when finding space for large data tables, such as text data for adventure games.

Bank 2
Half of this bank contains the Basic ROM but this will only cause problems to programs in Basic and only then if they want to PEEK graphics data. Machine code programs can use this bank just like bank 1, after switching out the Basic ROM using address 1.

Bank 3
This bank can be useful in some cases. The main problem in using this bank is the I/O area (VIC,SID,CIA chips and colour RAM) and that for read operations the kernal ROM lies over it. The simplest use for this bank is as a high resolution title screen during loading of a game. Also large games not using kernal can use it for a text screen with a suitable character set.

High resolution (bit map) screens

If the high resolution screen is being used only for a background or for a title display, then it is best placed out of the way behind a ROM. It can easily be flashed up on the screen using a quick change of bank. If it is to be used with a large number of sprites or continuous plotting onto it, don't put it behind the kernal if kernal is being used as well.

4.1.1 Character screens and character sets

If the ROM character set is being used and a small number of sprites the default screen position is the easiest to use. The only thing to consider when moving the bank and position of the character screen is that the kernal screen scroll and read routines cannot work behind a ROM that is switched in.

4.1.2 Bit map or character screen

A passive background or title display is best done on a high resolution screen. Games using moving line graphics or software sprites have to be done on a high resolution screen or plotted on a block of user defined characters. Most other types of games are normally done on character screens as this enables a whole host of games effects. These effects include smooth scrolling and changing character definitions. Also there is a large memory saving which is very important in games with a few screens.

4.1.3 Split screens

If you need a high resolution display and characters for text you will either have to use a block of user defined characters to plot on or split the screen using raster interrupts. A raster interrupt can also be used to give you more than 256 user defined characters or more than 8 sprites, or simply enable you to change the VIC colour registers.

4.1.4 Raster effects

The VIC chip makes the number of the current raster line being displayed available on the high bit of register 17 and register 18. A simple but not very useful method of using this is to make the program loop until the required raster is reached. The following Basic program (Program 21) just changes the

```
10 REM BASIC SPLIT
15 GOSUB 100 STOP TIMER IRQ
20 A=53265:B=53281
25 REM WAIT ON RASTER HI BIT
30 WAITA,128:WAITA,128,128
40 POKEB,4
50 POKEB,0
60 GOTO30
90 .
100 A=53266:B=53281:POKE56334,0
110 PRINT"[illegible]          USE "
120 PRINT"[illegible]       RUN STOP/RESTORE
130 PRINT"[illegible]       TO EXIT
140 RETURN
```

Program 21.

background colour of a band across the screen using the WAIT command to test the high bit of register 17. The only real use for this is as a simple screen wait to stop screen flicker on moving graphics.

A more useful method is to store the required raster number in registers 17 and 18 and enable raster interrupts. Before this is done normal interrupts must be disabled and the interrupt vector at $Ø314–$Ø315 changed to point to a routine to handle and clear the interrupts. Normal timer interrupts need to be disabled because they cause flicker on splits.

4.1.5 Raster IRQ routines

A routine to switch from normal timer IRQ to raster IRQ could be as follows:

```
    SEI      Set interrupt disable
```

Disable and clear CIA timer interrupts:

```
Disable IRQ:
    LDA #$1F
    STA $DCØD      CIA control registers
    STA DDØD
Clear IRQ:
    LDA $DCØD
    LDA $DDØD
```

Change IRQ vector (assuming kernal is being used):

```
    LDA #<SPLIT lsb of address IRQ routine
    STA $Ø314
    LDA #>SPLIT
    STA $Ø315
```

Enable interrupts:

```
    CLI
```

Change to hi bit of raster comparison register:

```
    LDA $DØ11
    AND #$7F
    ORA #Raster no. high bit (Ø or $8Ø)
    STA $DØ11
```

Set low byte of raster comparison:

```
    LDA #Raster low byte
    STA $DØ12
```

Clear processor interrupt flag:

```
    CLI
```

Enable VIC raster IRQ:

```
    LDA #1
    STA $DØ1A
```

The interrupt routine pointed to by $Ø314–$Ø315 should at one point contain the code:

```
LDA #1
STA $DØ19
```

If interrupts are being handled by kernal and $Ø314–$Ø315 then the routine may end with either a JMP $EA31 or:

```
PLA    Restore registers
TAY
PLA
TAX
PLA
RTI    Return from interrupt
```

The jump to $EA31 will scan the keyboard and update the clock (the clock will be slow as it is only updated every 1/50 sec). When exiting using an RTI, if the keyboard is required a JSR $FF9F can be included in the routine to scan the keyboard.

In most cases a single split on the screen will not be enough. For effects like smooth scrolling and two or more background colours etc., you need at least two splits. The first split is near the top of the screen to set the effect. The second split resets to the conditions used for the remainder of the screen. If you are using more than one split it is best that only one ends with a JMP $EA31 or uses a JSR to $FF9F.

4.2 Smooth scrolling

Horizontal smooth scrolling of part of a text screen is simple once you have mastered using splits. Scrolling looks best with the screen width reduced to 38 characters. Screen width may be reduced for the whole screen or just the part being scrolled, as suits the game.

Your first split routine will have to reduce screen width if this is not done for the whole screen. Then store a value for the smooth scroll position. The smooth scroll position can be either increased or decreased each time it is stored to register $DØ12. When this value crosses zero (from 7 to Ø or Ø to 7) the lines being scrolled must be moved physically one character sideways. This physical scroll is best done on the next or later split from the one that sets the smooth scroll position. The assembly listing in Program 22 is to smooth scroll 1Ø lines of

```
C000           *=$C000
C000           !SMOOTH SCROLL 10 LINES
C000           !RASTER CMP NUMBERS FOR SPLITS
C000           RAS1          =  114
C000           RAS2          =  194
C000           !
C000           !INITIALISE NEW IRQ
C000 78        SETUP         SEI
C001           !KILL CIA CHIPS
C001 A91F                    LDA #$1F
C003 8D0DDC                  STA $DC0D
```

```
C006 8D0DDD            STA $DD0D
C009 AD0DDC            LDA $DC0D
C00C AD0DDD            LDA $DD0D
C00F          !CLEAR HI BIT RASTER CMP
C00F AD11D0            LDA $D011
C012 297F              AND #$7F
C014 8D11D0            STA $D011
C017          !SET CMP VALUE FOR FIRST SPLIT
C017 A972              LDA #RAS1
C019 8D12D0            STA $D012
C01C          !POINT IRQ VECTOR TO SPLIT1
C01C A92D              LDA #<SPLIT1
C01E 8D1403            STA $0314
C021 A9C0              LDA #>SPLIT1
C023 8D1503            STA $0315
C026          !ENABLE INTERRUPTS BEFORE EXIT
C026 A901              LDA #1
C028 8D1AD0            STA $D01A
C02B 58                CLI
C02C 60                RTS
C02D          !
C02D          !FIRST INTERRUPT ROUTINE
C02D EA       SPLIT1   NOP                      !DELAY
C02E EA                NOP
C02F EA                NOP
C030 EE20D0            INC $D020                !BORDER
C033 EE21D0            INC $D021                !SCREEN
C036          !SET SMOOTH SCROLL POSITION
C036          !AND REDUCE SCREEN WIDTH
C036 AD04C1            LDA SCRP
C039 2907              AND #7
C03B 8D16D0            STA $D016
C03E          !NEW RASTER CMP VALUE
C03E A9C2              LDA #RAS2
C040 8D12D0            STA $D012
C043          !CLEAR IRQ
C043 A901              LDA #1
C045 8D19D0            STA $D019
C048          !POINT IRQ VECTOR TO SPLIT2
C048 A955              LDA #<SPLIT2
C04A 8D1403            STA $0314
C04D A9C0              LDA #>SPLIT2
C04F 8D1503            STA $0315
C052          !EXIT TO KERNAL IRQ ROUTINE
C052 4C31EA            JMP $EA31
C055          !
C055          !SECOND INTERRUPT ROUTINE
C055 EA       SPLIT2   NOP                      !DELAY
C056 EA                NOP
C057 EA                NOP
C058 CE20D0            DEC $D020                !BORDER
C05B CE21D0            DEC $D021                !SCREEN
C05E          !NORMAL SCREEN WIDTH
C05E A908              LDA #8
C060 8D16D0            STA $D016
C063          !BUMP SMOOTH SCROLL POSITION
C063 EE04C1            INC SCRP
C066          !PHYYSICAL SCROLL REQUIRED ?
C066 AD04C1            LDA SCRP
C069 2907              AND #7
C06B D07D              BNE NOSCRL
C06D          !PHYSICAL SCROLL
C06D A026              LDY #$26
C06F B94005 PHSCRL     LDA $0540,Y              !1
C072 994105            STA $0541,Y
C075 B96805            LDA $0568,Y              !2
C078 996905            STA $0569,Y
C07B B99005            LDA $0590,Y              !3
C07E 999105            STA $0591,Y
```

```
C081 B9B805           LDA $05B8,Y               !4
C084 99B905           STA $05B9,Y
C087 B9E005           LDA $05E0,Y               !5
C08A 99E105           STA $05E1,Y
C08D B90806           LDA $0608,Y               !6
C090 990906           STA $0609,Y
C093 B93006           LDA $0630,Y               !7
C096 993106           STA $0631,Y
C099 B95806           LDA $0658,Y               !8
C09C 995906           STA $0659,Y
C09F B98006           LDA $0680,Y               !9
C0A2 998106           STA $0681,Y
C0A5 B9A806           LDA $06A8,Y               !10
C0A8 99A906           STA $06A9,Y
C0AB 88               DEY
C0AC 10C1             BPL PHSCRL
C0AE         !MOVE LAST BYTE OF LINE TO FIRST
C0AE AD6705           LDA $0567                 !1
C0B1 8D4005           STA $0540
C0B4 AD8F05           LDA $058F                 !2
C0B7 8D6805           STA $0568
C0BA AD6705           LDA $0567                 !3
C0BD 8D9005           STA $0590
C0C0 ADDF05           LDA $05DF                 !4
C0C3 8DB805           STA $05B8
C0C6 AD0706           LDA $0607                 !5
C0C9 8DE005           STA $05E0
C0CC AD2F06           LDA $062F                 !6
C0CF 8D0806           STA $0608
C0D2 AD5706           LDA $0657                 !7
C0D5 8D3006           STA $0630
C0D8 AD7F06           LDA $067F                 !8
C0DB 8D5806           STA $0658
C0DE ADA706           LDA $06A7                 !9
C0E1 8D8006           STA $0680
C0E4 ADCF06           LDA $06CF                 !10
C0E7 8DA806           STA $06A8
C0EA         !SET INTERRUPT VECTOR TO SPLIT1
C0EA A92D    NOSCRL   LDA #<SPLIT1
C0EC 8D1403           STA $0314
C0EF A9C0             LDA #>SPLIT1
C0F1 8D1503           STA $0315
C0F4         !CHANGE RASTERCMP VALUE
C0F4 A972             LDA #RAS1
C0F6 8D12D0           STA $D012
C0F9 A901             LDA #1
C0FB 8D19D0           STA $D019
C0FE 68               PLA
C0FF A8               TAY
C100 68               PLA
C101 AA               TAX
C102 68               PLA
C103 40               RTI
C104 00      SCRP     BYT 0
```

Program 22.

the screen from left to right at 50 pixels a second. The routine also includes wrap around; that is, characters leaving the right edge of the screen appear on the left side of the screen. To enable the smooth scroll, enter SYS 49152.

4.3 More than eight sprites

Sprite registers can also be changed by splits. The normal method is for a main program to calculate the required sprite settings and store them in a reserved area. Then a split interrupt routine takes the settings and transfers them to the VIC registers and sprite block pointers. Note that the split routine may also read collision registers and store them to a reserved place. It will usually be necessary for the interrupt routine and the main program to set and wait for flags so that the main program won't change the sprite settings before the split routine has used them.

The assembly listing in Program 23 enables 8 sprites and changes the vertical position on each of its 6 splits giving you 48 sprites. To enable the routine, enter SYS 49152.

```
C000              *=$C000
C000 A9FF   START      LDA #$FF          !ENABLE ALL SPRITES
C002 8D15D0            STA $D015
C005 8D17D0            STA $D017
C008 A208              LDX #8
C00A 8A     CLOOP      TXA
C00B 9D1FD0            STA $D01F,X
C00E E8                INX
C00F E010              CPX #$10
C011 D0F7              BNE CLOOP
C013 A91E              LDA #30
C015 A200              LDX #0
C017 9D00D0 XLOOP      STA $D000,X
C01A E8                INX
C01B E8                INX
C01C 18                CLC
C01D 6926              ADC #38
C01F E010              CPX #$10
C021 D0F4              BNE XLOOP
C023 A9C0              LDA #$C0
C025 8D10D0            STA $D010
C028 A210              LDX #16
C02A 8A     PLOOP      TXA
C02B 9DE807            STA $07E8,X
C02E E8                INX
C02F E018              CPX #24
C031 D0F7              BNE PLOOP
C033 A91F              LDA #$1F          !KILL CIA CHIPS
C035 78                SEI
C036 8D0DDC            STA $DC0D
C039 8D0DDD            STA $DD0D
C03C AD0DDC            LDA $DC0D         !CLEAR ANY IRQ FLAGS
C03F AD0DDD            LDA $DD0D
C042 A900              LDA #0            !START WITH SPLIT 1
C044 8D66C0            STA SPLITN
C047 AD68C0            LDA SPVTAB+1      !RASTER NO. OF SPLIT 1
C04A 8D12D0            STA $D012         !RASTER CMP REG
C04D A96D              LDA #<SPLIT       !CHANGE IRQ VECTOR
C04F 8D1403            STA $0314
C052 A9C0              LDA #>SPLIT
C054 8D1503            STA $0315
C057 A901              LDA #1            !ENABLE RASTER IRQ
C059 8D1AD0            STA $D01A
C05C AD11D0            LDA $D011         !CLEAR HI BIT RASTER CM

C05F 297F              AND #$7F
C061 8D11D0            STA $D011
C064 58                CLI
C065 60                RTS
C066 00     SPLITN     BYT 0
C067 003264 SPVTAB     BYT 0,50,100,150,200,250
C06D AE66C0 SPLIT      LDX SPLITN
C070 EA                NOP
```

```
C071 EA               NOP
C072 EA               NOP
C073 EA               NOP
C074 EA               NOP
C075 EA               NOP
C076 EA               NOP
C077 8E20D0           STX $D020          !SCREEN & BORDER
C07A 8E21D0           STX $D021
C07D E8               INX
C07E E006             CPX #6             ;CHECK SPLIT NUMBER
C080 D005             BNE RNOK
C082 209FFF           JSR $FF9F          !SCAN KEYBOARD
C085 A200             LDX #0
C087 8E66C0 RNOK      STX SPLITN
C08A BD67C0           LDA SPVTAB,X
C08D 8D12D0           STA $D012
C090 A901             LDA #1             !CLEAR IRQ
C092 8D19D0           STA $D019
C095 AD12D0           LDA $D012          !CURRENT RASTER
C098 18               CLC
C099 6905             ADC #5
C09B 8D01D0           STA $D001          !STORE SPRITE Y
C09E 8D03D0           STA $D003
C0A1 8D05D0           STA $D005
C0A4 8D07D0           STA $D007
C0A7 8D09D0           STA $D009
C0AA 8D0BD0           STA $D00B
C0AD 8D0DD0           STA $D00D
C0B0 8D0FD0           STA $D00F
C0B3 68               PLA
C0B4 A8               TAY
C0B5 68               PLA
C0B6 AA               TAX
C0B7 68               PLA
C0B8 40               RTI
```

Program 23.

