

# ***HOT PROGRAMS TO FEED YOUR DRAGON***

*& Tandy Color Computer*



**Peter Robinson  
Mark Smith  
Neil Blacow**

**$\Sigma$  Sigma Technical Press**





# Hot Programs to Feed your Dragon ---- and Tandy Color Computer

**Peter Robinson, Mark Smith and  
Neil Blacow**

 Sigma Technical Press

Copyright © 1983 by Peter Robinson, Mark Smith and Neil Balcow

All rights reserved.

No part of this book may be reproduced or transmitted by any means without the prior permission of the publisher. The only exceptions are for the purposes of review, or as provided for by the Copyright (Photocopying) Act or in order to enter the programs herein onto a computer for the sole use of the purchaser of this book.

ISBN 0 905104 51 X

Published by:

SIGMA TECHNICAL PRESS,  
5 Alton Road,  
Wilmslow,  
Cheshire,  
UK.

Distributors:

Europe, Africa:  
JOHN WILEY & SONS LIMITED,  
Baffins Lane, Chichester,  
West Sussex, England.

Australia, New Zealand, South-East Asia:  
Jacaranda-Wiley Ltd., Jacaranda Press,  
JOHN WILEY & SONS INC.,  
GPO Box 859, Brisbane,  
Queensland 40001, Australia.

## Preface

So, you convinced yourself that buying a computer was the right thing to do because, not only would you learn about how it worked, but also it would be a useful thing to have about the house.

Now, be honest, has it really been useful? A quick look at the sales figures for software reveals that the majority of home computers are used for video games and very little else. This is not helped by the tremendous emphasis on games in the magazines and books published for home computers.

Computers can be useful and the programs in this book were written with this in mind. Looking at the Contents section reveals that we have included very few games. This was deliberate and, we hope, makes this book of programs for the Dragon 32 and Tandy Color Computer different from other books published for these computers.

We have included some games partly for our own amusement in writing them. Chapter 1 gets the games over with at the start.

The Graphics chapter tries to exploit the potential of the computer for producing line drawings, patterns and designs. We hope that the programs in this section will prove easy to use and should tempt even the most computer shy adult into trying out a few commands.

The rather grandly titled Business chapter tries to provide a few useful tools for use at home. The text editor and word processor can be used to type letters, prepare bills and so on. The telephone call programs could also prove useful when dialling long distance, both to recall the correct dialling sequence and to keep a parsimonious eye on the cost.

In the last chapter are included all those programs which we needed to help us write the book. In the jargon used by programmers of large computers such odds and ends of software are called "system utilities", or just plain utilities. We hope that these programs will prove useful to you, just as they did to us.

Finally, may you derive as much enjoyment from using these programs as we did in writing them.

**Acknowledgements:** The Authors wish to thank Dragon Data Ltd. for useful information on the BASIC interpreter and circuit diagrams; Tandy Corporation (UK) for the generous loan of a complete Color Computer system; the staff of the Lancaster branch of Tandy for their help and interest and Graham Beech for suggesting that we should write this book.

Peter Robinson,  
Mark Smith, Neil Blacow.

August 1983



# List of Contents

Introduction	vi
The Dragon 32	vi
The Tandy Color Computer	vii
Compatibility	vii
<b>Chapter 1 GAMES</b>	
Sub Hunt	1
Blocks	14
<b>Chapter 2 GRAPHICS</b>	
Sketchpad	21
Threedee	39
Logo	46
<b>Chapter 3 BUSINESS</b>	
Editor	74
W-Pro	90
Telecalc	110
Telenum	114
<b>Chapter 4 UTILITIES</b>	
Convertor	125
Monitor	138
Copy	143
Append	147
Catalog	152



## Introduction

In the last chapter are included some programs which we needed to help us write the book. In the jargon used by programmers of large computers such odds and ends of software are called "system utilities", or just plain utilities. We hope that these programs will prove useful to you just as they did to us.

Finally, may you derive as much enjoyment from these programs as we did in writing them.

# The Dragon 32

**Acknowledgements.** The Authors wish to thank Dragon Data Ltd. for useful information on the BASIC interpreter and circuit diagrams; Tandy Corporation (UK) for the generous loan of a complete Color Computer system; the

The Dragon 32 was introduced in 1982 as the first product of what is now Dragon Data Ltd. The Dragon is unusual in its use of the Motorola 6809 CPU, but this does not unduly concern the user of BASIC programs due to the choice of Microsoft BASIC as the Dragon's native language.

Although not the fastest microcomputer on the market, the Dragon does have a reasonable turn of speed and is blessed with colour graphics which are amongst the easiest to use, thanks to a powerful repertoire of commands. In particular, the provision of several graphics pages (thanks to the 6883 memory controller chip) can lead to some excellent animation effects. Hopefully the programs in this book will show some of the many graphics capabilities of the Dragon.

The sound capabilities of the Dragon are limited by the lack of a dedicated sound generation chip. The sound is produced by the processor outputting values representing a sine wave to a digital to analogue convertor. This means that the processor is completely committed to the execution of a sound command and can do nothing else. This can slow the speed of program execution down. In games programs where continuous action is needed it is not possible to execute a normal sound command. Later chapters will show how this problem can be overcome.

At the time of writing the Dragon disc system has not been formally announced, but it seems likely to be Microware's OS9. This operating system is a subject in its own right and this book will assume only that a normal tape cassette is being used for program and data storage on the Dragon.

## THE TANDY COLOR COMPUTER

Both the Color Computer and the Dragon appear to have been designed from a circuit suggested in a Motorola data book. It is therefore not surprising to find them to be virtually identical.

The above observations on the Dragon apply also to the Color Computer and the only differences as far as the user is concerned are the fact that the Color Computer has a serial printer connection as opposed to the Dragon's parallel interface.

One advantage of the Color Computer at the moment is the fact that disc drives are available which integrate with the standard BASIC language and augment it with extra commands. The Color Computer used in conjunction with this book was equipped with a single disc drive which proved most useful. Its only vice is a tendency to overheat, but then the summer of '83 has been unusually hot!

## COMPATIBILITY

In general BASIC programs will run on either machine (but see the last chapter concerning programs stored on tape). However, some other BASIC programs (NOT in this book!) might not run correctly if they make use of specific memory addresses which are different on the two machines. The programs included in this book will run on either machine.

To cater for the fact that Tandy owners might wish to use their disc drives for data storage, any programs which use data files have been written in such a way that editing one program line (i.e. the one used for file access) will allow the disc to be used. Dragon users will have to be content with cassette for the moment.



## Chapter 1

# Games

### SUB HUNT

A considerable amount of imagination helps in this game. For a start, the situation presented seems highly improbable. Secondly, due to speed limitations in BASIC, the three dimensional object which has to be found and destroyed is drawn in a fairly rudimentary way.

Having said that, here is the "scenario":-

As commander of a nuclear submarine you are told that a device has been placed on the sea bed near a fault in the earth's crust. Should this device explode then the result will be an earthquake of sufficient magnitude that a giant tidal wave will engulf all continents and their inhabitants will perish.

Of course, you and the rest of your crew would also perish due to the hull of your submarine being crushed in the blast. Hence there is some incentive for you to try to prevent this slightly messy situation from occurring.

How then can you find and destroy this menace? Easy, you have two aids to underwater vision. One is the 360 degree sonar scan. This is the display which presents itself when the game starts. Submariners may skip this explanation and go onto the next section. The sonar sweeps around the hull of the sub sending out sound pulses and listening for echoes. Targets are displayed on the TV screen in such a way that the arrow head on the display points in the direction of travel of the sub. Thus an object on the left appears on the left of the screen. If you spin the sub round the dot corresponding to an object would travel round a circular path in the opposite direction to your spin direction.



Horizontal movement of the right hand joystick will cause the sub to spin on its axis, i.e. this stick controls the submarine's heading. Run the program, select the easiest level, make sure the left hand stick is pulled as far back in the vertical plane as it will go, and try moving the right hand stick horizontally. You will see that the dot (which is the object to be located and destroyed) will appear to move in a circle about the centre of the screen. It is not really the object that is moving, but the fact that your sub is spinning about its axis. At the same time a beeping sound will indicate that the object has been located by your scanners.

Whilst spinning round, watch the dials at the bottom of the screen. The leftmost dial shows the heading of the submarine relative to the points of the compass (North at the top). This dial will rotate as your heading changes.

To move nearer to the object, use the heading control to bring the dot back to the dead ahead position (i.e. on the line with the arrow head). Now, holding this heading, move the left hand joystick forward and watch the dot move nearer to the centre of the screen. The pitch of the bleep goes up as the dot gets nearer. Don't get too near as the bomb has a proximity fuse which will be triggered if the range is less than 7 units. (The object is initially at 600 units away, due North).

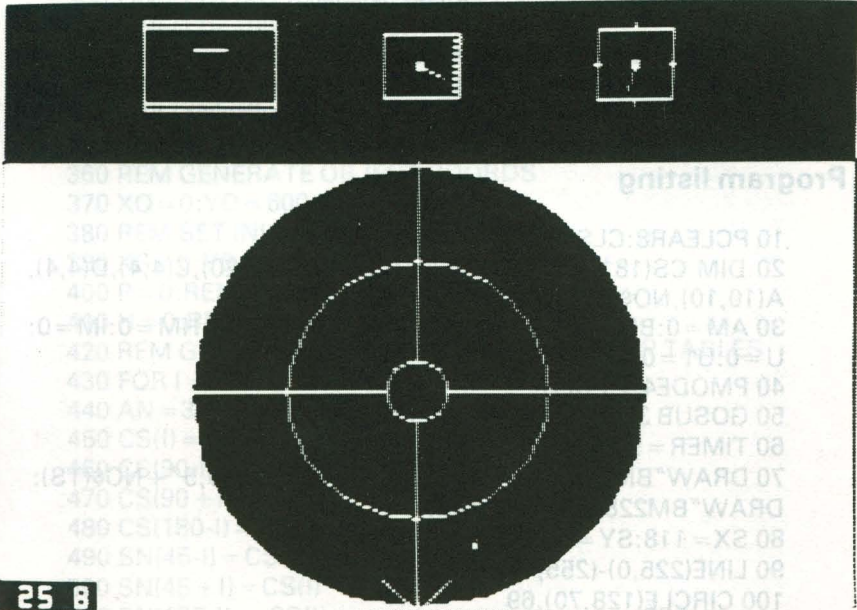
The pitch of the sub is controlled by vertical movement of the right hand stick. Pull this back and the sub rises in the water, push it forward and the sub dives. This control only takes effect if the sub is moving. The starting depth of the sub depends upon the level of difficulty selected, and at level 9 the sub is on the sea bed, as is the object. Thus, to destroy the object from level 1 (the hardest) the submarine commander is going to have to dive from the surface to the sea bed pretty quickly if the world is to be saved. The middle dial shows the pitch angle of the sub and the pointer will be level if the sub is neither rising nor diving. The right hand gauge shows the sub's depth, the pointer being at the top when the sub is at the surface and vice versa.

The time readout at the top right shows how much time is left until the bomb explodes. Keep an eye on this. It could be important to several people.

When the dot is within the innermost circle on the sonar display it should be visible on the second (TV) display. Press the button on the right hand joystick to turn on the TV display. This shows a 3D view of the bomb as would be seen by a TV camera mounted on the front of the sub. When the object is in view it will be seen that it is a rectangular box with a cross on one end. To destroy the



bomb the centre of this cross must be aligned with the cross hairs on the TV display. When all is lined up, press the button on the left hand joystick to fire.



**25 8**

**Note** that, due to the slowness of BASIC, the buttons have to be pressed for a considerable time before they take effect.

If all is well and you have aligned the cross hairs on the display with the cross on the box you will succeed in destroying it. The snag is that you can't fire **through** the box. This means that you must be sure to have the end of the box with the cross marked on it facing you when you fire. **Hint:** use the sonar display to help you move round the bomb. To move round it in an anticlockwise direction, keep the dot on the horizontal line to the left of centre and move the speed control forward. Adjust the controls to keep the dot on the line and watch the heading indicator turn. When the heading has changed by the required amount (say 90 degrees) bring the dot back to the dead ahead position and look at it on the TV display.

Remember not to get too close or the proximity fuse will detonate the bomb. A warning will sound when the sub is approaching the point where this will occur, so take note!

## Summary of controls

### *Left joystick*

Vertical = speed control  
Horizontal = not used  
Button = fire

### *Right joystick*

Vertical = pitch  
Horizontal = heading  
Button = change display

## Program listing

```
10 PCLEAR8:CLS:PRINT"PLEASE WAIT"
20 DIM CS(181),SN(181),V(20,3),E(20),B(20,20),C(4,4),D(4,4),
  A(10,10),NO$(11),Q(30,10)
30 AM=0:BM=0:DM=0:EM=0:FM=0:GM=0:HM=0:IM=0:
  U=0:U1=0:V=0:V1=0
40 PMODE4,1:PCLS
50 GOSUB 3200 : REM INITIALISE STRINGS
60 TIMER=50:UM=0:TS=0:US=9
70 DRAW"BM245,5"+NO$(US):DRAW"BM239,5"+NO$(TS):
  DRAW"BM228,5"+NO$(UM)
80 SX=118:SY=50:D=75:FL=1
90 LINE(225,0)-(255,11),PSET,B
100 CIRCLE(128,70),69
110 LINE(0,0)-(255,140),PSET,B
120 PAINT(79,5):GET(0,0)-(3,3),C:PAINT(180,5)
130 LINE(0,140)-(255,191),PSET,B
140 LINE(0,0)-(255,140),PRESET,B
150 CIRCLE(128,70),10
160 CIRCLE(128,70),40
170 LINE(128,0)-(128,60),PSET
180 LINE(128,140)-(128,80),PSET
190 LINE(48,70)-(118,70),PSET
200 LINE(138,70)-(255,70),PSET
210 LINE(118,10)-(128,0),PSET
220 LINE(138,10),PSET
230 REM SET UP STATUS DISPLAY
240 LINE(53,159)-(75,181),PSET,B
250 LINE(64,157)-(64,160),PSET
260 LINE(52,170)-(54,170),PSET
270 LINE(64,183)-(64,180),PSET
```

```

280 LINE(74,170)-(76,170),PSET
290 FOR I=160 TO 180 STEP 2
300 LINE(116,I)-(118,I),PSET:NEXT I
310 LINE(116,160)-(139,180),PSET,B
320 LINE(172,156)-(212,158),PSET,B
330 LINE(172,182)-(212,184),PSET,B
340 LINE(172,156)-(212,184),PSET,B
350 GOSUB 7000
360 REM GENERATE OBJECT COORDS
370 XO=0:YO=600:ZO=1000
380 REM SET INITIAL SUB POSITION
390 XS=0:YS=0
400 P=0:REM INITIAL PITCH= ZERO
410 H=0:REM INITIAL HEADING= ZERO
420 REM GENERATE SINE AND COSINE LOOKUP TABLES
430 FOR I=0 TO 45
440 AN=3.142*I/90
450 CS(I)=COS(AN)
460 CS(90-I)=-CS(I)
470 CS(90+I)=-CS(I)
480 CS(180-I)=CS(I)
490 SN(45-I)=CS(I)
500 SN(45+I)=CS(I)
510 SN(135-I)=-CS(I)
520 SN(135+I)=-CS(I)
530 NEXT I
540 REM READ OBJECT MATRICES
550 READ NV:REM NUMBER OF VERTICES
560 FOR P=1 TO NV
570 READ V(P,1),V(P,2),V(P,3)
580 V(P,1)=V(P,1)+XO
590 V(P,2)=V(P,2)+YO
600 V(P,3)=V(P,3)+ZO
610 NEXT P
620 READ NE:REM NUMBER OF EDGES
630 FOR E=1 TO NE
640 READ E(E)
650 NEXT E
660 REM SAMPLE CONTROLS
670 DP=JOYSTK(1):DH=JOYSTK(0)
680 VL=16-JOYSTK(3)/4

```



```
690 H = H-FIX(DH/8) + 4
700 IF DP < 32 THEN P = 164 + DP/2 ELSE P = DP/2-16
710 IF H > 180 THEN H = H-180
720 IF H < 0 THEN H = H + 180
730 REM UPDATE 3D VELOCITIES
740 VX = VL*SN(H)*CS(P)
750 VY = VL*CS(H)*CS(P)
760 VZ = VL*SN(P)
770 REM CALC NEW SUB POSITION
780 XS = XS + VX:YS = YS + VY:ZS = ZS-VZ
790 IF ZS < 0 THEN ZS = 0
800 IF ZS > 1000 THEN ZS = 1000
810 REM CHECK FOR OBJECT IN LONG RANGE SCAN
820 AX = XO-XS:AY = YO-YS
830 RO = SQR(AX*AX + AY*AY)
840 IF RO > 680 THEN 1090
850 IF RO < 7 AND ZS > 990 THEN 4000
860 B = PEEK(65280):IF B = 254 OR B = 126 THEN 870 ELSE 880
870 CF = 1:FL = FL-1:IF FL < 1 THEN FL = 2:PMODE
4,5:PCLS:GOSUB3010:SCREEN1,0
880 ON FL GOTO 1000,2000
1000 REM GENERATE LONG RANGE SCAN
1010 PMODE 4,1:SCREEN1,0:GOSUB 3010
1020 PUT(SX-2,SY-2)-(SX+2,SY+2),D
1030 SX = -(AX*CS(H)-AY*SN(H))/10 + 128:SY = -
AY*CS(H) + AX*SN(H))/10 + 70
1040 GET(SX-2,SY-2)-(SX+2,SY+2),D
1050 PSET(SX,SY):PSET(SX+1,SY+1)
1060 PSET(SX+1,SY):PSET(SX,SY+1)
1070 SOUND 255-RO/10,1
1080 REM UPDATE STATUS DISPLAYS
1090 PUT(57,160)-(71,180),B
1100 PUT(122,162)-(134,178),B
1110 PUT(182,160)-(202,180),B
1120 LINE(-8*SN(H) + 64,-8*CS(H) + 170)-(64,170),PSET
1130 LINE(63,169)-(65,171),PSET,B
1140 LINE(-8*CS(P) + 128,-8*SN(P) + 170)-(128,170),PSET
1150 LINE(127,169)-(129,171),PSET,B
1160 LINE(187,ZS/50 + 160)-(197,ZS/50 + 160),PSET
1170 GOSUB 3010
1180 IF EW = 1 THEN 4000 : REM BOMB EXPLODES
```

```

1190 GOTO670
2000 PMODE4,5:SCREEN1,0
2010 AM = CS(H):BM = -SN(H)
2020 DM = SN(H)*CS(P)
2030 EM = CS(P)*CS(H)
2040 FM = SN(P)
2050 GM = -SN(H)*SN(P)
2060 HM = -SN(P)*CS(H)
2070 IM = CS(P)
2080 PCLS:CF = 1:LINE(123,96)-(133,96),PSET:LINE(128,91)-
(128,101),PSET
2090 FOR E = 1 TO NE
2100 EA = ABS(E/2)
2110 X = V(EA,1)-XS
2120 Y = V(EA,2)-YS
2130 Z = V(EA,3)-ZS
2140 GOSUB 3010
2150 X3 = AM*X + BM*Y
2160 Y3 = DM*X + EM*Y + FM*Z
2170 Z3 = GM*X + HM*Y + IM*Z
2180 U = 128-150*X3/Y3
2190 V = 96-127.5*Z3/Y3
2200 IF U > 255 THEN U = 255
2210 IF U < 0 THEN U = 0
2220 IF V > 255 THEN V = 255
2230 IF V < 0 THEN V = 0
2240 IF E/2 > 0 THEN LINE(U1,V1)-(U,V),PSET
2250 B = PEEK(65280):IF B = 253 OR B = 125 THEN B2 = 1 ELSE B2 = 0
2260 IF E < > NE THEN 2320
2270 IF U1 > U THEN MU = (U1-U)/2 + U ELSE MU = (U-U1)/2 + U1
2280 IF V1 > V THEN MV = (V1-V)/2 + V ELSE MV = (V-V1)/2 + V1
2290 IF MU > 128 THEN DX = MU-128 ELSE DX = 128-MU
2300 IF MV > 96 THEN DY = MV-96 ELSE DY = 96-MV
2310 IF DX < 5 AND DY < 5 AND XS > 10 AND YS > 580 AND YS < 620
AND ZS > 900 THEN MP = 1 ELSE MP = 0
2320 U1 = U:V1 = V
2330 IF RO < 15 AND ZS > 990 THEN SOUND 255-RO,1
2340 NEXT E
2350 IF EW = 1 THEN 4000
2360 HT = 0

```



```

2370 IF B2 = 1 THEN GOSUB 5000 : REM FIRE
2380 IF HT = 1 THEN 6000 : REM HIT
2390 GOTO 670
2400 DATA 8
2410 DATA 5.25,-2,3.25
2420 DATA -5.25,-2,3.25
2430 DATA -5.25,-2,-3.25
2440 DATA 5.25,-2,-3.25
2450 DATA 5.25,2,3.25
2460 DATA -5.25,2,3.25
2470 DATA -5.25,2,-3.25
2480 DATA 5.25,2,-3.25
2490 DATA 19
2500 DATA -1,2,3,4,1,5,6,7,8,5
2510 DATA -2,6,-3,7,-4,8,1,-4,5
3000 REM COUNTDOWN
3010 IF CF = 1 THEN CF = 0 ELSE 3040
3020 IF FL = 1 THEN PUT(225,10)-(255,0),Q,PSET
3030 DRAW"BM244,5" + NO$(US) + "BM239,5" + NO$(TS) +
"BM228,5" + NO$(UM)
3040 IF TIMER < 50 THEN RETURN
3050 TIMER = 0:DP = JOYSTK(1):DH = JOYSTK(0)
3060 PUT(245,10)-(254,0),A,PSET
3070 S1 = 0:S2 = 0
3080 US = US-1
3090 IF US < 0 THEN US = 9:S1 = 1
3100 DRAW"BM245,5" + NO$(US)
3110 IF UM = 0 AND TS = 0 AND US = 0 THEN EW = 1
3120 IF S1 < > 1 THEN RETURN ELSE TS = TS-1:IF TS < 0 THEN
TS = 5:S2 = 1
3130 PUT(238,10)-(242,0),A,PSET
3140 DRAW"BM239,5" + NO$(TS)
3150 IF S2 < > 1 THEN RETURN ELSE UM = UM-1:IF UM < 0 THEN
UM = 0
3160 PUT(228,10)-(234,0),A,PSET
3170 DRAW"BM228,5" + NO$(UM)
3180 RETURN
3200 NO$(1) = "BU3BR3D3BL3BD3BR3U3BR1"
3210 NO$(2) = "BU3R3D3L3D3R3BU3BR1"
3220 NO$(3) = "BU3R3D3L3BD3R3U3BR1"
3230 NO$(4) = "U3BR3D3L3BD3BR3U3BR1"

```

```

3240 NO$(5) = "U3R3BD3L3BD3R3U3BR1"
3250 NO$(6) = "U3BR3BD3L3D3R3U3BR1"
3260 NO$(7) = "BU3R3D6BU3BR1"
3270 NO$(8) = "U3R3D3L3D3R3U3BR1"
3280 NO$(9) = "U3R3D3L3BD3BR3U3BR1"
3290 NO$(0) = "U3R3D3BL3D3R3U3BR1"
3300 RETURN
4000 REM EXPLODE BOMB
4010 POKE &HFF23,&H3C
4020 FOR I = 1 TO 4:PCOPY I TO I + 4:NEXT I
4030 PMODE 4,5:SCREEN 1,0
4040 FOR I = 1 TO 50
4050 CIRCLE(SX,SY),I
4060 FOR J = 1 TO RND(10):POKE&HFF20,RND(127):POKE&HFF20,
0:POKE&HFF22,RND(255): NEXT J
4070 NEXT I
4080 PMODE 3,5:SCREEN1,1
4090 DRAW"BM5,20S8C5BU8R8BL4D8BR8"
4100 DRAW"U8BR8D8BU4L8BD4BR12"
4110 DRAW"U8R8BD4L8BD4R8BR4"
4120 DRAW"BM193,130U8R8BD4L8BD4R8BR4"
4130 DRAW"U8F8U8BD8BR4"
4140 DRAW"U8R6F2D4G2L6BR12S4"
4150 FOR I = 1 TO 2000:NEXT I
4160 SCREEN 0,0:CLS
4170 PRINT "HAVING SINGULARLY FAILED TO ":PRINT"SAVE THE
WORLD"
4180 PRINT "WOULD YOU LIKE TO TRY AGAIN?"
4190 PRINT "ENTER Y OR N> "
4200 A$ = INKEY$
4210 IF A$ = "" THEN 4200
4220 IF A$ = "Y" THEN GOSUB
7000:FL = 1:XS = 0:YS = 0:H = 0:HT = 0:EW = 0:GOTO 670
4230 POKE 65494,0 : REM NORMAL PROCESSOR RATE
4240 END
5000 REM FIRE AT BOMB
5010 X = 0:Y = 128:POKE &HFF23,&H3C
5020 FOR I = 1 TO 32
5030 LINE(X,Y)-(X+4,Y-1),PSET
5040 LINE(256-X,Y)-(251-X,Y-1),PSET

```

```
5050 X = X + 4:Y = Y - 1:POKE&HFF20,255:POKE&HFF20,0
5060 NEXT I
5070 CIRCLE(128,96),4:PAINT(128,96),1:SOUND127,1:X = 0:Y = 128
5080 FOR I = 1 TO 32
5090 CIRCLE(128,96),4 + I/8
5100 LINE(X,Y)-(X + 4,Y - 1),PRESET
5110 LINE(256 - X,Y)-(251 - X,Y - 1),PRESET
5120 X = X + 8:Y = Y - 2
5130 NEXT I
5140 IF MP = 1 THEN HT = 1
5150 RETURN
6000 REM HIT BOMB
6010 PRINT"YOU HAVE SAVED THE WORLD"
6020 PRINT"WITH";STR$(UM);": ";RIGHT$(STR$(TS),
LEN(STR$(TS)) - 1); RIGHT$(STR$(US),LEN(STR$(US)) - 1);
"TO SPARE"
6030 GOTO 4180
7000 REM INSTRUCTIONS
7010 SCREEN 0,0:CLS
7020 CLS
7030 PRINT"LEVEL OF DIFFICULTY?"
7040 PRINT"ENTER A NUMBER 1 TO 9"
7050 PRINT"(1 = HARD,9 = EASY)"
7060 I$ = INKEY$
7070 IF I$ = "" THEN 7060
7080 IF I$ < "1" OR I$ > "9" THEN 7060
7090 UM = VAL(I$):TS = 5:US = 10
7100 CF = 1:TIMER = 50
7110 ZS = (VAL(I$) + 1)*100
7120 PRINT"LEVEL ";I$
7130 PRINT"PLEASE WAIT"
7140 RETURN
```

## Program description

Lines	Purpose
10	Reserve graphics screen memory and clear screen.
20-30	Dimension arrays, initialise variables.
40	Set graphics mode, clear graphics screen.



50	Set up strings for drawing numbers for timer.
60	Set timer value and initialise time to 10secs.
70	Draw time in top right of screen.
80	Set variables SX and SY to a clear part of screen, set viewer distance (D) to 75 units, set flag (FL) to select sonar display on entry.
90-220	Set up the sonar display in screen 1.
230-340	Draw the status displays for sonar screen.
350	Ask user to set level of difficulty.
360-370	Initialise the object's X,Y,Z position.
380-390	Initialise the sub's X,Y,Z position.
400-410	Set initial pitch and heading to zero.
420-530	Generate the look up tables for sines and cosines.
540-650	Read in the data to form the 3D picture of the bomb.
660-720	Sample joysticks, compute new values for pitch, heading and speed.
730-760	Form new X,Y and Z components of velocity.
770-800	Calculate new sub position using new velocities.
810-830	Calculate distance from sub to object (RO).
840	If out of range of sonar only update status, no sound.
850	Check for proximity explosion, but make sure sub depth is near bottom (ZS> 990).
860-870	If right hand button pressed then change display by altering value of flag FL.
880	Go to appropriate display routine.
1000-1190	Generate long range (sonar) scan with sound, check for end-of-world flag (EW) (set by timer routine).
2000	Select 3D display starting at page 5.
2010-2070	Calculate 3D projection parameters.
2080	Clear screen, set flag telling timer routine to re-display the time, put crosshairs at centre.
2090-2130	Move observer to origin.
2140	Call timer routine.
2150-2170	Transform coordinates of box edge (8 edges in all).
2180-2190	Project object edge on to 2D screen.
2200-2230	Clip object to lie within screen limits.
2240	Plot edge from end of last edge (U1,V1) to end of new edge (U,V).
2250	Check fire button.
2260	Skip next section if not last edge of object.
2270-2310	Calculate distance from centre of cross on bomb to

### *Hot programs to feed your Dragon*

crosshairs on screen,if within allowed limits set mid-point flag (MP) to 1 else reset it.

2320 Update U1,V1 ready for next time round.

2330 Check for proximity warning,sound if too near.

2340 Repeat above for all edges of box.

2350 If EW flag set then explode everything.

2360 Reset hit flag (HT).

2370 If button was pressed during drawing of bomb then go to fire routine.

2380 Fire routine will set HT to indicate a hit.

2390 Go and do it all again.

2400-2510 Data for drawing box (bomb).

3000 Timer routine counts down from initial setting to zero.Sets EW flag at zero.

3010 Check flag which signals screen cleared,if set then re-write entire time display and clear flag.

3020 If TV screen selected then must erase corner of screen to remove any object drawn there.

3030 Draw time.

3040 Only update time every 50 timer ticks (1 sec).

3050 Reset timer,sample joysticks (to provide better control sensing).

3060 Erase units of seconds.

3070 Zero flags.

3080 Decrement units of seconds count.

3090 Check for underflow,reset to 9 if  $< 0$ ,set flag to signal that tens of seconds must also change.

3100 Draw units of seconds.

3110-3180 Repeat for tens of seconds and units of minutes.

3200-3300 Strings for drawing numerals on graphics screen.

4000 Like it says!.

4010 Enable sound channel output from D-A convertor.

4020 Copy sonar screen to TV screen (to save re-writing the sonar screen ready for next game).

4030-4070 Draw expanding circle with random low frequency. sounds poked straight to the D-A convertor,flash random colours by changing modes on the 6847 TV controller chip.

4080-4240 Put up end message,invite further attempt.

5000-5150 Draw tracer to centre of screen,check for hit.

6000-6030 Congratulate and print safety margin.



7000-7140 Ask for difficulty level, adjust start time and initial sub depth accordingly.

## Variable definitions

Variable name	Function
CS, SN	Arrays used to store pre-calculated cosine and sine values.
V	Array of 3-D object vertices.
E	Array of 3-D object edges.
A, B, C, D	Blank arrays for screen erasure.
NO\$	Strings for graphics numerals.
Q	Array used to erase timer in TV mode.
AM, BM, DM, EM, FM, GM, HM, IM	Variables used to obtain 3-D image.
U, U1, V, V1	Used to plot each edge of 3-D object.
UM, TS, US	Three digit timer value, units of mins, tens of secs, units of secs.
SX, SY	X, Y position of dot on sonar screen.
D	Distance of viewer from TV screen.
FL	Flag used to change from sonar to TV display screen.
XO, YO, ZO	Object coordinates.
XS, YS, ZS	Submarine coordinates.
P, H	Pitch and heading of submarine.
AN	Angle used in calculation of sine and cosine arrays.
NV, NE	Number of vertices and number of edges in the 3-D object.
DP, DH	Changes in pitch and heading.
VL	Submarine's velocity.
VX, VY, VZ	Velocity components in three dimensions.
AX, AY	Components of object-sub distance in X, Y plane.
RO	Radial distance of object from sub.
B	Used to read button settings.
CF	Flag to denote screen cleared.
EW	Flag to denote end of world has come.
EA	Current edge value in 3-D projection.
X, Y, Z	Coordinates of object relative to sub.
X3, Y3, Z3	Transformed coordinates.
MU, MV, MP,	Used to check for hit when firing.
DX, DY	
HT	Flag to signal object hit.

## **Points of interest**

The 3D projection techniques are adapted from an article in "BYTE" magazine entitled "Interactive 3-D Graphics for the Apple II" which appeared in the issue of November 1982. The method used draws a 3-D image of a "wire frame" object (i.e. one which is transparent). The object to be drawn is described by means of its "edges" (the "wires") and "vertices" (the points joined by the edges). Any object can be drawn with this technique, but the process is slow, as the vertices have to be transformed to the 2-D screen for each edge. The variables U, V and U1, V1 are used to represent the end points of each edge and then a simple PLOT command is used to draw the edge. A negative number is used to signify an edge which is "invisible" and these are not plotted. The major modifications which have been adopted to speed up the projections concern the pre-calculation of sine and cosine values. These are placed in arrays indexed by the pitch and heading values. Also, variations in bank of the sub are not allowed and this simplifies the calculations considerably. The sub is allowed free movement in three dimensions according to the control settings and thus the equations are modified slightly from those in the original article where the observer was constrained to lie on the surface of a hemisphere surrounding the object.

It was found that the standard SOUND and PLAY commands did not allow a satisfactory explosion to be produced, so the sound generator was POKEd directly with random levels during the end of world explosion.

A final note: Level 1 is possible, just!

## **BLOCKS**

After the Sub Hunt this one is easy! The player is presented with a grid of 12 rectangles, of which 4 are coloured red and 4 are coloured yellow. The object is to exchange the red and yellow rectangles such that all those which were initially red become yellow and vice versa.

The rectangles are numbered 1 to 12 and the program makes moves by prompting the player to enter a FROM number (which must correspond to a coloured rectangle) and a TO number which must be one jump away from the previous rectangle and must not be occupied. Invalid entries will be ignored.



After each move a counter is incremented which represents the total number of valid moves made. The player who scores the lowest total count over a number of games is the winner.

### Program listing

```

10 REM BLOCKS
20 CLEAR 500
30 XF=0:YF=0:FXF=0:FYF=0
40 DIM BL(30,15),NO$(10),AB(12,3)
50 GOSUB 5000
60 PMODE3,1:SCREEN1,0:PCLS
70 A$="":B$=""
80 GET(100,50)-(130,60),BL,G
90 FOR Q=1 TO 12
100 FOR Z=1 TO 3
110 READ AB(Q,Z)
120 NEXT Z
130 NEXT Q
140 RESTORE
150 REM DRAW INITIAL SETUP
160 X=10:Y=5
170 COLOR 7
180 FOR I=1 TO 4
190 LINE(X,Y)-(X+35,Y+20),PSET,B
200 Y=Y+40
210 NEXT I
220 Y=Y-80
230 FOR I=1 TO 4
240 X=X+50
250 LINE(X,Y)-(X+35,Y+20),PSET,B
260 NEXT I
270 FOR I=1 TO 3
280 LINE(X,Y)-(X+35,Y+20),PSET,B
290 Y=Y-40
300 NEXT I
310 Y=Y+160
320 LINE(X,Y)-(X+35,Y+20),PSET,B
330 X=X-100
340 LINE(X,Y)-(X+35,Y+20),PSET,B
350 DRAW"BM50,12D8"

```



```
360 DRAW"BM51,50" + NO$(2)
370 DRAW"BM1,92" + NO$(3)
380 DRAW"BM51,132" + NO$(4)
390 DRAW"BM78,75" + NO$(5)
400 DRAW"BM126,75" + NO$(6)
410 DRAW"BM176,72R4D8"
420 DRAW"BM125,155" + NO$(8)
430 DRAW"BM200,15" + NO$(9)
440 DRAW"BM195,50" + NO$(1) + "BR3" + NO$(0)
450 DRAW"BM249,90D7BR6U7"
460 DRAW"BM195,135" + NO$(1) + "BR3" + NO$(2)
470 LINE(27,25)-(27,45),PSET
480 LINE(27,65)-(27,85),PSET
490 LINE(27,105)-(27,125),PSET
500 LINE(127,105)-(127,125),PSET
510 LINE(227,105)-(227,125),PSET
520 LINE(227,65)-(227,85),PSET
530 LINE(227,25)-(227,45),PSET
540 LINE(45,95)-(60,95),PSET
550 LINE(95,95)-(110,95),PSET
560 LINE(145,95)-(160,95),PSET
570 LINE(195,95)-(210,95),PSET
580 Y = 10
590 FOR I = 1 TO 4: PAINT(15,Y),0,3: Y = Y + 40: NEXT I
600 Y = 10: FOR I = 1 TO 4: PAINT(230,Y),2,3: Y = Y + 40: NEXT I
610 X = 100
700 REM INPUT "FROM" POSN.
710 REM
720 DRAW"BM30,180U8R4BD4L4BR8D4U8R4D4L4R1F4BR4U8R5
D8L5BR9U8F4E4D8BR20E4H4"
730 A$ = INKEY$
740 IF A$ = CHR$(13) THEN 900
750 IF A$ < "0" OR A$ > "9" THEN 730
760 IF F = 1 AND LEN(B$) = 2 THEN X = 200: GOTO 780
770 IF LEN(B$) = 2 THEN X = 100 ELSE 790
780 IF LEN(B$) = 2 THEN PUT(X-2,169)-(X+28,184),
BL,PSET: A$ = "": B$ = "": GOTO 700
790 DRAW"BM" + STR$(X) + ",176" + NO$(VAL(A$))
800 X = X + 15
810 B$ = B$ + A$
820 GOTO 730
```

```

900 REM EXAMINE INPUT
910 IF VAL(B$)> 0 AND VAL(B$)< 13 THEN 940
920 IF F = 1 THEN X = 200 ELSE X = 100
930 GOTO 780
940 ON VAL(B$) GOSUB 970,980,990,1000,1010,1020,1030,1040,
1050,1060,1070,1080
950 IF F = 0 THEN GOTO 1090
960 TVAL = VAL(B$):GOTO 1230
970 XF = 27:YF = 15:RETURN
980 XF = 27:YF = 55:RETURN
990 XF = 27:YF = 95:RETURN
1000 XF = 27:YF = 135:RETURN
1010 XF = 77:YF = 95:RETURN
1020 XF = 127:YF = 95:RETURN
1030 XF = 177:YF = 95:RETURN
1040 XF = 127:YF = 135:RETURN
1050 XF = 227:YF = 15:RETURN
1060 XF = 227:YF = 55:RETURN
1070 XF = 227:YF = 95:RETURN
1080 XF = 227:YF = 135:RETURN
1090 IF PPOINT(XF,YF) = 4 THEN BC = 0:GOTO 1120
1100 IF PPOINT(XF,YF) = 2 THEN BC = 2:GOTO 1120
1110 B$ = " ":X = 100:GOTO 780
1120 FVAL = VAL(B$):FXF = XF:FYF = YF
1200 REM INPUT "TO" POSN.
1210 DRAW"BM150,180U8L3R6BR3R6D8L6U8BR25F4G4"
1220 X = 210:A$ = "":B$ = "":F = 1:GOTO 730
1230 IF PPOINT(XF,YF) = 4 THEN 1260
1240 IF PPOINT(XF,YF) = 2 THEN 1260
1250 GOTO 1270
1260 PUT(100,169)-(135,184),BL,PSET:PUT(146,169)-(185,184),
BL,PSET:PUT(200,169)-(235,184),BL,PSET:F = 0:A$ = "":B$ = "":
X = 100:GO TO 730
1270 FOR Q = 1 TO 3
1280 IF AB(FVAL,Q) = TVAL THEN 1310
1290 IF Q = 3 THEN 1260
1300 NEXT Q
1310 REM PAINT NEW COLOURS
1315 GOSUB 6000
1320 PAINT(XF,YF),BC,3
1330 PAINT(FXF,FYF),1,3

```

```

1340 GOTO 1260
5000 REM NUMBER GRAPHICS
5010 NO$(1) = "BR2U4D8U4"
5020 NO$(2) = "BU4R4D4L4D4R4BU4"
5030 NO$(3) = "BU4R4D4L4BD4R4U4"
5040 NO$(4) = "U4BR4D4L4BD4BR4U4"
5050 NO$(5) = "U4R4BD4L4BD4R4U4"
5060 NO$(6) = "U4BR4BD4L4D4R4U4BR6"
5070 NO$(7) = "BU4R4D8U4"
5080 NO$(8) = "U4R6D4L6D4R6U4"
5090 NO$(9) = "U4R4D4L4BD4BR4U4"
5100 NO$(0) = "U4R4D8L4U4"
5110 RETURN
5120 DATA 2,2,2,1,3,3,2,4,5
5130 DATA 3,3,3,3,6,6,5,7,8
5140 DATA 6,11,11,6,6,6,10,10
5150 DATA 10,9,11,11,7,10,12
5160 DATA 11,11,11
6000 REM INCREMENT COUNTER
6010 CT = CT + 1
6030 HC = INT(CT/100)
6040 TC = INT((CT-HC*100)/10)
6050 UC = CT-HC*100-TC*10
6060 PUT(115,1)-(145,16),BL,PSET
6070 DRAW"BM115,5" + NO$(HC) + "BM125,5" + NO$(TC) +
    "BM135,5" + NO$(UC)
6080 RETURN

```

## Program description

Lines	Purpose
20	Clear space for numeral strings.
30	Initialise move coordinates.
40	Dimension arrays.
50	Set up strings which are used to draw numerals on graphics screen.
60	Set graphics mode,clear screen.



70	Initialise FROM and TO strings.
80	GET the blanking array (for erasure of rectangles).
90-140	Read in data to array AB which defines valid moves for each rectangle on grid.
150-600	Draw grids and numbers on screen, paint rectangles.
610-930	Prompt player to enter FROM value, check each entry to see if it lies in the range 1 to 12. Wait for a valid entry.
940	Go to routines dealing with each FROM rectangle.
970-1080	Routines which set the X,Y coordinates of the centre of each rectangle. These points are to be tested for colour:red,yellow or green(unoccupied).
1090-1100	Set variable BC to indicate red or yellow rectangle.
1110	Come here if trying to move from an unoccupied rectangle, blank out FROM entry and try again.
1200-1220	Prompt player to enter a TO value. This is done by using the same routine as in the FROM case. A flag "F" is set to 1 to get a TO value and to 0 if a FROM value is required.
1230-1260	Check for rectangle already filled with red or yellow. If so then wipe out both FROM and TO entries (line 1260) and start again.
1270	Come here if both FROM and TO are valid and the TO rectangle is not already coloured.
1270-1300	Check all moves in array AB to see if the move specified by the player is in the array (i.e is a valid move).
1310-1330	Paint colours of FROM and TO rectangles. Only come here on a valid move, so ok to increment the valid move counter (line 1315).
1340	Go back and do it all again.

## Variable definitions

<i>Variable name</i>	<i>Function</i>
BL	Array used to blank either rectangles or counter.
NO\$	Holds strings for drawing numerals on graphics screen.
AB	Array of possible moves from each rectangle. Used to check moves for validity.
XF,YF	Coordinates of centre of rectangle (used to test for colour of rectangle).

## Hot programs to feed your Dragon

FXF,FYF	As above but specific to FROM rectangle.
A\$,B\$	Strings used to read in a two digit value which specifies either the FROM or TO rectangle.
X,Y	Coordinates used in setting up screen and prompting for FROM and TO entries.
I,Q,Z	Loop counters.
F	Flag,initially zero,used to signify whether a FROM or a TO move is being input by the common entry routine.
FVAL,TVAL	FROM and TO positions used in making moves.
CT	Move counter,initially zero by default.
HC,TC,UC	Hundredths,tenths and units digits of count value.

Program description		Variable definitions	
Lines	Purpose	Function	Variable name
100	Array of possible moves from each rectangle. Used to check moves for validity.		AB
110	Coordinates of centre of rectangle (used to test for colour of rectangle).		XF,YF
120	Strings used for drawing numbers on graphics screen.		NO\$
130	Array used to blank either rectangles or counter.		BL

## Chapter 2

# Graphics

## SKETCHPAD

This is a line drawing package which allows its user to exploit the high resolution screen of the Dragon or Color Computer.

Most functions of the program are controlled by the joysticks and pushbuttons. This gives more flexible control than would be possible if the user was continually changing from joystick to keyboard and vice versa. To allow for cursor selection of functions the alternatives are displayed on the graphics screen when the program is run. Down the right hand side of the screen are displayed the letters D,E,N,S,L,P,H,T corresponding to the following options:-

Draw	-the normal drawing mode (selected on entry)
Erase	-erase the last operation performed
New	-clear the screen and select Draw mode
Save	-save the current list of commands to disc or tape
Load	-load a new list of commands from disc or tape
Paint	-paint an area of the screen with solid colour
Hard copy	-dump the screen contents to a printer
Text	-enter text and draw it in the screen

Along the bottom of the screen are symbols which are used to denote what action will be performed when a Draw command is executed-the alternatives are straight lines, rectangles, circles and alphanumeric text.



## **Controls**

### **1) Cursors**

The right hand joystick controls the position of the "dynamic cursor". Try moving this control and you will see that one of the two crosses (the + sign) will follow the movements of the stick. This is the dynamic cursor. The "static cursor" is shown by a X symbol and this does not move with the stick. However, if the left hand button is pressed the static cursor will move to lie on top of the dynamic one. Keep the left button pressed and wiggle the right joystick, both cursors now move.

The two cursors are used to define various aspects of the drawings such as the end points of lines, the radii of circles etc. These will be described under sections dealing with each command.

### **2) Selecting functions**

The button on the right hand joystick is used to signify that an action is to be performed. Thus, to enter Paint mode, say, the dynamic cursor should be placed as close as possible to the "P" symbol on the right hand side of the screen and the right hand button pressed. If the "P" appears in inverse mode, then the function has been selected. If not, then the dynamic cursor is not near enough to the correct position and you should try again. The program checks the position of the cursor and will only allow functions to be selected if the cursor is to the right of the 225th screen dot. Push the stick hard to the right and move the cursor up and down the line next to the letters to ensure reliable selection.

The currently selected function is denoted by the fact that the appropriate letter is inverted (i.e. appears as a black letter on a green background).

## **The functions**

### **1) Draw**

This is the mode which is selected at the start of the program. This mode allows straight lines, rectangles and circles to be drawn, (it also allows text to be positioned, but this will be covered later).

Initially, line mode is selected and this is shown by the inverted triangle at the bottom of the screen pointing to the "line" symbol. Lines are drawn which join the centres of the two cursors.

To draw a line, first position the two cursors (using the left hand pushbutton to move the static cursor). Pressing the right hand button will cause a line to be drawn.

Note that the static cursor now automatically moves to the dynamic cursor position. This allows connecting lines to be drawn easily, just by moving the dynamic cursor and pressing the right hand push button. If separate lines are required the static cursor's position should be changed after each line is drawn.

Mistakes can be corrected by selecting the Erase function. Notice that the whole drawing is re-drawn each time a line is erased. This is because the sequence of actions is stored in an array and erasure is performed by reducing the count of array entries by one. The penalty for using this simple method is that the whole picture must be re-drawn each time an erasure is made. Notice also that it is the **last** action which is erased each time the Erase function is selected. This means that an error several moves back can be erased by repeatedly selecting the Erase function, but that all subsequent moves must be re-entered.

To change from drawing lines to drawing either rectangles or circles, simply move the dynamic cursor to a point just above the symbol for the required function and press the right hand button. Selection of the new function will be denoted by the triangle pointer moving to a position above the selected function.

Control of rectangle drawing is achieved by placing the two cursors at the opposite corners of the rectangle. The static cursor is taken to indicate the required position of the top left hand corner of the rectangle and the dynamic cursor will mark the bottom left corner. The two cursors can be positioned at will and the rectangle will not be drawn until the right hand button is pressed. Again, select Erase to remove any unwanted rectangle.

Circles are drawn in a similar way to rectangles, but the dynamic cursor marks the centre and the static cursor the radius of the circle drawn. Erase works with circles too.

## 2) Erase

Most of the features of erase have been described above, but it is worth noting that the method adopted to implement erasure means that it will work with any command. Note, however, that all commands are stored and this



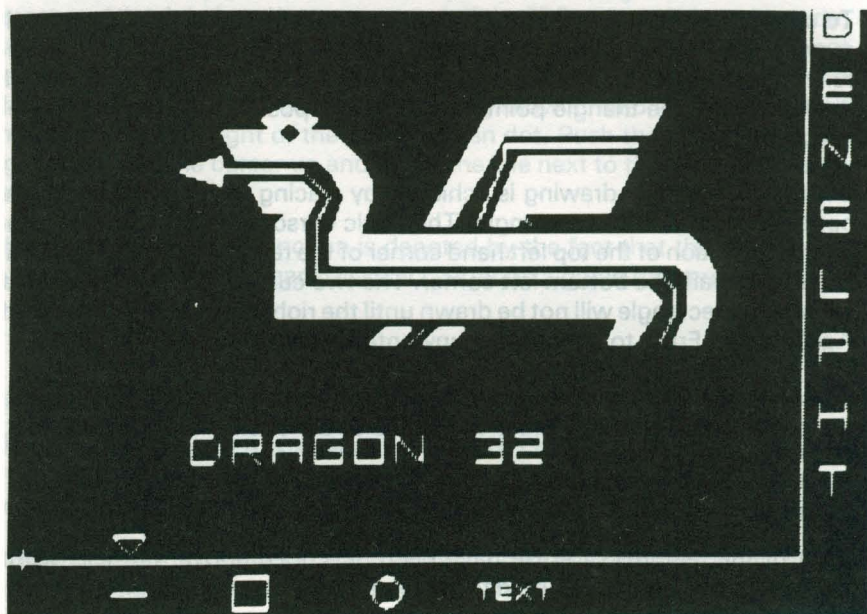
includes any "messaging about" with cursors whilst they are being correctly positioned. This must be remembered when erase appears not to work: you may be erasing cursor movements, not the line drawing that you expected to erase.

### 3) New

Be careful with this one!. Not only does it erase the screen, it also forgets all commands entered to date. It is an easy way to lose everything. Use it when you want to scrap everything and start again.

### 4) Save

Selecting the Save option will cause a change over to the text screen and the user will be asked to enter a filename under which the list of commands is to be saved. The data will be saved to tape or to disc, depending upon the value given to the variable DV in line 30. Set DV to 1 for disc or to -1 to use tape for storage.





### 5) Load

The user will be asked for a filename to load a command list from. There must be a file of this name on the filing system in use (tape or disc). If no file is found the program will fail with an error message. Unfortunately, due to the lack of an ON ERROR GOTO command in Dragon or Color Computer BASIC, no error trapping is performed. This could be solved by using the tape read routines directly (as in the Convertor program in Chapter 4).

Once the file has been found the data will be loaded, the screen cleared, and the picture re-drawn. Note that this will clear any existing drawing.

### 6) Paint

Once the Paint function has been selected the dynamic cursor can be positioned anywhere on the screen and will define the area to be "painted". Pressing the right hand button will cause "paint" to spread outwards from the cursor towards the edges of the screen. It will stop at any lines. Be careful, as "paint" can spill past any boundaries which are not properly closed. See the description of the PAINT command in the computer's manual for a further discussion on this.

As many areas as necessary can be painted simply by re-positioning the cursor and pressing the right hand button. To cancel Paint mode, position the cursor next to the Draw command (D) and press the right hand button. Erase will remove the last area painted.

### 7) Hard copy

The part of the program dealing with hard copy assumes that a Tandy Model DMP 100 printer is being used. Selection of the "H" option will ask the user to ready the printer before data is sent to it. Pressing "ENTER" will cause the picture part of the screen to be printed (i.e. excluding the menu of commands and the symbols at the bottom of the screen).

The Tandy DMP100 printer will enter graphics mode when sent the character CHR\$(18). Following this it expects the top bit of each 8 bit word to be set to 1 and the lower 7 bits to represent the dot pattern to be printed. The part of the program which dumps the screen to the printer (lines 12080 to 12180) examines each dot on the screen and uses this information to form 8 bit words to send to the printer, the top bit always being set. It would be quite easy to change this part of the program to allow any graphics printer to be used (e.g.

one of the Epson MX80 FT series). This could be done by sending the correct character needed to put the printer in graphics mode (do this at line 12070), and rewriting lines 12080 to 12180 to assemble characters in the form expected by the printer (see printer manual).

Note that this method of dumping to the printer is slow, because BASIC takes a long time to examine each point on the screen.

## 8) Text entry

Text can be entered and placed at any point on the screen by use of the "T" command.

The user will be asked to enter a message to be displayed. This can be up to 36 characters long (i.e. one text screen line + 4 characters). Type in the text and press "ENTER". The screen display will revert to graphics and the text can then be positioned by means of the two cursors. The program will start the text from the static cursor position and will write it from left to right. The size of the displayed text will be controlled by the distance between the static and dynamic cursors. The program will attempt to end the text at the dynamic cursor, but this will not always be possible. There are 16 different text sizes and it is useful to try writing the same letters in the different sizes by changing the distance between cursors. Only letters A to Z, numbers 0 to 9 and space are currently allowed.

Text mode will be retained (with the same message) until another function is selected. The next function could be another Text command with a different message. As Text mode is actually part of Draw mode, further Draw commands will result in the last text being displayed. Text mode is cancelled when line, rectangle or circle drawing is selected.

## Program listing

```
10 CLEAR 1000:PCLEAR4
20 DIM L$(38),B(10,10),T(10,10),C(10,10),D(10,10),S(10,10),E(10,
10),Z(5,5),Q(400,3),Q$(400)
30 EX = 1:QP = 1:EF = 0:DV = 1:REM DISK = 1,TAPE = -1
40 PMODE 4,1:PCLS:SCREEN1,0
50 GOSUB 20120 : REM SET UP CHARACTER STRINGS
60 CX = 108:CY = 80:DF = 1:TX = 30:OX = 5:OY = 5:FU = 1:OF = 1
70 GET(0,0)-(9,9),D,G:GET(0,0)-(9,9),E,G
```



```

80 LINE(0,0)-(235,170),PSET,B
90 LINE(30,180)-(40,180),PSET : REM LINE DRAWING SYMBOL
100 LINE(65,175)-(75,185),PSET,B : REM SQUARE SYMBOL
110 CIRCLE(110,180),5 : REM CIRCLE SYMBOL
120 DRAW "S2BM137,182" + L$(20) + L$(5) + L$(24) + L$(20)
130 DRAW "S4" : REM SET SCALE
140 DRAW "BM240,10" + L$(4)
150 DRAW "BM240,30" + L$(5)
160 DRAW "BM240,50" + L$(14)
170 DRAW "BM240,70" + L$(19)
180 DRAW "BM240,90" + L$(12)
190 DRAW "BM240,110" + L$(16)
200 DRAW "BM240,130" + L$(8)
210 DRAW "BM240,150" + L$(20)
220 REM SELECT LINE DRAWING
230 DRAW "BM35,168E5L10F5" : REM TRIANGLE SELECTION
    SYMBOL
240 GET(30,159)-(39,168),T : REM PICK UP TRIANGLE
250 PUT(237,1)-(250,12),Z,NOT
260 DRAW "BM108,80D6U3R3L6" : REM DYNAMIC CURSOR
270 GET(104,80)-(112,90),C,G : REM PICK UP DYNAMIC CURSOR
    SYMBOL
280 DRAW "BM0,0F9BM10,0G10"
290 GET(1,1)-(9,9),S,G : REM PICK UP STATIC CURSOR SYMBOL
300 GOSUB 1000 : REM SAMPLE CONTROLS AND UPDATE
    CURSORS
310 ON FU GOSUB 3000,4000,5000,6000,7000,8000,12000,13000,
    9000,10000,11000
320 REM ABOVE STATEMENT GOES TO FUNCTION SELECTED BY
    VALUE
330 REM OF "FU" RETURNED BY THE CONTROL SCAN ROUTINE
340 IF FU = 3 THEN 30
350 IF FU = 2 OR FU = 5 THEN 40 ELSE 300
1000 REM SAMPLE JOYSTICKS AND BUTTONS
1010 REM CHECK FOR FUNCTION SELECT AND RETURN VALUE OF
    FUNCTION IN THE VARIABLE "FU"
1020 IF EF = 1 THEN 1400
1030 PX = JOYSTK(0):PY = JOYSTK(1)
1040 B = PEEK(65280)
1050 IF B = OB THEN 1100
1060 OB = B:IF B = 255 OR B = 127 THEN 1100