4.4 Moving objects in interrupts

It may be desirable as part of a game that an object moves continuously with little interaction with the main program. The routine to move the sprite (or block of pixels etc.) as part of an interrupt routine will enable you to set it moving and then leave it going until it is stopped. The assembly listing in Program 24 moves a block shaped sprite around the screen bouncing off characters. The routine runs on raster interrupts, using background collision to detect characters. To start the sprite moving, enter SYS 49152.

```
C000        *=$C000
C000        !ENABLE A SPRITE
C000 A901   START       LDA #$01
C002 8D15D0             STA $D015
C005 A901               LDA #1
C007        !SET COLOUR
C007 8D27D0             STA $D027
C00A A900               LDA #0
C00C        !NO EXPANSION
C00C 8D17D0             STA $D017
C00F 8D1DD0             STA $D01D
C012        !SPRITE 11 BLOCK
C012 A90B               LDA #11
```

```
C014 8DF807             STA $07F8
C017 A9FF               LDA #255
C019 A23E               LDX #62
C01B         !SOLID BLOCK
C01B 9DC002 FSPRL       STA $02C0,X
C01E CA                 DEX
C01F 10FA               BPL FSPRL
C021         !START SPRITE AT 100,100
C021 A900               LDA #0
C023 8DEEC0             STA XP+1
C026 8D10D0             STA $D010
C029 A964               LDA #100
C02B 8DEDC0             STA XP
C02E 8DEFC0             STA YP
C031 8D00D0             STA $D000
C034 8D01D0             STA $D001
C037         !KILL CIA CHIPS
C037 A91F               LDA #$1F
C039 78                 SEI
C03A 8D0DDC             STA $DC0D
C03D 8D0DDD             STA $DD0D
C040         !CLEAR ANY IRQ FLAGS
C040 AD0DDC             LDA $DC0D
C043 AD0DDD             LDA $DD0D
C046         !RASTER NO.
C046 A500               LDA 0
C048         !RASTER CMP REG
C048 8D12D0             STA $D012
C04B         !CHANGE IRQ VECTOR
C04B A964               LDA #<SPLIT
C04D 8D1403             STA $0314
C050 A9C0               LDA #>SPLIT
C052 8D1503             STA $0315
C055         !ENABLE RASTER IRQ
C055 A901               LDA #1
C057 8D1AD0             STA $D01A
C05A         !CLEAR HI BIT RASTER CMP
C05A AD11D0             LDA $D011
C05D 297F               AND #$7F
C05F 8D11D0             STA $D011
C062 58                 CLI
C063 60                 RTS
C064         ! IRQ ROUTINE
C064         !CLEAR IRQ
C064 A901   SPLIT       LDA #1
C066 8D19D0             STA $D019
C069         !MOVE SPRITE ?
C069 ADECC0             LDA ENABLE
C06C D003               BNE CONT
C06E 4C31EA             JMP $EA31
C071         !CHECK FOR COLLISION
C071 AD1FD0 CONT        LDA $D01F
C074 F020               BEQ NOCOLL
C076 ADF0C0             LDA CO
C079         !LAST MOVE IN X OR Y DIR ?
C079 2901               AND #1
C07B F00E               BEQ YCOL
C07D         !REVERSE X MOVEMENT
C07D A901               LDA #1
C07F 4DF1C0             EOR XV
C082 8DF1C0             STA XV
C085         !AND MOVE
C085 20C0C0             JSR XMOVE
C088 4C96C0             JMP NOCOLL
C08B         !REVERSE Y MOVEMENT
C08B A901   YCOL        LDA #1
C08D 4DF2C0             EOR YV
C090 8DF2C0             STA YV
C093         !AND MOVE
```

```
C093 20DBC0             JSR YMOVE
C096        !NO COLLISION OCCURRED
C096        !MOVE ALTERNATE X OR Y DIR.
C096 EEF0C0 NOCOLL      INC CO
C099 A901               LDA #1
C09B 2DF0C0             AND CO
C09E F006               BEQ YMOV
C0A0 20C0C0             JSR XMOVE
C0A3 4CA9C0             JMP PLACE
C0A6 20DBC0 YMOV        JSR YMOVE
C0A9        !PUT COORDINATES IN VIC REG.
C0A9 ADEDC0 PLACE       LDA XP
C0AC 8D00D0             STA $D000
C0AF ADEEC0             LDA XP+1
C0B2 2901               AND #1
C0B4 8D10D0             STA $D010
C0B7 ADEFC0             LDA YP
C0BA 8D01D0             STA $D001
C0BD 4C31EA             JMP $EA31
C0C0        !INC. OR DEC. X POSITION
C0C0 A901   XMOVE       LDA #1
C0C2 A000               LDY #0
C0C4 AEF1C0             LDX XV
C0C7 D003               BNE XADD
C0C9 88                 DEY
C0CA A9FF               LDA #$FF
C0CC 18     XADD        CLC
C0CD 6DEDC0             ADC XP
C0D0 8DEDC0             STA XP
C0D3 98                 TYA
C0D4 6DEEC0             ADC XP+1
C0D7 8DEEC0             STA XP+1
C0DA 60                 RTS
C0DB        !INC. OR DEC. Y POSITION
C0DB A901   YMOVE       LDA #1
C0DD AEF2C0             LDX YV
C0E0 D002               BNE YADD
C0E2 A9FF               LDA #$FF
C0E4 18     YADD        CLC
C0E5 6DEFC0             ADC YP
C0E8 8DEFC0             STA YP
C0EB 60                 RTS
C0EC        !ENABLE FLAG
C0EC 00     ENABLE      BYT 0
C0ED        !HORIZONTAL POSITION
C0ED 0000   XP          WOR 0
C0EF        !VERTICAL POSITION
C0EF 00     YP          BYT 0
C0F0        !BIT 0 FLAG MOVE X OR Y
C0F0 00     CO          BYT 0
C0F1        !FLAG 0=DEC X , 1=INC X
C0F1 00     XV          BYT 0
C0F2        !FLAG 0=DEC Y , 1=INC Y
C0F2 00     YV          BYT 0
```

Program 24.

It is also possible to create animation effects by switching between slightly different images stored on separate screens within memory. This is shown in Program 25.

```
10 REM ****************************
20 REM MULTIPLE TEXT SCREEN
30 REM ANIMATION
40 REM ****************************
50 DIM CH(15),P2(3)
55 REM READ CHAR CODE TABLE FOR PLOT ROUTINE
60 FORI=0TO15
70 READCH(I)
```

```
80 NEXT
82 REM TABLE POWERS OF 2
85 FORI=0TO3:P2(I)=2↑I:NEXT
89 REM MAKE SCREEN POKES VISIBLE ON OLD C64
90 POKE53281,14:POKE646,14
92 :
95 REM CLEAR SCREENS 8 TO 15
97 REM AND PRINT SCREEN NUMBER AT TOP
100 FOR S=8 TO 15
110 GOSUB1000
120 PRINT"♥";S
130 NEXT:POKE53281,6
131 :
135 REM DRAW PATTER ON SCREENS 8 TO 15
140 FORS=8TO15
150 GOSUB1000
160 FORU=0TO20
170 FORJ=0TO3
180 A=J*π/5+U/20+S*π/4
190 X=40+U*SIN(A)
200 Y=25+U*COS(A)
210 GOSUB2000 Q SQUARE PLOT AT X,Y
220 NEXTJ,U,S
221 :
225 REM LOOP DISPLAYING SCREENS
230 FORZ=0TO2000
240 FORS=8TO15
250 GOSUB1000
260 GETA$:IFA$<>""THENZ=2000:REM DELAY OR END LOOP
270 NEXTS,Z
280 S=1:GOSUB1000 NORMAL SCREEN
290 END
1000 REM ***************************
1010 REM VIC AND KERNAL FOR SCREEN S
1020 REM ***************************
1030 IF(S>1 AND S<8)OR S=0THEN S=1
1040 POKE53272,5+S*16:REM SET VIC
1050 POKE648,4*S:REM TELL KERNAL
1060 RETURN
1070 .
2000 REM QUARTER SQUARE PLOT X,Y
2001 IFX<0ORY<0ORX>79ORY>49THENRETURN
2010 A=INT(X/2)+INT(Y/2)*40+S*1024
2020 P=P2((X AND1)+(Y AND 1)*2)
2030 C=0
2040 V=PEEK(A)
2050 FORC1=0TO15
2060 IFV=CH(C1)THENC=C1
2070 NEXT
2080 POKEA,CH(C OR P)
2090 RETURN
10000 DATA32,126,124,226,123,97,255,236
10010 DATA108,127,225,251,98,252,254
10020 DATA160
```

Program 25.

4.5 Changing characters in interrupts

Interrupt routines are also used for changing the definition of characters or sprite block pointers while they are being displayed. This can give the effect of constant movement or rotation. This can be used for scrolling short messages or for moving walkways etc. The assembly listing in Program 26 scrolls the word 'READY.' sideways and '∅123456789' vertically. Both are scrolled with wrap around from the first character to the last. This routine runs in normal timer interrupts but is well suited for use on raster. To start this routine, enter SYS 16384.

```
033C          !************************
033C          !CHANGING CHAR DEFINITIONS
033C        . !FROM INTERRUPTS
033C          !************************
033C          !
4000          * = $4000
4000 A938     ENTRY          LDA #$38         !PROTECT MEMORY
4002 8538                    STA $38
4004 85FE                    STA $FE
4006 A900                    LDA #0
4008 8537                    STA $37
400A 85FD                    STA $FD
400C A9D0                    LDA #$D0
400E 85FC                    STA $FC
4010 A900                    LDA #$00
4012 85FB                    STA $FB
4014 78                      SEI              !COPY CHAR SET INTO RAM
4015 A933                    LDA #$33
4017 8501                    STA $01
4019 A000                    LDY #0
401B B1FB     MOVE           LDA ($FB),Y
401D 91FD                    STA ($FD),Y
401F C8                      INY
4020 D0F9                    BNE MOVE
4022 E6FC                    INC $FC
4024 E6FE                    INC $FE
4026 A5FE                    LDA $FE
4028 C940                    CMP #$40
402A D0EF                    BNE MOVE
402C A937                    LDA #$37
402E 8501                    STA $01
4030 A91F                    LDA #$1F         !USER CHAR AT $3800
4032 8D18D0                  STA $D018
4035 A941                    LDA #<IRQ        !CHANGE IRQ VECTOR
4037 8D1403                  STA $0314
403A A940                    LDA #>IRQ
403C 8D1503                  STA $0315
403F 58                      CLI              !RETURN TO BASIC
4040 60                      RTS
4041 A200     IRQ            LDX #0           !SCROLL 'READY.'
4043 BD9038 LOOP             LDA $3890,X      ! 'R'
4046 0A                      ASL A
4047 3E7039                  ROL $3970,X      ! '.'
404A 3EC838                  ROL $38C8,X      ! 'Y'
404D 3E2038                  ROL $3820,X      ! 'D'
4050 3E0838                  ROL $3808,X      ! 'A'
4053 3E2838                  ROL $3828,X      ! 'E'
4056 3E9038                  ROL $3890,X      ! 'R'
4059 E8                      INX
405A E008                    CPX #8
405C D0E5                    BNE LOOP
405E A24E                    LDX #78          !SCROLL '0123456789'
4060 ACCF39                  LDY $39CF
4063 BD8039 LOOP1            LDA $3980,X      ! '0'
4066 9D8139                  STA $3981,X
4069 E000                    CPX #0
406B F004                    BEQ EXIT
406D CA                      DEX
406E 4C6340                  JMP LOOP1
4071 8C8039 EXIT             STY $3980
4074 4C31EA                  JMP $EA31        !CONTINUE NORMAL IRQ
```

Program 26.

Chapter Five

Some Advanced Aspects of Sound on the CBM 64

The CBM 64 has excellent sound generation capabilities and it is possible to program the computer to generate very acceptable quality music. This chapter covers two advanced aspects of music generation on the 64. The first is adding a real music keyboard to allow easy input and playing of a musical score. The hardware is quite simple and cheap to construct and can easily be undertaken by anyone with a small amount of electronics experience. The software is capable of a considerable amount of expansion with potential applications ranging from the teaching of music to computerised score writing. The second part of this chapter is devoted to the techniques of playing music in the background whilst running a program, a feature frequently encountered in games programs.