```



```
1070 Q(QP,1)=PX:Q(QP,2)=PY:Q(QP,3)=B
1080 QP=QP+1
1090 IF QP>400 THEN PRINT "OUT OF COMMAND SPACE":STOP
1100 IF B=255 OR B=251 OR B=127 THEN B1=0:B2=0:GOTO1150
1110 IF (B=254 OR B=126) AND B1=0 THEN SOUND 127,1:B1=1:
B2=0:CA=0:GOTO 1150
1120 IF (B=253 OR B=125) AND B2=0 THEN SOUND 64,1:B2=1:
B1=0:CA=0:GOTO 1150
1130 IF B=252 OR B=124 THEN 1140 ELSE 1150
1140 B1=1:B2=1:CA=1 : REM CANCEL IF BOTH BUTTONS
PUSHED
1150 PUT(CX-4,CY-4)-(CX+6,CY+6),D,PSET
1160 CX=4*PX:CY=4*PY
1170 IF CX>230 THEN CX=230
1180 IF CX<5 THEN CX=5
1190 IF CY>170 THEN CY=170
1200 IF CY<5 THEN CY=5
1210 GET(CX-4,CY-4)-(CX+6,CY+6),D,G
1220 PUT(CX-4,CY-4)-(CX+4,CY+4),C,OR
1230 PUT(CX-4,CY-4)-(CX+4,CY+4),C,AND
1240 IF CX<225 AND CY<160 THEN 1270
1250 IF CX>225 AND B1=1 THEN GOSUB 1330
1260 IF CY>160 AND B1=1 THEN GOSUB 1370
1270 IF CA=1 THEN FU=1 : REM BACK TO DRAW MODE ON
CANCEL
1280 IF OF=FU OR FU>8 THEN RETURN
1290 PUT(237,(OF-1)*20+1)-(250,(OF-1)*20+12),Z,NOT
1300 PUT(237,(FU-1)*20+1)-(250,(FU-1)*20+12),Z,NOT
1310 OF=FU
1320 RETURN
1330 IF CY<10 THEN FU=1 ELSE FU=INT((CY-10)/20)+2:IF FU>8
THEN FU=8
1340 IF FU<1 THEN FU=1
1350 IF FU=4 OR FU=5 OR FU=7 THEN QP=QP-2:IF QP<1 THEN
QP=1
1360 RETURN
1370 IF CX<30 THEN FU=9 ELSE FU=INT((CX-20)/40)+9: IF
FU>11 THEN FU=11
1380 IF FU<9 THEN FU=9
1390 RETURN
1400 IF EX=QP THEN EF=0:GOTO 1030
```

```

1410 PX = Q(EX,1):PY = Q(EX,2):B = Q(EX,3)
1420 EX = EX + 1
1430 GOTO1100
3000 REM DRAW ROUTINE
3010 IF B1<>1 AND B2<>1 THEN RETURN
3020 IF CX>225 OR CY>160 THEN RETURN
3030 PUT(CX-4,CY-4)-(CX+6,CY+6),D,PSET
3040 PUT(OX-4,OY-4)-(OX+6,OY+6),E,PSET
3050 IF B2 = 1 THEN 3150
3060 REM DRAW LINE,SQUARE,CIRCLE OR TEXT
3070 ON DF GOTO 3080,3110,3130,3240
3080 REM DRAW A LINE FROM STATIC ORIGIN TO DYNAMIC
CURSOR
3090 LINE(OX,OY)-(CX,CY),PSET
3100 GOTO3190
3110 LINE(OX,OY)-(CX,CY),PSET,B
3120 GOTO 3190
3130 CIRCLE(CX,CY),SQR((OX-CX)*(OX-CX)+(OY-CY)*(OY-CY))
3140 GOTO 3190
3150 REM MOVE STATIC ORIGIN
3160 IF OX = CX AND OY = CY THEN RETURN
3170 PUT(CX-4,CY-4)-(CX+6,CY+6),D,PSET
3180 PUT(OX-4,OY-4)-(OX+6,OY+6),E,PSET
3190 OX = CX:OY = CY
3200 GET(OX-4,OY-4)-(OX+6,OY+6),E,G
3210 PUT(OX-4,OY-4)-(OX+4,OY+4),S,OR
3220 GET(OX-4,OY-4)-(OX+6,OY+6),D,G
3230 RETURN
3235 REM DRAW TEXT STRING BETWEEN CURSORS
3240 PUT(CX-4,CY-4)-(CX+6,CY+6),D,PSET
3250 PUT(OX-4,OY-4)-(OX+6,OY+6),E,PSET
3260 Z$ = "BM" + RIGHT$(STR$(OX),LEN(STR$(OX))-1) + "," +
RIGHT$(STR$(OY),LEN(STR$(OY))-1)
3270 S = ABS(OX-CX)
3280 S = INT(4*S/(LEN(X$)*12)):IF INT(S/2)<S/2 THEN S = S + 1
3290 IF S<2 THEN S = 2
3300 Z$ = Z$ + "S" + RIGHT$(STR$(S),LEN(STR$(S))-1)
3310 DRAW Z$
3320 FOR I = 1 TO LEN(X$)
3330 TP = 0
3340 CD = ASC(MID$(X$,I,1))

```

```
3350 IF CD = 32 THEN TP = 27
3360 IF CD > 47 AND CD < 58 THEN TP = 27 + CD - 47
3370 IF CD > 64 AND CD < 91 THEN TP = CD - 64
3380 IF TP > 0 THEN DRAW L$(TP)
3390 NEXT I
3400 GOTO 3190
4000 REM ERASE BY MOVING QUEUE POINTER BACK TWO
PLACES AND THEN RE-DRAWING
4010 QP = QP - 2: IF QP < 1 THEN QP = 1
4020 EF = 1: EX = 1
4030 RETURN
5000 RETURN
6000 REM SAVE PICTURE TO DISK
6010 SCREEN 0,0:CLS
6020 PRINT "SAVE SCREEN TO ";
6030 IF DV = 1 THEN PRINT "DISK" ELSE PRINT "TAPE"
6040 GOSUB 20000
6050 IF N$ = "" THEN QP = QP + 1: GOTO 6170
6060 OPEN "O", #DV, N$
6070 FOR I = 1 TO QP
6080 PRINT #DV, Q(I,1)
6090 PRINT #DV, Q(I,2)
6100 PRINT #DV, Q(I,3)
6110 PRINT #DV, Q$(I)
6120 NEXT I
6130 CLOSE #DV
6140 PRINT "SAVE COMPLETE"
6150 PRINT "TYPE SPACE TO RETURN TO GRAPHICS"
6160 IF INKEY$ < > " " THEN 6160
6170 SCREEN 1,0
6180 FU = 1
6190 RETURN
7000 REM LOAD PICTURE
7010 SCREEN 0,0:CLS
7020 PRINT "LOAD SCREEN FROM ";
7030 IF DV = 1 THEN PRINT "DISK" ELSE PRINT "TAPE"
7040 GOSUB 20000
7050 IF N$ = "" THEN QP = QP + 1: FU = 1: GOTO 7180
7060 OPEN "I", #DV, N$
7070 QP = 1
7080 IF EOF(DV) THEN 7130
```



```

7090 INPUT #DV,Q(QP,1),Q(QP,2),Q(QP,3):LINE INPUT #DV,Q$
(QP)
7100 QP=QP+1
7110 IF QP>400 THEN 7190
7120 GOTO 7080
7130 CLOSE #DV
7140 PRINT"LOAD COMPLETE"
7150 PRINT"TYPE SPACE TO RETURN TO GRAPHICS"
7160 IF INKEY$<>" " THEN 7160
7170 EF=1:EX=1
7180 SCREEN 1,0:RETURN
7190 PRINT"OUT OF SPACE":STOP
8000 IF CX>225 OR CY>160 THEN RETURN
8010 IF B1<>1 THEN RETURN
8020 PUT(CX-4,CY-4)-(CX+6,CY+6),D,PSET
8030 PUT(OX-4,OY-4)-(OX+6,OY+6),E,PSET
8040 PAINT(CX+2,CY+2)
8050 GET(OX-4,OY-4)-(OX+6,OY+6),E,G
8060 PUT(OX-4,OY-4)-(OX+4,OY+4),S,OR
8070 GET(CX-4,CY-4)-(CX+6,CY+6),D,G
8080 RETURN
9000 REM SELECT LINE DRAWING
9010 DF=1
9020 PUT(TX,158)-(TX+10,168),B
9030 TX=30
9040 PUT(TX,158)-(TX+9,167),T
9050 FU=1
9060 RETURN
10000 REM SELECT SQUARE DRAWING
10010 DF=2
10020 PUT(TX,158)-(TX+10,168),B
10030 TX=62
10040 GOTO 9040
11000 REM SELECT CIRCLE DRAWING
11010 DF=3
11020 PUT(TX,158)-(TX+10,168),B
11030 TX=100
11040 GOTO 9040
12000 REM HARD COPY
12010 SCREEN 0,0:CLS
12020 PRINT"READY PRINTER"

```

```
12030 PRINT "TYPE SPACE TO BEGIN HARD COPY"
12040 PRINT "TYPE E TO EXIT"
12050 A$ = INKEY$: IF A$ = "E" THEN QP = QP + 1: GOTO 12200
12060 IF A$ < > " " THEN 12050
12070 PRINT #2, CHR$(18);
12080 FOR SY = 0 TO 168 STEP 7
12090 FOR SX = 0 TO 235
12100 PL = 128: W = 64
12110 FOR BT = 0 TO 6
12120 IF PPOINT(SX, SY + 6 - BT) < > 0 THEN PL = PL + W
12130 W = W / 2
12140 NEXT BT
12150 PRINT #2, CHR$(PL);
12160 NEXT SX
12170 PRINT #2
12180 NEXT SY
12190 PRINT #2, CHR$(30)
12200 FU = 1: SCREEN 1, 0
12210 RETURN
13000 REM TEXT ENTRY
13010 IF EF = 1 THEN FU = 1: DF = 4: X$ = Q$(EX): RETURN
13020 SCREEN 0, 0
13030 CLS
13040 PRINT "TEXT ENTRY(< RETURN> TO EXIT)"
13050 LINE INPUT "ENTER TEXT> "; X$
13060 IF X$ = "" THEN QP = QP - 2: FU = 1: RETURN
13070 Q$(QP) = X$
13080 DF = 4
13090 PUT(TX, 158) - (TX + 10, 168), B
13100 TX = 140
13110 PUT(TX, 158) - (TX + 10, 168), T
13120 SCREEN 1, 0
13130 FU = 1
13140 RETURN
20000 REM GET FILENAME
20010 PRINT "TYPE < RETURN> TO EXIT"
20020 LINE INPUT "FILENAME? > "; N$
20030 IF N$ = "" THEN RETURN
20040 PRINT "IS "; N$; " CORRECT? ";
20050 A$ = INKEY$
20060 IF A$ = "" THEN 20050
```

```

20070 IF A$ < > "N" AND A$ < > "Y" THEN 20050
20080 PRINT A$
20090 IF A$ = "N" THEN 20020
20100 RETURN
20110 REM CHARACTER SET
20120 L$(1) = "U8R8D4L8BR8D4BR4"
20130 L$(2) = "U8R6F2D2L8BR8D2G2L6BR12"
20140 L$(3) = "U8R8BD8L8BR12"
20150 L$(4) = "U8R6F2D4G2L6BR12"
20160 L$(5) = "U8R8BD4L8BD4R8BR4"
20170 L$(6) = "U8R8BD4L8BD4BR12"
20180 L$(7) = "U8R8BD4L4BR4D4L8BR12"
20190 L$(8) = "U8BR8D8BU4L8BD4BR12"
20200 L$(9) = "BU8R8BL4D8BL4R8BR4"
20210 L$(10) = "U4BU4BR8D8L8BR12"
20220 L$(11) = "U8BR8G4L4BR4F4BR4"
20230 L$(12) = "U8BD8R8BR4"
20240 L$(13) = "U8F4E4D8BR4"
20250 L$(14) = "U8F8U8BD8BR4"
20260 L$(15) = "U8R8D8L8BR12"
20270 L$(16) = "U8R8D4L8BD4BR12"
20280 L$(17) = "U8R8D8H4BG4R8BR4"
20290 L$(18) = "U8R8D4L8BR4F4BR4"
20300 L$(19) = "BU4U4R8BD4L8BR8D4L8BR12"
20310 L$(20) = "BU8R8BL4D8BR8"
20320 L$(21) = "U8BR8D8L8BR12"
20330 L$(22) = "BU8D4F4E4U4BD8BR4"
20340 L$(23) = "U8BR8D8H4G4BR12"
20350 L$(24) = "E8BL8F8BR4"
20360 L$(25) = "BU8F4E4BG4D4BR8"
20370 L$(26) = "BU8R8G8R8BR4"
20380 L$(27) = "BR12"
20390 L$(28) = "U8R8G8R8U8BD8BR4"
20400 L$(29) = "R8BL4U8G4BE4BD8BR8"
20410 L$(30) = "BU8R8D4L8D4R8BR4"
20420 L$(31) = "BU8R8D4L4BR4D4L8BR12"
20430 L$(32) = "BR8U4L8U4BR8D4BD4BR4"
20440 L$(33) = "R8U4L8U4R8BD8BR4"
20450 L$(34) = "U4R8D4L8BU4U4R8BD8BR4"
20460 L$(35) = "BU8R8D8BR4"
20470 L$(36) = "U8R8D8L8BU4R8BD4BR4"

```



```
20480 L$(37) = "BR8U8L8D4R8BD4BR4"
20490 RETURN
```

### Program description

<i>Lines</i>	<i>Purpose</i>
10	Clear lots of string space (for the text). Use four graphics pages.
20	Dimension arrays.
30	Set erase execution pointer (EX) to 1. Set normal mode command queue pointer (QP) to 1. Set storage device (DV) to tape or disc.
40	Set up graphics screen and turn it on.
50	Initialise alphanumeric strings for text on graphics screen.
60	Set cursor X,Y position (CX,CY). Set Draw mode to line drawing (DF = 1). Set X coordinate of triangle symbol (TX = 30). Set static cursor position to 5,5 (OX = 5,OY = 5). Select Draw mode (FU = 1). Set "last function executed" to Draw (OF = 1).
70	Initialise the cursor blanking arrays (D and E).
80-290	Draw menus, symbols, and cursors. Pick up arrays C and S containing the dynamic and static cursors respectively.
300	Call routine which samples joysticks, moves cursors & checks for function select.
310	Despatch functions to individual routines.
340	If "New" then clear everything and re-start.
350	If "Erase" or "Load" then just clear screen and re-start.
1000	Main routine which is being called continually to check and update cursors and buttons.
1020	Special case if the Erase flag is set (EF = 1).
1030	Get joystick settings.
1040	Test buttons.
1050	Check if new button setting (B) is same as the previous one (OB).
1060	Come here if button just pressed, update OB.
1070	Place joystick settings (PX, PY) and button value (B) in command queue (Q).
1080	Update queue pointer (QP).

- 1090 Complain if too many commands (Can anyone enter this number of commands?).
- 1100 If no buttons pressed then set button flags (B1 and B2) to zero.
- 1110 If button 1 (right hand) pressed then beep and set flags.
- 1120 Likewise for button 2 (left hand) but lower beep.
- 1130-1140 Check for both buttons pressed.
- 1150 Remove the dynamic cursor from its current position by PUT-ting the array D on the screen. (The array D will always hold what was on the screen "beneath" the dynamic cursor).
- 1160 Update dynamic cursor coordinates. (Note that the X,Y position of the cursor can only move in 4 screen dot increments, due to the joystick only returning values 0..63).
- 1170-1200 Check for screen limits and make sure the cursor cannot enter the menu borders of the screen.
- 1210 Take a copy of the screen below the new cursor position.
- 1220 Draw the dynamic cursor symbol by OR-ing it with the screen.
- 1230 Make sure the cursor still shows up even on areas of the screen which have been painted.
- 1240 Check if the cursor is near either the right hand edge or the bottom line of the screen, don't check for a command entry if not.
- 1250 If on right hand edge and button 1 is pressed then must be entering a new command.
- 1260 Same, but changing the drawing mode.
- 1270 Cancel command if both buttons pressed (only useful when in Paint mode).
- 1280 See if the new function (FU) is the same as the old one (OF) or if the function selected is not valid (i.e.  $FU > 8$ ). If either of these conditions prevail then leave the scan routine.
- 1290 Make the letter corresponding to the old function appear in normal mode (white on black).
- 1300 Invert the letter of the new function.
- 1310 Update the old function variable.
- 1320 Leave the scan routine.



1330-1360	Check for function selected on right hand edge of screen, return its value in variable FU.
1370-1390	As above but for bottom line of screen (i.e. the drawing functions).
1400	Come here when erasing the last command. This is done by re-executing all the commands in the queue but one. The variable EX is used to scan through Q and pick out the values of PX, PY and B, just as if they were being entered from the joysticks. When EX = QP the erase is completed and the erase flag is reset (EF = 0).
1410	Pick up commands from the queue.
1420	Step to next entry.
1430	Execute the command via the normal routines.
3000	The Draw routines.
3010	Don't do anything until a button is pressed.
3020	Don't trample on the borders.
3030-3040	Wipe out the dynamic and static cursors.
3050	Check for movement of static cursor.
3070	Use variable DF to select line drawing, rectangles circles or text.
3090	Draw a line joining the two cursors.
3110	As for line but draw a box.
3130	Circle of radius equal to separation of the two cursors.
3160	Check if two cursors already superimposed, don't bother moving them if they are.
3170-3180	Remove both cursors from the screen.
3190	Move the static cursor to the dynamic cursor position.
3200	Get the background behind the static cursor.
3210	Draw the static cursor.
3220	Draw the dynamic cursor.
3240-3250	Remove cursors prior to drawing the text.
3260	Start the draw command from the static cursor position.
3270-3290	Set the drawing scale (S) to approximate the distance between cursors.
3300	Form the string (Z\$) defining start position and scale.
3310	Draw it.
3320-3390	Take the entered text (X\$) and check for characters other than A-Z, 0-9. If valid, draw character using array L\$ for draw commands.
3400	Replace cursors and leave.



4000 Erase function.

4010 Move queue pointer back two places. (Two places because we want to skip the Erase command as well as the command before it).

4020 Set Erase flag and Erase queue pointer.

4030 Returning with EF set will cause next call to routine at 1000 to take commands from the queue array (Q) rather than from the joysticks.

5000 New does nothing but return with FU = 3 which causes line 340 to clear everything down.

6000 Save command.

6010 Use text screen.

6020-6030 Put up Save message.

6040 Call routine to get a filename into the string N\$.

6050 Allow for false entry to Save by allowing a null string to terminate the command.

6060 Open the file.

6070-6120 Save the queue data to tape/disc. The strings for text entry are stored separately in the array Q\$.

6130 Close the file.

6140-6190 Reselect graphics screen, go back to draw mode by setting FU = 1.

7000-7190 Load is handled generally as for Save. Data is read from the file into the arrays Q and Q\$.

8000 The Paint command. Check for cursor on either border area.

8010 If button 1 not pressed then loop back until it is.

8020-8030 Remove cursors.

8040 Paint the required area.

8050-8080 Pick up the new screen under the cursors. Re-write the cursors and return.

9000 Come here if changing draw mode to line drawing.

9010 Set Draw mode to 1 (i.e. lines).

9020 Blank out the current triangle symbol.

9030-9040 Re-draw the triangle symbol over the line symbol.

9050-9060 Select Draw mode and return.

10000-11040 As above but select rectangle or circle modes.

12000 Hard copy routine.

12010-12060 Select text screen, prompt user. Allow for escape.

12070 Put printer into graphics mode (Tandy Model DMP100 only).

12080-12180	Pick up screen display dot by dot and form 8 bit characters with the top bit set (PL = 128 does this). Send characters to printer.
12190	Finished, put printer back in normal mode (DMP100).
12200-12210	Back to Draw mode and graphics screen.
13000	Text entry function.
13010	If in Erase mode (EF = 1) then set command to Draw with DF = 4 (i.e. draw text). Return and let other routines handle the drawing.
13020	Come here if entering new text string.
13030-13060	Prompt user and get a string into X\$. Allow for a null entry to terminate this mode.
13070	Put new string into string array.
13080	Set Draw command to text mode.
13090-13140	Move triangle symbol to point to text symbol. Back to graphics and Draw function.
20000-20100	Get a filename and ask user to check it.
20110-20490	Subroutine to set up graphics text array L\$.

## Variable definitions

Variable name	Function
L\$	String array used to hold commands which draw text on the graphics screen.
B,T,C,D, S,E,Z	Arrays used to blank screen and hold symbols for cursors etc.
Q	Array of commands entered.
Q\$	Array of text strings entered.
QP,EX	Queue pointer and erase function pointer.
EF	Flag which denotes Erase mode when set (EF = 1).
DV	Storage device number.
CX,CY	Dynamic cursor coordinates.
OX,OY	Static cursor coordinates.
DF	Draw function mode selector (1 = line, 2 = rectangle, 3 = circle, 4 = text).
TX	Position of triangle symbol at bottom of screen.
FU	Function selector (1..8 = D,E,N,S,L,P,H,T).
OF	Last function selected.



PX,PY	Joystick X,Y values.
B	Button value.
OB	Last button value.
B1	Right hand button pressed flag.
B2	Left hand button pressed flag.
CA	Both buttons pressed flag.
Z\$	String used in drawing text.
S	Scale factor for text drawing.
X\$	Text entry string.
CD	Used to check characters in X\$ for A-Z,0-9.
N\$	File name.
A\$	String used in hard copy routine.
SX,SY	Coordinates of screen point being examined in hard copy routine.
PL 8	bit value built up from screen dots.
W	Weighting factor used in building PL.

## Points of interest

It is worth noting the technique of using a main routine for checking command entries and separate routines for command execution. This is a useful technique as it allows the functions to be separated and tested independently. Notice that other programs in later chapters make extensive use of this method.

Note also the way in which the cursors are handled to allow them to be positioned anywhere on the screen without affecting the screen content. This was probably the most awkward part of the program to develop, as the GET and PUT routines can sometimes have unexpected results.

## THREED

This short program allows any "wire frame" object to be drawn on the graphics screen in three dimensional representation. It formed the starting point for the routines in the Sub Hunt game in Chapter 1 and is based on the article in BYTE magazine mentioned earlier.

A wire frame object is one apparently made up of many interconnected wires. It can be three dimensional, and is always transparent. The program listed



below describes such an object in terms of it's "vertices" and it's "edges". The vertices are the points at which two wires meet and the edges are the wires themselves.

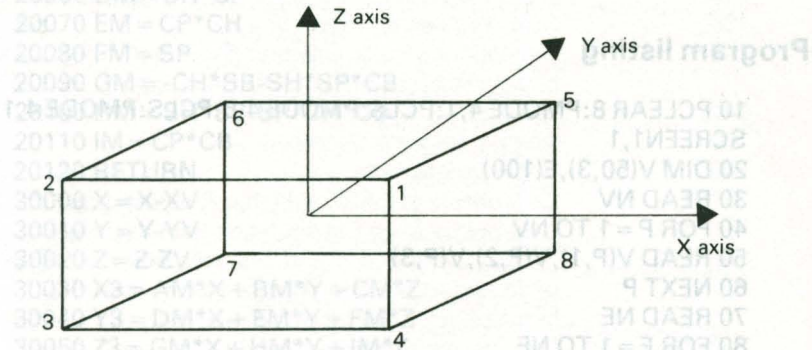
To describe an object it is best to draw it on graph paper and assign numbers to the vertices. In the case of the rectangular box drawn in the program below, the vertices are numbered 1 to 8. The X, Y and Z coordinates of each vertex must now be written down and used to build data statements describing the object. Finally the edges must be defined by specifying which vertices are joined together.

Take the DATA statements describing the box as an example:-

```
30090 DATA 8
30100 DATA 5.25,-2,3.25
30110 DATA -5.25,-2,3.25
30120 DATA -5.25,-2,-3.25
30130 DATA 5.25,-2,-3.25
30140 DATA 5.25,2,3.25
30150 DATA -5.25,2,3.25
30160 DATA -5.25,2,-3.25
30170 DATA 5.25,2,-3.25
30180 DATA 19
30190 DATA -1,2,3,4,1,5,6,7,8,5,-2,6,-3,7,-4,8
30200 DATA 6,-7,5
```

Line 30090 states that there are 8 vertices forming the object. Line 30100 defines the X,Y and Z coordinates of vertex 1 to be 5.25,-2 and 3.25 respectively. Similarly lines 30110 to 30170 describe the other 7 vertices as shown in the diagram overleaf.

Line 30180 defines the number of edges. Note that not all of these are plotted. Any entry in lines 30190 and 30200 which is preceded by a negative sign means that a line is not plotted, but plotting will continue from the vertex number given. This is necessary in order to minimise the number of moves needed to plot the object. Thus the first item (-1 on line 30190) means "move to the first vertex without drawing a line". The next item (2 on the same line) means draw a line to vertex 2 and so on until the data is exhausted. Note that the last four edges (8,6,-4,5) put a cross on one face of the box to make visualisation easier.



The reader is referred to the original article (see reference in Chapter 1, Sub Hunt) for a detailed description of the projection from a three dimensional object to the two dimensional computer screen. However, the information given above should be sufficient to allow the reader to create other objects.

## Controls

The viewing angle is controlled by the joysticks. The viewer is assumed to move on the surface of a hemisphere surrounding the object. Vertical movement of the right hand joystick controls the viewer's pitch. The heading is altered by horizontal movement of the left joystick. As the viewer moves on a hemisphere he or she is always facing the object and at a constant distance from it. Try moving one stick slowly and then the other.

Line 290 ensures that the screen always shows the last view plotted. It does this by flipping pages on the graphics screen so that the plotting is always done invisibly (apart from the first time after typing RUN). If you prefer to see the plotting going on, change line 290 to read:-

```
290 FOR I=1 TO 1000:NEXT I
```

This delay loop will give time to see the results before the screen is cleared ready for the new plot.

## Program listing

```
10 PCLEAR 8:PMODE 4,1:PCLS:PMODE 4,5:PCLS:PMODE 4,1:
SCREEN1,1
20 DIM V(50,3),E(100)
30 READ NV
40 FOR P=1 TO NV
50 READ V(P,1),V(P,2),V(P,3)
60 NEXT P
70 READ NE
80 FOR E=1 TO NE
90 READ E(E)
100 NEXT E
110 PG=1
120 PMODE 4,PG:PCLS:LINE -(0,0),PRESET
130 D=75:Z=JOYSTK(0)
140 P=6.28*(JOYSTK(1)+1)/64-3.1416
150 B=0
160 H=6.28*(JOYSTK(2)+1)/64
170 GOSUB 20000
180 XV=-D*CP*SH
190 YV=-D*CP*CH
200 ZV=-D*SP
210 FOR E=1 TO NE
220 X=V(ABS(E(E)),1)
230 Y=V(ABS(E(E)),2)
240 Z=V(ABS(E(E)),3)
250 GOSUB 30000
260 IF E(E)>0 THEN LINE (U1,V1)-(U,V),PSET
270 U1=U:V1=V
280 NEXT E
290 SCREEN1,1:PG=PG+4:IFPG>5 THEN PG=1
300 GOTO120
20000 CH=COS(H):SH=SIN(H)
20010 CP=COS(P):SP=SIN(P)
```



```

20020 CB = COS(B):SB = SIN(B)
20030 AM = CB*CH-SH*SP*SB
20040 BM = -CB*SH-SP*CH*SB
20050 CM = CP*SB
20060 DM = SH*CP
20070 EM = CP*CH
20080 FM = SP
20090 GM = -CH*SB-SH*SP*CB
20100 HM = SH*SB-SP*CH*CB
20110 IM = CP*CB
20120 RETURN
30000 X = X-XV
30010 Y = Y-YV
30020 Z = Z-ZV
30030 X3 = AM*X + BM*Y + CM*Z
30040 Y3 = DM*X + EM*Y + FM*Z
30050 Z3 = GM*X + HM*Y + IM*Z
30060 U = 135 + 13.5*D*X3/Y3
30070 V = 80 - 11.5*D*Z3/Y3
30080 RETURN
30090 DATA 8
30100 DATA 5.25,-2,3.25
30110 DATA -5.25,-2,3.25
30120 DATA -5.25,-2,-3.25
30130 DATA 5.25,-2,-3.25
30140 DATA 5.25,2,3.25
30150 DATA -5.25,2,3.25
30160 DATA -5.25,2,-3.25
30170 DATA 5.25,2,-3.25
30180 DATA 19
30190 DATA -1,2,3,4,1,5,6,7,8,5,-2,6,-3,7,-4,8
30200 DATA 6,-7,5

```

## Program description

Lines	Purpose
10	Clear 8 pages to allow two screens in MODE 4. Clear both screens. Start on screen 1.
20	Dimension arrays for object description.

## Hot programs to feed your Dragon

30	Read number of vertices.
40-60	Read in vertex coordinates.
70	Read number of edges.
80-100	Read in the edge data.
110	PG is the screen number. Start on page 1.
120	Select screen and clear it.
130	Set distance from viewer to TV screen. Dummy read of Joystick 0 (erratic results otherwise).
140	Get pitch angle.
150	Always set bank to zero.
160	Get heading angle.
170	Call routine at 20000 which calculates projection parameters from the pitch and heading angles.
180-200	Set the viewers X, Y and Z coordinates according to pitch and heading so that viewer moves on a hemisphere surrounding the object.
210-240	Get coordinates of each edge.
250	Transform them and project them onto the TV screen.
260	Draw the line if not a negative edge value.
270	Update the last point plotted (variables U1 and V1).
280	Do it for all edges.
290	Turn the screen on. Flip the page so that next plot occurs on screen not being displayed.
300	Keep doing it.
20000-20120	Routine which transforms the coordinates of a point on the object to the coordinates of a point on an object which has been moved into the "standard position". The standard position occurs when the viewer is at the origin, and the line of sight is along the positive Y axis.
30000-30080	Project one point of the object onto the TV screen. Returns screen coordinates U,V.
30090-30200	Data for box as described above.

## Variable definitions

Variable name	Purpose	Lines
V	Array of vertices.	10
E	Array of edges.	20

NV	Number of vertices.
NE	Number of edges.
PG	Graphics screen page in use.
D	Distance from viewer to TV screen.
P	Pitch angle.
B	Bank angle.
H	Heading angle.
XV,YV,ZV	Viewer's X,Y,Z coordinates.
X,Y,Z	Coordinates of point to be plotted, prior to transformation to the "standard position".
CP,CB,CH	Cosines of pitch,bank and heading.
SP,SB,SH	Sines of pitch,bank and heading.
U,V,U1,V1	Coordinates used in screen plotting.
AM,BM...	Transformation parameters used to move point on object to the standard position.
..IM	
X3,Y3,Z3	Coordinates of point when moved to the standard position.