5.1 The keyboard decoder and player

To go with the interface for the keyboard, a program is required to ask for and accept data from the interface. This program must be able to run fast enough to accept a keypress the instant that it is made and must therefore be written in machine code. To make it even better, the routine in Program 27 has been encoded to run within the hardware interrupts of the Commodore 64 and can thus check the keys 6Ø times a second and be totally independent of the rest of the computer.

```
LOC   CODE       LINE

0000             ; KEYBOARD SCANNER AND PLAYER
0000             ; FOR USE WITH THE KEYBOARD
0000             ; INTERFACE
0000             ;
0000             ;
0000             *       =$C000
C000             ; TABLE FOR THE ACTUAL NOTES TO
C000             ; BE PLAYED.
C000             ;
C000  00         TABLE   .BYT 0,0,147,8,21,9,159,9
C001  00
C002  93
C003  08
C004  15
C005  09
C006  9F
C007  09
C008  32                 .BYT 50,10,205,10,113,11,32,12
C009  0A
C00A  CD
C00B  0A
```

```
LOC   CODE        LINE

C00C  71
C00D  0B
C00E  20
C00F  0C
C010  D8          .BYT 216,12,156,13,107,14,71,15
C011  0C
C012  9C
C013  0D
C014  6B
C015  0E
C016  47
C017  0F
C018  2F          .BYT 47,16,38,17,43,18,63,19
C019  10
C01A  26
C01B  11
C01C  2B
C01D  12
C01E  3F
C01F  13
C020  64          .BYT 100,20,155,21,227,22,64,24
C021  14
C022  9B
C023  15
C024  E3
C025  16
C026  40
C027  18
C028  B1          .BYT 177,25,56,27,215,28,142,30
C029  19
C02A  38
C02B  1B
C02C  D7
C02D  1C
C02E  8E
C02F  1E
C030  5F          .BYT 95,32,76,34,86,36,127,38
C031  20
C032  4C
C033  22
C034  56
C035  24
C036  7F
C037  26
C038  C9          .BYT 201,40,54,43,199,45,128,48
C039  28
C03A  36
C03B  2B
C03C  C7
C03D  2D
C03E  80
C03F  30
C040  63          .BYT 99,51,113,54,174,57,28,61
C041  33
C042  71
C043  36
C044  AE
C045  39
C046  1C
C047  3D
C048  BE          .BYT 190,64,152,68,172,72,254,76
C049  40
C04A  98
C04B  44
C04C  AC
```

```
LOC   CODE        LINE

C04D  48
C04E  FE
C04F  4C
C050  92                 .BYT 146,81,108,86,143,91,1,97
C051  51
C052  6C
C053  56
C054  8F
C055  5B
C056  01
C057  61
C058  C6                 .BYT 198,102,226,108,92,115,56,122
C059  66
C05A  E2
C05B  6C
C05C  5C
C05D  73
C05E  38
C05F  7A
C060  7C                 .BYT 124,129,48,137
C061  81
C062  30
C063  89
C064
C064              ; INITIALISE THE KEYBOARD
C064              ;
C064              *      =$C080
C080  A9 00              LDA #$00
C082  A0 18              LDY #24
C084  99 00 D4    LOOP   STA $D400,Y
C087  88                 DEY
C088  10 FA              BPL LOOP
C08A  A9 0F              LDA #$0F        ;SET UP THE D.D.R
C08C  8D 03 DD           STA $DD03       ; FOR THE USER PORT
C08F  A9 09              LDA #$09        ;ATTACK=0,DECAY=9
C091  8D 05 D4           STA $D405
C094  8D 0C D4           STA $D40C
C097  8D 13 D4           STA $D413
C09A  A9 00              LDA #$00        ;SUSTAIN=0,RELEASE=0
C09C  8D 06 D4           STA $D406
C09F  8D 0D D4           STA $D40D
C0A2  8D 14 D4           STA $D414
C0A5  A9 40              LDA #$40        ;WAVEFORM=PULSE
C0A7  8D B8 C2           STA WAVE
C0AA  8D 04 D4           STA $D404
C0AD  8D 0B D4           STA $D40B
C0B0  8D 12 D4           STA $D412
C0B3  A9 00              LDA #$00        ;LOW BYTE OF P WIDTH
C0B5  8D 02 D4           STA $D402
C0B8  8D 09 D4           STA $D409
C0BB  8D 10 D4           STA $D410
C0BE  A9 08              LDA #$08        ;HI BYTE OF P WIDTH
C0C0  8D 03 D4           STA $D403
C0C3  8D 0A D4           STA $D40A
C0C6  8D 11 D4           STA $D411
C0C9  78          ENABLE SEI
C0CA  A9 0F              LDA #$0F
C0CC  8D 18 D4           STA $D418       ;SET MAX VOLUME
C0CF  A9 ED              LDA #<TESTER    ;PUT ADDRESS OF
C0D1  8D 14 03           STA $0314       ; SCANNER INTO IRQ
C0D4  A9 C0              LDA #>TESTER    ; VECTOR
C0D6  8D 15 03           STA $0315
C0D9  58                 CLI             ;ENABLE KEYBOARD
C0DA  60                 RTS
C0DB              ;
C0DB              ;DISABLE KEYBOARD
```

```
LOC   CODE        LINE

C0DB              ;
C0DB  78          DISABL SEI
C0DC  A9 00              LDA #$00
C0DE  8D 18 D4           STA $D418      ;NO VOLUME
C0E1  A9 31              LDA #$31
C0E3  8D 14 03           STA $0314      ;RESET IRQ
C0E6  A9 EA              LDA #$EA
C0E8  8D 15 03           STA $0315
C0EB  58                 CLI
C0EC  60                 RTS
C0ED              ;
C0ED              ; ACTUAL SCANNER FOR KEYBOARD
C0ED              ;
C0ED  A0 00       TESTER LDY #$00       ;# OF NOTES FOUND
C0EF  84 FB              STY $FB        ; EQUALS ZERO
C0F1  8C 01 DD    INPUT  STY $DD01      ;TEST FOUR KEYS
C0F4  AD 01 DD           LDA $DD01
C0F7  85 02              STA $02        ;STORE VALUE OFF
C0F9  29 10              AND #$10       ;TEST CHIP 1
C0FB  F0 0C              BEQ NOT1       ;NO KEY
C0FD  18                 CLC
C0FE  98                 TYA
C0FF  69 01              ADC #$01       ;CONVERT TO KEY #
C101  A6 FB              LDX $FB        ;POINTER TO INPUT BUFFER
C103  9D D2 C2           STA INPUTB,X   ;STORE KEY #
C106  E8                 INX
C107  86 FB              STX $FB
C109  A5 02       NOT1   LDA $02
C10B  29 20              AND #$20       ;TEST CHIP 2
C10D  F0 0C              BEQ NOT2       ;NO KEY
C10F  18                 CLC
C110  98                 TYA
C111  69 11              ADC #$11       ;CONVERT TO KEY #
C113  A6 FB              LDX $FB        ;GET POINTER
C115  9D D2 C2           STA INPUTB,X   ;STORE KEY #
C118  E8                 INX
C119  86 FB              STX $FB
C11B  A5 02       NOT2   LDA $02
C11D  29 40              AND #$40       ;TEST CHIP 3
C11F  F0 0C              BEQ NOT3       ;NO KEY
C121  18                 CLC
C122  98                 TYA
C123  69 21              ADC #$21       ;CONVERT TO KEY #
C125  A6 FB              LDX $FB        ;GET POINTER
C127  9D D2 C2           STA INPUTB,X   ;STORE KEY #
C12A  E8                 INX
C12B  86 FB              STX $FB
C12D  C0 00       NOT3   CPY #$00       ;IS IT A SWITCH?
C12F  D0 12              BNE NOTKEY     ;YES
C131  A5 02              LDA $02
C133  29 80              AND #$80       ;TEST CHIP 4
C135  F0 1D              BEQ NOT4       ;NO KEY
C137  A9 31              LDA #$31       ;KEY #
C139  A6 FB              LDX $FB        ;GET POINTER
C13B  9D D2 C2           STA INPUTB,X   ;STORE KEY #
C13E  E8                 INX
C13F  86 FB              STX $FB
C141  D0 11              BNE NOT4       ;ALWAYS
C143  98          NOTKEY TYA
C144  AA                 TAX
C145  A5 02              LDA $02
C147  29 80              AND #$80
C149  D0 03              BNE NOSW
C14B  A9 FF              LDA #$FF
C14D  2C                 .BYT $2C       ;SKIP NEXT
C14E  A9 00       NOSW   LDA #$00
```

```
LOC   CODE        _INE

C150  CA                DEX
C151  9D C3 C2          STA SWITCH,X
C154  C8          NOT4  INY              ;DO NEXT FOUR KEYS
C155  C0 10             CPY #$10         ;ALL DONE?
C157  F0 03             BEQ COMPLT       ;YES
C159  4C F1 C0          JMP INPUT
C15C              ;
C15C              ; SCANNING COMPLETE, TEST SLIDERS
C15C              ;
C15C  A2 01       COMPLT LDX #$01
C15E  AD 02 DC          LDA $DC02        ;GET CURRENT DDR
C161  8D B7 C2          STA DDRSAV
C164  A9 C0             LDA #$C0
C166  8D 02 DC          STA $DC02
C169  A9 80             LDA #$80         ;READ PORT B
C16B  8D 00 DC    READ  STA $DC00
C16E  A0 80             LDY #$80         ;PAUSE FOR READING
C170  EA          PAUSE NOP
C171  88                DEY
C172  10 FC             BPL PAUSE
C174  AD 19 D4          LDA $D419        ;GET POT 1
C177  9D BF C2          STA SLIDES,X
C17A  AD 1A D4          LDA $D41A        ;GET POT 2
C17D  9D C1 C2          STA SLIDES+2,X
C180  A9 40             LDA #$40         ;READ PORT A
C182  CA                DEX
C183  10 E6             BPL READ
C185  AD B7 C2          LDA DDRSAV       ;RESTORE DDR
C188  8D 02 DC          STA $DC02
C18B              ;
C18B              ; SLIDERS COMPLETE, PLAY NOTES
C18B              ; THAT HAVE BEEN READ.
C18B              ;
C18B  A6 FB             LDX $FB
C18D  E0 03       PAD   CPX #$03         ;3 KEYS AT LEAST?
C18F  B0 08             BCS ENOUGH       ;YES
C191  A9 00             LDA #$00
C193  9D D2 C2          STA INPUTB,X
C196  E8                INX
C197  D0 F4             BNE PAD          ;ALWAYS
C199              ;
C199  86 FB       ENOUGH STX $FB
C19B  20 3F C2          JSR CHKHLD       ;CHECK FOR HELD DOWN
C19E  AD BC C2          LDA V1ENAB       ;VOICE 1 ENABLED?
C1A1  D0 03             BNE DOVC2
C1A3  20 B9 C1          JSR PLAY1
C1A6  AD BD C2    DOVC2 LDA V2ENAB       ;VOICE 2 ENABLED?
C1A9  D0 03             BNE DOVC3
C1AB  20 DE C1          JSR PLAY2
C1AE  AD BE C2    DOVC3 LDA V3ENAB       ;VOICE 3 ENABLED?
C1B1  D0 03             BNE ALLDNE
C1B3  20 03 C2          JSR PLAY3
C1B6  4C 31 EA    ALLDNE JMP $EA31       ;NORMAL IRQ
C1B9              ;
C1B9              ; PLAY THE NOTE IN VOICE 1
C1B9              ;
C1B9  20 28 C2    PLAY1 JSR GETVCE       ;GET A NOTE
C1BC  8D B9 C2          STA VOICE1
C1BF  0A                ASL A
C1C0  A8                TAY
C1C1  AD B8 C2          LDA WAVE
C1C4  8D 04 D4          STA $D404        ;TURN GATE OFF
C1C7  98                TYA
C1C8  F0 13             BEQ PL1EXT
C1CA  B9 00 C0          LDA TABLE,Y      ;SET NEW NOTE
C1CD  8D 00 D4          STA $D400        ; VALUE
```

```
C1D0  B9 01 C0         LDA TABLE+1,Y
C1D3  8D 01 D4         STA $D401
C1D6  8E B8 C2         LDX WAVE
C1D9  E8               INX
C1DA  8E 04 D4         STX $D404      ;SET GATE ON
C1DD  60        PL1EXT RTS
C1DE            ;
C1DE            ; PLAY VOICE 2
C1DE            ;
C1DE  20 28 C2  PLAY2  JSR GETVCE     ;GET A NOTE
C1E1  8D BA C2         STA VOICE2
C1E4  0A               ASL A
C1E5  A8               TAY
C1E6  AD B8 C2         LDA WAVE
C1E9  8D 0B D4         STA $D40B      ;TURN GATE OFF
C1EC  98               TYA
C1ED  F0 13            BEQ PL2EXT
C1EF  B9 00 C0         LDA TABLE,Y    ;SET NEW NOTE
C1F2  8D 07 D4         STA $D407      ; VALUE
C1F5  B9 01 C0         LDA TABLE+1,Y
C1F8  8D 08 D4         STA $D408
C1FB  AE B8 C2         LDX WAVE
C1FE  E8               INX
C1FF  8E 0B D4         STX $D40B
C202  60        PL2EXT RTS
C203            ;
C203            ; PLAY VOICE 3
C203            ;
C203  20 28 C2  PLAY3  JSR GETVCE     ;GET A NOTE
C206  8D BB C2         STA VOICE3
C209  0A               ASL A
C20A  A8               TAY
C20B  AD B8 C2         LDA WAVE
C20E  8D 12 D4         STA $D412      ;TURN GATE OFF
C211  98               TYA
C212  F0 13            BEQ PL3EXT
C214  B9 00 C0         LDA TABLE,Y    ;SET NEW NOTE
C217  8D 0E D4         STA $D40E      ; VALUE
C21A  B9 01 C0         LDA TABLE+1,Y
C21D  8D 0F D4         STA $D40F
C220  AE B8 C2         LDX WAVE
C223  E8               INX
C224  8E 12 D4         STX $D412
C227  60        PL3EXT RTS
C228            ;
C228            ;GET A VALUE FROM THE INPUT
C228            ;BUFFER
C228            ;
C228  A0 00     GETVCE LDY #$00       ;LOOP START
C22A  B9 D2 C2  GET1   LDA INPUTB,Y   ;GET VALUE
C22D  F0 08            BEQ GETNXT
C22F  48               PHA
C230  A9 00            LDA #$00
C232  99 D2 C2         STA INPUTB,Y
C235  68               PLA
C236  60               RTS
C237            ;
C237  C8        GETNXT INY
C238  C4 FB            CPY $FB
C23A  D0 EE            BNE GET1
C23C  A9 00            LDA #$00
C23E  60               RTS
C23F            ;
C23F            ;CHECK ALL VOICES FOR KEYS HELD
C23F            ;DOWN
C23F            ;
C23F  AD BC C2  CHKHLD LDA V1ENAB
C242  29 7F            AND #$7F
C244  8D BC C2         STA V1ENAB
C247  AD BD C2         LDA V2ENAB
```

```
LOC    CODE          LINE

C24A   29 7F                AND #$7F
C24C   8D BD C2             STA V2ENAB
C24F   AD BE C2             LDA V3ENAB
C252   29 7F                AND #$7F
C254   8D BE C2             STA V3ENAB
C257   A0 00                LDY #$00
C259   B9 D2 C2      CHECK1 LDA INPUTB,Y
C25C   F0 05                BEQ NEXT1
C25E   CD B9 C2             CMP VOICE1
C261   F0 07                BEQ CONT1
C263   C8            NEXT1  INY
C264   C4 FB                CPY $FB
C266   D0 F1                BNE CHECK1
C268   F0 0D                BEQ CHCKV2
C26A   AD BC C2      CONT1  LDA V1ENAB
C26D   09 80                ORA #$80
C26F   8D BC C2             STA V1ENAB
C272   A9 00                LDA #$00
C274   99 D2 C2             STA INPUTB,Y
C277   A0 00         CHCKV2 LDY #$00
C279   B9 D2 C2      CHECK2 LDA INPUTB,Y
C27C   F0 05                BEQ NEXT2
C27E   CD BA C2             CMP VOICE2
C281   F0 07                BEQ CONT2
C283   C8            NEXT2  INY
C284   C4 FB                CPY $FB
C286   D0 F1                BNE CHECK2
C288   F0 0D                BEQ CHCKV3
C28A   AD BD C2      CONT2  LDA V2ENAB
C28D   09 80                ORA #$80
C28F   8D BD C2             STA V2ENAB
C292   A9 00                LDA #$00
C294   99 D2 C2             STA INPUTB,Y
C297   A0 00         CHCKV3 LDY #$00
C299   B9 D2 C2      CHECK3 LDA INPUTB,Y
C29C   F0 05                BEQ NEXT3
C29E   CD BB C2             CMP VOICE3
C2A1   F0 06                BEQ CONT3
C2A3   C8            NEXT3  INY
C2A4   C4 FB                CPY $FB
C2A6   D0 F1                BNE CHECK3
C2A8   60                   RTS
C2A9   AD BE C2      CONT3  LDA V3ENAB
C2AC   09 80                ORA #$80
C2AE   8D BE C2             STA V3ENAB
C2B1   A9 00                LDA #$00
C2B3   99 D2 C2             STA INPUTB,Y
C2B6   60                   RTS
C2B7   00            DDRSAV .BYT 0
C2B8   00            WAVE   .BYT 0
C2B9   00            VOICE1 .BYT 0
C2BA   00            VOICE2 .BYT 0
C2BB   00            VOICE3 .BYT 0
C2BC   00            V1ENAB .BYT 0
C2BD   00            V2ENAB .BYT 0
C2BE   00            V3ENAB .BYT 0
C2BF   00            SLIDES .BYT 0,0,0,0
C2C0   00
C2C1   00
C2C2   00
C2C3   00            SWITCH .BYT 0,0,0,0,0,0,0,0
C2C4   00
C2C5   00
C2C6   00
C2C7   00
C2C8   00
```

```
LOC   CODE       LINE

C2C9  00
C2CA  00
C2CB  00                  .BYT 0,0,0,0,0,0,0
C2CC  00
C2CD  00
C2CE  00
C2CF  00
C2D0  00
C2D1  00
C2D2  00         INPUTB .BYT 0
C2D3             .END

SYMBOL VALUE
 ALLDNE  C1B6    CHCKV2  C277    CHCKV3  C297    CHECK1  C259
 CHECK2  C279    CHECK3  C299    CHKHLD  C23F    COMPLT  C15C
 CONT1   C26A    CONT2   C28A    CONT3   C2A9    DDRSAV  C2B7
 DISABL  C0DB    DOVC2   C1A6    DOVC3   C1AE    ENABLE  C0C9
 ENOUGH  C199    GET1    C22A    GETNXT  C237    GETVCE  C228
 INPUT   C0F1    INPUTB  C2D2    LOOP    C084    NEXT1   C263
 NEXT2   C283    NEXT3   C2A3    NOSW    C14E    NOT1    C109
 NOT2    C11B    NOT3    C12D    NOT4    C154    NOTKEY  C143
 PAD     C18D    PAUSE   C170    PL1EXT  C1DD    PL2EXT  C202
 PL3EXT  C227    PLAY1   C1B9    PLAY2   C1DE    PLAY3   C203
 READ    C16B    SLIDES  C2BF    SWITCH  C2C3    TABLE   C000
 TESTER  C0ED    V1ENAB  C2BC    V2ENAB  C2BD    V3ENAB  C2BE
 VOICE1  C2B9    VOICE2  C2BA    VOICE3  C2BB    WAVE    C2B8
```