## Points of interest

Because of the speed limitations of BASIC the plotting is quite slow. There are two ways of speeding things up. One is to try running the CPU at 1.8 MHz by adding a line thus:-

```
5 POKE 65495,0
```

The snag is that not all Dragon and Color Computer 6809 CPUs are capable of working at 1.8 MHz, so there is just a chance this might cause your computer to "crash". No harm will result if it does, so try it. Remember to enter POKE 65494,0 to revert to normal working before using any input/output devices such as the cassette.

Another way would be to adapt the routines in "Sub Hunt" which pre-calculate the sine and cosine values. This speeds things up a lot, (although in Sub Hunt things are slowed down again by checking things such as push buttons and displaying the timer so you might not notice the speed increase).

Of course, if you really feel adventurous, why not re-write everything in machine code? Send us the listings when you do, please!



## LOGO

The language LOGO has been widely accepted as an ideal way to introduce beginners to the world of computing. This is because it is an interactive language with powerful, easily understood commands. LOGO is being used extensively to teach young children to program and to explore the world of mathematics in a way that fascinates them. We hope that the program presented here will give some insight into this language and perhaps encourage others to explore its potential.

To say that LOGO is a beginner's language is not to belittle it. There are features of LOGO which make it a powerful and useful language capable of being set alongside any of the other languages in current use. Features such as recursion and the well known "Turtle Graphics" are the most notable.

The implementation listed below is by no means a full one. It is the bare minimum needed to begin using LOGO. Nonetheless it could quite easily be used to introduce children (and adults!) to programming in a pleasant way.

### The LOGO environment

Loading and running the program will produce a display with a question mark at the bottom left corner and a strange triangular symbol in the centre of the screen. This is the "turtle". More about turtles and their tricks later on.

The question mark is inviting the user to enter a command. There are several of these and they will be described later. Typing in a command will cause the program to analyse what has been typed to see if it corresponds to an executable command. If the entry is a good one then the program will execute the command and return with the question mark, waiting for further instructions.

Try typing:-

**FORWARD 20< ENTER>**

(Here < ENTER> means press the ENTER key, not type the symbols and letters shown). Make sure there is a space between FORWARD and 20.

The turtle moves up the screen, leaving a line behind it. This means that the program accepted the command "FORWARD 20" and executed it.

When an error is detected in a command, the word "ERROR" will appear at the right hand side of the screen and, after a pause, the command will be erased and the question mark replaced. The command should be re-entered correctly.

Typing errors can be corrected before pressing < ENTER> by pressing the back arrow key, as used in BASIC. To erase a whole line and start again, press < SHIFT> and back arrow at the same time.

## Turtle Graphics

Turtle Graphics is a term used to describe a set of graphics commands which allow patterns and shapes to be drawn with fewer commands than if normal techniques such as PLOT and DRAW were used.

To understand Turtle Graphics it is useful to consider an imaginary turtle which can crawl across the TV screen. In the version presented here the turtle is shown as a triangular symbol with a dot to represent the head. This is vital as it shows which way the turtle is facing.

He or she (remember, it matters to other turtles) has a pen which can write on the TV screen. The turtle can obey commands from the keyboard such as FORWARD, BACK, RIGHT and LEFT. Of course, the distance to travel or the amount to by which to turn has to be specified in the command.

Using these simple commands, complex shapes can be drawn. What is more, the turtle will obey the command right away, which is why the technique proves so fascinating.

## Turtle Graphics commands

**FORWARD < distance>** The turtle will move forward in the current direction by the amount specified. The value of < distance> must be an integer in the range 1 to 999 inclusive. One unit of distance is equivalent to one screen dot. (The screen is 256 dots wide by 192 dots high). Note that there must be a space between the word FORWARD and the value of < distance>. There must be no spaces before the command and none after it.



**BACK < distance>** As for FORWARD, but the turtle moves backwards by the specified amount.

**RIGHT < angle>** The turtle will rotate right (i.e. clockwise) through < angle> degrees.

**LEFT < angle>** The turtle rotates < angle> degrees anti-clockwise.

**PENUP** The pen will be raised and no trace left behind when the turtle moves.

**PENDOWN** The pen is lowered and tracing begins.

**HIDETURTLE** This removes the triangular turtle symbol from the screen, leaving everything else unchanged. This command should be used when the turtle symbol would obscure detail on the drawing. As a side effect, the program speeds up as it no longer has to draw the turtle.

**SHOWTURTLE** The turtle is revealed.

**CIRCLE < radius>** Draws a circle of radius equal to the value supplied. The centre of the circle will be < radius> units beyond the turtle's head.

**CLEARSCREEN** Removes all graphics detail, but leaves the turtle at its current position.

**DRAW** Clears the screen and places the turtle in the centre of the screen, facing vertically upwards.

## **Immediate mode**

All the commands described above can be used in what is called "immediate mode". This means that they can be entered from the keyboard and will produce some obvious effect straight away. There are other immediate commands which will be described later.

To create a LOGO program needs another mechanism whereby commands can be stored and then rapidly executed, just as BASIC does.



## LOGO procedures

A simple LOGO procedure would be:-

```
TO LINE
FORWARD 20
END
```

Here, a procedure called LINE has been created which moves the turtle forward 20 units. Two new commands "TO" and "END" have been defined which signify the start and end of the procedure. The procedure name must follow the word "TO", remembering to include a space, as in the FORWARD command. The length of the procedure name can be up to 16 characters comprising any combination of letters A to Z and numbers 0 to 9, but excluding reserved words such as TO or FORWARD. Examples of valid names are:-

```
FRED JIM 123 SQUARE1 POLYGON QUITEALARGENAME
```

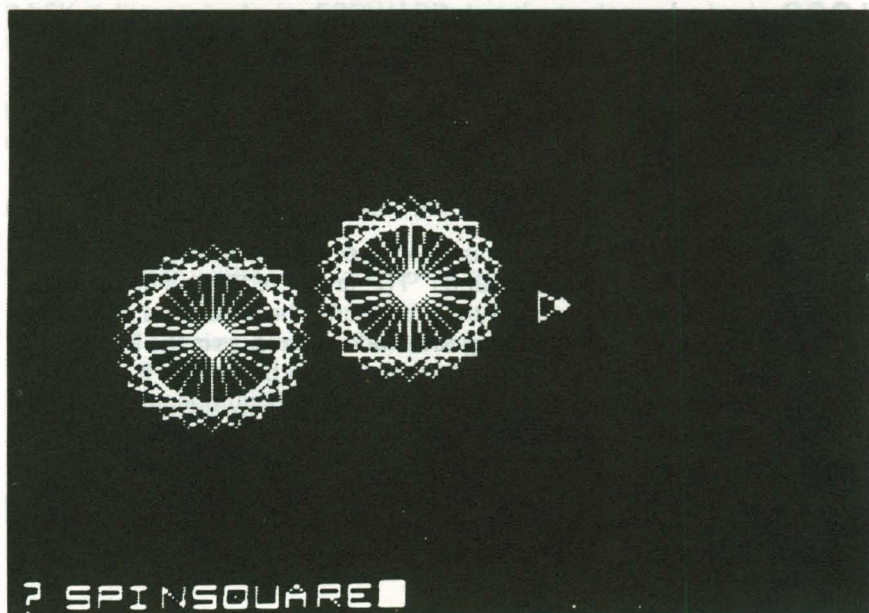
The following are invalid:-

```
RIGHT SUPER-NOVA (**JIM**)
```

After a procedure name (such as FRED) has been declared in the TO command, subsequent commands are stored in an array associated with the name FRED. Commands are executed as they are typed in.

There are two ways of executing a procedure. Either by entering its name in immediate mode, or by calling it from another procedure. To investigate this, type in the following procedure:-

```
TO SQUARE
FORWARD 20
RIGHT 90
FORWARD 20
RIGHT 90
FORWARD 20
RIGHT 90
FORWARD 20
RIGHT 90
END
```



As the procedure square was entered, a square should have been drawn. To execute the procedure in immediate mode, type:-

```
DRAW
SQUARE
```

The DRAW command clears the screen and the procedure SQUARE causes a square to be drawn, with sides 20 units long. To show how one procedure can call another, enter this procedure:-

```
TO SPINSQUARE
DRAW
SQUARE
RIGHT 90
SQUARE
RIGHT 90
SQUARE
```

```

RIGHT 90
SQUARE
RIGHT 90
END

```

Run the SPINSQUARE routine and it should produce four adjacent squares, similar to a house window. This is because the turtle makes a 90 degree turn after drawing each square. This could be modified to produce more interesting patterns by reducing the amount of turn at the end of each square. However, reducing the turn from 90 degrees to, say, 15 degrees would require  $360/15$  (i.e. 24) calls to the SQUARE routine and 24 "RIGHT 15" commands. This could prove tedious to type.

## The REPEAT command

REPEAT allows one or more commands to be repeated a specified number of times. It can be used to duplicate actions as many times as required.

The form of the command is:-

```

REPEAT <repeat count>
[<command 1>
<command 2>
<command 3>
... ]

```

The opening square bracket is typed as <SHIFT> down arrow and the closing bracket is <SHIFT> forward arrow.

The <repeat count> is an integer in the range 1 to 999 which governs the number of times the commands enclosed in square brackets are executed. So, armed with this new command, the SQUARE routine now becomes:-

```

TO SQUARE
REPEAT 4
[FORWARD 20
RIGHT 90]
END

```

The SPINSQUARE routine can be modified to use REPEAT and the smaller turn thus:-



```
TO SPINSQUARE  
REPEAT 24  
[SQUARE  
RIGHT 15]  
END
```

This shows how a small number of LOGO commands can be used to produce an interesting result. Remember, things happen much faster if you use HIDE TURTLE. Finally, to really impress people, type in the following which calls SPINSQUARE (and hence indirectly calls SQUARE):-

```
TO PATTERN  
DRAW  
HIDE TURTLE  
PENUP  
LEFT 90  
FORWARD 25  
RIGHT 90  
REPEAT 8  
[PENDOWN  
SPINSQUARE  
PENUP  
RIGHT 45  
FORWARD 30]  
END
```

## Things to remember about REPEAT

- 1) REPEAT can only be used in a procedure and will produce an error if used in immediate mode.
- 2) The closing bracket must be on the same line as the last command. It cannot appear on a line alone.
- 3) REPEATs may be nested e.g.:-

```
REPEAT 4  
[REPEAT 2  
[FORWARD 10  
RIGHT 10]  
BACK 15]
```

4) More than one closing bracket may appear at the end of a command when nested loops are used e.g.:-

```
REPEAT 4
[FORWARD 10
REPEAT 2
[SQUARE
RIGHT 15]]
```

5) The number of closing brackets must always equal the number of REPEAT commands.

## Recursion

"Recursion" is the name used to describe the action of a procedure calling itself. For example:-

```
TO FRED
FORWARD 20
RIGHT 90
FRED
END
```

Notice that the last statement of FRED is a call to FRED. This is perfectly acceptable in LOGO, and can be a powerful technique. There are limitations in using recursion in the version of LOGO presented here. One drawback is that there is no way of stopping the recursion, so the process could carry on for ever. This won't happen, however, as the program will eventually run out of memory for "stack" storage.

The stack (in this program) is an array used to store information every time a procedure call is met. The program needs to know what was happening when such an event occurs, so that it can carry on with the main program after the procedure has finished. In the example above, information is saved in the stack array every time FRED is called. Eventually the stack array reaches the limit of memory allocated to it, and, when this happens, LOGO will terminate the procedure and control will return to immediate mode.

In full implementations of LOGO, conditional statements such as "IF.. THEN" allow recursive loops to be terminated when a given condition occurs.

## Other commands

**CANCEL** This allows the last command entered in a procedure to be cancelled. It can only be used when typing in a procedure (i.e. between TO and END).

**DELETE** This asks for a procedure name to delete. This procedure will be destroyed along with any entered after it.

**LIST** This command allows procedures to be listed on the screen. If a listing is longer than one screen, depress the < SHIFT > and @ simultaneously to halt the listing. To resume the listing, press any key other than < SHIFT > and @.

**SAVE** Use this to save a specified procedure to tape under the same name as the procedure. Thus procedure FRED will appear as a tape file FRED.

**LOAD** Reloads procedures saved with the SAVE command.

## Program listing

```
10 POKE65495,0
20 CLEAR 1000
30 OB=0:BRP=0:SR=1:HT=0:P=1:PI=3.14159:NRW=14:NE=
NRW:CE=NRW+1:SP=1
40 OCE=0
50 PL=0:UP=1:H=0:R=5:Y=90:X=120:D=0:I=0:K=1:RC=0:
RP=1
60 RESTORE
70 F$="":G$="":N$="":C$="":S$="":S1$="":S2$="":O$=""
80 DIM Z$(NRW+10,2),A(17,17),R(10,10),UF$(10,40),L$(29),NO$(
10),UDP$(50),Q$(100),Q2$(100),Q3$(100)
90 DATA "TO","0","END","0","FORWARD","1","BACK","1",
"RIGHT","1","LEFT","1","PENUP","0","PENDOWN","0","HIDE
TURTLE","0","SHOWTURTLE","0","CLEARSCREEN","0",
"REPEAT","1","DRAW","0"
95 DATA "CIRCLE","1"
100 FOR J=1 TO NRW
110 READ Z$(J,1),Z$(J,2)
120 NEXT J
```



```

130 PMODE4,1:PCLS:SCREEN1,1
140 GOSUB 14000
150 GET(100,50)-(110,40),R,G
160 GOSUB 20000
170 C$="":N$=C$:GOSUB 19000
180 GOSUB 21000
190 GOSUB 9000
200 GOTO 170
9000 IF LEN(O$)=3 AND G$="" THEN O$=RIGHT$(O$,2)
9010 IF S$="LIST" OR S$="DELETE" OR S$="SAVE" OR S$=
"LOAD" THEN S$="":RETURN
9020 IF G$="" AND F$="TO" THEN 10000
9030 IF S$="CANCEL" AND SP>1 THEN S$="":RETURN
9040 IF O$="00002" THEN 9090 ELSE CD=VAL(LEFT$(O$,2)):
N$=RIGHT$(O$,LEN(O$)-2)
9050 C$=LEFT$(O$,2)
9060 IF C$="02" AND UDP$(CE-NRW)="1" AND UP>1 THEN
10000
9070 IF CD=99 THEN 10000
9080 IF CD>NRW AND UDP$(VAL(Q$(2)))<>"1" THEN CE=CD:
DP$(CD-NRW)="2":K=1:GOTO 11200
9090 IF UDP$(VAL(Q$(2)))="1" THEN 11000
9100 IF O$="00002" AND UDP$(CE-NRW)="2" THEN 9110 ELSE
9180
9110 IF Q2$(RP-1)="" THEN Q3$(SR)=STR$(K):SR=SR+1
9120 RC=VAL(Q2$(RP-1))
9130 RC=RC+1
9140 Q2$(RP-1)=STR$(RC)
9150 IF VAL(Q2$(RP-1))<>VAL(Q2$(RP-2)) THEN K=VAL(Q3$
(SR-2))+1:GOTO 11200
9160 IF RP=3 THEN RP=1:REPT=0:RC=0:K=VAL(Q3$(SR-1))+
1:SR=1:GOTO 11220
9170 K=VAL(Q3$(SR-1))+1:SR=SR-2:Q2$(RP-1)="" :RP=RP-2:
GOTO 11200
9180 ON CD GOTO 12000,12500,13000,13500,14000,14500,15000,
15500,16000,16500,17000,17500,10500,15700
9500 REM CANCEL
9510 IF BRP=1 THEN BRP=0:OB=1
9520 IF S2$<>" " THEN 9640
9530 IF SP>1 THEN K=K-1
9540 UF$(CE-NRW,K)="" :C$="" :N$=""

```

*Hot programs to feed your Dragon*

```
9550 CD = 0:F$ = ""
9560 IF VAL(O$)>NRW AND VAL(O$)=<NE THEN 9730
9570 ON VAL(LEFT$(O$,2)) GOTO 9580,9730,9730,9730,9730,9730,
15500,15000,16500,16000,9730,9630,9730,9730
9580 SP = 1:Q$(SP+1) = "" : Q$(SP) = ""
9590 Z$(CE,1) = "" : Z$(CE,2) = ""
9600 NE = NE-1:UDP$(CE-NRW) = ""
9610 K = 1:CE = OCE
9620 GOTO 9730
9630 REPT = 0:SR = SR-1:Q3$(SR) = "" : UP = UP-1:GOTO 9730
9640 PL = 1
9650 FOR I = LEN(S2$)-1 TO 0 STEP -1
9660 IF RP = 3 THEN REPT = 0
9670 UF$(VAL(Q$(SP-1)),K) = ""
9680 IF UP = 1 THEN REPT = 1
9690 UP = UP + 1:K = K-1
9700 IF I = 0 THEN K = K + 1
9710 NEXT I
9720 GOTO 9530
9730 O$ = "08":RETURN
10000 REM ERROR
10010 DRAW"BM200,185"+L$(5)+L$(18)+L$(18)+L$(15)+L$(18)
10020 FOR W = 1 TO 200:NEXT W
10030 GET(200,100)-(209,91),R,G
10040 FOR J = 1 TO 6
10050 PUT(190+J*9,185)-(199+J*9,176),R,PSET
10060 NEXT J
10070 RETURN
10500 REM DRAW
10510 H = 0:X = 115:Y = 90:PCLS
10520 GET(X-7,Y+7)-(X+9,Y-10),A,G
10530 GOTO 14000
11000 REM USR DEF. PROCEDURE
11010 IF UF$(CE-NRW,K)<>" " THEN K = K + 1
11020 IF UDP$(CE-NRW) = "1" AND VAL(O$) = 2 THEN GOTO 12500
11030 IF VAL(C$)>NRW AND UDP$(CE-NRW) = "1" THEN GOTO
11050
11040 IF UDP$(CE-NRW) = "1" AND VAL(O$)<>2 THEN UF$(CE-
NRW,K) = O$:K = K + 1:GOTO 9180
11050 UF$(CE-NRW,VAL(Q$(SP-2))+K-1) = STR$(CD):K = K + 1
```

```

11060 IF CD>NRW AND UDP$(CE-NRW)="1" AND PL=1 THEN
19510 ELSE RETURN
11200 REM EXECUTE USR.DEF.PROCEDURE
11210 IF SP=1 THEN Q$(1)="2":Q$(2)=STR$(CE-NRW):SP=
SP+2
11220 IF VAL(UF$(CE-NRW,K))=<NE AND VAL(UF$(CE-NRW,K))
>NRW THEN GOTO 11260
11230 O$=UF$(CE-NRW,K):GOTO 9040
11240 K=K+1:IF UF$(CE-NRW,K)="02" THEN 12500
11250 GOTO 11220
11260 Q$(SP)=STR$(K+1):SP=SP+1
11270 Q$(SP)=STR$(CE-NRW):SP=SP+1
11280 IF SP>98 THEN O$="02":SP=1:GOTO 12500
11290 CE=VAL(UF$(CE-NRW,K))
11300 UDP$(CE-NRW)="2"
11310 K=1:GOTO 11220
11500 REM SAVE TO TAPE
11510 SCREEN 0:CLS
11520 PRINT"POSN. TAPE-PRESS PLAY & RECORD"
11530 INPUT"PROCEDURE NAME";RT$
11540 POKE65494,0
11550 OPEN "O",#-1,RT$
11560 FOR T=1 TO 10
11570 FOR P=1 TO 40
11580 PRINT#-1,UF$(T,P)
11590 NEXT P
11600 FOR Q=1 TO 2
11610 PRINT#-1,Z$(NRW+T,Q)
11620 NEXT Q
11630 NEXT T
11640 PRINT#-1,CD,F$,G$,O$,S1$
11650 CLOSE#-1
11660 POKE65495,0
11670 PMODE4,1:SCREEN1,1:RETURN
11700 REM INPUT DATA FROM TAPE
11710 SCREEN 0:CLS
11720 PRINT"REWIND TAPE-PRESS PLAY"
11730 INPUT"PROCEDURE NAME";RT$
11740 POKE65494,0
11750 OPEN"I",#-1,RT$
11760 FOR T=1 TO 10

```



```

11770 FOR P = 1 TO 40
11780 INPUT #1,UF$(T,P)
11790 NEXT P
11800 FOR Q = 1 TO 2
11810 INPUT #1,Z$(NRW + T,Q)
11820 NEXT Q
11830 NEXT T
11840 INPUT #1,CD,F$,G$,O$,S1$
11850 CLOSE #1
11860 POKE65495,0
11870 FOR T = 1 TO 10
11880 IF Z$(NRW + T,1) = "" THEN CE = NRW + T - 1:NE = CE:GOTO
11910
11890 NEXT T
11900 CE = NRW + 11:NE = NRW + 10
11910 P = 1:PMODE4,1:SCREEN1,1:RETURN
12000 REM EXECUTE "TO"
12010 FOR I = 1 TO NE
12020 IF N$ = Z$(I,1) THEN 10000
12030 NEXT I
12031 IF INSTR(1,N$," ") = 0 THEN 12040
12032 FOR I = 1 TO NE
12033 IF LEFT$(N$,INSTR(1,N$," ") - 1) = Z$(I,1) THEN 10000
12034 NEXT I
12040 IF N$ = "[" OR N$ = "]" THEN 10000
12050 OCE = CE:CE = NE + 1:UDP$(CE-NRW) = "1"
12060 Z$(CE,1) = N$
12070 Z$(CE,2) = "0"
12080 NE = NE + 1:K = 1
12090 Q$(SP) = STR$(K)
12100 Q$(SP + 1) = STR$(CE-NRW)
12110 SP = SP + 2
12120 RETURN
12500 REM EXECUTE "END"
12510 IF UDP$(CE-NRW) = "1" AND K = 1 OR UDP$(CE-NRW) = ""
THEN O$ = "99":GOTO 10000
12520 IF UDP$(CE-NRW) = "1" THEN UDP$(CE-NRW) = "":UF$(CE-
NRW,K) = "02":K = 1:SP = 1:SR = 1:RETURN
12530 IF SP > 1 THEN SP = SP - 2
12540 IF SP < 1 THEN PRINT"STACK UNDERFLOW":END
12550 IF K = 1 THEN 10000

```

```

12560 UDP$(CE-NRW) = ""
12570 K = VAL(Q$(SP)):CE = VAL(Q$(SP + 1)) + NRW
12580 IF SP = 1 THEN 12600
12590 GOTO 11220
12600 UP = 1:SR = 1:K = 1:RC = 0:RP = 1
12610 FOR I = 1 TO 99
12620 Q$(I) = ""
12630 Q2$(I) = ""
12640 Q3$(I) = ""
12650 NEXT I
12660 RETURN
13000 REM EXECUTE FORWARD
13010 IF N$ = "000" THEN 10000
13020 D = VAL(N$)
13030 PUT(X-7,Y+7)-(X+9,Y-10),A,PSET
13040 X = X + D*SIN(H):Y = Y-D*COS(H)
13050 IF X-INT(X)>0.5 THEN X = INT(X) + 1 ELSE X = INT(X)
13060 IF Y-INT(Y)>0.5 THEN Y = INT(Y) + 1 ELSE Y = INT(Y)
13070 X1 = X-D*SIN(H):Y1 = Y + D*COS(H)
13080 IF Y1-INT(Y1)>0.5 THEN Y1 = INT(Y1) + 1 ELSE Y1 = INT(Y1)
13090 IF X1-INT(X1)>0.5 THEN X1 = INT(X1) + 1 ELSE X1 = INT(X1)
13100 IF X<247 THEN 13150
13110 IF X>247 AND Y<>Y1 THEN Y = Y1-(247-X1)/TAN(H)
13120 IF X=247 THEN 13150
13130 IF X>247 THEN X=247
13140 IF X=247 AND X1=247 THEN Y=Y1
13150 IF X>10 THEN 13200
13160 IF X<10 THEN Y = Y1-(X1-7)/TAN(2*PI-H)
13170 IF X=10 THEN 13200
13180 IF X<10 THEN X=10
13190 IF X=10 AND X1=10 THEN Y=Y1
13200 IF Y>=10 THEN 13230
13210 IF Y<10 THEN X = (Y1-7)*TAN(H) + X1:Y=10
13220 IF Y=10 AND Y1=10 THEN X=X1
13230 IF Y<170 THEN 13280
13240 IF Y>170 THEN X = X1 + (170-Y1)*TAN(PI-H)
13250 IF Y=170 THEN 13280
13260 IF Y>170 THEN Y=170
13270 IF Y=170 AND Y1=170 THEN X=X1
13280 IF X-INT(X)>0.5 THEN X = INT(X) + 1 ELSE X = INT(X)
13290 IF Y-INT(Y)>0.5 THEN Y = INT(Y)

```

```

13300 IF P = 0 THEN 13320
13310 LINE(X1,Y1)-(X,Y),PSET
13320 GET(X-7,Y+7)-(X+9,Y-10),A,G
13330 IF HT = 1 THEN 13380
13340 LINE((X + R*SIN(H)),(Y-R*COS(H)))-((X + R*SIN(H+2*PI/3)),
,(Y-R*COS(H+2*PI/3))),PSET
13350 LINE-((X + R*SIN(H-2*PI/3)),(Y-R*COS(H-2*PI/3))),PSET
13360 LINE-((X + R*SIN(H)),(Y-R*COS(H))),PSET
13370 CIRCLE((X + R*SIN(H)),(Y-R*COS(H))),2=
13380 IF PL = 1 THEN 19510
13390 IF UDP$(CE-NRW) = "2" THEN 11240 ELSE RETURN
13500 REM EXECUTE BACK
13510 IF N$ = "000" THEN 10000
13520 D = -VAL(N$)
13530 GOTO 13030
14000 REM EXECUTE RIGHT
14010 IF N$ = "000" THEN 10000
14020 N = VAL(N$)/57.29577951:H = H + N
14030 IF UDP$(CE-NRW) = "2" AND HT = 1 THEN 11240
14040 PUT(X-7,Y+7)-(X+9,Y-10),A,PSET
14050 GET(X-7,Y+7)-(X+9,Y-10),A,G
14060 IF HT = 1 THEN 14110
14070 LINE((X + R*SIN(H)),(Y-R*COS(H)))-((X + R*SIN(H+2*PI/3)),
,(Y-R*COS(H+2*PI/3))),PSET
14080 LINE-((X + R*SIN(H-2*PI/3)),(Y-R*COS(H-2*PI/3))),PSET
14090 LINE-((X + R*SIN(H)),(Y-R*COS(H))),PSET
14100 CIRCLE((X + R*SIN(H)),(Y-R*COS(H))),2=
14110 IF PL = 1 THEN 19510
14120 IF UDP$(CE-NRW) = "2" THEN 11240 ELSE RETURN
14500 REM EXECUTE LEFT
14510 IF N$ = "000" THEN 10000
14520 N = 2*PI-(VAL(N$)/57.29577951):H = H + N
14530 GOTO 14040
15000 REM EXECUTE PENUP
15010 P = 0
15020 IF PL = 1 THEN 19510
15030 IF UDP$(CE-NRW) = "2" THEN 11240 ELSE RETURN
15500 REM EXECUTE PENDOWN
15510 P = 1
15520 IF UDP$(CE-NRW) = "2" THEN 11240
15530 IF PL = 1 THEN 19510 ELSE RETURN

```



```

15700 REM CIRCLE
15710 CR=VAL(N$)
15720 CX=X+(CR*COS(PI/2-H)):CY=Y-(CR*SIN(PI/2-H))
15721 IF CX>254 THEN 10000
15722 IF CX=<0 THEN 10000
15723 IF CY=<0 THEN 10000
15724 IF CY>190 THEN 10000
15730 PUT(X-7,Y+7)-(X+9,Y-10),A,PSET
15740 CIRCLE(CX,CY),CR
15750 GOTO 13320
16000 REM EXECUTE HIDETURTLE
16010 IF HT=1 AND UDP$(CE-NRW)=" THEN RETURN
16020 HT=1
16030 PUT(X-7,Y+7)-(X+9,Y-10),A,PSET
16040 IF PL=1 THEN 19510
16050 IF UDP$(CE-NRW)="2" THEN 11240 ELSE RETURN
16500 REM EXECUTE SHOWTURTLE
16510 HT=0
16520 GET(X-7,Y+7)-(X+9,Y-10),A,G
16530 LINE((X+R*SIN(H)),(Y-R*COS(H)))-((X+R*SIN(H+2*PI/3)),
,(Y-R*COS(H+2*PI/3))),PSET
16540 LINE-((X+R*SIN(H-2*PI/3)),(Y-R*COS(H-2*PI/3))),PSET
16550 LINE-((X+R*SIN(H)),(Y-R*COS(H))),PSET
16560 CIRCLE((X+R*SIN(H)),(Y-R*COS(H))),2
16570 IF PL=1 THEN 19510
16580 IF UDP$(CE-NRW)="2" THEN 11240 ELSE RETURN
17000 REM CLEARSCREEN
17010 PCLS:PMODE4,1:SCREEN1,1
17020 PUT(X-7,Y+7)-(X+10,Y-9),R,PSET
17030 GET(X-7,Y+7)-(X+10,Y-9),A,G
17040 IF HT=1 THEN 17100
17050 LINE((X+R*SIN(H)),(Y-R*COS(H)))-((X+R*SIN(H+2*PI/3)),
,(Y-R*COS(H+2*PI/3))),PSET
17060 LINE-((X+R*SIN(H-2*PI/3)),(Y-R*COS(H-2*PI/3))),PSET
17070 LINE-((X+R*SIN(H)),(Y-R*COS(H))),PSET
17080 CIRCLE((X+R*SIN(H)),(Y-R*COS(H))),2
17090 IF PL=1 THEN 19510
17100 IF UDP$(CE-NRW)="2" THEN 11240 ELSE RETURN
17500 REM TO REPEAT
17510 IF N$="000" THEN 10000
17520 UP=UP+1

```

*Hot programs to feed your Dragon*

```
17530 IF UDP$(CE-NRW) = "" THEN 10000
17540 Q3$(SR) = STR$(K):SR = SR + 1
17550 RC = 0:REPT = 1
17560 IF UDP$(CE-NRW) = "1" THEN OB = 1
17570 IF UDP$(CE-NRW) = "2" THEN K = K + 1:Q2$(RP) = N$:RP =
RP + 2:GOTO 11200
17580 RETURN
18000 REM TO DELETE A UDP
18010 SCREEN0,0:CLS
18020 INPUT"PROCEDURE TO BE DELETED";RT$
18030 FOR I = 1 TO 10
18040 IF RT$ = Z$(NRW + I,1) THEN 18070
18050 IF I = 10 THEN O$ = "99":SCREEN1,1:GOTO 10000
18060 NEXT I
18070 FOR J = I TO NE-NRW
18080 FOR B = 1 TO 40
18090 UF$(J,B) = ""
18100 NEXT B
18110 FOR B = 1 TO 2
18120 Z$(NRW + J,B) = ""
18130 NEXT B
18140 NEXT J
18150 O$ = "08":CE = NRW + I - 1:NE = CE
18160 SCREEN1,1:PMODE4,1:RETURN
18500 REM LIST PROCEDURES
18510 SCREEN0,0:CLS
18520 PRINT"USER PROCEDURES:-"
18530 PRINT
18540 IF NE = NRW THEN 18900
18550 FOR I = 1 TO NE-NRW
18560 PRINT Z$(NRW + I,1)
18570 NEXT I
18580 PRINT
18590 INPUT"DO YOU WISH TO LIST A PROCEDURE?(Y/N)";RT$
18600 IF RT$ = "N" THEN SCREEN1,1:RETURN
18610 IF RT$ <> "Y" THEN 18590
18620 CLS:INPUT"PROCEDURE TO BE LISTED";RT$
18630 PRINT
18640 FOR I = 1 TO 10
18650 IF RT$ = Z$(NRW + I,1) THEN 18680
18660 IF I = 10 THEN SCREEN1,1:GOTO 10000
```



```

18670 NEXT I
18680 FOR B = 1 TO 40
18690 RT$ = UF$(I,B):IF RT$ = "" THEN 18900
18700 IF VAL(RT$) > NRW AND VAL(RT$) = < NE THEN PRINT Z$
(VAL(RT$),1):GOTO 18880
18710 IF RT$ = "00002" THEN 18870
18720 IF VAL(LEFT$(RT$,2)) > 9 THEN 18740
18730 ON VAL(LEFT$(RT$,2))-1 GOTO 18750,18760,18770,18780,
18790,18800,18780,18820
18740 ON VAL(LEFT$(RT$,2))-9 GOTO 18830,18840,18850,18860
18750 PRINT"END":GOTO 18880
18760 PRINT"FORWARD";STR$(VAL(RIGHT$(RT$,3))):GOTO
18880
18770 PRINT"BACK";STR$(VAL(RIGHT$(RT$,3))):GOTO 18880
18780 PRINT"RIGHT";STR$(VAL(RIGHT$(RT$,3))):GOTO 18880
18790 PRINT"LEFT";STR$(VAL(RIGHT$(RT$,3))):GOTO 18880
18800 PRINT"PENUP":GOTO 18880
18810 PRINT"PENDOWN":GOTO 18880
18820 PRINT"HIDETURTLE":GOTO 18880
18830 PRINT"SHOWTURTLE":GOTO 18880
18840 PRINT"CLEARSCREEN":GOTO 18880
18850 PRINT"REPEAT" + STR$(VAL(RIGHT$(RT$,3))):PRINT "[";
:GOTO 18880
18860 PRINT"DRAW":GOTO 18880
18870 PRINT"]"
18880 FOR J = 1 TO 100:NEXT J
18890 NEXT B
18900 PRINT:PRINT"PRESS (ENTER) TO RETURN"
18910 RT$ = INKEY$
18920 IF RT$ = CHR$(13) THEN SCREEN1,1:RETURN ELSE 18910
19000 REM GRAPHICS & PRODUCTION OF S$
19010 S$ = "":J = 0
19020 DRAW"BM5,184BU4R4D4L3D2BD3D1"
19030 FOR V = 0 TO 20
19040 PUT(17 + V*10,187)-(27 + V*10,177),R,PSET
19050 NEXT V
19060 B$ = INKEY$
19070 LINE(17 + J*9,187)-(24 + J*9,180),PSET,BF
19080 IF B$ = "" THEN GOTO 19060
19090 LINE(17 + J*9,187)-(24 + J*9,180),PRESET,BF
19100 IF B$ < > CHR$(21) THEN 19150

```



```

19110 FOR V= 1 TO 25
19120 PUT(7 + V*10,187)-(17 + V*10,177),R,PSET
19130 NEXT V
19140 GOTO 19010
19150 IF B$ < > CHR$(13) THEN 19170
19160 RETURN
19170 IF B$ = CHR$(8) THEN J = J-1 ELSE 19190
19180 IF J < 0 THEN J = 0: S$ = LEFT$(S$,J): GOTO 19060 ELSE S$ =
LEFT$(S$,J): GOTO 19060
19190 IF B$ = CHR$(91) OR B$ = CHR$(93) OR B$ = " " OR B$ = > "A"
AND B$ = < "Z" OR B$ = > "0" AND B$ = < "9" OR B$ = CHR$(13)
THEN 19200 ELSE 19060
19200 IF LEN(S$) = > 19 THEN O$ = "99": RETURN
19210 IF B$ = CHR$(13) AND S$ = "" THEN 10000
19220 IF B$ = CHR$(13) THEN 21000
19230 S$ = S$ + B$
19240 U = ASC(B$)
19250 IF B$ = > CHR$(48) AND B$ = < CHR$(57) THEN 19300
19260 IF B$ = CHR$(91) THEN 19310
19270 IF B$ = CHR$(32) THEN 19320
19280 IF B$ = CHR$(93) THEN 19330
19290 DRAW"BM"+STR$(17+J*9)+",187"+L$(U-ASC("@")):
GOTO 19340
19300 DRAW"BM"+STR$(17+J*9)+",184"+NO$(U-ASC("0")):
GOTO 19340
19310 DRAW"BM"+STR$(17+J*9)+",187"+L$(28):GOTO 19340
19320 DRAW"BM"+STR$(17+J*9)+",187"+L$(27):GOTO 19340
19330 DRAW"BM"+STR$(17+J*9)+",187"+L$(29)
19340 J = J + 1
19350 GOTO 19060
19500 REM REGISTER "J" IN ARRAY
19510 FOR I=0 TO LEN(S2$)-1
19520 IF I=0 THEN K=K-1
19530 K=K+1:OB=0
19540 IF UP=1 THEN 21590
19550 IF UP>1 THEN UP=UP-1
19560 IF UP=1 THEN REPT=0
19570 UF$(VAL(Q$(SP-1)),K)="00002"
19580 IF RP=3 THEN REPT=0
19590 NEXT I
19600 PL=0:RETURN

```

```

20000 REM CHARACTER GRAPHICS
20010 L$(1) = "U4E2R1F2D1L5BR5D3BR2"
20020 L$(2) = "U6R4D3L4R4F1D2L5BR10"
20030 L$(3) = "BU1U4E1R4BD6L4H1BD1BR9"
20040 L$(4) = "U6R4F2D2G2L4BR10"
20050 L$(5) = "U6R6BD3BL2L4BD3R6BR2"
20060 L$(6) = "U6R6BD3L6BD3BR10"
20070 L$(7) = "U6R6BD3L2BR2D3L6BR10"
20080 L$(8) = "U6BR6D6BU3L6BD3BR10"
20090 L$(9) = "BU6R4BL2D6BL2R4BR1"
20100 L$(10) = "U2BU4BR5D6L5BR8"
20110 L$(11) = "U6BR6G3F3BR2"
20120 L$(12) = "U6BD6R6BR2"
20130 L$(13) = "U6F3E3D6BR2"
20140 L$(14) = "U6F6U6BD6BR2"
20150 L$(15) = "BU1U4E1R4F1D4G1L4H1F1BR8"
20160 L$(16) = "U6R6D3L6BD3BR8"
20170 L$(17) = "U6R6D6H3BG3R6BR2"
20180 L$(18) = "U6R6D3L6BR3F3BR3"
20190 L$(19) = "BU3U3R6BD3L6BR6D3L6BR8"
20200 L$(20) = "BU6R6BL3D6BR6"
20210 L$(21) = "U6BR6D6L6BR10"
20220 L$(22) = "BU6D3F3E3U3BD6BR2"
20230 L$(23) = "U6BR6D6H3G3BR10"
20240 L$(24) = "E6BL6F6BR2"
20250 L$(25) = "BU6F3E3BG3D3BR4"
20260 L$(26) = "BU6R6G6R6BR2"
20270 L$(27) = "BR10"
20280 L$(28) = "BR2U6R2L2D6R2BR4"
20290 L$(29) = "BR2R2U6L2R2D6BR2"
20300 NO$(1) = "BU3BR3D3BL3BD3BR3U3BR1"
20310 NO$(2) = "BU3R3D3L3D3R3BU3BR1"
20320 NO$(3) = "BU3R3D3L3BD3R3U3BR1"
20330 NO$(4) = "U3BR3D3L3BD3BR3U3BR1"
20340 NO$(5) = "U3R3BD3L3BD3R3U3BR1"
20350 NO$(6) = "U3BR3BD3L3D3R3U3BR1"
20360 NO$(7) = "BU3R3D6BU3BR1"
20370 NO$(8) = "U3R3D3L3D3R3U3BR1"
20380 NO$(9) = "U3R3D3L3BD3BR3U3BR1"
20390 NO$(0) = "U3R3D3BL3D3R3U3BR1"
20400 RETURN

```



```
21000 REM SYNTAX ANALYSER
21010 IF S$ = "DELETE" THEN 18000
21020 IF S$ = "SAVE" THEN 11500
21030 IF S$ = "LOAD" THEN 11700
21040 IF S$ = "LIST" THEN 18500
21050 IF S$ = "CANCEL" AND SP > 1 THEN GOTO 9500
21060 BRP = 0: S2$ = ""
21070 IF UDP$(CE-NRW) = "1" AND LEFT$(S$,2) = "TO" THEN 21590
21080 O$ = "": I = 0
21090 IF REPT = 0 THEN S1$ = S$: GOTO 21200
21100 IF INSTR(1,S$,"J") = 0 THEN S1$ = S$ ELSE S1$ = LEFT$(S$,
INSTR(1,S$,"J")-1): S2$ = RIGHT$(S$,LEN(S$)-LEN(S1$))
21110 IF SP > 3 AND INSTR(1,S$,"[") = 0 AND VAL(Q$(SP-1)) = K
THEN O$ = "99": RETURN
21120 IF OB = 0 THEN 21140
21130 IF INSTR(1,S1$,"[") = 1 THEN BRP = 1: GOTO 21150 ELSE
21590
21140 IF INSTR(1,S1$,"[") = 1 THEN 21590 ELSE 21160
21150 S1$ = RIGHT$(S1$,LEN(S1$)-1)
21160 IF S2$ = "" THEN GOTO 21200
21170 FOR I = 1 TO LEN(S2$)
21180 IF MID$(S2$,I,1) < > "]" THEN 21590
21190 NEXT I
21200 IF INSTR(1,S1$," ") = 0 THEN F$ = S1$ ELSE 21220
21210 GOTO 21310
21220 IF S1$ = "" THEN 21590
21230 F$ = LEFT$(S1$,INSTR(1,S1$," ")-1)
21240 G$ = RIGHT$(S1$,LEN(S1$)-INSTR(1,S1$," "))
21250 IF F$ < > "TO" THEN 21270
21260 IF INSTR(1,G$," ") = 0 THEN 21310 ELSE 21590
21270 FOR T = 1 TO LEN(G$)
21280 IF MID$(G$,T,1) < "0" OR MID$(G$,T,1) > "9" THEN 21590
21290 NEXT T
21300 I = 0
21310 I = I + 1
21320 IF I > NE THEN 21590
21330 IF F$ < > Z$(I,1) THEN 21310
21340 IF I > = 10 THEN 21360 ELSE C$ = "0" + RIGHT$(STR$(I),LEN
(STR$(I))-1)
21350 GOTO 21380
21360 C$ = STR$(I)
```



```

21370 IF VAL(C$) = > 10 THEN C$ = RIGHT$(C$,2)
21380 IF F$ = "TO" THEN 21510
21390 IF Z$(I,2) = "0" AND INSTR(1,S$," ") < > 0 OR N$ < > "" THEN
21590
21400 IF Z$(I,2) = "0" THEN 21500
21410 IF Z$(I,2) = "1" AND INSTR(1,S$," ") = 0 THEN 21590
21420 IF I = 2 OR I = 7 OR I = 8 OR I = 9 OR I = 10 OR I = 11 OR I = 13
THEN 221425
21422 IF VAL(G$) = 0 THEN 21590
21425 IF LEN(G$) = 3 THEN N$ = G$ ELSE 21440
21430 GOTO 21500
21440 IF LEN(G$) > 3 AND F$ < > "TO" THEN 21590 ELSE 21450
21450 IF LEN(G$) = 2 THEN N$ = "0" + G$ ELSE 21470
21460 GOTO 21500
21470 IF LEN(G$) = 1 THEN N$ = "00" + G$ ELSE 21490
21480 GOTO 21500
21490 IF LEN(G$) = 0 THEN 21590 ELSE 21510
21500 O$ = C$ + N$:GOTO 21520
21510 IF F$ < > "TO" THEN O$ = C$ ELSE O$ = C$ + G$
21520 IF REPT = 1 AND UDP$(CE-NRW) = "2" THEN RC = RC + 1
21530 IF S2$ = "" THEN OB = 0:RETURN
21540 FOR I = 1 TO LEN(S2$)
21550 IF MID$(S2$,I,1) < > "]" THEN 21590
21560 NEXT I
21570 IF LEN(S2$) = > UP THEN 21590
21580 OB = 0:PL = 1:RETURN
21590 O$ = "99"
21600 BRP = 0
21610 RETURN

```

## Program description

<i>Lines</i>	<i>Purpose</i>
10	Speeds up calculations and graphics.
20	Clear string space.
30-50	Set up variables.
60	Restores all data statements.
70	Set up strings.
80	Dimension arrays.

90-95	Define reserved words.
100-120	Set up array of reserved words.
130	Turn on graphics screen.
140	Draw turtle symbol.
150	Get blank screen into array R.
160	Set up character arrays.
170-200	Main loop.
9000	The command interpreter.
9000	Reduces coded command string, O\$, to two characters.
9010-9030	Check for LIST, DELETE, SAVE, LOAD and CANCEL. If any one of these found then execute command at 21010-21050. Check that, if command is "TO", a valid procedure name follows.
9040	The coded instruction is split into CD, the command code and N\$, the value following the command.
9050	Reduces a command to its coded form, CD.
9060	Checks if END command is valid.
9070	Checks for ERROR code.
9080	Is command a valid procedure name?
9090	Checks if command is to be put into a procedure's command array.
9100-9170	When an closing square bracket is reached in a procedure, its REPEAT loop is evaluated and K, the position pointer, is reset.
9180	Executes appropriate subroutine.
9500-9730	Cancels last command. Resets all arrays.
10000-10020	Error routine.
10030	Get graphics screen into array R.
10040-10070	Blank over "ERROR" message, return to main loop.
10050	Subroutine for DRAW.
10510	Turtle heading, H, reset to zero. Turtle X and Y coordinates set to screen centre. Screen cleared.
10520	Draw turtle.
11000	Subroutine which enters commands under a procedure name.
11010-11030	Check for valid command.
11040-11050	Put coded form of command in procedure array.
11060	Return to main loop.
11200	Subroutine which executes procedures.

11210-11310	Takes coded command from the procedure array and assigns it to O\$. This is then passed to the main program for execution.
11500	SAVE to tape.
11510	Displays and clears text screen.
11520-11660	Display instructions. Open file on tape, write data. Close file.
11670	Reselect graphics screen. Back to main loop.
11700-11910	LOAD from tape. See SAVE above.
12000	Subroutine to execute the "TO" command.
12010-12030	Check if name given is already assigned to a procedure.
12031-12040	Check if name given is valid.
12050-12110	Put new name in procedure array, Z\$. Increase the count of entries in Z\$. Increment stack pointer.
12120	Back to main loop.
12500-12650	Puts code for "END" in procedure array when in immediate mode. If executing a procedure, then "END" is executed.
12660	Back to main loop.
13000	Execute "FORWARD".
13010	Check for zero travel.
13020	Obtain D (the distance to travel) from N\$.
13030	Replace graphics detail present on screen before the turtle was drawn.
13040-13290	Calculate new turtle position.
13300-13310	If pen is down then draw line from old position to new position.
13320	Save screen "under" turtle to array A.
13330-13370	If turtle not hidden then draw it.
13380	If end bracket of REPEAT included in command, put it into array.
13390	Return to main loop.
13500	Subroutine for "BACK".
13510	Check for zero travel.
13520	Change sign of D, the distance to travel.
13350	From now on, use "FORWARD" routine.
14000	Execute "RIGHT".
14010	Zero rotation?.



*Hot programs to feed your Dragon*

14020	Calculate N, the amount of rotation and H, the new direction in which the turtle points.
14030	Check for hidden turtle.
14040-14100	Draw turtle at new angle, if not hidden.
14120	Return to main loop.
14500-14530	Subroutine for "LEFT". See "RIGHT".
15000-15350	"PENUP" and "PENDOWN". Set the pen flag to zero if pen is up, or one if pen down.
15700-15570	Executes "CIRCLE" command.
16000-16050	Hide turtle by setting the "turtle hidden" flag, H, to zone.
16500-16580	Show turtle by setting H to zero.
17000-17100	Execute "CLEARSCREEN".
17500	Execute "REPEAT".
17510	Check number of repeats is not zero.
17520-17570	Set up variables and arrays.
17580	Back to main loop.
18000-18160	"DELETE".
18500-18920	"LIST".
19000	Draw graphics characters.
19010	Set string S\$ to null. Zero count, J.
19020	Draw question mark.
19030-19050	Blank out the bottom line of the screen.
19060-19080	Scan keyboard for depression. Draw cursor.
19090	Erase cursor.
19100-19140	Check for shift + backspace. Erase entire line if these keys pressed.
19150-19160	Return to main loop when < ENTER > is pressed.
19170-19180	Check for backspace.
19190	Is key valid?
19200	Check length of string, S\$, which holds the command entered.
19210-19220	Check for < ENTER > .
19230	Add character B\$ to string S\$.
19240-19340	Draw character on screen.
19350	Wait for next key.
19500-19510	Counts the closing square brackets.
19520-19560	Checks and alters variables.
19570-19580	Enters code for "END" in array UF\$.

19590	Returns to 19510 to enter next closing bracket (if any).
19600	Resets closing bracket pointer. Back to main loop.
20000-20400	Set up strings for graphics characters.
21000	The syntax analyser.
21010-21050	Check for commands. Go to appropriate routines.
21060-21580	Evaluate the command string, S\$, and return a coded equivalent, O\$.
21590-21610	Come here on error. Signal error by setting the command code, O\$, to 99.

## Variable definitions

<i>Variable name</i>	<i>Function</i>
A,R	Screen arrays.
Z\$	Array of procedure names.
UF\$	Holds procedure commands.
L\$,NO\$	Character drawing strings.
UDP\$	Array which determines if a procedure is to be executed or added to.
Q\$,Q2\$,Q3\$	Arrays used when executing procedures.
F\$	String holding command as a word.
C\$	Coded form of above.
N\$	Coded version of value following a command.
G\$	Number string.
S\$,S1\$,S2\$	Command strings.
O\$	Coded command string.
I,J,T,V,Q	Counters.
PI	The constant "Pi".
X,Y	New turtle coordinates.
X1,Y1	Old turtle coordinates.
N	Angle to turn.
H	Absolute angle of turtle relative to screen.
D	Distance travelled by turtle.

R	Distance from centre of turtle to vertices of triangular turtle symbol.
NE	Number of command entries.
NRW	Number of reserved words.
CE	Current entry.
CD	Current command in coded form.
P	Pen flag.
HT	Turtle hidden flag.
SP	Stack pointer.
SR,RP,UP,	Hide turtle by setting the "turtle hidden" flag, H, to
REPT	Repeat pointers.
RC	Repeat counter.
PL	Flag set when closing bracket included in a command.
OB,OCE,BRP	Variables associated with "CANCEL".

## Points of interest

It is interesting to notice the way in which the program invokes a routine to input a command and then passes this command to a syntax analyser for analysis. The syntax analyser checks to see if the command is a built in command (such as TO, FORWARD etc) or if it is a call to a user defined procedure.

The syntax analyser converts valid commands to a coded form which is then passed to a separate routine (the run time interpreter) for execution. This process mirrors that found in "real" implementations of languages such as BASIC and LOGO.

In order to persuade BASIC to run faster it will be seen that use has been made of the POKE 65495,0 command which makes the machine run at twice its normal rate. If strange results occur when running the program, remove any such lines. The program does, however, revert to normal speed during the SAVE and LOAD commands. If you press break, type POKE65494,0 to revert to normal speed.



After running the program, type ? MEM to see how much room is left in memory. LOGO uses nearly all available RAM. A side effect of this is that the program listed above is now too big to renumber using the standard RENUM command. Some of the line numbers look strange as a result of this.

The text insertion command will insert text from the current line number (pointed to by the edit pointer) onwards, displaying a new incremental line number at the start of every line of text entered. To stop text insertion type (backarrow) and < ENTER >

e.g. \* I< ENTER>

0: THIS IS A TEST LINE OF TEXT < ENTER>

1: < ENTER>

## EDITOR

This is a simple line oriented editor. It enables the user to edit a text file, save and load the file to cassette tape and list the file to a printer. Used in conjunction with W-Pro it provides a simple word processing system.

## Brief command summary

The first command will look for the first occurrence of the requested string in a given file and will point to it. It is valid to use any file name.

F < string > Find next occurrence of < string > in file  
R < string1 > < string2 > Replace first occurrence of < string1 > with < string2 >  
e.g. \* RTEST FRED < ENTER>  
0: THIS IS A TEST LINE OF TEXT

K Delete a line of text  
< number > Move down file < number > of lines e.g. 10 would move ten lines down the file  
< number > Move up file < number > of lines e.g. 10 would move ten lines up the file  
# < number > Move to absolute line number e.g. #10 means go to line ten  
B Move to end of file  
R replace a string

The replace command will search for the first occurrence of a string and replace it with another string. The strings are delimited by double quotes. The command is initiated by typing < ENTER > after the command.  
e.g. \* RTEST FRED < ENTER>  
0: THIS IS A TEST LINE OF TEXT  
1: < ENTER>  
2: Save text file to cassette tape  
3: Load text file from cassette tape

All commands are initiated by typing < ENTER > after the command. Anything shown above enclosed in < > signs does not mean that the signs should be typed in. The signs are used to show the format of the command and should not be typed in. The signs are used to show the format of the command and should not be typed in. The signs are used to show the format of the command and should not be typed in.

## Chapter 3

# Business

## EDITOR

This is a simple line orientated editor. It enables the user to edit a text file, save and load the file to cassette tape and list the file to a printer. Used in conjunction with W-Pro it provides a simple word processing system.

## Brief command summary

Command	Function
I	Insert text into file
F< string>	Find next occurrence of < string> in file
R< string1> ^< string2>	Replace first occurrence of < string1> with < string2>
K	Delete a line of text
< number>	Move down file < number> of lines e.g 10 would move ten lines down the file
-< number>	As above but moves up the file
#< number>	Move to absolute line number e.g #10 means go to line ten
B	Move to start of file
-B	Move to end of file
P	Send text to the printer
S	Save text file to cassette tape
L	Load text file from cassette tape

All commands are initiated by typing <ENTER> after the command. Anything shown above enclosed in < > signs does not mean that the < and > should be typed in. The ^ sign is used throughout this description to signify the backarrow key.

## Detailed command descriptions

### I -insert text

The text insertion command will insert text from the current line number (pointed to by the edit pointer) onwards, displaying a new incremented line number at the start of every line of text entered. To stop text insertion type ^ (backarrow) and < ENTER>

e.g. \* I< ENTER>

0: THIS IS A TEST LINE OF TEXT.< ENTER>

1: ^< ENTER>

\*

Lines of text may be up to 255 characters long. Insertion can take place starting at any line in the file.

### F -find a string

The find command will look for the first occurrence of the requested string in the text and place the edit pointer at that line and display it.

e.g. \* FTEST< ENTER>

0: THIS IS A TEST LINE OF TEXT

\*

The find command usually searches down the file from the current edit pointer position. To reverse the direction of search precede the command with a "-" (minus sign).

### R -replace a string

The replace command will search for the first occurrence of a string and replace it with another string, displaying the altered line of text. The two strings are delimited by a ^ (backarrow).

e.g. \* RTESTFRED< ENTER>

0: THIS IS A FRED LINE OF TEXT

As with the find command, preceding the command with a "-" (minus sign) reverses the direction of search



## **K** -delete command

The delete command removes the current line pointed to by the edit pointer from the file. For deletion of more than one line the command is followed by the number of lines to delete.

e.g. \* **K10**< ENTER> deletes the next ten lines of text

The direction in which lines are deleted may also be reversed by preceding the command with a "-" (minus sign).

## **D** -display line command

The display command displays the current line of text (pointed to by the edit pointer) on the screen. More than one line may be displayed by following the command with a number. The required number of lines will be displayed but the position of the edit pointer will not be changed.

e.g. \* **D10**< ENTER> displays next ten lines

## **NUMBER**-move down file

Typing a number simply moves the edit pointer down the file that number of lines and displays the new current line. The edit pointer may also be moved up the file by preceding the command with a "-" (minus sign)

e.g. \* **-10**< ENTER> moves ten lines up the file

## **#NUMBER** -move to line number

This command simply places the edit pointer at the requested line and displays that line.

e.g. \* **#10**< ENTER>

10: THIS IS LINE TEN

\*

## **B** -move to beginning of file

The **B** command moves the text pointer to the top of the file and displays the top line. If the command is preceded by a "-" (minus sign) the edit pointer will be placed at the last line of the file and that line will be displayed.

e.g. \* **B< ENTER>**  
 0: THIS IS THE START OF THE FILE

\*

-**B< ENTER>**  
 0: THIS IS THE END OF THE FILE

\*

**P** -print text

The **P** command allows the text currently being edited to be listed to a printer. There are several additional options to the command which can be summarised as follows:-

<b>P</b>	Print all the text being edited
<b>P100-200</b>	Print text lines 100-200 inclusive
<b>100-</b>	Print all lines of text from line 100 onwards
<b>-200</b>	Print text lines up to and including line 200
<b>PN</b>	Addition of the <b>N</b> directive in all of the above will cause all lines to be printed preceded by their respective line numbers

e.g. \* **PN< ENTER>** will print all the text with line numbers

**S** -save text to cassette tape

To save the text file currently being edited to tape, type the following:-

```
*S< ENTER>
ENTER FILNAME> TEST
IS TEST CORRECT? Y
PLACE BLANK TAPE IN CASSETTE
DRIVE AND PRESS PLAY AND RECORD
TYPE SPACE WHEN READY < SPACE>
SAVE O.K.
*
```

After the above sequence a data file called "TEXT" should have been saved to tape.

**L** -load text from tape

To load a text file from tape issue the following commands:

```
*L< ENTER>
ENTER FILNAME> TEST
IS TEST CORRECT ? Y
PLACE TAPE IN CASSETTE DRIVE
AND PRESS PLAY
TYPE SPACE WHEN READY < SPACE>
LOAD COMPLETED O.K.
```

\*

The requested file is now loaded into the editor and the edit pointer will be pointing at the first line of text.