Program 27.

In addition to reading the keyboard to determine keys pressed, the same routine plays the relevant notes in the SID chip. There is also a routine within the scanner to read four potentiometers (slides) connected to the joystick ports. The slide values are stored in a four byte table at location $C2BF (49855).

The routine's playing section uses a table for the frequency values to be put into the SID chip. This table has values for 49 notes (a 49 key keyboard or 4 octaves). It must be noted, however, that the interface can handle 64 keys, so the extra 15 connections have been wired up to on/off switches and the code has been written to determine between keys and switches. The switch values are stored in a 15 byte table at location $C2C3 (49859).

The playing part of the routine is set up to play notes on all three voices of the SID chip taking into account any keys held down to make full use of the envelopes. This means that the attack/decay/sustain is working while the key is held down and when released the release part of the envelope starts. Any of the three voices of the SID chip can be disabled by putting a non zero (not 128) value into its disable register:

Voice 1 : $C2BC (49852)
Voice 2 : $C2BD (49853)
Voice 3 : $C2BE (49854)

This will allow these voices to be controlled by your program for such things as special effects.

5.2 Scanning the keyboard

The main principle for scanning the keyboard is to send a value (Ø–15) to the multiplexers which in turn tests the relevant line to the keys. The result is sent back as either a 1 or Ø on one of the four input lines (1 bit per multiplexer). Therefore, four keys are tested each time through the loop and the loop is repeated 16 times for a total of 64 keys. After receiving the results of one pass each bit is tested, and if it is 1 the value for that key is stored in an input buffer to be dealt with later. With the fourth multiplexer, only an output of zero is considered as a key; the other 15 connections are switches which are stored in the switch table (255 or Ø).

Once all 64 lines have been tested and the results stored away, the routine tests the four potentiometers (slides). This is performed using the routine given in *The Commodore 64 Programmer's Reference Guide* and the results are stored in the slide table.

5.3 Playing the notes read in

In this section of the routine, the first action is to test each of the three voices to see if the note is still being played. If the value is found in the table it is removed and the voice is temporarily disabled using the highest bit of the enable register. After all three voices have been checked for hold down, each voice is dealt with. If the enable register is non zero the voice is ignored, if the value is zero the gate is first turned off and the first non zero note in the table is removed and played. If no non zero note is available the gate is left off to allow the release to work. If a note is found the new value is looked up from the frequency table and stored in the relevant SID chip locations and the gate is turned on. Normal IRQ is then jumped to and the routine is repeated every 1/60th of a second.

The keyboard routine is first initialised by SYS(4928Ø). To disable the keyboard use SYS(49371) and to re-enable use SYS(49353).

5.4 Constructing the keyboard interface

The interface board uses four 7415Ø multiplexer chips, each of which is connected to 16 keys, one on each of its input lines. The 7415Ø chips are fed with a 4 bit address via a 7417 buffer driver (as shown in the circuit diagram in Fig. 5.1). This address is used to select one input key whose state is reflected on the multiplexer output line. The output line of each of the multiplexers is connected to one of the four lines used for input on the user port by organising the keyboard such that pressing a key connects one of the multiplexer input lines to ground. The four multiplexers are then scanned by outputting a value between Ø and 15 on the user port output lines. While simultaneously testing the four input lines it is possible to determine which key or keys are being pressed.

The keyboard interface is constructed on a single sided printed circuit board which is etched and drilled according to the pattern shown in Fig. 5.2. The

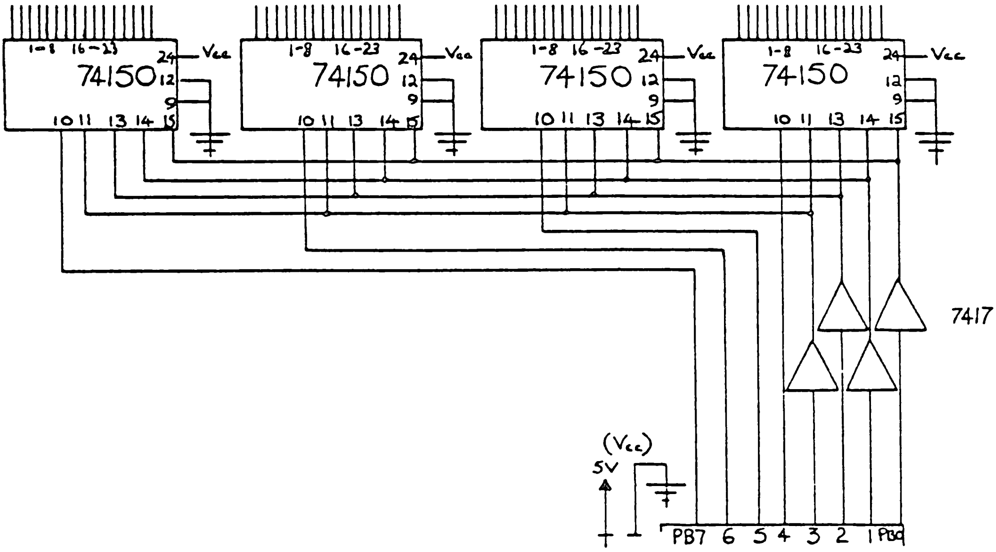


Fig. 5.1. Circuit diagram of keyboard interface.

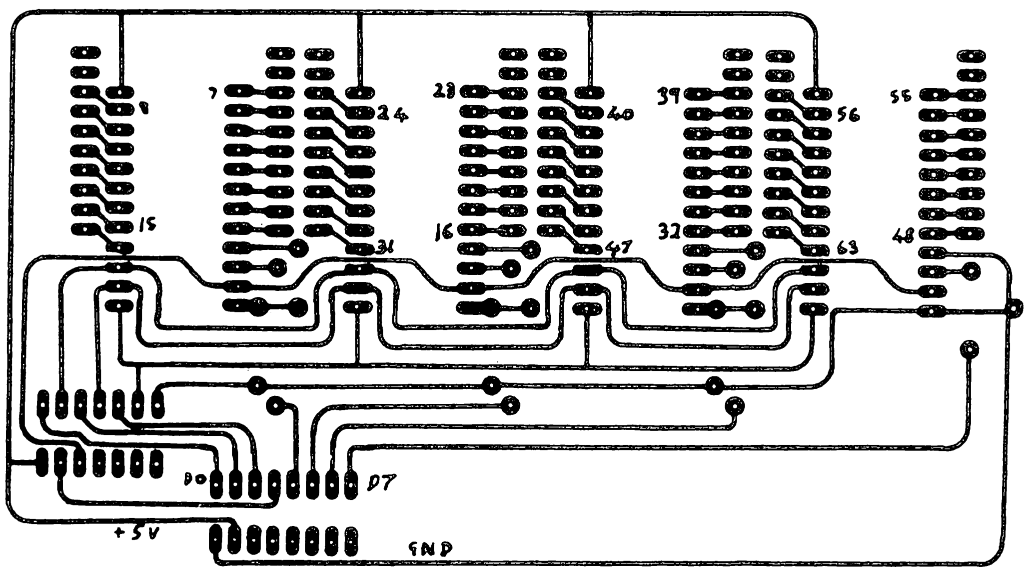


Fig. 5.2. Printed circuit board design.

components and wire links are attached as shown in Fig. 5.3. There are two cables which connect the keyboard interface board to the CBM 64, a 16 way ribbon cable connects the user port to the interface board and an 8 way cable connects the slider potentiometers to the joystick ports. The connections between the board and computer are shown in Fig. 5.4 which illustrates all the interconnections in detail.

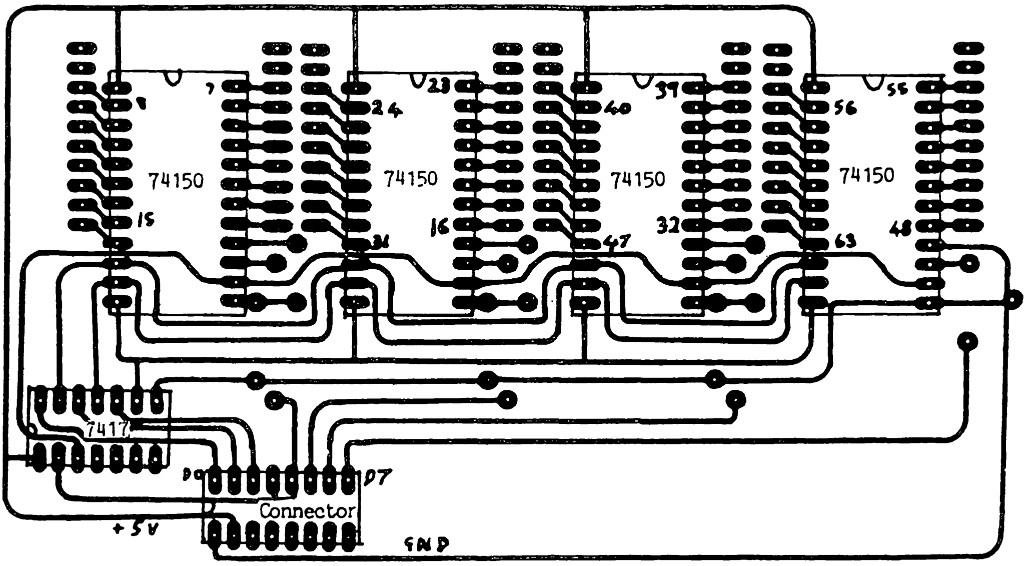


Fig. 5.3. Component layout for keyboard interface circuit (top view).

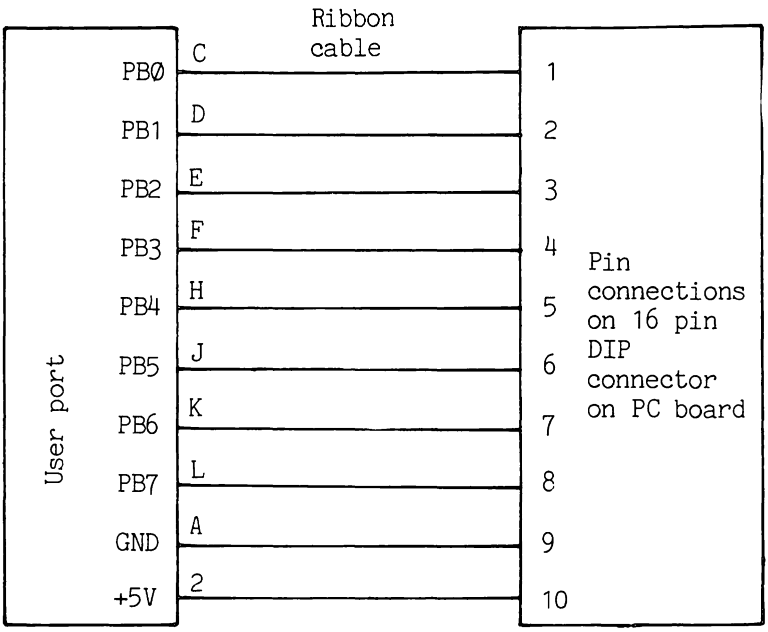


Fig. 5.4. Interconnection between user port and interface circuit.