## Program listing

```
10 REM LINE ORIENTATED TEXT EDITOR
20 CLEAR 14000:CLS
30 PRINT"TEXT EDITOR VER 1.0"
40 DV = -1
100 EP = 0:ET = 0:NL = 500:REM MAX NUMBER OF LINES
110 DIM TS$(NL)
120 FOR I = 1 TO NL
130 TS$(I) = ""
140 NEXT I
1000 REM COMMAND INTERP
1010 LINE INPUT"*";CM$
1020 IF LEN(CM$)< 1 THEN CM$ = "1"
1030 TP$ = LEFT$(CM$,1)
1040 CM$ = RIGHT$(CM$,LEN(CM$)-1)
1050 IF TP$ = "-" THEN 1500
1060 IF TP$ = "B" THEN 2300
1070 IF TP$ = "F" THEN 3000
1080 IF TP$ = "R" THEN 3500
1090 IF TP$ = "K" THEN 4000
1100 IF TP$ = "I" THEN 5000
1110 IF TP$ = "#" THEN 2500
1120 IF TP$ = "D" THEN 1800
1130 IF TP$ = "P" THEN 4500
1140 IF TP$ = "S" THEN 5500
1150 IF TP$ = "L" THEN 6000
```



```

1160 IF TP$ = "E" THEN END
1200 REM COMMAND A LINE NUMBER ?
1210 I = 1
1220 CM$ = TP$ + CM$
1230 IF MID$(CM$,I,1) < "0" OR MID$(CM$,I,1) > "9" THEN 1400
ELSE I = I + 1
1235 IF I <= LEN(CM$) THEN 1230
1240 I = VAL(CM$)
1250 I = I + EP
1255 IF I > ET THEN 1300 ELSE EP = I
1260 GOTO 1700
1300 PRINT "RUN OFF END OF FILE"
1305 IF EP = ET THEN EP = EP + 1
1310 GOTO 1000
1350 PRINT "RUN OFF TOP OF FILE"
1360 GOTO 1000
1400 PRINT "INVALID COMMAND"
1410 GOTO 1000
1420 PRINT "TEXT SPACE FULL"
1430 GOTO 1000
1440 PRINT "STRING NOT FOUND"
1450 GOTO 1000
1460 PRINT "OUT OF MEMORY"
1470 GOTO 1000
1500 REM NEGATIVE COMMAND INTERP
1510 IF LEN(CM$) < 1 THEN CM$ = "1"
1520 TP$ = LEFT$(CM$,1)
1530 CM$ = RIGHT$(CM$,LEN(CM$)-1)
1540 IF TP$ = "B" THEN 3200
1550 IF TP$ = "F" THEN 3100
1560 IF TP$ = "R" THEN 3700
1570 IF TP$ = "K" THEN 4200
1600 REM COMMAND A LINE NUMBER ?
1610 I = 1
1620 CM$ = TP$ + CM$
1630 IF MID$(CM$,I,1) < "0" OR MID$(CM$,I,1) > "9" THEN 1400
ELSE I = I + 1
1640 IF I <= LEN(CM$) THEN 1630
1650 I = VAL(CM$)
1660 I = EP - I
1670 IF I < 1 THEN 1350 ELSE EP = I

```

```
1700 REM DISPLAY LINE
1710 IF (EP>0 AND EP<=ET) THEN PRINT EP;" ";TS$(EP)
1720 GOTO 1000
1800 REM DISPLAY TEXT
1810 IF LEN(CM$)<1 THEN 1700
1820 I=1
1830 IF MID$(CM$,I,1)<"0" OR MID$(CM$,I,1)>"9" THEN 1400
ELSE I=I+1
1835 IF VAL(CM$)=0 THEN 1000
1840 IF I<=LEN(CM$) THEN 1830
1850 I=0
1860 PRINT EP+I;" ";TS$(EP+I)
1870 I=I+1
1890 IF I+EP>ET THEN 1300 ELSE IF I=VAL(CM$) THEN 1000
1900 GOTO 1860
2300 REM GO TO START OF FILE
2310 EP=1
2320 GOTO 1700
2500 REM GOT TO ABS LINE NUMBER
2510 IF LEN(CM$)<1 THEN 1400
2520 I=1
2530 IF MID$(CM$,I,1)<"0" OR MID$(CM$,I,1)>"9" THEN 1400
ELSE I=I+1
2540 IF I<=LEN(CM$) THEN 2530
2550 I=VAL(CM$)
2560 IF I>ET THEN 1300
2565 IF I<1 THEN 1400
2570 EP=I
2580 GOTO 1700
3000 REM FIND STRING IN TEXT
3010 I=EP
3020 IF INSTR(1,TS$(I),CM$)>0 THEN 3040 ELSE I=I+1
3030 IF I>ET THEN 1300 ELSE 3020
3040 EP=I
3050 GOTO 1700
3100 REM FIND STRING BACKWARDS
3110 I=EP
3120 IF INSTR(1,TS$(I),CM$)>0 THEN 3140 ELSE I=I-1
3130 IF I<0 THEN 1350 ELSE 3120
3140 EP=I
3150 GOTO 1700
```

```

3200 REM TO END OF FILE
3210 EP = ET
3220 GOTO 1700
3500 REM FIND AND REPLACE STRING
3510 ES = INSTR(1,CM$,"__")
3520 IF ES < 1 THEN 1400
3530 TP$ = LEFT$(CM$,ES-1)
3540 CM$ = RIGHT$(CM$,LEN(CM$)-ES)
3550 I = EP
3560 IF INSTR(1,TS$(I),TP$) > 0 THEN 3580 ELSE I = I + 1
3570 IF I > ET THEN 1300 ELSE 3560
3580 ES = INSTR(1,TS$(I),TP$)-1
3590 TS$(I) = LEFT$(TS$(I),ES) + CM$ + RIGHT$(TS$(I),LEN(TS$(I))
-ES-LEN(TP$))
3600 EP = I
3610 GOTO 1700
3700 REM REPLACE STRING BACKWARDS
3710 ES = INSTR(1,CM$,"__")
3720 IF ES < 1 THEN 1400
3730 TP$ = LEFT$(CM$,ES-1)
3740 CM$ = RIGHT$(CM$,LEN(CM$)-ES)
3750 I = EP
3760 IF INSTR(1,TS$(I),TP$) > 0 THEN 3780 ELSE I = I - 1
3770 IF I < 0 THEN 1350 ELSE 3760
3780 ES = INSTR(1,TS$(I),TP$)-1
3790 TS$(I) = LEFT$(TS$(I),ES) + CM$ + RIGHT$(TS$(I),LEN(TS$(I))
-ES-LEN(TP$))
3800 EP = I
3810 GOTO 1700
4000 REM DELETE LINES DOWNWARD
4010 ND = 1
4020 IF LEN(CM$) < 1 THEN 4080
4030 I = 1
4050 IF MID$(CM$,I,1) < "0" OR MID$(CM$,I,1) > "9" THEN 1400
ELSE I = I + 1
4060 IF I <= LEN(CM$) THEN 4050
4070 ND = VAL(CM$)
4080 IF ND + EP > ET THEN ND = ET - EP + 1
4085 IF ET = 0 THEN TS$(0) = ""
4090 FOR I = EP TO EP + ET
4100 TS$(I) = TS$(ND + I)

```



```
4110 NEXT I
4120 ET = ET-ND
4130 GOTO 1700
4200 REM DELETE LINES UPWARDS
4210 ND = 1
4220 IF LEN(CM$)< 1 THEN 4280
4230 I = 1
4250 IF MID$(CM$,I,1)< "0" OR MID$(CM$,I,1)> "9" THEN 1400
ELSE I = I + 1
4260 IF I <= LEN(CM$) THEN < 250
4270 ND = VAL(CM$)
4280 IF EP-ND< 1 THEN ND = EP
4285 IF EP = 1 THEN TS$(1) = ""
4290 FOR I = EP-ND TO ET
4300 TS$(I) = TS$(I + ND)
4310 NEXT I
4320 ET = ET-ND
4330 GOTO 1700
4500 REM LIST TEXT IN BUFFER
4510 SL = 1
4520 EL = ET
4530 LN = 0
4540 IF LEN(CM$)< 1 THEN 4900
4550 TP$ = LEFT$(CM$,1)
4560 IF TP$ = "N" THEN 4850
4570 IF INSTR(1,CM$,"-")< 0 THEN 1400
4580 IF LEN (CM$)< 2 THEN 1400
4610 TP$ = LEFT$(CM$,INSTR(1,CM$,"-")-1)
4620 IF MID$(CM$,LEN(CM$),1) = "-" THEN 4670 ELSE CM$ =
RIGHT$(CM$,LEN(CM$)-LEN(TP$)-1)
4630 I = 1
4640 IF MID$(CM$,I,1)< "0" OR MID$(CM$,I,1)> "9" THEN 1400
ELSE I = I + 1
4650 IF I <= LEN(CM$) THEN 4640
4660 EL = VAL(CM$)
4670 IF LEN(TP$)< 1 THEN 4900
4680 I = 1
4690 IF MID$(TP$,I,1)< "0" OR MID$(TP$,I,1)> "9" THEN 1400 ELSE
I = I + 1
4700 IF I <= LEN(TP$) THEN 4690
4800 SL = VAL(TP$)
```

```

4810 GOTO 4900
4850 LN = 1
4855 IF CM$ = "N" THEN 4900
4860 CM$ = RIGHT$(CM$, LEN(CM$)-1)
4870 GOTO 4570
4900 IF SL > ET THEN 1400
4910 IF SL < 0 THEN 1400
4920 IF EL > ET THEN ML = ET
4930 IF EL < SL THEN 1400
4940 FOR I = SL TO EL
4950 IF LN THEN PRINT #2, I, ":"; ":";
4960 PRINT #2, TS$(I)
4970 NEXT I
4980 GOTO 1000
5000 REM INSERT TEXT
5010 PRINT EP; ":"; ":";
5020 LINE INPUT L$
5030 IF L$ = " _ " THEN 5200
5035 IF ET = NL THEN 1420
5050 FOR I = ET TO EP STEP -1
5060 TS$(I + 1) = TS$(I)
5070 NEXT I
5075 ET = ET + 1
5080 TS$(EP) = L$
5090 EP = EP + 1
5100 GOTO 5000
5200 REM FINISHED EDIT
5220 GOTO 1000
5500 REM SAVE FILE
5510 GOSUB 10000
5515 IF DV = 1 THEN 5560
5520 PRINT "PLACE BLANK TAPE IN CASSETTE"
5530 PRINT "DRIVE AND PRESS PLAY AND RECORD"
5540 PRINT "TYPE SPACE WHEN READY"
5550 IF INKEY$ < "> " THEN 5550
5560 OPEN "O", #DV, N$
5570 FOR I = 1 TO ET
5580 PRINT #DV, TS$(I)
5590 NEXT I
5600 CLOSE #DV
5610 PRINT "SAVE O.K."

```

```

5620 GOTO 1000
6000 REM LOAD TEXT FILE
6010 GOSUB 10000
6020 ET = 0:EP = 0
6025 IF DV = 1 THEN 6060
6030 PRINT "PLACE TAPE IN CASSETTE DRIVE"
6035 PRINT "AND PRESS PLAY"
6040 PRINT "TYPE SPACE WHEN READY"
6050 IF INKEY$ < > " " THEN 6050
6060 OPEN "I", #1, N$
6070 FOR I = 1 TO NL
6080 TS$(I) = ""
6090 NEXT I
6100 IF EOF(DV) THEN 6150
6110 LINE INPUT #1, TS$(ET)
6120 ET = ET + 1
6140 GOTO 6100
6150 CLOSE #DV
6155 ET = ET - 1
6160 PRINT "LOAD COMPLETED O.K."
6180 CLOSE #1
6190 GOTO 1000
10000 REM GET FILENAME
10010 LINE INPUT "ENTER FILENAME> "; N$
10020 PRINT "IS "; N$; " CORRECT?";
10030 A$ = INKEY$
10040 IF A$ = "" THEN 10030
10050 IF A$ < > "N" AND A$ < > "Y" THEN 10030
10060 PRINT A$
10070 IF A$ = "N" THEN 10000
10080 RETURN

```

## Program description

Lines	Purpose
10	Line orientated text editor.
20	Clear string space for text and clear text screen.
30	Display text editor title.
40	Sets text source/destination device number.



100 Set edit pointer to zero, end of text pointer to zero and maximum number of text lines to 500.  
 110 Dimension string array that will contain edited text.  
 120-140 Set all entries in array to null.  
 1000 Command interpreter starts here.  
 1010 Output prompt "\*" and get command from keyboard.  
 1020 If just < ENTER > typed, set a default command of move one line down text.  
 1030 Place command parameters in TP\$.  
 1040 Isolate command character in CM\$.  
 1050-1160 Check for command, if a match is found go to the relevant routine.  
 1200 Else if command fell through, check to see if it was a valid number.  
 1210 Set pointer to first character of string.  
 1220 Reconstruct full command string.  
 1230-1235 Validate command string as a number, if invalid display message to say so.  
 1240 Convert the string to a number.  
 1250 Add current edit pointer to the number.  
 1255 If number is past the last line of text display an error message else update the edit pointer.  
 1260 Display the current line and return to get another command.  
 1300-1310 Display error message, set edit pointer beyond end of text if required and return to get another command.  
 1350-1470 Display error messages.  
 1500 Interpret a command starting with a minus sign, ie negative command.  
 1510 If no other parameter, set a default command of move one line up text.  
 1520 Place command parameters in TP\$.  
 1530 Isolate command character in CM\$.  
 1540-1570 Check for command, if a match is found go to the relevant routine.  
 1600 Else if command fell through, check to see if it was a valid number.  
 1610 Set pointer to first character of string.  
 1620 Reconstruct full command string.  
 1630-1640 Validate command string as a number, if invalid display message to say so.

1650	Convert the string to a number.
1660	Subtract number from edit pointer.
1670	If number is above the first line of text, display an error message else update the edit pointer.
1700	Display the line of text pointed to by the edit pointer.
1710	Display line number and line if a valid line.
1720	Return to command interpreter.
1800	Display text command.
1810	If no parameters then just display single line of text.
1820	Point to first character in command parameter.
1830	Validate character as numeric, if not display appropriate error message else point to next character.
1835	If value of parameter is zero then ignore and return to command interpreter.
1840	Repeat for all characters in parameter.
1850	Start display at current line.
1860	Display line number and line of text.
1870	Point to next line.
1890-1900	If run off the end of the text then display appropriate error message, if all lines required displayed then return to command interpreter else display another line.
2300	Go to start of file command.
2310	Set edit pointer to first line.
2320	Display the line.
2500	Set edit pointer to absolute line number command.
2510	Make sure command has a parameter, if not display appropriate error message.
2520-2540	Validate parameter as a number, if not display appropriate error message.
2550	Convert to numeric value.
2560-2565	Check if line requested exists.
2570-2580	If it does, set edit pointer to the required line and display it.
3000	Find a specified string in the text from the current edit pointer position downwards.
3010	Set pointer to current edit pointer.
3020	Search for string on current line, if not found point to next line down.
3030	If end of text reached display appropriate message else search the next line.



3040-3050	String found, set edit pointer to line and display line.
3100	Find a specified string in the text from the current edit pointer position upwards.
3110	Set pointer to current edit pointer.
3120	Search for string on current line, if not found point to next line up.
3130	If end of text reached display appropriate message else search the next 3140-3150 String found, set edit pointer to line and display line.
3500	Find and replace string command down the file.
3510-3520	Check if string delimiter is present, if not display appropriate error message.
3530	TP\$ becomes string to find.
3540	CM\$ becomes string to replace it with.
3550-3570	Search found string, if not found display appropriate error message.
3580	Get position of string to be replaced in line.
3590	Replace string.
3600-3610	Point to and display line with new string in it.
3700	Find and replace string command up the file.
3710-3720	Check if string delimiter is present, if not display appropriate error message.
3730	TP\$ becomes string to find.
3740	CM\$ becomes string to replace it with.
3750-3770	Search for string, if not found display appropriate error message.
3780	Get position of string to be replaced in line.
3790	Replace string.
3800-3810	Point to and display line with new string in it.
4000	Delete lines downward command.
4010	Set number of lines to delete to one.
4020	If no parameter then take appropriate action.
4030	Else point to first character of parameter.
4050-4060	Validate parameter as numerical, if not display appropriate message.
4070	Convert to a number.
4080	If deletions run off the end of the file, set number of deletions to end of file only.
4085	Delete line zero ?.
4090-4110	Delete required number of lines by shuffling up.
4120	Update end of text pointer accordingly.
4130	Display current line.



4200	Delete lines upwards command.
4210	Set number of lines to delete to one.
4220	If no parameter then take appropriate action.
4230	Else point to first character of parameter.
4250-4260	Validate parameter as numerical, if not display appropriate message.
4270	Convert to a number.
4280	If deletions run off the top of the file, set number of deletions to top of file only.
4285	Delete line one ?.
4290-4310	Delete required number of lines by shuffling up.
4320	Update end of text pointer accordingly.
4330	Display current line.
4500	List text in buffer to printer command.
4510-4520	Set start line for list at start of text and end line at the end of text.
4530	Set line numbers flag to false.
4540	Check if any parameters.
4550	If so, get first character of parameters in TP\$.
4560	Check if line numbers required.
4570	If not, check for a list to (-) character, not present then error.
4580	If less than 2 characters then error.
4610	Extract first parameter.
4620	Check if second parameter is present, if so extract it in CM\$ else skip this parameters processing .
4630-4650	Validate second parameter as a number, if not, error.
4660	Set end line of listing to this parameter.
4670	If no first parameter to evaluate.
4680-4700	Else validate first parameter as a number.
4800	Set start line of listing to this parameter.
4810	Skip.
4850	Set line number flag to true.
4855	If no parameters remaining.
4860	Else isolate parameters.
4870	And evaluate.
4900-4930	Check all parameters are legal, if not display appropriate error message.
4940-4970	Print all required lines with line numbers if requested.
4980	Return to command interpreter.

5000	Text insertion command.
5010	Display line number text will be inserted at.
5020	Get line of text to insert.
5030	If text contains ^ then end insertion.
5035	If no space for text display error message.
5050-5070	Shuffle down text.
5075	Update end of text pointer.
5080	Insert text in buffer.
5090	Update edit pointer.
5100	See if any more text insertion is required.
5200-5220	Finished insertion, return to command interpreter.
5500	Save file command.
5510	Get name of file to save to.
5520-5550	Wait for tape to be placed in cassette drive.
5560	Open the file.
5570-5590	Write the file to tape.
5600	Close the file.
5610-5620	Display save o.k. message and return to command interpreter.
6000	Load file command.
6010	Get filename from which to load.
6020	Reset text pointers.
6030-6050	Wait for tape to be place in cassette drive.
6060	Open the file.
6070-6090	Clear down buffer space.
6100	If end of file then finished reading.
6110	Else read in next line.
6120	Update text length record.
6140	Read in new line ?.
6150	Close file.
6155	Correct end of text pointer.
6160-6190	Display load completed message and return to command interpreter.
10000	Get a filename.
10010	Get the name.
10020-10080	Double check it, if correct return.

## Variable definitions

Variable name	Function
TS\$	String array which contains the text to be edited.

CM\$	String which contains the most recent command.
TP\$	String used in command analysis.
L\$	String used for input of a text line.
A\$	String used for the answer to a question.
DV	Device number used as source and destination for edited files.
NL	Maximum number of editable lines.
EP	Edit pointer.
ET	End of current file of text.
ES	Used for the finding and replacement of strings.
ND	Number of lines to delete.
LN	Line number flag, set if line numbers required when printing a file.
SL	First line to print.
EL	Last line to print.

## **W-PRO**

W-Pro is a simple word processor which includes such features as proportional spacing of text, underlining of text and boldface type. W-Pro operates on a source file of text which can be produced using Editor. Format of W-Pro's output is controlled by commands embedded in the text. The device (e.g. tape,disk or printer) from which the source file is taken and to which the output or destination file is written may be reconfigured within the program. Initial values of such functions as paragraph indent, margin positions and page length can also be setup before source file processing takes place.

### **Brief command summary**

W-Pro commands take the form of special sequences of characters in the source text,such as:-

**#N**

The hash (" #") character indicates to W-Pro that the following word is a command.(If the one hash character itself is needed,simply use two consecutive hashes in the source file).

e.g. ## in the source file appears as # in the output



N	New paragraph
R	Set right hand margin
L	Set left hand margin
P	New page
S	Toggle proportional spacing
U	Toggle underline
B	Toggle boldface type
A	Turn on/off page numbering
T	Tabulate to column relative to left hand margin

## Detailed command description

### N—new paragraph command

This command takes the form:

#Nn

Where "n" is a number representing the required paragraph indent relative to the current left hand margin. If the numerical parameter "n" is omitted the last set or default paragraph indent is used.

The command starts a new line in the output text and indents the start of text by the required number of spaces.

e.g. #N0 This is an #10 example of #N a new #N5 paragraph  
in the source text would produce:

This is an  
    example of  
    a new  
    paragraph

in the output text.

### R—set right hand margin command

This command takes the form:

#Rn

The command sets the absolute position of the right hand margin in the output text (not relative to left hand margin) to value "n". No output text line will go beyond this margin and if proportional spacing is enabled text will be justified to this margin.

e.g. #R40 will set the right hand margin to the 40th column

**L**—set left hand margin command

This command takes the form:

#Ln

The command sets the absolute position of the left hand margin in the output text to value "n". No output text line will start before this margin and if proportional spacing is enabled text will be justified to this margin.

e.g. #L10 will set the right hand margin to the 10th column

**P**—new page command

This command takes the form:

#Pn

Where "n" is an optional parameter for page length. This command forces the output text to the start of the next page.

e.g. #P66 would set the page length to 66 lines and set the output text to the start of the next page.

**S**—proportional spacing command

This command takes the form:

#S

The command changes the state of proportional spacing enable. For example, if proportional spacing is disabled when the command is encountered, proportional spacing will then be enabled. When the command is next encountered proportional spacing will be disabled again.

**U**—underline text command

This command takes the form:

#U

The command changes the state of text underline enable. For example, if text underline is disabled when the command is encountered, text underline will then be enabled and the following text underlined. When the command is next encountered text underline will be disabled again.

**B**—boldface type command

This command takes the form:

#B

The command changes the state of boldface type enable. For example, if boldface type is disabled when the command is encountered, boldface type will then be enabled and the following text will appear in boldface. When the command is next encountered boldface type will be disabled again.

**A**—page numbering command

This command takes the form:

#An

When no numerical parameter ("n") accompanies this command it changes the state of page numbering enable. If it is currently disabled it is enabled and vice versa. The optional numerical parameter governs the number which will be printed at the end of the current page of text.

e.g. #A20 would set the next page number to 20.

**T**—tabulate command

This command takes the form:

#Tn



The tabulate command works relative to the left hand margin. It pads out the output text to column "n" with spaces.

e.g. This #T10 is a test line of text.

in the source file would produce:

This is a test line of text.

in the output file.

## Program listing

```
10 REM WORD PROCESSOR 1.0
20 CLEAR 1000
100 REM DEFINE VARS, SET UP DEFAULT STATUS
110 DIM O$(2,255),S(255)
120 CM$=" #"
130 PL=67
140 L=0
150 R=40
160 PS=1
170 U=0
180 B=0
190 DP=5
200 A=0
210 N=0
220 S=1
230 D=-2
240 F=0
250 NW=0
260 LO=0
270 SU=95
280 C=0
290 F1=0
300 EC=0
310 N$=""
320 C$=""
330 L$=""
340 W$=""
350 OV$=""
360 OU$=""
```

```

370 T$ = ""
380 CR = 26
390 SL = 0
400 M = 0
410 P = 1
420 REM
430 CLS
440 PRINT " WORD PROCESSOR 1.0"
450 PRINT " ===== "
460 PRINT
470 PRINT " 0: QUIT."
480 PRINT
490 PRINT " 1: SELECT I/O DEVICES."
500 PRINT
510 PRINT " 2: SETUP OPTIONS."
520 PRINT
530 PRINT " 3: START PROCESSING."
540 PRINT
550 PRINT
560 PRINT " ENTER OPTION> "
570 A$ = INKEY$
580 IF A$ = "" THEN 570
590 IF A$ < "0" OR A$ > "3" THEN 420
600 PRINT A$
610 IF A$ = "0" THEN END
620 ON VAL(A$) GOSUB 11500,9000,640
630 GOTO 420
640 GOSUB 12000
650 GOSUB 10000
660 IF F THEN 690
670 GOSUB 1000
680 GOTO 650
690 CLOSE #S
700 PRINT "SOURCE FILE CLOSED"
710 GOSUB 1300
720 CLOSE #D
730 PRINT "DESTINATION FILE CLOSED"
740 FOR I = 1 TO 1000
750 NEXT I
760 RETURN
1000 REM WORD INTERP

```

*Hot programs to feed your Dragon*

```
1010 IF MID$(W$,1,1) = CM$ THEN 2200
1020 IF LEN(W$) > R-L THEN 1200
1030 LO = LO + LEN(W$) + 1
1040 IF LO-1 > R-L THEN 1300
1050 NW = NW + 1
1060 O$(1,NW) = W$
1070 O$(2,NW) = CHR$(1)
1080 IF U THEN O$(2,NW) = CHR$(ASC(O$(2,NW))OR 64)
1090 IF B THEN O$(2,NW) = CHR$(ASC(O$(2,NW))OR 128)
1100 RETURN
1200 REM WORD TOO LONG
1210 L$ = RIGHT$(W$,LEN(W$)-R-L+1) + " " + L$
1220 IF R-L < 2 THEN EC = 100:GOSUB 2400
1230 W$ = LEFT$(W$,R-1-L) + "-"
1240 GOTO 1030
1250 PRINT
1300 REM OUTPUT LINE FILLED
1310 IF NW < 1 THEN 1670
1320 IF L = 0 THEN 1360
1330 PRINT #D,STRING$(L," ");
1340 OV$ = OV$ + STRING$(L," ")
1350 OU$ = OU$ + STRING$(L," ")
1360 F1 = 0
1370 IF NP = 0 THEN L$ = W$ + " " + L$
1380 LO = LO - LEN(W$) - 1
1390 IF ((PS AND (F = 0)) AND (NP = 0)) THEN GOSUB 1900
1400 IF NW < 1 THEN 1680
1410 FOR I = 1 TO NW
1420 IF (ASC(O$(2,I)) AND 64) THEN OU$ = OU$ + STRING$(
(LEN(O$(1,I)),CHR$(SU)) ELSE OU$ = OU$ +
STRING$(LEN(O$(1,I))," ")
1500 IF I < NW THEN O$(1,I) = O$(1,I) + STRING$(ASC(O$(2,I))
AND 63," ")
1510 IF I < NW THEN OU$ = OU$ + STRING$(ASC(O$(2,I))
AND 63," ")
1520 PRINT #D,O$(1,I);
1530 IF (ASC(O$(2,I))) < 128 THEN 1570 ELSE OV$ = OV$ + O$(1,I)
1540 F1 = 1
1550 NEXT I
1560 GOTO 1600
1570 OV$ = OV$ + STRING$(LEN(O$(1,I))," ")
```



```

1580 NEXT I
1600 IF F1=0 THEN 1660
1610 FOR I=1 TO 3
1620 IF INSTR(1,OU$,CHR$(SU))>0 THEN PRINT #D,CHR$(CR)
;OU$;
1630 PRINT #D,CHR$(CR);OV$;
1640 NEXT I
1650 GOTO 1670
1660 IF INSTR(1,OU$,CHR$(SU))>0 THEN PRINT #D,CHR$(CR)
;OU$
1670 PRINT #D
1680 LO=0
1690 OV$=""
1700 OU$=""
1730 NW=0
1740 P=P+1
1750 IF P<PL-4 THEN RETURN
1760 PRINT #D
1770 PRINT #D
1780 IF A THEN 1830
1790 PRINT #D
1800 PRINT #D
1810 P=1
1820 RETURN
1830 N$=STR$(N)
1840 N$=STRING$(INT((R-L)/2)-INT(LEN(N$)/2)+L,"") + N$
1850 PRINT #D,N$
1860 N=N+1
1870 GOTO 1800
1900 REM PROP SPACING ROUTINE
1910 IF NW<2 THEN RETURN
1920 FOR I=1 TO NW-1
1930 S(I)=1
1940 NEXT I
1950 FOR I=1 TO NW-1
1960 J=1
1970 IF LEN(O$(1,I))>=LEN(O$(1,S(J))) THEN 2020 ELSE J=J+1
1980 IF J<I THEN 1970
1990 S(J)=I
2000 NEXT I
2010 GOTO 2060