5.5 Playing music in interrupts

In the majority of Commodore 64 games that are on the market some sort of background music is played. The routine in Program 28 plays music in

```
033C          !MUSIC IN INTERRUPTS.
033C          !*******************
033C          !
033C          !
033C          !
033C          !
033C          SID           =  54272
C000          *=$C000
C000 A018                   LDY #24
C002 A900                   LDA #$00
C004 9900D4 LOOP1           STA SID,Y             !ZERO SID
C007 88                     DEY
C008 10FA                   BPL LOOP1
C00A A90F                   LDA #$0F              !VOLUME
C00C 8D18D4                 STA SID+24
C00F A900                   LDA #$00              !ATTACK/DECAY
C011 8D05D4                 STA SID+5             !VOICE 1
C014 8D0CD4                 STA SID+12            !VOICE 2
C017 8D13D4                 STA SID+19            !VOICE 3
C01A A9F1                   LDA #$F1              !SUSTAIN/RELEASE
C01C 8D06D4                 STA SID+6             !VOICE 1
C01F 8D0DD4                 STA SID+13            !VOICE 2
C022 8D14D4                 STA SID+20            !VOICE 3
C025 A907                   LDA #$07
C027 8D10D4                 STA SID+16
C02A 8D0AD4                 STA SID+10
C02D A90F                   LDA #$0F
C02F 8D11D4                 STA SID+17
C032 8D09D4                 STA SID+9
C035 AD8FC0                 LDA TEMTMP            !TEMPO VALUE
C038 8D8AC1                 STA TEMPO
C03B A901                   LDA #$01              !DURATION
C03D 8D8BC1                 STA V1DUR
C040 8D8CC1                 STA V2DUR
C043 8D8DC1                 STA V3DUR
C046 A963                   LDA #<VOICE1          !START OF MUSIC V1
C048 8DD0C0                 STA V1PNT+1
C04B A9C2                   LDA #>VOICE1
C04D 8DD1C0                 STA V1PNT+2
C050 A963                   LDA #<VOICE2          !START OF MUSIC V2
C052 8D0EC1                 STA V2PNT+1
C055 A9C2                   LDA #>VOICE2
C057 8D0FC1                 STA V2PNT+2
C05A A9F0                   LDA #<VOICE3          !START OF MUSIC V3
C05C 8D4CC1                 STA V3PNT+1
C05F A9C1                   LDA #>VOICE3
C061 8D4DC1                 STA V3PNT+2
C064 A9F6                   LDA #<DURAT1          !VOICE 1 DURATION TBL
C066 8DC2C0                 STA V1TBL+1
C069 A9C3                   LDA #>DURAT1
C06B 8DC3C0                 STA V1TBL+2
C06E A9F6                   LDA #<DURAT2          !VOICE 2 DURATION TBL
C070 8D00C1                 STA V2TBL+1
C073 A9C3                   LDA #>DURAT2
C075 8D01C1                 STA V2TBL+2
C078 A984                   LDA #<DURAT3          !VOICE 3 DURATION TBL
C07A 8D3EC1                 STA V3TBL+1
C07D A9C3                   LDA #>DURAT3
C07F 8D3FC1                 STA V3TBL+2
C082 78                     SEI
C083 A993                   LDA #<PLAYER          !IRQ POINTER
C085 8D1403                 STA $0314
C088 A9C0                   LDA #>PLAYER
C08A 8D1503                 STA $0315
C08D 58                     CLI
C08E 60                     RTS
C08F          !
C08F 01       TEMTMP        BYT 1
C090 21       V1WAVE        BYT 33                !SAWTOOTH
```

```
C091 41      V2WAVE      BYT 65              !SAWTOOTH
C092 41      V3WAVE      BYT 65              !TRIANGLE
C093         !
C093         !IRQ PLAYER
C093         !
C093 CE8AC1 PLAYER       DEC TEMPO           !TIME TO UPDATE?
C096 F003                BEQ PLAY01          !YES
C098 4C31EA EXIT         JMP $EA31           !NO
C09B         !
C09B AD8FC0 PLAY01       LDA TEMTMP          !RESET TEMPO
C09E 8D8AC1              STA TEMPO
C0A1 CE8BC1              DEC V1DUR           !TIME FOR V1?
C0A4 D003                BNE PLAY02          !NO
C0A6 20BCC0              JSR DOV1            !YES, PLAY NEXT V1
C0A9 CE8CC1 PLAY02       DEC V2DUR           !TIME FOR V2?
C0AC D003                BNE PLAY03          !NO
C0AE 20FAC0              JSR DOV2            !YES, PLAY NEXT V2
C0B1 CE8DC1 PLAY03       DEC V3DUR           !TIME FOR V3?
C0B4 D0E2                BNE EXIT            !NO
C0B6 2038C1              JSR DOV3            !YES, PLAY NEXT V3
C0B9 4C31EA              JMP $EA31           !DO NORMAL IRQ
C0BC         !
C0BC         !PLAY VOICE 1
C0BC         !
C0BC A900   DOV1         LDA #$00
C0BE 8D04D4              STA SID+4           !GATE OFF
C0C1 ADF6C3 V1TBL        LDA DURAT1          !GET DURATION
C0C4 8D8BC1              STA V1DUR           !SET DURATION
C0C7 EEC2C0              INC V1TBL+1
C0CA D003                BNE V1PNT
C0CC EEC3C0              INC V1TBL+2
C0CF AD63C2 V1PNT        LDA VOICE1          !GET NOTE #
C0D2 EED0C0              INC V1PNT+1
C0D5 D003                BNE DOV12
C0D7 EED1C0              INC V1PNT+2
C0DA C900   DOV12        CMP #$00
C0DC F01B                BEQ DOV14
C0DE C9FF                CMP #$FF            !END OF MUSIC?
C0E0 D003                BNE DOV13
C0E2 4C76C1              JMP MUSEXT          !EXIT PLAYER
C0E5 0A     DOV13        ASL A
C0E6 AA                  TAX
C0E7 BD8EC1              LDA FREQTBL,X
C0EA 8D01D4              STA SID+1
C0ED BD8FC1              LDA FREQTBL+1,X
C0F0 8D00D4              STA SID
C0F3 AD90C0              LDA V1WAVE
C0F6 8D04D4              STA SID+4
C0F9 60     DOV14        RTS
C0FA         !
C0FA         !PLAY VOICE 2
C0FA         !
C0FA A900   DOV2         LDA #$00
C0FC 8D0BD4              STA SID+11          !GATE OFF
C0FF ADF6C3 V2TBL        LDA DURAT2          !GET DURATION
C102 8D8CC1              STA V2DUR           !SET DURATION
C105 EE00C1              INC V2TBL+1
C108 D003                BNE V2PNT
C10A EE01C1              INC V2TBL+2
C10D AD63C2 V2PNT        LDA VOICE2          !GET NOTE #
C110 EE0EC1              INC V2PNT+1
C113 D003                BNE DOV22
C115 EE0FC1              INC V2PNT+2
C118 C900   DOV22        CMP #$00
C11A F01B                BEQ DOV24
C11C C9FF                CMP #$FF            !END OF MUSIC?
C11E D003                BNE DOV23
C120 4C76C1              JMP MUSEXT          !EXIT PLAYER
C123 0A     DOV23        ASL A
```

```
C124 AA                TAX
C125 BD8EC1            LDA FREQTBL,X
C128 8D08D4            STA SID+8
C12B BD8FC1            LDA FREQTBL+1,X
C12E 8D07D4            STA SID+7
C131 AD91C0            LDA V2WAVE
C134 8D0BD4            STA SID+11
C137 60     DOV24      RTS
C138        !
C138        !PLAY VOICE 3
C138        !
C138 A900   DOV3       LDA #$00
C13A 8D12D4            STA SID+18              !GATE OFF
C13D AD84C3 V3TBL      LDA DURAT3              !GET DURATION
C140 8D8DC1            STA V3DUR               !SET DURATION
C143 EE3EC1            INC V3TBL+1
C146 D003              BNE V3PNT
C148 EE3FC1            INC V3TBL+2
C14B ADF0C1 V3PNT      LDA VOICE3              !GET NOTE #
C14E EE4CC1            INC V3PNT+1
C151 D003              BNE DOV32
C153 EE4DC1            INC V3PNT+2
C156 C900   DOV32      CMP #$00
C158 F01B              BEQ DOV34
C15A C9FF              CMP #$FF                !END OF MUSIC?
C15C D003              BNE DOV33
C15E 4C76C1            JMP MUSEXT              !EXIT PLAYER
C161 0A     DOV33      ASL A
C162 AA                TAX
C163 BD8EC1            LDA FREQTBL,X
C166 8D0FD4            STA SID+15
C169 BD8FC1            LDA FREQTBL+1,X
C16C 8D0ED4            STA SID+14
C16F AD92C0            LDA V3WAVE
C172 8D12D4            STA SID+18
C175 60     DOV34      RTS
C176        !
C176 68     MUSEXT     PLA
C177 68                PLA
C178 A931              LDA #$31
C17A 8D1403            STA $0314
C17D A9EA              LDA #$EA
C17F 8D1503            STA $0315
C182 A900              LDA #$00
C184 8D18D4            STA SID+24
C187 4C31EA            JMP $EA31
C18A        !
C18A 00     TEMPO      BYT 0
C18B 00     V1DUR      BYT 0
C18C 00     V2DUR      BYT 0
C18D 00     V3DUR      BYT 0
C18E        !
C18E 000003 FREQTBL    BYT 0,0,3,155,3,210,4,12
C196 044904            BYT 4,73,4,139,4,208,5,25
C19E 056705            BYT 5,103,5,185,6,16,6,108
C1A6 06CE07            BYT 6,206,7,53,7,163,8,23
C1AE 089309            BYT 8,147,9,21,9,159,10,60
C1B6 0ACD0B            BYT 10,205,11,114,12,32,12,216
C1BE 0D9C0E            BYT 13,156,14,107,15,70,16,47
C1C6 112512            BYT 17,37,18,42,19,63,20,100
C1CE 159A16            BYT 21,154,22,227,24,63,25,177
C1D6 1B381C            BYT 27,56,28,214,30,141,32,94
C1DE 224B24            BYT 34,75,36,85,38,126,40,200
C1E6 2B342D            BYT 43,52,45,198,48,127,51,97
C1EE 366F              BYT 54,111
C1F0        !
C1F0 00     VOICE3     BYT 0
C1F1 030003            BYT 3,0,3,0,3,0,3,0
C1F9 030003            BYT 3,0,3,0,3,0,3,0
```

```
C201 080C0F              BYT 8,12,15,17,18,17,15,12
C209 080C0F              BYT 8,12,15,17,18,17,15,12
C211 010508              BYT 1,5,8,10,11,10,8,5
C219 080C0F              BYT 8,12,15,17,18,17,15,12
C221 0F1613              BYT 15,22,19,22,13,20,17,20
C229 080C0F              BYT 8,12,15,17,18,17,15,12
C231 080C0F              BYT 8,12,15,17,18,17,15,12
C239 080C0F              BYT 8,12,15,17,18,17,15,12
C241 010508              BYT 1,5,8,10,11,10,8,5
C249 080C0F              BYT 8,12,15,17,18,17,15,12
C251 0F1613              BYT 15,22,19,22,13,20,17,20
C259 080C0F              BYT 8,12,15,17,18,17,15,12
C261 08FF                BYT 8,$FF
C263            !
C263 00         VOICE2   BYT 0
C263            *=*-1
C263 00         VOICE1   BYT 0
C264 1B001B              BYT 27,0,27,0,27,0,0,27
C26C 001B00              BYT 0,27,0,27,27,27,27,0
C274 140014              BYT 20,0,20,24,0,24,27,0,27,29,0,29
C280 1E001E              BYT 30,0,30,29,27,23,24,27,23
C289 181B                BYT 24,27
C28B 140014              BYT 20,0,20,24,0,24,27,0,27,29,0,29
C297 1E001E              BYT 30,0,30,29,27,23,24,27,23
C2A0 181B                BYT 24,27
C2A2 190019              BYT 25,0,25,29,0,29,32,0,32,34,0,34
C2AE 230023              BYT 35,0,35,34,32,28,29,32,28
C2B7 1D20                BYT 29,32
C2B9 140014              BYT 20,0,20,24,0,24,27,0,27,29,0,29
C2C5 1E001E              BYT 30,0,30,29,27,23,24,27,23
C2CE 181B                BYT 24,27
C2D0 1B001B              BYT 27,0,27,39,37,0,37,34
C2D8 190019              BYT 25,0,25,37,35,35,32,29
C2E0 140014              BYT 20,0,20,24,0,24,27,0,27,29,0,29
C2EC 1E001E              BYT 30,0,30,29,27,23,24,27,23
C2F5 181B                BYT 24,27
C2F7 140014              BYT 20,0,20,23,24,27
C2FD 140014              BYT 20,0,20,23,24,27
C303 140014              BYT 20,0,20,23,24,27
C309 140014              BYT 20,0,20,23,24,27
C30F 140014              BYT 20,0,20,23,24,27
C315 140014              BYT 20,0,20,23,24,27
C31B 140014              BYT 20,0,20,23,24,27
C321 140014              BYT 20,0,20,23,24,27
C327 190019              BYT 25,0,25,28,29,32
C32D 190019              BYT 25,0,25,28,29,32
C333 190019              BYT 25,0,25,28,29,32
C339 190019              BYT 25,0,25,28,29,32
C33F 140014              BYT 20,0,20,23,24,27
C345 140014              BYT 20,0,20,23,24,27
C34B 140014              BYT 20,0,20,23,24,27
C351 140014              BYT 20,0,20,23,24,27
C357 1B001B              BYT 27,0,27,30,31,34
C35D 1B001B              BYT 27,0,27,30,31,34
C363 190019              BYT 25,0,25,28,29,32
C369 190019              BYT 25,0,25,28,29,32
C36F 140014              BYT 20,0,20,23,24,27
C375 140014              BYT 20,0,20,23,24,27
C37B 140014              BYT 20,0,20,23,24,27
C381 1400FF              BYT 20,0,$FF
C384 60         DURAT3   BYT 96
C385 060606              BYT 6,6,6,6,6,6,6,6
C38D 060606              BYT 6,6,6,6,6,6,6,6
C395 0C0C0C              BYT 12,12,12,12,12,12,12,12
C39D 0C0C0C              BYT 12,12,12,12,12,12,12,12
C3A5 0C0C0C              BYT 12,12,12,12,12,12,12,12
C3AD 0C0C0C              BYT 12,12,12,12,12,12,12,12
C3B5 0C0C0C              BYT 12,12,12,12,12,12,12,12
C3BD 0C0C0C              BYT 12,12,12,12,12,12,12,12
```

```
C3C5 0C0C0C                     BYT 12,12,12,12,12,12,12,12
C3CD 0C0C0C                     BYT 12,12,12,12,12,12,12,12
C3D5 0C0C0C                     BYT 12,12,12,12,12,12,12,12
C3DD 0C0C0C                     BYT 12,12,12,12,12,12,12,12
C3E5 0C0C0C                     BYT 12,12,12,12,12,12,12,12
C3ED 0C0C0C                     BYT 12,12,12,12,12,12,12,12
C3F5 30                         BYT 48
C3F6             !
C3F6 00          DURAT2         BYT 0
C3F6             *=*-1
C3F6 60          DURAT1         BYT 96
C3F7 060606                     BYT 6,6,6,6,6,6,6,6
C3FF 060606                     BYT 6,6,6,6,6,6,6,6
C407 080103                     BYT 8,1,3,8,1,3,8,1,3,8,1,3
C413 080103                     BYT 8,1,3,9,3,4,4,4,4,4,4
C41E 080103                     BYT 8,1,3,8,1,3,8,1,3,8,1,3
C42A 080103                     BYT 8,1,3,9,3,4,4,4,4,4,4
C435 080103                     BYT 8,1,3,8,1,3,8,1,3,8,1,3
C441 080103                     BYT 8,1,3,9,3,4,4,4,4,4,4
C44C 080103                     BYT 8,1,3,8,1,3,8,1,3,8,1,3
C458 080103                     BYT 8,1,3,9,3,4,4,4,4,4,4
C463 080103                     BYT 8,1,3,6,9,3,6,12
C46B 080103                     BYT 8,1,3,6,9,3,6,12
C473 080103                     BYT 8,1,3,8,1,3,8,1,3,8,1,3
C47F 080103                     BYT 8,1,3,9,3,4,4,4,4,4,4
C48A 060303                     BYT 6,3,3,4,4,4,6,3,3,4,4,4
C496 060303                     BYT 6,3,3,4,4,4,6,3,3,4,4,4
C4A2 060303                     BYT 6,3,3,4,4,4,6,3,3,4,4,4
C4AE 060303                     BYT 6,3,3,4,4,4,6,3,3,4,4,4
C4BA 060303                     BYT 6,3,3,4,4,4,6,3,3,4,4,4
C4C6 060303                     BYT 6,3,3,4,4,4,6,3,3,4,4,4
C4D2 060303                     BYT 6,3,3,4,4,4,6,3,3,4,4,4
C4DE 060303                     BYT 6,3,3,4,4,4,6,3,3,4,4,4
C4EA 060303                     BYT 6,3,3,4,4,4,6,3,3,4,4,4
C4F6 060303                     BYT 6,3,3,4,4,4,6,3,3,4,4,4
C502 060303                     BYT 6,3,3,4,4,4,6,3,3,4,4,4
C50E 060303                     BYT 6,3,3,4,4,4,6,74
```

Program 28.

interrupts. The main IRQ routine can be standard but the initialisation and the actual tables have to be coded from sheet music (for the best results).

The following are the main principles behind the playing of music in interrupts:

1. The piece of music has an overall 'tempo' value which is a counter for how regularly the music IRQ is carried out. A value of 1 means each time a system IRQ occurs, 2 – every other system IRQ, etc.
2. Each voice has two reference tables. The first is the actual note number (one byte) which is used to look up the SID chip value from the frequency table. The second table is a note duration table and holds the value for how many music IRQ routines must be carried out before the next note is played. For example, if the tempo value is 2 and the note duration value is 6, the next note to be played on that voice will be after 12 IRQs have been called.

It is best to keep the duration values as low as possible. This is done by going through the music and finding the shortest note. The duration for that note is 1 and other notes have greater durations. When the duration table and the note table have been entered, the final adjustment required is to change the tempo value (to speed up or slow down). In most cases the tempo value would be less than 6.

In the demonstration piece (which is part of the full piece) the overall tempo value is set to 2. The reason for this is that in the music there is an unusual duration value for some notes (4/3rds) and so all duration values have to be multiplied by 3 to give whole numbers. This means that the tempo value must be decreased.

The coding of the piece of music is a very tedious job and requires at least some ability to read music. Each note must be written down along with its duration value for each voice to play. The notes must then be converted to numbers corresponding to the position in the frequency table. The frequency table below is set up in a two byte hi-lo order table and the values required follow.

Position	Note	Position	Note	Position	Note	Position	Note
1	A1	13	A2	25	A3	37	A4
2	A#1	14	A#2	26	A#3	38	A#4
3	B1	15	B2	27	B3	39	B4
4	C2	16	C3	28	C4	4Ø	C5
5	C#2	17	C#3	29	C#4	41	C#5
6	D2	18	D3	3Ø	D4	42	D5
7	D#2	19	D#3	31	D#4	43	D#5
8	E2	2Ø	E3	32	E4	44	E5
9	F2	21	F3	33	F4	45	F5
1Ø	F#2	22	F#3	34	F#4	46	F#5
11	G2	23	G3	35	G4	47	G5
12	G#2	24	G#3	36	G#4	48	G#5

Position Ø causes the gate to be turned off and no change made to the frequency (setting the release in action).

This table was made up by finding out the highest and lowest notes to be played and including all notes in the range. The table could be decreased in size by containing just the notes that are played.

5.6 Music IRQ operation

The actual IRQ routine first decrements the tempo value and, if not zero, the normal IRQ is done. If the tempo value is down to zero the music IRQ routine is called. In this routine, the tempo value is reset and the duration value for each voice is decremented. If it reaches zero a subroutine is called to put the next value into the voice and set up the next note's duration. If the next note found for any of the voices is $FF, a routine to reset the IRQ vector to normal is jumped to and the music ends. After all three voice durations have been tested the normal IRQ routine is called.

Note: The total of durations for each voice should be equal so that all voices end at the same time.

Appendix A

Sprite Editor

Program 29 is a simple sprite editor for the CBM 64. It allows adding and deleting dots, in either multicolour or standard sprites.

```
1000 REM ********SPRITE EDITOR********
1010 REM  FOR MULTI COLOUR OR STANDARD
1020 REM  SPRITES.
1030 REM ******************************
1040 DEF FNS(Z)=32763+(R+1)*40+C+1
1050 DEF FNC(Z)=FNS(Z)+22528
1060 SP=0:POKE2,0:VIC=53248:POKE650,128
1070 DIM P2(7):FORI=0TO7:P2(I)=2↑I:NEXT
1080 CD$="▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓"
1090 GOSUB2630: REM PUT IN MACHINE CODE
1100 REM ******************************
1110 REM       PROTECT MEMORY
1120 REM ******************************
1130 POKE56,128:POKE54,128:POKE52,128
1140 REM ******************************
1150 REM SELECT BANK 2
1160 REM ******************************
1170 POKE56578,PEEK(56578)OR3
1180 POKE56576,(PEEK(56576)AND252)OR1
1190 POKE53272,PEEK(53272)AND15:POKE648,128
1200 PRINT"▓▓":POKE53281,11:POKE53280,9
1210 GOSUB2070: REM DO YOU WANT TO LOAD
1220 REM ******************************
1230 REM SET VIC CHIP SETTINGS
1240 REM ******************************
1250 POKEVIC+21,1:POKEVIC,240:POKEVIC+1,59
1260 POKEVIC+16,0:POKEVIC+23,1:POKEVIC+29,1
1270 POKEVIC+39,7:POKEVIC+37,0:POKEVIC+38,0
1280 POKEVIC+28,0:GOSUB1790
1290 REM ******************************
1300 REM MAIN INPUT LOOP FOR COMMANDS
1310 REM ******************************
1320 POKEFNC(0),0
1330 GETA$:IFA$=""THEN1330
1340 POKEFNC(0),1
1350 IFA$="+"THEN1570
1360 IFA$="-"THEN1640
1370 IFA$="▓"THENR=R+1:IFR=21THENR=0:GOTO1320
1380 IFA$="]"THENR=R-1:IFR=-1THENR=20:GOTO1320
1390 IFA$="▌"THENC=C+1:IFC=24THENC=0:GOTO1320
1400 IFA$="▐"THENC=C-1:IFC=-1THENC=23:GOTO1320
1410 IFA$="N"THENSP=(SP+1)AND31:GOSUB2020:GOTO1320
1420 IFA$="Q"THEN2220
1430 IFA$="M"THENPOKEVIC+28,PEEK(VIC+28)OR1:GOTO1320
1440 IFA$="O"THENPOKEVIC+28,PEEK(VIC+28)AND254:GOTO1320
1450 IFA$="S"THEN1710:REM SELECT
1460 IFA$="■"THENPOKE53281,(PEEK(53281)+1)AND15:GOTO1320
1470 IFA$="▀"THENPOKE53280,(PEEK(53280)+1)AND15:GOTO1320
1480 IFA$="▐"THENPOKEVIC+39,(PEEK(VIC+39)+1)AND15:GOTO1320
1490 IFA$="▌"THENPOKEVIC+37,(PEEK(VIC+37)+1)AND15:GOTO1320
1500 IFA$="■"THENPOKEVIC+38,(PEEK(VIC+38)+1)AND15:GOTO1320
1510 IFA$="X"THENPOKEVIC+29,(PEEK(VIC+29)+1)AND1:GOTO1320
```

```
1520 IFA$="Y"THENPOKEVIC+23,(PEEK(VIC+23)+1)AND1:GOTO1320
1530 IFA$="□"THENFORI=0TO63:POKE32768+(SP+16)*64+I,0:NEXT:GOSUB2020:A$="■"
1540 IFA$="■"THENR=0:C=0:GOTO1320
1550 GOTO1320
1560 REM ******************************
1570 REM ADD A POINT TO THE SPRITE
1580 REM ******************************
1590 AD=32768+R*3+INT(C/8)+64*(SP+16)
1600 POKEAD,PEEK(AD)ORP2(7-(CAND7))
1610 POKEFNS(0),81
1620 A$="■":GOTO1390
1630 REM ******************************
1640 REM DELETE A POINT FROM THE SPRITE
1650 REM ******************************
1660 AD=32768+R*3+INT(C/8)+64*(SP+16)
1670 POKEAD,PEEK(AD)AND(255-P2(7-(CAND7)))
1680 POKEFNS(0),46
1690 A$="■":GOTO1390
1700 REM ******************************
1710 REM SELECT A SPRITE TO DISPLAY
1720 REM ******************************
1730 OPEN1,0
1740 PRINTCD$;"  SPRITE NUMBER (0-31) ";:INPUT#1,SP$:CLOSE1
1750 PRINTCD$"                                    ■■■■■■ ";
1760 SP=VAL(SP$):IFSP>31ORSP<0THEN1710
1770 GOSUB2020:GOTO1320
1780 REM ******************************
1790 REM      PUT UP DISPLAY
1800 REM ******************************
1810 PRINT"□┌────────────────────────┐          SPRITE";
1820 FORI=1TO21
1830 PRINT"|........................|"
1840 NEXT
1850 PRINT"└────────────────────────┘"
1860 PRINT"■■■■■■■"SPC(27)"■CONTROLS:"
1870 PRINTSPC(27)"■+■ ADD DOT    "
1880 PRINTSPC(27)"■-■ DELETE DOT"
1890 PRINTSPC(27)"■X■ EXPAND     "
1900 PRINTSPC(27)"■Y■ EXPAND     "
1910 PRINTSPC(27)"■M■ MULTI ON   "
1920 PRINTSPC(27)"■O■ MULTI OFF "
1930 PRINTSPC(27)"■S■ SELECT     "
1940 PRINTSPC(27)"■N■ NEXT       "
1950 PRINTSPC(27)"■Q■ QUIT       "
1960 PRINTSPC(29)"COLOURS:  "
1970 PRINTSPC(27)"■F1■ SCREEN    "
1980 PRINTSPC(27)"■F3■ BORDER    "
1990 PRINTSPC(27)"■F5■ SPRITE    "
2000 PRINTSPC(27)"■F7■ M-C 1     "
2010 PRINTSPC(27)"■F8■ M-C 2     ■"
2020 POKE33784,SP+16
2030 PRINT"■■"SPC(34)"■#"RIGHT$("    "+STR$(SP),4)
2040 SYS49152,SP+16
2050 R=0:C=0:RETURN
2060 REM ******************************
2070 REM LOAD A SPRITE FILE FROM TAPE
2080 REM  OR DISK.
2090 REM ******************************
2100 POKEVIC+21,0:PRINT"□              SPRITE EDITOR"
2110 PRINT"■■DO YOU WISH TO LOAD A SPRITE FILE ?";
2120 GETA$:IFA$<>"Y"ANDA$<>"N"THEN2120
2130 IFA$="N"THENRETURN
2140 PRINT"Y":GOSUB2480:REM SET UP FILE DETAILS
2150 POKE780,0:SYS(65493)
2160 IF(PEEK(783)AND1)=0THENRETURN
2170 PRINT"■■■LOAD ERROR■ DO YOU WANT TO TRY AGAIN?"
2180 GETA$:IFA$<>"Y"ANDA$<>"N"THEN2180
2190 IFA$="N"THENRETURN
```

```
2200 GOTO2070
2210 REM ******************************
2220 REM END PROGRAM AND SAVE SPRITES
2230 REM ******************************
2240 POKEVIC+21,0:PRINT"▯DO YOU WISH TO SAVE THE SPRITES ?";
2250 GETA$:IFA$<>"Y"ANDA$<>"N"THEN2250
2260 IFA$="N"THEN2420
2270 PRINT"Y":PRINT"▯▯ENTER RANGE OF SPRITES:"
2280 PRINT"▯START SPRITE NUMBER:";:OPEN1,0:INPUT#1,SS$:CLOSE1:PRINT
2290 SS=VAL(SS$):IFSS<0ORSS>31THENPRINT"▯▯▯":GOTO2280
2300 PRINT"▯END SPRITE NUMBER:";:OPEN1,0:INPUT#1,ES$:CLOSE1:PRINT
2310 ES=VAL(ES$):IFES<0ORES>31THENPRINT"▯▯▯":GOTO2300
2320 IFSS>ESTHENPRINT"▯▯▯▯▯":GOTO2280
2330 SS=(SS+16)*64+32768:ES=(ES+17)*64+32768
2340 POKE252,INT(SS/256):POKE251,SS-INT(SS/256)*256
2350 POKE780,251:POKE782,INT(ES/256):POKE781,ES-INT(ES/256)*256
2360 GOSUB2480
2370 SYS(65496):REM SAVE
2380 IFST=0THEN2420
2390 PRINT"▯▯▯SAVE ERROR▯ DO YOU WANT TO TRY AGAIN ?"
2400 GETA$:IFA$<>"Y"ANDA$<>"N"THEN2400
2410 IFA$="Y"THEN2220
2420 POKE53280,14:POKE53281,6
2430 POKE648,4:PRINT"▯▯"
2440 POKEVIC+24,(PEEK(VIC+24)AND15)OR16
2450 POKE56578,PEEK(56578)AND252
2460 END
2470 REM ******************************
2480 REM GET FILE DETAILS FOR LOAD
2490 REM AND SAVE
2500 REM ******************************
2510 PRINT"▯▯PLEASE ENTER FILENAME:";
2520 OPEN1,0:INPUT#1,FM$:CLOSE1:PRINT
2530 PRINT"▯▯▯T▯APE OR ▯D▯ISK:";
2540 GETA$:IFA$<>"T"ANDA$<>"D"THEN2540
2550 PRINTA$:D=8:IFA$="T"THEND=1
2560 IFLEN(FM$)>16THENFM$=LEFT$(FM$,16)
2570 POKE183,LEN(FM$)
2580 POKE186,D:POKE185,D:POKE184,D
2590 POKE188,2:POKE187,192
2600 FORI=1TOLEN(FM$):POKE703+I,ASC(MID$(FM$,I,1)):NEXT
2610 RETURN
2620 REM ******************************
2630 REM STORE MACHINE CODE INTO
2640 REM MEMORY
2650 REM ******************************
2660 I=49152
2670 READA:IFA=-1THEN2700
2680 POKEI,A:I=I+1
2690 T=T+A:GOTO2670
2700 IFT<>14011THENPRINT"▯DATA ERROR":END
2710 RETURN
2720 REM ******************************
2730 REM DATA FOR MACHINE CODE SPRITE
2740 REM DISPLAY
2750 REM ******************************
2760 DATA32,253,174,32,158,183,134
2770 DATA251,169,0,133,252,160,5
2780 DATA6,251,38,252,136,16,249
2790 DATA165,252,105,128,133,252,32
2800 DATA102,229,169,17,32,210,255
2810 DATA160,0,169,29,32,210,255
2820 DATA32,64,192,200,32,64,192
2830 DATA200,32,64,192,200,169,13
2840 DATA32,210,255,192,63,208,230
2850 DATA96,177,251,133,254,169,128
2860 DATA133,253,37,254,240,8,169
2870 DATA209,32,210,255,76,89,192
2880 DATA169,46,32,210,255,70,253
2890 DATA165,253,208,233,96,-1
```

Program 29.