```

```

2020 FOR K = NW TO J STEP -1
2030 S(K) = S(K-1)
2040 NEXT K
2050 GOTO 1990
2060 J = 1
2070 IF R-L-(LO-1) < 1 THEN RETURN
2080 FOR I = 1 TO R-L-(LO-1)
2090 K = ASC(0$(2,S(J)))
2100 IF (K AND 63) > 62 THEN NEXT I ELSE K = K + 1
2110 0$(2,S(J)) = CHR$(K)
2120 IF J > NW-1 THEN J = 1 ELSE J = J + 1
2130 NEXT I
2140 RETURN
2200 REM COMMAND INTERP
2210 IF LEN(W$) < 2 THEN 2350
2220 W$ = RIGHT$(W$, LEN(W$)-1)
2230 IF MID$(W$, 1, 1) = CM$ THEN 1020
2240 C$ = LEFT$(W$, 1)
2250 W$ = RIGHT$(W$, LEN(W$)-1)
2260 IF C$ = "N" THEN 2500
2270 IF C$ = "R" THEN 2800
2280 IF C$ = "L" THEN 2900
2290 IF C$ = "P" THEN 3000
2300 IF C$ = "S" THEN 3500
2310 IF C$ = "U" THEN 3600
2320 IF C$ = "B" THEN 3700
2330 IF C$ = "A" THEN 3800
2340 IF C$ = "T" THEN 4000
2350 EC = 1
2400 REM RUNTIME ERROR
2410 IF M THEN 2450
2420 PRINT "ERROR "; EC; " AT LINE "; SL
2430 L$ = " #B ERROR" + STR$(EC) + ">" + "$" + C$ + W$ + "
    #B" + L$
2440 RETURN
2450 PRINT "ERROR "; EC
2460 FOR I = 1 TO 500
2470 NEXT I
2480 RETURN
2500 REM NEW PARAGRAPH
2510 IF LEN(W$) > 0 THEN 2530

```

```

2520 GOTO 580
2530 GOSUB 20000
2540 IF V>=0 THEN 2570
2550 EC=2
2560 GOTO 2400
2570 IF V>R-L THEN 2550 ELSE DP=V
2580 IF M THEN RETURN ELSE F1=0
2590 NP=1
2600 GOSUB 1300
2610 NP=0
2620 OV$=OV$+STRING$(DP," ")
2630 OU$=OU$+STRING$(DP," ")
2640 PRINT #D,STRING$(DP," ");
2650 LO=LO+DP
2660 RETURN
2800 REM SET RIGHT HAND MARGIN
2810 IF LEN(W$)>0 THEN 2840
2820 R=80
2830 RETURN
2840 GOSUB 20000
2850 IF V<L OR V>255 THEN 2550
2860 R=V
2870 RETURN
2900 REM SET LEFT HAND MARGIN
2910 IF LEN(W$)>0 THEN 2940
2920 L=0
2930 RETURN
2940 GOSUB 20000
2950 IF V>R OR V<0 THEN 2550
2960 L=V
2970 RETURN
3000 REM NEW PAGE
3010 IF LEN(W$)>0 THEN 3100
3020 IF M<>0 THEN RETURN ELSE NP=1
3030 GOSUB 1300
3040 NP=0
3050 FOR I=P TO PL-4
3060 PRINT #D
3070 NEXT I
3080 P=PL-4
3090 GOTO 1740

```



```

3100 GOSUB 20000
3110 IF V < 1 OR V > 255 THEN 2550
3120 PL = V
3130 GOTO 3020
3500 REM TURN ON/OFF PROP SPACING
3510 IF LEN(W$) > 0 THEN 2550
3520 IF PS THEN PS = 0 ELSE PS = 1
3530 RETURN
3600 REM TURN ON/OFF UNDERLINE
3610 IF U THEN U = 0 ELSE U = 1
3620 RETURN
3700 REM TURN ON/OFF OVERPRINT
3710 IF B THEN B = 0 ELSE B = 1
3720 RETURN
3800 REM TURN ON/OFF PAGE NUMBERING
3810 IF LEN(W$) < 1 THEN 3850
3820 GOSUB 20000
3830 IF V < 0 THEN 2550
3840 N = V
3850 IF A THEN A = 0 ELSE A = 1
3860 RETURN
4000 REM TAB TO COL REL TO L-MARGIN
4010 IF LEN(W$) < 1 THEN 2550
4020 GOSUB 20000
4030 IF V < 1 THEN 2550
4040 IF LO-1 < L + V AND V + L < R THEN 4060 ELSE EC = 3
4050 GOTO 2400
4060 IF NW > 0 THEN 4110 ELSE OV$ = OV$ + STRING$(L + V - LO, " ")
4070 OU$ = OU$ + STRING$(L + V - LO, " ")
4080 PRINT #D, STRING$(L + V - LO, " ");
4090 LO = LO + (L + V - LO)
4100 RETURN
4110 O$(2, NW) = CHR$(L + V - LO)
4120 GOTO 4090
9000 REM ENTER STARTUP CONFIG
9010 M = 1
9020 CLS
9030 PRINT " STARTUP CONFIGURE MODE"
9040 PRINT " = = = = = "
9050 PRINT
9060 PRINT "PROPORTIONAL SPACING ";

```

```

9070 IF PS THEN PRINT"ENABLED" ELSE PRINT"DISABLED"
9080 PRINT"UNDERLINE.....";
9090 IF U THEN PRINT"ENABLED" ELSE PRINT"DISABLED"
9100 PRINT"BOLD PRINT.....";
9110 IF B THEN PRINT"ENABLED" ELSE PRINT"DISABLED"
9120 PRINT"PAGE NUMBERING.....";
9130 IF A THEN PRINT"ENABLED" ELSE PRINT"DISABLED"
9140 PRINT"LEFT HAND MARGIN....";L = 0
9150 PRINT"RIGHT HAND MARGIN....";R = 0
9160 PRINT"PARAGRAPH INDENT....";DP = 0
9170 PRINT"PAGE LENGTH.....";PL = 0
9180 PRINT"PAGE NUMBER.....";N = 0
9190 PRINT
9200 PRINT"ENTER COMMAND (ENTER TO QUIT)";
9210 LINE INPUT"> ";W$
9220 IF W$ = "" THEN 9250
9230 GOSUB 2200
9240 GOTO 9000
9250 M = 0
9260 RETURN
10000 REM READ IN WORD
10010 IF LEN(L$) > 0 THEN 10060
10020 IF EOF(S) THEN F = 1 ELSE 10040
10030 RETURN
10040 LINE INPUT "#S,L$
10050 IF L$ = "" THEN 10020
10060 IF INSTR(1,L$," ") > 0 THEN 10100
10070 W$ = L$
10080 L$ = ""
10090 RETURN
10100 WP = 1:W$ = ""
10110 IF MID$(L$,WP,1) < > " " THEN 10130 ELSE WP = WP + 1
10120 IF WP < LEN(L$) THEN 10110 ELSE 10020
10130 W$ = W$ + MID$(L$,WP,1)
10140 IF WP > = LEN(L$) THEN 10070 ELSE WP = WP + 1
10150 IF MID$(L$,WP,1) < > " " THEN 10130 ELSE L$ = RIGHT$(L$,
LEN(L$)-WP)
10160 RETURN
11000 REM GET FILENAME
11010 LINE INPUT"ENTER FILENAME> ";N$
11020 PRINT"IS ";N$;" CORRECT?";

```

```

11030 A$ = INKEY$
11040 IF A$ = "" THEN 11030
11050 IF A$ <> "N" AND A$ <> "Y" THEN 11030 ELSE PRINT A$
11060 IF A$ = "N" THEN 11000 ELSE RETURN
11500 REM GET DEVICE ASSIGNMENTS
11510 CLS
11520 PRINT "DATA DEVICE ASSIGNMENT"
11530 PRINT "===== "
11540 PRINT
11550 PRINT "VALID DEVICE NUMBERS ARE:-"
11560 PRINT
11570 PRINT "1: DISK DRIVE."
11580 PRINT "-1: TAPE DRIVE."
11590 PRINT "-2: PRINTER."
11600 PRINT "0: CONSOLE."
11610 PRINT
11620 INPUT "ENTER SOURCE";S
11630 INPUT "ENTER DESTINATION";D
11640 IF S = 1 OR S = -1 OR S = 0 THEN 11650 ELSE 11500
11650 IF D = -2 OR D = 1 OR D = 0 THEN 11670 ELSE IF D <> -1 THEN
11500
11660 IF D = S THEN 11500
11670 PRINT "IS ABOVE CORRECT?";
11680 A$ = INKEY$
11690 IF A$ = "" THEN 11680
11700 IF (D = 1) AND (S = 1) THEN D = 2
11710 IF A$ <> "N" AND A$ <> "Y" THEN 11680
11720 PRINT A$
11730 IF A$ = "N" THEN 11500 ELSE RETURN
12000 REM OPEN DATA PATHS
12010 CLS
12020 PRINT TAB(7); "DATA PATH OPENER"
12030 PRINT TAB(7); "===== "
12040 PRINT
12050 PRINT
12060 IF S = 0 THEN 12150
12070 PRINT TAB(7); "source filename"
12080 PRINT
12090 GOSUB 11000
12100 IF S <> -1 THEN 12130
12110 PRINT "HIT SPACE WHEN TAPE READY"

```



```

12120 IF A$ < > " " THEN 12120
12130 OPEN "I", #S, N$
12140 PRINT "SOURCE "; N$; " OPENED"
12150 IF D = -2 OR D = 0 THEN 12210
12160 PRINT
12170 PRINT TAB(7); "destination filename"
12180 PRINT
12190 GOSUB 11000
12200 OPEN "O", #D, N$
12210 PRINT "DESTINATION OPENED O.K."
12220 FOR I = 1 TO 1000
12230 NEXT I
12240 RETURN
20000 REM VALIDATE NUMERIC PARAM
20010 K = 1
20020 IF MID$(W$, K, 1) < "0" OR MID$(W$, K, 1) > "9" THEN 20060
ELSE K = K + 1
20030 IF K <= LEN(W$) THEN 20020
20040 V = VAL(W$)
20050 RETURN
20060 V = -1
20070 RETURN

```

## Program description

Lines	Purpose
20	Reserve space for string variables.
110	Setup string arrays for text processing.
120	Set command character.
130	Set default page length.
140-150	Set default left and right hand margin positions.
160	Set proportional spacing flag to default (0 = off, 1 = on).
170	Set underline off.
180	Set boldface off.
190	Set default paragraph indent.
200	Page numbering off.
210	Set page number to 0.
220-230	Set default data source and destination devices.
240	End of file flag to not end of file.

250	Number of words to zero.
260	Length of destination line to zero.
270	Set underline character code.
280-370	General variable initialisation.
380	Set carriage return code (without line feed) for printer.
390	Set output line count to zero.
400	Clear manual configure flag.
410	Source line count = 1.
430-560	Clear text screen and display main menu.
570-600	Wait till valid option number entered from keyboard.
610	Terminate execution if end option was selected.
620	Call the selected options subroutine.
630	Re-start main menu sequence.
640	Beginning of text processing sequence, open data paths.
650	Read a word of source text.
660	Check if end of source file.
670	Interpret word and take required action.
680	Repeat word read and interpretation sequence until end of file.
690-750	Output last line, close source and destination files.
760	Return to main menu.
1000	Word interpreter starts here.
1010	Check word for command character, if present go to command interpreter.
1020	Test if word is too long to fit between margins.
1030	Add in word length to output line length total.
1040	Check if line has reached required destination line length.
1050	Increment output line word count.
1060	Add word into output line string.
1070	Set trailing spaces associated with word to 1.
1080	Add in current underline flag setting (0 = off, 1 = on).
1090	As above but for boldface type.
1100	Word interpretation completed.
1200	Deal with over-long word.
1210-1240	Split up word and hyphenate it.
1300	Output line filled, ready to send to destination device.
1310	Make sure there is at least one word to send.
1320-1350	Add in space to pad out to left hand margin.
1360	Clear flag which indicates if line has any boldface type.

- 1370 If not at the start of a paragraph, return unused word for processing as part of the next line.
- 1380 Remove unused word length from output line length.
- 1390 Proportionally space line if required.
- 1400 Skip if no words left in output line.
- 1410 Repeat for each word in output line.
- 1420 If word requires underline, pack underline string with underline characters for length of word, else pack with spaces.
- 1500 Pack word in output string with required number of spaces.
- 1510 Add required spaces to underline string as well.
- 1520 Output text word.
- 1530-1580 If boldface type was selected for word, add word to overlay string and set boldface flag, else pack to word length with spaces.
- 1600 All words done. Skip if no boldface type present in the line.
- 1610-1650 Overprint with boldface and underline overlay strings.
- 1660 If required print underline overlay.
- 1670 Send a new line to destination.
- 1680 Zero output line length count.
- 1690-1700 Clear down overlay strings.
- 1740-1750 Increment page line count, if not new page return.
- 1760-1770 Line feeds to destination file.
- 1780 Check if page numbering enabled.
- 1790-1800 Line feeds to destination.
- 1810 Reset page line count.
- 1820 Return.
- 1830 Convert page number into a string.
- 1840 Center page number string within left and right hand margins.
- 1850-1870 Print page number, increment page number and set destination file to next page.
- 1900 Routine to proportionally space a line starts here.
- 1910 If only one word on the line, proportional spacing is impossible.
- 1920-1940 Set an element in array S to 1 for each word in the output line.
- 1950-2060 Each word in the line is represented by a number in array S, the first word in the line being represented by



	number one. The words are sorted into order with the longest word entered first. (i.e. at the end of the sort the largest word will be pointed to by the first element of array S).
2070	If no space is left on the line then no space addition is required.
2080	Repeat trailing space addition to words for all of the spaces left on the line.
2090	K gets the current number of spaces added to the currently selected word.
2100	If no room for space to be added to this word go to next word. Else add in an extra space.
2110	Update words space record.
2120	Set next word for space addition.
2130	Repeat ?.
2140	End of space addition, return.
2200	Command interpreter starts here. Takes appropriate action on a command planted in the source file.
2210	If word less than 2 characters long, must be an invalid command.
2220	Remove command character from string.
2230	Check for successive command characters, if so treat as one command character in text.
2240	Extract command.
2250	Delete command from rest of command word.
2260-2340	Check for commands and call appropriate subroutine.
2350	Command was not valid, set error code.
2400	Runtime error handler.
2410	If in manual parameter setup.
2420	Else display error code and source line number at which it occurred on the console.
2430	Plant error message with error code in boldface type at the point it occurred in the destination file.
2440	Return to text processing.
2500	New paragraph command.
2510	Check if any more characters accompany the command.
2520	If not, exit.
2530	Else validate parameter as a numeric value (paragraph indent).
2540	If parameter numerically valid.

2550-2560	Else set and process error code.
2570	If numeric value for paragraph indent is within allowable range set new indent value, else set and display error code.
2580	If in manual setup mode return.
2590-2660	Else flag new paragraph/new line and process.
2800	Right hand margin command.
2810-2830	Check if any parameter accompanies command if not, set default margin width.
2840	Validate as numeric parameter (right hand margin indent).
2850	Check if in allowable range, if not set error code.
2860-2870	Set new right hand margin and return.
2900	Left hand margin command.
2910-2930	Check if any parameter accompanies command if not, set default margin width.
2940	Validate as numeric parameter (left hand margin width).
2950	Check if in allowable range, if not set error code.
2960-2970	Set new left hand margin and return.
3000	New page command.
3010	Check if any parameter accompanies command.
3020	Else if manual mode return else set new page.
3030	Process new page.
3040	Set not new page.
3050-3070	Send line feeds to end of current page.
3080-3090	Output page number if required.
3100	Validate numeric parameter (page length).
3110	Check if within range, if not set error code.
3120	Set value of new page length.
3130	Output new page data.
3500	Proportional spacing on/off command.
3510	Check if any parameter accompanies command, if so then set error.
3520-3530	Turn on or off as required and return.
3600	Underline on/off command.
3610-3620	Turn on or off as required and return.
3700	Boldface type on/off command.
3710-3720	Turn on or off as required and return.
3800	Turn on/off page numbering.
3810	Check if any parameter accompanies command.



3820-2830	If so, validate numeric parameter, set error code if required.
3840	Set new page number.
3850-3860	Turn on or off as required and return.
4000	Tabulate to column relative to left hand margin command.
4010	Check if any parameter accompanies command, if not set error code.
4020-4030	If so, validate numeric parameter, set error code if required.
4040-4050	Check to see if tab value in range, if not set error code.
4060-4080	If no words have been processed output number of spaces needed to tab to correct column.
4090-4100	Set output line length accordingly and return.
4110-4120	Else add on required number of spaces to current word.
9000	Startup configuration mode.
9010	Set manual mode flag.
9020	Clear text screen.
9030-9200	Display menu of options.
9210	Get command.
9220	If null command then leave configure mode.
9230	Interpret command.
9240	Repeat sequence.
9250-9260	Clear manual mode flag and return.
10000	Read in a word from source file.
10010	If last line has all been processed, read in a new line.
10020-10030	If end of source file, set flag and return.
10040	Read in new line from source file.
10050	If empty line, try again.
10060	If more than one word on line.
10070-10090	Else set next word, empty line and return.
10100	Set word pointer to first character in line, clear word.
10110-10120	Remove all leading spaces from word.
10130-10160	Build up word until a trailing space is detected or the end of the line is reached. Remove that word from the line and return.
11000-11060	Get a filename.
11500	Get source and destination file device assignments.
11510	Clear text screen.
11520-11610	Display menu of options.



11620	Prompt for and get source device number.
11630	Prompt for and get destination device number.
11640	Check for valid source device.
11650-11660	Check for valid destination device.
11670-11730	Get and act on yes/no response from keyboard.
12000	Open files (data paths) for source and destination.
12010-12050	Display heading.
12060-12090	Get source filename if required.
12100-12120	Display tape prompt and wait for response if device selected is cassette.
12130-12140	Open source file and display file opened message.
12150-12210	As above but for destination file.
12220-12240	Short delay and return.
20000	Validate a numeric parameter (W\$).
20010	Set pointer to first character in string.
20020	Check that character is a numeral, if so point to next character else return error.
20030	Repeat for all characters in parameter.
20040	Convert to numerical value in V.
20050	Return.
20060	V set to -1 indicates invalid number.
20070	Return.

## Variable definitions

<i>Variable name</i>	<i>Function</i>
A\$	Holds the response to a question.
C\$	This string contains the isolated command letter from a command word.
CM\$	A string used to store command character (normally #).
L\$	A string which contains a line of text read from the source file.
V\$	This string is used for the build up of overlay text for a boldface type on a line.
OU\$	A string used for the generation of underline and space characters for underlining text.
W\$	A string containing a word read from the source file.
O\$	This two dimensional array of strings contains in one dimension (1) the words to be output. In the other dimension (2) the bottom 6 bits of the first character

	of the string contain the number of trailing spaces required after that word. The top bit contains the setting of the boldface flag at the time the word was processed, and the next bit down contains the underline flag status at that time.
A	Flag to indicate page numbering required
B	Flag to indicate boldface type
M	Manual setup flag, when set indicates not in text processing mode.
U	Flag to indicate when underline required.
PS	Flag to indicate proportional spacing required.
NP	Flag to indicate start of new paragraph.
F	End of source file flag.
F1	General purpose flag.
PL	Contains the current required page length.
L	Current position of the left hand margin.
R	Current position of the right hand margin.
DP	Current paragraph indent (relative to left hand margin).
N	Page number.
S	Number of source file device.
D	Number of destination file device.
NW	Number of words in output line.
WP	Output line word pointer.
LO	Length of output line.
SU	Character code of underline character.
EC	Last recorded error code.
CR	Character code of carriage return without line feed for printer.
SL	Source line count.
P	Page line count.
V	Value of last numerical string.
S	Array used to sort words into length order.

## TELECALC

Tired of paying exorbitant 'phone bills?. Try this program to keep a check on those long distance calls.

The program will ask you to select the appropriate charging rate. These rates depend upon the time of day at which the call is connected, and can be found in the telephone charges leaflet issued to all subscribers.

The program also needs to know the call distance, and uses the conventions defined in the STD code booklet.

"Special rate" is for overseas calls and will ask for the time allowed (in seconds) for the "standard charge unit". The latter is currently fixed at 4.3 pence + VAT (i.e. 4.945 pence). Note that this will change in late 1983, so be ready to edit line 40!.

### Program listing

```

10 REM TELEPHONE CALL COST CALCULATOR
20 REM CHARGE RATES IN COST IN PENCE PER MINUTE
30 F = 49.6911249
40 MC = 0.043*1.15 : REM CALCULATE RATE INCLUDING VAT
50 CLS:R=0
60 PRINT " TELEPHONE CALL COST CALCULATOR"
70 PRINT " = = = = = "
  = = = = = "
80 PRINT
90 PRINT TAB(7)"0) QUIT."
100 PRINT
110 PRINT TAB(7)"1) PEAK RATE."
120 PRINT
130 PRINT TAB(7)"2) STANDARD RATE."
140 PRINT
150 PRINT TAB(7)"3) CHEAP RATE."
160 PRINT
170 PRINT TAB(7)"4) SPECIAL RATE."
180 PRINT
190 PRINT
200 PRINT TAB(2)"ENTER RATE> ";
210 A$ = INKEY$
220 IF A$> = "0" AND A$< "5" THEN 230 ELSE 210
230 PRINT A$;
240 IF A$ = "0" THEN END
250 IF A$ = "4" THEN 480
260 ON VAL(A$) GOSUB 1000,1050,1100
270 CLS

```



```
280 PRINT TAB(6)"CALL DISTANCE SELECT"
290 PRINT TAB(6)" = = = = = "
300 PRINT
310 PRINT
320 PRINT TAB(7)"L) LOCAL CALL."
330 PRINT
340 PRINT TAB(7)"A) UP TO 56KM."
350 PRINT
360 PRINT TAB(7)"B) OVER 56KM."
370 PRINT
380 PRINT
390 PRINT" ENTER OPTION> ";
400 A$ = INKEY$
410 IF A$ = "" THEN 400
420 PRINT A$
430 IF A$ = "L" THEN R = L
440 IF A$ = "A" THEN R = A
450 IF A$ = "B" THEN R = B
460 IF R = 0 THEN 50
470 GOTO 650
480 REM DEAL WITH SPECIAL CHARGES
490 CLS
500 PRINT TAB(8)"SPECIAL CHARGES"
510 PRINT TAB(8)" = = = = = "
520 PRINT:PRINT
530 PRINT" ENTER TIME IN SECS/CHARGE UNIT"
540 PRINT
550 PRINT" (CHARGE UNIT = ";MC*100;"PENCE)"
560 PRINT:PRINT:PRINT
570 PRINT TAB(8);
580 INPUT R
590 IF R > 0 THEN 610
600 GOTO 480
610 PRINT
620 PRINT"IS THE ABOVE CORRECT(Y/N)?";
630 A$ = INKEY$
640 IF A$ < > "N" AND A$ < > "Y" THEN 630 ELSE IF A$ = "N" THEN
480
650 REM SET UP CHARGE/TIME HEADINGS
660 CLS
670 PRINT@ 129,"CHARGE"
```

```

680 PRINT " RATE TIME COST"
690 PRINT "(SEC/UNIT) (PENCE)"
700 PRINT " = = = = = "
710 PRINT@416
720 PRINT " type space when call starts";
730 PRINT@257,R;
740 IF INKEY$ < " " THEN 740
750 TIMER = 0
760 PRINT@416
770 PRINT " type space when call finished";
780 REM DISPLAY ROUTINE
790 T = TIMER/F
800 M = INT(T/60)
810 S = T - (M*60)
820 PRINT@269, USING " # #.# # #";M + (INT(T/R) + 1) ELSE PRINT
"$";MC*100
830 PRINT TAB(23);
840 IF T' = R THEN PRINT "$";100*MC*(INT(T/R) + 1) ELSE
PRINT "$";MC*100
850 IF INKEY$ < " " THEN 790
860 PRINT@416
870 PRINT " type space to return to menu ";
880 IF INKEY$ < " " THEN 880 ELSE 50
1000 REM SET PEAK RATES
1010 L = 90
1020 A = 30
1030 B = 12
1040 RETURN
1050 REM SET STANDARD RATES
1060 L = 120
1070 A = 45
1080 B = 16
1090 RETURN
1100 REM SET CHEAP RATES
1110 L = 480
1120 A = 144
1130 B = 48
1140 RETURN

```

### Program description

Lines	Purpose
30	Set timer calibration variable.

40	Set standard charge unit (4.3p + VAT).
50-260	Print menu,select option.
270	Come here in modes 1,2,3 to select distance.
280-460	Select value of rate (R).
470	Skip next if not special rate.
480-580	Ask for special rate for overseas calls.
590	Can't have zero rate.
610-720	Check rates,set up display for timing.
740	Wait for space bar to be pressed before starting.
750	Reset timer.
760-770	Remind how to end timer.
790	Calibrate timer with correction factor.
800	Calculate minutes.
810	Calculate seconds.
820	Print time.
830	Move to cost column.
840	If less then 1 unit of charge then print one unit,else print cost = rate*time.
850	Check for end.
860-880	Pause to allow meter to be read.
1000-1140	Subroutines which define rates for various time bands and distances (see British Telecom leaflets).

## Variable definitions

<i>Variable name</i>	<i>Function</i>
F	Correction factor for timer.
MC	Standard charge unit in pence.
R	Charge rate.
A\$	Used for replies.
T	Elapsed time.
M	Elapsed minutes.
S	Elapsed seconds.
L,A,B	Distance weightings for charges.

## TELENUM

Telenum is a telephone directory program. It allows the user to add and delete entries of names and telephone numbers. Telephone numbers can be found by entering the name of the person who's number is required. A search is then



made of the directory for this name and possible matches are displayed. The program can also match telephone numbers to find the name of a person. Directories can be saved to, and re-loaded from, cassette tape. The program will also allow any fraction of the required name to be entered and display any possible matches to that fragment.