All colour combinations (except text) can be changed at the press of a button. In total, 32 sprites may be edited/created using this program and they can then be saved to disk or tape and reloaded to be edited further or for the final program.

For speed, a short machine code program is first poked into memory which displays the enlarged sprite grid onto the screen. All editing commands available are displayed on the screen continuously, and when you have pressed 'Q' to quit you will be asked if you wish to save the sprites. The sprites are stored as memory (not as a data file) and are located at $8400.

Appendix B

Character Editor

This character editor (Program 3∅) allows the editing of 128 characters in the Commodore 64. The characters may be standard, multicolour, or a mixture of both. The characters are edited on a large 8 by 8 grid and all characters are

```
1000 REM ******CHARACTER EDITOR*******
1010 REM  FOR MULTICOLOUR OR STANDARD
1020 REM  CHARACTERS.
1030 REM ******************************
1040 DEF FNS(Z)=32768+(R+1)*40+C+1
1050 DEF FNC(Z)=FNS(Z)+22528
1060 CH=0:CO=1:POKE2,0:VIC=53248:POKE650,128
1070 DIM P2(7):FORI=0TO7:P2(I)=2↑I:NEXT
1080 CD$="■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
1090 GOSUB2550: REM PUT IN CHARACTERS
1100 REM ******************************
1110 REM       PROTECT MEMORY
1120 REM ******************************
1130 POKE56,128:POKE54,128:POKE52,128
1140 REM ******************************
1150 REM SELECT BANK 2
1160 REM ******************************
1170 POKE56578,PEEK(56578)OR3
1180 POKE56576,(PEEK(56576)AND252)OR1
1190 POKE53272,2:POKE648,128
1200 PRINT"■":POKE53281,11:POKE53280,9
1210 GOSUB2040: REM DO YOU WANT TO LOAD
1220 GOSUB1710
1230 REM ******************************
1240 REM MAIN INPUT LOOP FOR COMMANDS
1250 REM ******************************
1260 POKEFNC(0),0
1270 GETA$:IFA$=""THEN1270
1280 POKEFNC(0),1
1290 IFA$="+"THEN1490
1300 IFA$="-"THEN1560
1310 IFA$="■"THENR=R+1:IFR=8THENR=0:GOTO1260
1320 IFA$="■"THENR=R-1:IFR=-1THENR=7:GOTO1260
1330 IFA$="■"THENC=C+1:IFC=8THENC=0:GOTO1260
1340 IFA$="■"THENC=C-1:IFC=-1THENC=7:GOTO1260
1350 IFA$="N"THENCH=(CH+1)AND127:GOSUB1960:GOTO1260
1360 IFA$="Q"THEN2190
1370 IFA$="M"THENPOKEVIC+22,PEEK(VIC+22)OR16:GOTO1260
1380 IFA$="O"THENPOKEVIC+22,PEEK(VIC+22)AND239:GOTO1260
1390 IFA$="S"THEN1630:REM SELECT
1400 IFA$="■"THENPOKE53281,(PEEK(53281)+1)AND15:GOTO1260
1410 IFA$="■"THENPOKE53280,(PEEK(53280)+1)AND15:GOTO1260
1420 IFA$="■"THEN2650: REM CHANGE CHARACTER COLOURS
1430 IFA$="■"THENPOKEVIC+34,(PEEK(VIC+34)+1)AND15:GOTO1260
1440 IFA$="■"THENPOKEVIC+35,(PEEK(VIC+35)+1)AND15:GOTO1260
1450 IFA$="■"THENFORI=0TO7:POKE34816+CH*8+I,0:NEXT:GOSUB1960:A$="■"
1460 IFA$="■"THENR=0:C=0:GOTO1260
1470 GOTO1260
1480 REM ******************************
1490 REM ADD A POINT TO THE CHARACTER
1500 REM ******************************
```

```
1510 AD=34816+R+8*CH
1520 POKEAD,PEEK(AD)ORP2(7-(CAND7))
1530 POKEFNS(0),81+128
1540 A$="▶▌":GOTO1330
1550 REM ******************************
1560 REM DELETE A POINT FROM THE CHAR
1570 REM ******************************
1580 AD=34816+R+8*CH
1590 POKEAD,PEEK(AD)AND(255-P2(7-(CAND7)))
1600 POKEFNS(0),46+128
1610 A$="▶▌":GOTO1330
1620 REM ******************************
1630 REM SELECT A CHAR TO DISPLAY
1640 REM ******************************
1650 OPEN1,0
1660 PRINTCD$;"▚CHAR NUMBER (0-127) ";:INPUT#1,CH$:CLOSE1
1670 PRINTCD$"                          ▚"
1680 CH=VAL(CH$):IFCH>127ORCH<0THEN1630
1690 GOSUB1960:GOTO1260
1700 REM ******************************
1710 REM       PUT UP DISPLAY
1720 REM ******************************
1730 PRINT"▣":FORI=0TO999:POKE32768+I,160:NEXT
1740 PRINT"▚■▚┌────────┐            ■┌─┐▚        CHARACTER"
1750 FORI=1TO8
1760 PRINT"■▚|▚........■|"
1770 NEXT
1780 PRINT"■▚└────────┘"
1790 PRINT"▚▚▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶|| |"
1800 PRINT"▚▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶▶|└─┘"
1810 PRINT"▚=▚▚▚▚▚▚▚"SPC(20)"▚CONTROLS:"
1820 PRINTSPC(20)"▚+ ADD DOT    "
1830 PRINTSPC(20)"▚- DELETE DOT"
1840 PRINTSPC(20)"▚M MULTI ON   "
1850 PRINTSPC(20)"▚O MULTI OFF "
1860 PRINTSPC(20)"▚S SELECT     "
1870 PRINTSPC(20)"▚N NEXT       "
1880 PRINTSPC(20)"▚Q QUIT       "
1890 PRINTSPC(22)"▚COLOURS:  "
1900 PRINTSPC(20)"▚F1 SCREEN    "
1910 PRINTSPC(20)"▚F3 BORDER    "
1920 PRINTSPC(20)"▚F5 CHARACTER"
1930 PRINTSPC(20)"▚F7 M-C 1     "
1940 PRINTSPC(20)"▚F8 M-C 2    ▚"
1950 GOSUB2720
1960 PRINT"▚▚"SPC(29)"▚NUMBER"RIGHT$("     "+STR$(CH),4)
1970 FORR=0TO7:A=PEEK(34816+R+CH*8)
1980 FORC=0TO7
1990 IF(AANDP2(7-C))=0THENPOKEFNS(0),46+128:GOTO2010
2000 POKEFNS(0),81+128
2010 NEXT:NEXT:POKE32828,CH:POKE55356,CO
2020 R=0:C=0:RETURN
2030 REM ******************************
2040 REM LOAD A CHARACTER FILE FROM
2050 REM  TAPE OR DISK.
2060 REM ******************************
2070 PRINTCHR$(8)"▣▚            CHARACTER EDITOR            "
2080 PRINT"▚▚▚DO YOU WISH TO LOAD A CHARACTER FILE ?";
2090 GETA$:IFA$<>"Y"ANDA$<>"N"THEN2090
2100 IFA$="N"THENRETURN
2110 PRINT"Y":GOSUB2400:REM SET UP FILE DETAILS
2120 POKE780,0:SYS(65493)
2130 IF(PEEK(783)AND1)=0THENRETURN
2140 PRINT"▚▚▚LOAD ERROR DO YOU WANT TO TRY AGAIN?"
2150 GETA$:IFA$<>"Y"ANDA$<>"N"THEN2150
2160 IFA$="N"THENRETURN
2170 GOTO2040
2180 REM ******************************
2190 REM END PROGRAM AND SAVE CHARS
```

```
2200 REM ******************************
2210 PRINT"DO YOU WISH TO SAVE THE CHARACTERS ?";
2220 GETA$:IFA$<>"Y"ANDA$<>"N"THEN2220
2230 IFA$="N"THEN2340
2240 PRINTA$
2250 SS=34816:ES=35840
2260 POKE252,INT(SS/256):POKE251,SS-INT(SS/256)*256
2270 POKE780,251:POKE782,INT(ES/256):POKE781,ES-INT(ES/256)*256
2280 GOSUB2400
2290 SYS(65496):REM SAVE
2300 IFST=0THEN2340
2310 PRINT"SAVE ERROR DO YOU WANT TO TRY AGAIN ?"
2320 GETA$:IFA$<>"Y"ANDA$<>"N"THEN2320
2330 IFA$="Y"THEN2190
2340 POKE53280,14:POKE53281,6
2350 POKE648,4:PRINT""
2360 POKEVIC+24,21:POKEVIC+22,200
2370 POKE56578,PEEK(56578)AND252
2380 END
2390 REM ******************************
2400 REM GET FILE DETAILS FOR LOAD
2410 REM AND SAVE
2420 REM ******************************
2430 PRINT"PLEASE ENTER FILENAME:";
2440 OPEN1,0:INPUT#1,FM$:CLOSE1:PRINT
2450 PRINT"TAPE OR DISK:";
2460 GETA$:IFA$<>"T"ANDA$<>"D"THEN2460
2470 PRINTA$:D=8:IFA$="T"THEND=1
2480 IFLEN(FM$)>16THENFM$=LEFT$(FM$,16)
2490 POKE183,LEN(FM$)
2500 POKE186,D:POKE185,D:POKE184,D
2510 POKE188,2:POKE187,192
2520 FORI=1TOLEN(FM$):POKE703+I,ASC(MID$(FM$,I,1)):NEXT
2530 RETURN
2540 REM ******************************
2550 REM COPY CHARACTERS INTO RAM FOR
2560 REM  EDITING
2570 REM ******************************
2580 A=34816:B=35840:POKE56334,PEEK(56334)AND254
2590 POKE1,PEEK(1)AND251
2600 FORI=53248TO54272
2610 POKEA,PEEK(I):POKEB,PEEK(I):A=A+1:B=B+1:NEXT
2620 POKE1,PEEK(1)OR4
2630 POKE56334,PEEK(56334)OR1
2640 RETURN
2650 REM ******************************
2660 REM CHANGE COLOUR OF USER DEF
2670 REM  CHARACTERS
2680 REM ******************************
2690 CO=(CO+1)AND15:POKE55356,CO
2700 GOSUB2800:GOTO1260
2710 REM ******************************
2720 REM PUT UP CHARACTER DISPLAY
2730 REM ******************************
2740 PRINT"   0123456789 "
2750 PRINT"  ------------"
2760 FORI=0TO11
2770 PRINT""RIGHT$("   "+STR$(I),2);"|          |"
2780 NEXT
2790 PRINT"12|          ";
2800 FORRR=0TO11
2810 FORCC=0TO9
2820 POKE33251+RR*40+CC,RR*10+CC
2830 POKE55779+RR*40+CC,CO
2840 NEXT:NEXT:FORI=0TO7
2850 POKE33731+I,I+120:POKE56259+I,CO
2860 NEXT:RETURN
```

Program 30.

displayed on the screen. The character, like the sprites in Appendix A, can be loaded and saved as blocks of memory. The saved characters reside in memory at locations $88ØØ–$9ØØØ and are the full 128 characters.

Editing is performed with the same commands as for the sprite editor, with the exception of expanding.

Appendix C
Sound Editor

The sound editor in Program 31 is designed to allow you to experiment with the SID chip settings to find out what sounds good and what doesn't.

```
1000 REM ******************************
1010 REM ** SOUND EFFECTS EDITOR -64 **
1020 REM ******************************
1030 REM
1040 REM ZERO THE SID CHIP
1050 REM
1060 SID=54272:FORI=0TO24:POKESID+I,0:NEXT
1070 DIM K%(64),FL%(33),FH%(33)
1080 FORI=0TO64:READK%(I):NEXT:REM GET KEY TABLE
1090 FORI=0TO33:READFL%(I):NEXT:REM GET FREQUENCY TABLE LOW
1100 FORI=0TO33:READFH%(I):NEXT:REM GET FREQUENCY TABLE HIGH
1110 REM
1120 REM SET INITIAL VALUES
1130 REM
1140 AT=0:DE=9:SU=0:RE=0:03=0:FF=0:FR=0:FP$=" OFF":S$="NOTHING":WV$="PULSE    "
1150 FP=0:POKE SID+5,9:POKE SID+6,0
1160 POKE SID+24,15
1170 WV=64:PW=2048:POKESID+3,PW/256:POKESID+2,PW-INT(PW/256)*256:GOSUB2470
1180 REM
1190 REM MAIN LOOP STARTS HERE WITH
1200 REM KEYBOARD INPUT
1210 REM
1220 K=PEEK(197):IFK=LKTHEN1220
1230 LK=K:POKE SID+4,WV
1240 IFK%(K)=64 THEN 1220
1250 IFK%(K)>64 THEN 1380
1260 REM
1270 REM CHANGE FREQUENCY SETTINGS FOR
1280 REM NEW KEY
1290 REM
1300 POKE SID,FL%(K%(K))
1310 POKE SID+1,FH%(K%(K))
1320 POKE SID+4,WV+1
1330 GOTO 1220
1340 REM
1350 REM EITHER A FUNCTION KEY OR
1360 REM RETURN
1370 REM
1380 ON (K%(K)-64) GOTO 1430,1480,1530,1580,1640
1390 STOP
1400 REM
1410 REM F1
1420 REM
1430 AT=(AT+1)AND15:POKESID+5,AT*16+DE
1440 GOTO1590
1450 REM
1460 REM F3
1470 REM
1480 DE=(DE+1)AND15:POKESID+5,AT*16+DE
1490 GOTO1590
1500 REM
1510 REM F5
```

```
1520 REM
1530 SU=(SU+1)AND15:POKESID+6,SU*16+RE
1540 GOTO1590
1550 REM
1560 REM F7
1570 REM
1580 RE=(RE+1)AND15:POKESID+6,SU*16+RE
1590 LK=64:GOSUB2680:GOTO1220
1600 REM
1610 REM RETURN KEY, CHOOSE OTHER
1620 REM VALUES TO CHANGE
1630 REM
1640 POKE198,0:PRINT"■QQQQQQQQQQQQQQQQQQQQQ";
1650 PRINT"ENTER REQUIRED NUMBER:";:OPEN2,0:INPUT#2,Q$:CLOSE2
1660 PRINT:PRINT"]                                          "
1670 Q=INT(VAL(Q$)):IFQ<1ORQ>7THEN1640
1680 ONQGOTO1730,1800,1940,2050,2150,2250,2350
1690 STOP
1700 REM
1710 REM 1, SPECIAL EFFECT
1720 REM
1730 PRINT"]■R■ING MOD, ■S■YNC, ■N■OTHING"
1740 NEWA$:IFA$<>"N"ANDA$<>"S"ANDA$<>"R"THEN1740
1750 PRINT"]                      ";
1760 IFA$="N"THENS$="NOTHING":WV=WVAND240:GOTO1790
1770 IFA$="S"THENS$="  SYNC  ":WV=(WVAND240)OR2:GOTO1790
1780 S$="  RING  ":WV=(WVAND240)OR4
1790 POKESID+4,WV:GOSUB2680:GOTO1220
1800 PRINT"]ENTER NEW VALUE FOR OSC3 (0-65535)"
1810 REM
1820 REM 2, OSC 3 SETTING USED IN
1830 REM CONJUNCTION WITH #1
1840 REM
1850 OPEN2,0:INPUT#2,O3$:PRINT:CLOSE2
1860 PRINT"]]                                      "
1870 PRINT"                                          "
1880 O3=INT(VAL(O3$)):IFO3<0ORO3>65535THENPRINT"]";:GOTO1800
1890 POKESID+15,O3/256:POKESID+14,O3-INT(O3/256)*256
1900 GOSUB2680:GOTO1220
1910 REM
1920 REM 3, CHANGE WAVEFORM
1930 REM
1940 PRINT"]■T■RIANGLE, ■S■AWTOOTH, ■P■ULSE, ■N■OISE"
1950 NEWA$:IFA$<>"N"ANDA$<>"P"ANDA$<>"S"ANDA$<>"T"THEN1950
1960 PRINT"]                          "
1970 IFA$="T"THENWV$="TRIANGLE":WV=(WVAND15)OR16:GOTO2010
1980 IFA$="S"THENWV$="SAWTOOTH":WV=(WVAND15)OR32:GOTO2010
1990 IFA$="P"THENWV$="PULSE   ":WV=(WVAND15)OR64:GOTO2010
2000 WV$="NOISE   ":WV=(WVAND15)OR128
2010 GOSUB2680:GOTO1220
2020 REM
2030 REM 4, CHANGE PULSE WIDTH
2040 REM
2050 PRINT"]ENTER NEW PULSE WIDTH (0-4095)"
2060 OPEN2,0:INPUT#2,PW$:PRINT:CLOSE2
2070 PRINT"]]                                "
2080 PRINT"                                          "
2090 PW=INT(VAL(PW$)):IFPW<0ORPW>4095THENPRINT"]";:GOTO2050
2100 POKESID+3,PW/256:POKESID+2,PW-INT(PW/256)*256
2110 GOSUB2680:GOTO1220
2120 REM
2130 REM 5, CHANGE FILTER FREQUENCY
2140 REM
2150 PRINT"]ENTER NEW FILTER FREQUENCY (0-2047)"
2160 OPEN2,0:INPUT#2,FF$:PRINT:CLOSE2
2170 PRINT"]]                                   "
2180 PRINT"                                       "
2190 FF=INT(VAL(FF$)):IFFF<0ORFF>2047THENPRINT"]";:GOTO2150
2200 POKESID+21,FFAND7:POKESID+22,INT(FF/8)
```

```
2210 GOSUB2680:GOTO1220
2220 REM
2230 REM 6, CHANGE FILTER RESONANCE
2240 REM
2250 PRINT"ENTER NEW FILTER RESONANCE (0-15)"
2260 OPEN2,0:INPUT#2,FR$:PRINT:CLOSE2
2270 PRINT"                                    "
2280 PRINT"                                        "
2290 FR=INT(VAL(FR$)):IFFR<0ORFR>15THENPRINT"";:GOTO2250
2300 POKESID+23,FR*16+FP
2310 GOSUB2680:GOTO1220
2320 REM
2330 REM 7, CHANGE FILTER PASS
2340 REM
2350 PRINT"HIGH, LOW, BAND, OFF"
2360 GETA$:IFA$<>"O"ANDA$<>"B"ANDA$<>"L"ANDA$<>"H"THEN2360
2370 PRINT"                            "
2380 FP=1:IFA$="H"THENFP$="HIGH":VC=79:GOTO2420
2390 IFA$="L"THENFP$=" LOW":VC=31:GOTO2420
2400 IFA$="B"THENFP$="BAND":VC=47:GOTO2420
2410 FP$=" OFF":VC=15:FP=0
2420 POKESID+24,VC:POKESID+23,FR*16+FP
2430 GOSUB2680:GOTO1220
2440 REM
2450 REM PUT UP SCREEN DISPLAY
2460 REM
2470 POKE53281,14:POKE53280,14
2480 PRINT"                SOUND EDITOR 64                ";
2490 PRINT"                                               ";
2500 PRINT"          2  3  | 5  6";
2505 PRINT"  7  | 9  0  | -  \              ";
2510 PRINT"          Q W E R T Y U I O P @ * ↑          ";
2520 PRINT"                                               ";
2530 PRINT"          S  D  | G  H";
2535 PRINT"  J  | L  :                 ";
2540 PRINT"          Z X C V B N M , . /              ";
2550 PRINT"                                               ";
2560 PRINT"F1:D ATTACK  :       1 SPECIAL:"
2570 PRINT"F3:D DECAY   :       2 OSC 3   :"
2580 PRINT"F5:D SUSTAIN:"
2590 PRINT"F7:D RELEASE:"
2600 PRINT"  3 WAVEFORM    :"
2610 PRINT"  4 PULSE WIDTH:"
2620 PRINT"  5 FILTER FREQUENCY:"
2630 PRINT"  6 FILTER RESONANCE:"
2640 PRINT"  7 FILTER PASS      :"
2650 REM
2660 REM DISPLAY VALUES AFTER CHANGES
2670 REM
2680 PRINT""RIGHT$(STR$(AT),2)
2690 PRINT""RIGHT$(STR$(DE),2)
2700 PRINT""RIGHT$(STR$(SU),2)
2710 PRINT""RIGHT$(STR$(RE),2)
2720 PRINT"";S$
2730 PRINT""RIGHT$("      "+STR$(O3),5)
2740 PRINT"";WV$
2750 PRINT""RIGHT$("    "+STR$(PW),4)
2760 PRINT""RIGHT$("    "+STR$(FF),4)
2770 PRINT""RIGHT$(STR$(FR),2)
2780 PRINT"";FP$
2790 RETURN
2800 REM
2810 REM KEYBOARD TABLE, VALUE 64 MEANS
2820 REM NOT USED, <64 IS A NOTE, >64
2830 REM IS FOR MODIFICATION ROUTINES
2840 REM
2850 DATA 64,69,64,68,65,66,67,64
2860 DATA 15,14,64,64,0,1,16,64
2870 DATA 18,17,3,20,4,64,19,2
```

```
2880 DATA 22,21,6,64,7,8,23,5
2890 DATA 25,24,10,27,11,64,26,9
2900 DATA 64,28,13,30,14,15,29,12
2910 DATA 32,31,64,64,64,64,33,16
2920 DATA 64,64,64,13,64,64,12,64,64
2930 REM
2940 REM FREQUENCY LOW BYTE TABLE
2950 REM
2960 DATA 147,21,159,60,205,114,32,216
2970 DATA 156,107,70,47,37,42,63,100
2980 DATA 154,227,63,177,56,214,141,94
2990 DATA 75,85,126,200,52,198,127,97
3000 DATA 111,172
3010 REM
3020 REM FREQUENCY HIGH BYTE TABLE
3030 REM
3040 DATA 8,9,9,10,10,11,12,12
3050 DATA 13,14,15,16,17,18,19,20
3060 DATA 21,22,24,25,27,28,30,32
3070 DATA 34,36,38,40,43,45,48,51
3080 DATA 54,57
```

Program 31.

Notes can be played on voice 1 and you can change the filter values, set ring mod or sync with its corresponding oscillator 3 frequency, and change waveform, pulse width and the envelopes until you find the required sound.

Most of the keys on the keyboard are taken up with notes:

```
 2 3   5 6 7   9 Ø   — £
Q W E R T Y U I O P @ * ↑
```

and

```
 S D   G H J   L :
Z X C V B N M , . /
```

which give in total just over two and a half octaves to work with.

The function keys are set up to change the envelope settings and all the other values are changed by first pressing RETURN and then the number next to the required parameter.

Index

aperture, 136

bank select, 3, 4, 5
BORDER, 34

changing characters in interrupts, 158
CHAR, 52
character
    editor, 182
    generator, 1, 4, 5, 6
    memory, 1, 4
    plot, 46
    screen, 149
CIA#2, 3
CIRCLE, 73
circle plotting, 125
CLC, 31
CLG, 30
colour memory, 1, 3
combined 2D transformation using matrices, 132
combined transformations in Basic, 132
concatenated 3D matrix, 139
coordinates, 121, 136, 137
crunch to tokens wedge, 20

DFILL, 60
Diamond FILL demo program, 61
digital differential analyser algorithm, 123
display modes, 2
DRAW, 42

execute arithmetic wedge, 24
execute statement wedge, 23
extended graphics
    commands, 13 *et seq.*
    demo program, 119
    wedges, 15 *et seq.*

FILL, 55
FILL demo program, 56

games graphics planning, 148
GLOAD, 114
GRAPH, 29
graphics registers, 1, 2
GSAVE, 114

hidden line elimination, 144
hidden surface program, 144
high resolution bit mapped screen, 149
high resolution display storage, 9
HIRES, 27

image viewing coordinates, 137
initialisation wedge, 15
interrupts, 155, 158, 170
IRQ wedge, 24

keyboard, 160, 168
kinetic depth, 144

line intensity, 144
line plot in basic, 124
line plotting, 123

MAT, 78
mathematics of 3D displays, 138
matrix arithmetic, 78, 132, 139
matrix 2D transforms, 132
memory usage, 1, 2
moving 2D shape, 127
moving objects in interrupts, 155
multiple text screen animation, 157
music in interrupts, 170
music keyboard circuit, 168
music keyboard software, 160

NORM, 32

OFF, 107
ORIGIN, 33

PDRAW, 101
perspective cube - Basic program, 142
perspective projection, 142
PLACE, 108
PLOT, 38
POINT, 53
point plot in Basic, 122
point plotting and coordinates, 121
POLYGON, 66
polygon in Basic, 127
polyhedrons, 136
PORIGIN, 100
PPLOT, 101

raster effects, 149
rotation
  2D shape, 128
  3D shape, 139

scaling
  2D shape, 130
  3D shape, 139
SCREEN, 34
screen memory, 1, 3
scrolling, 151
shading surfaces, 145
SHAPE, 91
SHAPE demo program, 93
shape
  rotation in Basic, 129
  scaling in Basic, 130
  stretching in Basic, 131
  tables, 126, 137
  translation in Basic, 128
SID, 9
SID registers, 10, 11, 12
smooth scrolling, 151
sound editor, 186
split screens, 149
SPRITE, 108
  demo program, 109
  display storage, 9
  editor, 177
sprites - more than 8, 153
step quantisation, 123
stereoscoping views, 145
stretching 2D shape, 130

three D
  coordinates, 136
  display mathematics, 138
  image representation, 135
  matrix demo in Basic, 141
  plotting routines, 100 *et seq.*
  shape tables, 137
  shapes, 134
tokens to text wedge, 22
translation
  2D shape, 127
  3D shape, 138
two D
  plotting routines, 35 *et seq.*
  shapes, 126

VIC-11, 1, 2, 3, 4, 5, 6, 148
VIC-11 registers, 6, 7, 8
video bank select, 3, 148
view point, 136
viewing
  angle, 136
  coordinates, 137
  direction, 136

WINDOW, 25
wire frame display, 141

The graphics and sound capabilities of the Commodore 64 are one of the prime reasons for its success, but these features are some of the least understood by programmers. This book deals with the advanced techniques required to use the machine's capabilities to the full. Included is a full package of machine code routines to add graphics commands to the 64. These range from simple point plot to advanced three dimensional shape plotting. Besides a whole range of graphics utilities, many advanced subjects are covered ranging from smooth scrolling to split screens. Music and sound generation are dealt with in detail, from background music to adding a full keyboard to your computer.

Any programmer wishing to make full use of the graphics and sound capability of the Commodore 64 will find this book an invaluable mine of essential information – much of it previously unpublished.

The Authors

Nick Hampshire is a well-known author and microcomputer expert who has specialised in Commodore computer equipment. He started the first hobby microcomputer magazine, later absorbed into *Practical Computing,* of which he was technical editor for several years. He was the co-founder of *Popular Computing Weekly* and founder and managing editor of *Commodore Computing International* magazine. He is also the author of over a dozen books on popular computing, including the very successful and widely acclaimed *PET Revealed* and *VIC Revealed.*

Richard Franklin and Carl Graham are programmers with Zifra Software Ltd and together with Nick Hampshire have written some of the software included in this book.

Also by Nick Hampshire

THE COMMODORE 64 KERNAL AND HARDWARE REVEALED

0 00 383090 X

THE COMMODORE 64 DISK DRIVE REVEALED

0 00 383091 8

THE COMMODORE 64 ROMs REVEALED

0 00 383087 X

ADVANCED COMMODORE 64 BASIC REVEALED

0 00 383088 8

ISBN 0-00-383089-6

9 780003 830897

COLLINS
Printed in Great Britain

£10.95 net