### Program listing

```

10 REM TELEPHONE DIRECTORY
15 CLEAR 10000
20 N = 100
30 DIM D$(N,2)
40 NE = 2
45 NU = 1
50 FOR I = 1 TO N
60 D$(I,NE) = ""
70 D$(I,NU) = ""
80 NEXT I
200 CLS
210 PRINT TAB(5)"TELEPHONE DIRECTORY"
220 PRINT TAB(5)" = = = = = "
230 PRINT:PRINT
240 PRINT TAB(6)"0) QUIT."
250 PRINT TAB(6)"1) LOAD DIRECTORY."
260 PRINT TAB(6)"2) SAVE DIRECTORY."
270 PRINT TAB(6)"3) ENTER A NUMBER"
280 PRINT TAB(6)"4) FIND A NUMBER."
290 PRINT TAB(6)"5) DELETE A NUMBER."
400 PRINT:PRINT
410 PRINT TAB(3) "ENTER OPTION' ";
420 A$ = INKEY$
430 IF A$ < "0" OR A$ = "5" THEN 420
440 IF A$ = "0" THEN END
450 ON VAL(A$) GOSUB 1000,2000,3000,4000,5000
460 GOTO 200
1000 REM LOAD IN DIRECTORY
1005 CLS
1010 PRINT TAB(5)"DIRECTORY LOAD"
1020 PRINT TAB(5)" = = = = = "
1030 PRINT@ 5*32
1040 PRINT"ENTER FILENAME('ENTER' TO QUIT)"
1045 PRINT:PRINT TAB(5);

```

*Hot programs to feed your Dragon*

```
1050 LINE INPUT " ";F$
1060 IF F$ = "" THEN RETURN
1070 PRINT
1080 PRINT "IS ";F$;" CORRECT?";
1090 A$ = INKEY$
1100 IF A$ < "Y" AND A$ < "N" THEN 1090
1110 IF A$ = "N" THEN 1000 ELSE PRINT "YES"
1120 GOSUB 12000
1130 OPEN "I",1,F$
1140 PRINT F$;" OPENED O.K."
1150 PRINT "READING IN DIRECTORY"
1160 FOR I = 1 TO N
1170 IF EOF(1) THEN 1300
1180 LINE INPUT 1,D$(I,NU)
1185 LINE INPUT 1,D$(I,NE)
1190 NEXT I
1200 CLOSE 1
1210 RETURN
1300 PRINT "PREMATURE END OF DIRECTORY"
1310 PRINT "AT ENTRY ";I
1320 FOR J = 1 TO 3000
1330 NEXT J
1340 CLOSE 1
1350 RETURN
2000 REM SAVE DIRECTORY
2010 CLS
2020 PRINT TAB(7)"DIRECTORY SAVE"
2030 PRINT TAB(7)" = = = = = "
2040 PRINT @ 5*32
2050 PRINT "ENTER FILENAME('ENTER' TO QUIT)"
2060 PRINT:PRINT TAB(5);
2070 LINE INPUT " ";F$
2080 IF F$ = "" THEN RETURN
2090 PRINT
2100 PRINT "IS ";F$;" CORRECT?";
2110 A$ = INKEY$
2120 IF A$ < "Y" AND A$ < "N" THEN 2110
2130 IF A$ = "N" THEN 2000 ELSE PRINT "YES"
2140 GOSUB 12000
2150 OPEN "O",1,F$
2160 PRINT F$;" OPENED O.K."
```

```

2170 PRINT"WRITING DIRECTORY"
2180 FOR I= 1 TO N
2190 PRINT 1,D$(I,NU)
2200 PRINT 1,D$(I,NE)
2210 NEXT I
2220 CLOSE 1
2230 RETURN
3000 REM ENTER TELEPHONE NUMBER
3010 CLS
3020 PRINT"TELEPHONE NUMBER ENTRY"
3030 PRINT"===== "
3040 PRINT @3*32
3050 PRINT"ENTER NAME('ENTER' TO QUIT)"
3060 PRINT:PRINT TAB(14);
3070 LINE INPUT " ";N$
3080 IF N$ = "" THEN RETURN
3090 REM GOT ENTRY NAME
3100 PRINT
3110 LINE INPUT "ENTER TEL. NO. ";T$
3120 PRINT
3130 PRINT"IS ABOVE CORRECT?";
3140 A$ = INKEY$
3150 IF A$ < "Y" AND A$ < "N" THEN 3140
3160 IF A$ = "N" THEN 3000 ELSE PRINT"YES"
3170 GOSUB 10000
3180 I = H
3190 IF D$(I,NE) = "" THEN 3250
3200 IF I < N THEN I = I + 1 ELSE I = 1
3210 IF I = H THEN 3300 ELSE GOTO 3190
3250 D$(I,NE) = N$
3260 D$(I,NU) = T$
3270 GOTO 3000
3300 PRINT"NO SPACE FOR ANY MORE ENTRIES"
3310 FOR I = 1 TO 3000
3320 NEXT I
3330 RETURN
4000 REM FIND A NUMBER
4010 CLS
4020 PRINT TAB(8)"ENTRY FINDER"
4030 PRINT TAB(8)"===== "

```



```

4040 PRINT @3*32
4050 PRINT TAB(4)"0) QUIT."
4060 PRINT
4070 PRINT TAB(4)"1) FIND ENTRY BY NAME."
4080 PRINT
4090 PRINT TAB(4)"2) FIND ENTRY BY NUMBER."
4100 PRINT
4110 PRINT"ENTER OPTION> ";
4120 A$=INKEY$
4130 IF A$<"0" OR A$>"2" THEN 4120
4140 IF A$="0" THEN RETURN
4150 IF A$="2" THEN 4500
4160 REM FIND BY NAME
4170 CLS
4175 PRINT TAB(8)"ENTRY FINDER"
4176 PRINT TAB(8)"===== "
4180 PRINT @3*32
4190 PRINT"ENTER NAME";
4200 INPUT N$
4210 PRINT"IS ";N$;" CORRECT?";
4220 A$=INKEY$
4230 IF A$>"N" AND A$<>"Y" THEN 4220
4240 IF A$="N" THEN 4180
4250 PRINT"YES"
4260 PRINT"SEARCHING"
4270 GOSUB 10000
4280 I=H
4290 IF INSTR(1,D$(I,NE),N$) THEN 4320 ELSE I=I+1
4300 IF I>N THEN I=1
4310 IF I=H THEN 4400 ELSE GOTO 4290
4320 PRINT @8*32,"ENTRY NO.";I
4325 PRINT
4330 PRINT TAB(3)"NUMBER=";D$(I,NU)
4340 PRINT TAB(5)"NAME=";D$(I,NE)
4350 PRINT @13*32,"DO YOU WISH TO CONTINUE?";
4360 A$=INKEY$
4370 IF A$="" THEN 4360
4375 IF A$="N" THEN 4000
4380 IF A$<>"Y" THEN 4360
4390 PRINT"YES":I=I+1:GOTO 4300
4400 PRINT"ALL ENTRIES SEARCHED"

```

```

4410 FOR I = 1 TO 2000
4420 NEXT I
4430 GOTO 4000
4500 REM FIND BY NUMBER
4510 CLS
4515 PRINT TAB(8)"ENTRY FINDER"
4516 PRINT TAB(8)" = = = = = "
4520 PRINT @3*32
4530 PRINT "ENTER NUMBER";
4540 INPUT T$
4550 PRINT "IS ";T$;" CORRECT?";
4560 A$ = INKEY$
4570 IF A$ = "" THEN 4560
4580 IF A$ = "N" THEN 4520
4590 IF A$ < > "Y" THEN 4560
4595 PRINT "YES"
4600 PRINT "SEARCHING"
4610 I = 1
4620 IF D$(I,NU) = T$ THEN 4700 ELSE I = I + 1
4630 IF I <= N THEN 4620
4640 PRINT "ENTRY NOT FOUND"
4650 FOR I = 1 TO 2000
4660 NEXT I
4670 GOTO 4000
4700 PRINT
4710 PRINT "ENTRY NO. ";I
4720 PRINT
4730 PRINT "NUMBER = ";D$(I,NU)
4740 PRINT
4750 PRINT TAB(2)"NAME = ";D$(I,NE)
4760 PRINT
4770 PRINT "TYPE SPACE TO CONTINUE"
4780 IF INKEY$ < > " " THEN 4780 ELSE 4000
5000 REM DELETE ENTRY
5010 CLS
5020 PRINT "DELETE ENTRY"
5030 PRINT " = = = = = "
5040 PRINT @3*32
5050 PRINT "TYPE ENTRY NUMBER";
5060 INPUT I
5070 IF I < 0 OR I > N THEN 5250

```

```

5075 PRINT
5080 PRINT TAB(3)"NUMBER = ";D$(I,NU)
5090 PRINT TAB(5)"NAME = ";D$(I,NE)
5095 PRINT
5100 PRINT"DO YOU WISH TO DELETE ABOVE?";
5110 A$=INKEY$
5120 IF A$<>"Y" AND A$<>"N" THEN 5110
5130 IF A$="N" THEN RETURN
5140 D$(I,NU)=" "
5150 D$(I,NE)=" "
5160 RETURN
5250 PRINT
5260 PRINT"INVALID ENTRY NUMBER"
5270 FOR I=1 TO 1000
5280 NEXT I
5290 GOTO 5000
10000 REM GENERATE HASHED ENTRY NUMBER FOR STRING IN
N$ IN RANGE 1 TO N
10010 H=0
10020 I=LEN(N$)
10030 IF I<0 OR MID$(N$,I,1)>"0" THEN 10050 ELSE I=I-1
10040 GOTO 10030
10050 C$="A"
10060 IF I<0 OR C$<"/" THEN 10090 ELSE C$=MID$(N$,I,1)
10070 IF C$>"@" THEN H=H+ASC(C$)-65
10080 I=I+1
10090 T=SQR((H*.592362284)2)
10100 T=T-INT(T)
10110 H=INT((N-2)*T)+1
10120 RETURN
12000 REM GET CASSETTE IN PLACE
12010 PRINT"PLACE CASSETTE IN TAPE DRIVE"
12020 PRINT"AND TYPE SPACE"
12030 IF INKEY$<>" " THEN 12030
12040 RETURN

```

## Program description

<i>Lines</i>	<i>Purpose</i>
10-20	Set maximum number of entries in the directory.



25	Reserve string space for directory entries.
30	Dimension string array of directory entries.
40-45	Set pointers to name entry (NE) and number entry (NU).
50-80	Initialise all directory entries to a null value.
200	Clear text screen.
210-400	Display main menu of possible options.
410-430	Prompt for and get valid option number from keyboard.
440	If option to end selected terminate program execution.
450	Call the relevant subroutine.
460	Re-display main menu etc.
1000	Directory load option, allows loading of previously entered directory from cassette tape.
1005-1020	Clear text screen and display headings.
1030	Move print position to 5 lines down the screen.
1040-1050	Display prompt for and get filename of directory to be loaded.
1060	If filename is a null string then quit.
1070-1110	Double check filename was correct.
1120	Get cassette ready in drive.
1130	Open directory source file.
1140-1150	Tell user file was opened and directory is now being read.
1160	Read all directory entries.
1170	Check if end of file, if so then not enough entries in file.
1180-1190	Read in entry and repeat.
1200-1210	Close file and return.
1300-1350	Display premature end of directory message and entry number at which it occurred. Wait for user to see message, close file and return.
2000	Directory save option, allows the saving of the current directory to cassette tape.
2010-2030	Clear text screen and display options.
2040	Move print position 5 line down screen.
2050-2070	Display prompt for filename that directory is to be saved to.
2080	If null string entered quit.
2090-2130	Double check filename was correct.
2140	Get cassette ready in drive.
2150	Open directory destination file.

2160-2170	Tell user file was opened and directory is now being written.
2180	Write all directory entries.
2190-2210	Write entry and repeat.
2220-2230	Close file and return.
3000	Telephone number entry option.
3010-3030	Clear text screen and display heading.
3040-3070	Prompt for and get name related to telephone number.
3080	If name is a null string then quit.
3100	Line feed.
3110	Prompt for and get telephone number.
3120	Line feed.
3130-3160	Double check entry is correct.
3170	Calculate ideal entry position number.
3180-3190	Get value into I and check if entry position is already occupied.
3200	Already occupied, point to next entry.
3210	If all entries tried display directory full message else check if this entry is free.
3250-3260	Set selected entry position to entered value.
3270	See if more entries are required.
3300-3330	Display entry space full message, wait for user to see it and return to main menu.
4000	Telephone number finder option.
4010-4100	Clear text screen and display menu of possible options.
4110-4130	Prompt for and get a valid option number from the keyboard.
4140	If quit option selected return to main menu.
4150	If option 2 selected jump to routine else must be option 1.
4160	Find entry by name.
4170-4190	Clear text screen, display title and prompt for name.
4200-4250	Get entry name and double check it.
4260	Display searching message.
4270	Calculate ideal entry position number.
4280	Put position in I.
4290-4300	Check if entered name is part of entry, if not point to next entry.
4310	If all entries searched display message else try again with new entry.
4320-4340	Display this entry.

4350-4370	Prompt for and get search continuation response from keyboard.
4375	If response "N" then check if another search is required.
4380	If response not "Y" then try again.
4390	Restart search of entries.
4400-4430	Display all entries searched message, wait for user to see it and check if another search is required.
4500	Find entry by number.
4510-4516	Clear text screen and display title.
4520-4595	Prompt, get entry number and double check.
4600	Tell user search taking place.
4610	Start with first entry.
4620-4630	Search for entry with that number.
4640-4670	Entry not found, wait for user to see message and check if another search is required.
4700-4780	Display entry, wait for response (space) and check if another search is required.
5000	Entry delete option.
5010-5030	Clear text screen and display heading.
5040 Print	position 3 lines down.
5050-5070	Get valid entry number.
5075-5095	Display entry.
5100-5130	Double check deletion is required.
5140-5160	Delete entry and return to main menu.
5250-5290	Display invalid entry number message, wait for user to see it and try again.
10000	This routine generates an entry number in the range 1 to N where N is the maximum number of entries, related to the value of string N\$.
10010	Set initial entry number to zero.
10020	I gets length of string.
10030-10040	Allow only characters bellow "0" to be evaluated.
10050	Make character "A".
10060	Extract character if required.
10070	Add value in to hashing code.
10080-10120	Compute number and return.



12000-12040 Display prompt and wait for cassette to be placed in tape drive.

## Variable definitions

<i>Variable name</i>	<i>Function</i>
A\$	Contains answer to a question.
D\$	String array containing all directory entries.
F\$	Directory filename.
N\$	String containing entry name.
T\$	String containing entry telephone number.
N	Maximum number of directory entries.
NE	Pointer to name entry in D\$.
NU	Pointer to telephone number entry in D\$.

## Chapter 4

# Utilities

## CONVERTOR

This program converts cassette tapes written on a Color Computer into Dragon format and vice versa. It should prove useful to owners of either machine, as it will give access to software which may not be available in the correct format. It should be noted that the program is only designed to work with files containing BASIC programs.

## Tokens

The need for conversion is brought about by the fact that BASIC on the Dragon uses different tokens to those found on the Color Computer. A token is a means of storing BASIC keywords (such as DIM, PRINT, REM etc.) and consists of a one byte code which is assigned to a particular keyword. For example, on the Dragon, the token which represents "PAINT" is the hexadecimal code B2 (178 decimal), whereas PAINT on the Color Computer is denoted by C3 (195). Tokens are used in order to reduce the program size and to make programs run faster.

A Color Computer tape will load into a Dragon, but listing it reveals a strange mixture of keywords. Some will be correct, as not all tokens have been re-assigned on the Dragon. The problem of conversion is simply that of scanning through the program, token by token, and correcting any which differ between the two machines. In order to do this, we must examine the way in which a BASIC program is stored.

## Program storage

The format of a single line of BASIC on either of the two machines is as follows:-

Memory location(s)	Contents
1-2	Two byte address of start of next line (the link address)
3-4	Two byte line number
5 +	The program line. Consists of text + tokens and is terminated by a zero byte

So, a typical one line program such as :-

10 PRINT"HELLO"

would appear in memory as:-

Memory location	Contents	Purpose
2601	26	High byte of link address
2602	0E	Low byte of link address
2603	00	High byte of line number
2604	0A	Low byte of line number
2605	87	The token for PRINT
2606	22	ASCII code for double quote
2607	48	"H"
2608	45	"E"
2609	4C	"L"
260A	4C	"L"
260B	4F	"O"
260C	22	ASCII code for double quote
260D	00	End of line marker
260E	00	Two bytes of zeroes which
260F	00	mark the end of the program

For a program of more than one line, the pair of zero bytes at the end of the above example would be altered to the link address of the next line. Thus, a link address of zero can be used to find the end of a program. The particular memory locations quoted above are actually for the Color Computer with disc fitted, and will differ for the Dragon. In any case, the exact part of memory used will depend upon how many pages of graphics are in use, etc. The program may be relocated in a different part of memory simply by changing the link address at the start of each line and copying the line to the new part of memory.



The start of the program text can be found by examining the contents of two memory locations. These are locations 0019 and 001A (hexadecimal) or 25 and 26 decimal. In the example quoted above, location 0019 would contain the hexadecimal value 26 and location 001A the value 01.

A program may be converted by scanning through it, skipping the link address and line number (these do not need converting), and looking for tokens. Tokens differ from text in always having a value greater than 80 hexadecimal (128 decimal), and thus may easily be recognised. Having found a token, it's value is looked up in a table for the source computer to see what the equivalent value is on the other machine.

## Using the tape convertor program

The Converter program asks what machine the source tape was recorded on. The user replies with a T (Tandy) or D (Dragon). The program next prompts for a filename and then searches for a program saved under that name. If it finds such a file then the data is loaded and the conversion process starts. A dot is written to the screen each time an end of line marker is detected during conversion. Note that the conversion process is fairly slow and it is probably a good idea to make a cup of tea or coffee whilst waiting. Tokens are converted using arrays for lookup tables and conversion stops when the pair of zero bytes denoting the end of the program are detected.

The user is then invited to enter a filename under which the converted program will be saved. The program will then be saved and the tape can be loaded using the normal CLOAD command.

## Program Listing

```

10 REM MUST SET ALL VARIABLES TO ZERO FIRST TO ALLOCATE
   SPACE
20 DIM DT(128),DF(128),TT(128),TF(128)
30 Z$="":N$="":C$="":I=0:J=0:EP=0:TB=0:RV=0:S=0:
   CD=0:FF=0:PL=58:U1=49:LA=0:RF=0:BC=0:TA=0:NF=0
40 EP=PEEK(&H001F)*256+PEEK(&H0020)
50 DEFUSR0=EP:DEFUSR1=EP+U1:DEFUSR2=PEEK(&HA00C)*
   256+PEEK(&HA00D):DEFUSR3=PEEK(&HA008)*256+PEEK
   (&HA009)

```

```
60 FOR I = EP TO EP + PL - 1
70 READ CD
80 POKE I, CD
90 NEXT I
100 CLS:PRINT"IS SOURCE TAPE IN TANDY (T)"
110 PRINT"OR DRAGON (D) FORMAT?"
120 PRINT"ENTER T OR D> ";
130 C$ = INKEY$:IF C$ <> "T" AND C$ <> "D" THEN 130
140 IF C$ = "T" THEN TA = 0:PRINT"TANDY" ELSE TA = 1:PRINT
"DRAGON"
150 GOSUB 4000 : REM SET UP CODE CONVERSION TABLES
160 PRINT"ENTER SOURCE FILE NAME"
170 GOSUB 1000 : REM GET FILENAME
180 PRINT"INSERT TAPE TO BE CONVERTED"
190 PRINT"PRESS PLAY KEY"
200 PRINT"TYPE < SPACE> WHEN READY"
210 IF INKEY$ <> " " THEN 210
220 TB = EP + PL : REM TAPE BUFFER AFTER MACHINE CODE
230 POKE &H007E, TB/256:POKE &H007F, TB-FIX(TB/256)*256
240 RV = USR0(0) : REM READ HEADER + FILE
250 IF PEEK(&H0081) <> 0 THEN 5000 : REM ERROR IF &H0081 <> 0
260 IF PEEK(&H007C) <> 255 THEN 220
270 I = 0:Z$ = ""
280 C$ = CHR$(PEEK(I + TB))
290 IF C$ > " " AND I < 8 THEN Z$ = Z$ + C$
300 I = I + 1
310 IF I < 8 AND C$ <> " " THEN 280
320 REM Z$ NOW HAS FILE HEADER
330 REM NOW COMPARE WITH NAME N$
340 PRINT"FOUND ";Z$
350 IF Z$ = N$ THEN 380
360 PRINT"SEARCHING"
370 GOTO 220 : REM GET NEXT BLOCK
380 TB = EP + PL + 255 : REM SKIP HEADER BLOCK
390 S = TB + 4 : REM POINT TO PROGRAM AFTER LINE NUMBER
400 LA = PEEK(TB)*256 + PEEK(TB + 1)
410 TB = TB + 3 : REM SKIP LINE NUMBER
420 TB = TB + 1 : REM START LOOKING FOR END OF FIRST LINE
430 IF PEEK(TB) <> 0 THEN 420 : REM ZERO DENOTES END OF LINE
440 TB = TB + 1 : REM SKIP ZERO BYTE
450 RF = TB - LA : REM RELOCATION FACTOR = THIS ADDRESS-
```



```

LOAD ADDRESS
460 TB = EP + PL + 255 : REM RESTORE TB
470 LA = LA + RF : REM DO FIRST LINE
480 POKE TB,LA/256
490 POKE TB + 1,LA-FIX(LA/256)*256
500 GOSUB 2000 : REM CONVERT CODES
510 PRINT:PRINT"ENTER DESTINATION FILE NAME"
520 GOSUB 1000 : REM GET FILE NAME
530 PRINT"INSERT TAPE FOR OUTPUT DATA"
540 PRINT"PRESS RECORD AND PLAY KEYS"
550 PRINT"TYPE < SPACE> WHEN READY"
560 IF INKEY$<>" " THEN 560
570 TB = EP + PL : REM RESET TAPE BUFFER POINTER
580 I=0
590 POKE (TB + I),ASC(MID$(N$,I + 1,1)) : REM POKE NEW
FILENAME INTO BUFFER
600 I=I + 1 : REM POINT TO NEXT BUFFER POSITION
610 IF I< LEN(N$) AND I< 8 THEN 590
620 IF I=8 THEN 640 : REM DONT SPACE PACK
630 FOR J=I TO 7 : POKE (TB + J),32 : NEXT J : REM SPACE PACK
BUFFER
640 POKE &H007E,TB/256
650 POKE &H007F,TB-FIX(TB/256)*256
660 S = S + 2
670 RV = USR2(0)
680 POKE&H007C,0
690 POKE &H007D,15
700 RV = USR3(0)
710 RV = USR2(0)
720 TB = TB + 255:POKE&H007E,TB/256:POKE&H007F,TB-FIX
(TB/256)*256
730 POKE &H007C,1
740 BC = S-TB + 1
750 IF BC> 255 THEN BC = 255
760 POKE &H007D,BC
770 RV = USR3(0)
780 TB = TB + BC:POKE&H007E,TB/256:POKE&H007F,TB-FIX
(TB/256)*256
790 IF BC = 255 THEN 740
800 POKE &H007C,255
810 POKE &H007D,0

```



```
820 RV = USR3(0)
830 RV = USR1(0) : REM TURN TAPE OFF
840 PRINT "MORE TAPES TO CONVERT?(Y/N)"
850 C$ = INKEY$
860 IF C$ = "N" THEN END
870 IF C$ < > "Y" THEN 850
880 PRINT "SAME FORMAT?(Y/N)"
890 C$ = INKEY$
900 IF C$ = "" THEN 890
910 IF C$ = "N" THEN 100 ELSE 160
1000 REM GET FILENAME
1010 PRINT "FILENAME> ";
1020 LINE INPUT N$
1030 PRINT "IS "; N$; " CORRECT?(Y/N)"
1040 C$ = INKEY$
1050 IF C$ = "" THEN 1040
1060 IF C$ < > "Y" THEN 1010
1070 RETURN
2000 REM CONVERSION ROUTINE
2010 CD = PEEK(S)
2020 IF CD = 0 AND PEEK(S + 1) = 0 AND PEEK(S + 2) = 0 THEN
RETURN
2030 IF CD = 0 THEN PRINT ".": GOSUB 6000: S = S + 5: GOTO 2010
2040 IF CD < 127 THEN S = S + 1: GOTO 2010
2050 IF CD = 255 THEN FF = 1: S = S + 1: GOTO 2010
2060 IF TA = 1 AND FF = 1 THEN CD = TF(CD-127): FF = 0: GOTO 2100
2070 IF TA = 1 AND FF = 0 THEN CD = TT(CD-127): GOTO 2100
2080 IF FF = 1 THEN CD = DF(CD-127): FF = 0: GOTO 2100
2090 CD = DT(CD-127)
2100 POKE S, CD : REM POKE IN CONVERTED CODE
2110 S = S + 1
2120 GOTO 2010
3000 REM MACHINE CODE ROUTINE TO READ TAPE
3010 REM LOOKS FOR HEADER AND THEN READS REST OF FILE
3020 REM RETURNS ON ERROR OR END OF FILE
3030 DATA &HAD,&H9F,&HA0,&H04
3040 DATA &HAD,&H9F,&HA0,&H06
3050 DATA &HB6,&H00,&H81
3060 DATA &H81,&H00
3070 DATA &H26,&H22
3080 DATA &HB6,&H00,&H7C
```

```

3090 DATA &H81,&H00
3100 DATA &H26,&HEE
3110 DATA &HFC,&H00,&H7E
3120 DATA &HC3,&H00,&HFF
3130 DATA &HFD,&H00,&H7E
3140 DATA &HAD,&H9F,&HA0,&H06
3150 DATA &HB6,&H00,&H81
3160 DATA &H81,&H00
3170 DATA &H26,&H07
3180 DATA &HB6,&H00,&H7C
3190 DATA &H81,&H01
3200 DATA &H27,&HE5
3210 DATA &HB6,&HFF,&H21
3220 DATA &H84,&HF7
3230 DATA &HB7,&HFF,&H21
3240 DATA &H39
4000 IF TA = 1 THEN 4170
4010 REM MUST BE DRAGON
4020 FOR I = 1 TO 128
4030 CD = I + 127
4040 IF CD > 127 AND CD < 142 THEN 4140
4050 IF CD > 141 AND CD < 151 THEN CD = CD + 1:GOTO 4140
4060 IF CD > 150 AND CD < 164 THEN CD = CD + 2:GOTO 4140
4070 IF CD > 166 AND CD < 181 THEN CD = CD + 24:GOTO 4140
4080 IF CD > 186 AND CD < 204 THEN CD = CD - 17:GOTO 4140
4090 IF CD > 163 AND CD < 167 THEN CD = CD + 23:GOTO 4140
4100 IF CD > 180 AND CD < 185 THEN CD = CD - 15:GOTO 4140
4110 IF CD = 185 THEN CD = 152:GOTO 4140
4120 IF CD = 186 THEN CD = 142:GOTO 4140
4130 IF CD = 204 THEN CD = 190
4140 DT(I) = CD
4150 NEXT I
4160 GOTO 4310
4170 FOR I = 1 TO 128
4180 CD = I + 127
4190 IF CD > 127 AND CD < 142 THEN 4290
4200 IF CD = 190 THEN CD = 204
4210 IF CD > 141 AND CD < 152 THEN CD = CD - 1:GOTO 4290
4220 IF CD > 152 AND CD < 166 THEN CD = CD - 2:GOTO 4290
4230 IF CD > 169 AND CD < 187 THEN CD = CD + 17:GOTO 4290
4240 IF CD > 190 AND CD < 205 THEN CD = CD - 24:GOTO 4290

```

*Hot programs to feed your Dragon*

```
4250 IF CD> 165 AND CD< 170 THEN CD = CD + 15:GOTO 4290
4260 IF CD> 186 AND CD< 191 THEN CD = CD-23:GOTO 4290
4270 IF CD = 152 THEN CD = 185:GOTO 4290
4280 IF CD = 142 THEN CD = 186:GOTO 4290
4290 TT(I) = CD
4300 NEXT I
4310 REM NOW SET UP FUNCTION ARRAY
4320 READ NF
4330 FOR I = NF + 1 TO 128
4340 DF(I) = I + 127:TF(I) = I + 127
4350 NEXT I
4360 FOR I = 1 TO NF
4370 READ DF(I)
4380 NEXT I
4390 FOR I = 1 TO NF
4400 READ TF(I)
4410 NEXT I
4420 RETURN
4430 DATA 34
4440 DATA 128,129,130,161,132
4450 DATA 136,140,141,142,143
4460 DATA 144,145,146,147,150
4470 DATA 151,152,153,154,155
4480 DATA 139,137,138,135,148
4490 DATA 134,131,133,149,156
4500 DATA 157,158,159,160
4510 DATA 128,129,130,154,132
4520 DATA 155,153,151,133,149
4530 DATA 150,148,134,135,136
4540 DATA 137,138,139,140,141
4550 DATA 152,156,142,143,144
4560 DATA 145,146,147,157,158
4570 DATA 159,160,161,131
5000 REM I/O ERROR
5010 PRINT"I/O ERROR.REWIND TAPE"
5020 PRINT"PRESS <SPACE> WHEN READY"
5030 IF INKEY$<" "> THEN 5030
5040 GOTO 220
6000 LA = PEEK(S + 1)*256 + PEEK(S + 2) : REM PICK UP LOAD
ADRESS
6010 LA = LA + RF : REM CALCULATE RELOCATED ADDRESS
```



```

6020 POKE (S + 1),LA/256
6030 POKE (S + 2),LA-FIX(LA/256)*256
6040 REM POKE RELOCATED ADDRESS IN
6050 RETURN

```

## Program description

<i>Lines</i>	<i>Purpose</i>
20	Dimension arrays which serve as lookup tables for tokens and functions.
30	Set all variables before they are used in the program. This is to ensure that the variables will not overwrite the machine code program and the tape buffer which both follow directly after the variable space
40	Find the end of the storage used by the Convertor program itself and assign it to the variable EP. This is given by the contents of locations 001F and 0020 (hexadecimal).
50	Define entry points of machine code routines. The routines used are:-
USR0, USR1 - included at end of program. See below	
USR2 - operating system subroutine which prepares the cassette for writing data	
USR3 - operating system routine which writes one block to tape	
60-90	Read in the machine code subroutines after the end of the storage used by "Convertor".
100-140	Find the type of source data to be converted. Set a flag "TA" to indicate Tandy (TA = 1), else TA = 0 and data is from a Dragon.
150	Use value of TA to set up conversion tables.
160-210	Get source file name.
220	Set start of tape buffer to first free byte after the end of the machine code routines.
230	Update the tape buffer pointer in memory (i.e. locations 007E and 007F).
240	Call machine code routine which looks for file header and loads rest of file into buffer.

### *Hot programs to feed your Dragon*

250	Location 0081 is an error code which should be zero if no tape read error.
260	Location 007C should be FF to indicate that the last block read was data.
270	Come here if data read correctly. The header block has the filename in it, so next form the filename in string Z\$.
280	Pick up name character by character from the tape buffer.
290	Check for end of file name.
300	Point to next byte in buffer.
310	Loop round if not end of name.
340	Print name found.
350	Same as name requested?
360	If not, keep looking.
370	By getting next file.
380	Found correct file, position TB pointer to data after the header block (i.e. the first byte of the program).
390	Set the program data pointer (S) to four bytes (i.e. link address + line number) after the start of the program.
400	Form the load address (LA) by looking for the first link address, pointed to by TB. This will be used to calculate the relocation factor (RF). RF is the amount by which all link addresses must be altered to take account of the fact that the program has been loaded into a different part of memory to that from which it was saved.
410-440	Search for the actual link address which is at the end of the first line.
450	Calculate the relocation factor.
460	Put TB back to the start of the program.
470	Relocate the link address of the first line.
480-490	Poke in the relocated link address.
500	Call code conversion routine.
510-560	Get filename for output data.
570	Reset pointer to start of program header block.
580-630	Update output file name, space pack to 8 chars.
640-650	Update the tape buffer pointer to point to start of header.
660	Move pointer S to position after the pair of zero bytes marking end of program.



670 Prepare cassette for writing.  
 680 Set block type to "header" (location 007C).  
 690 Set block length (location 007D) to 15 (length of header).  
 700 Write header block.  
 710 Prepare for next block.  
 720 Update tape buffer pointer to point to next data (i.e. the program itself).  
 730 Set block type to "data".  
 740 Calculate byte count (BC) as number of bytes left to be written to tape.  
 750 If more than 255 then set BC to 255 (can only write maximum of 255 bytes per block).  
 760 Set byte count to value of BC.  
 770 Write block.  
 780 Update pointer.  
 790 If BC less than 255 then must be last block.  
 800 If done last block then set block type to end of file (007C = 255).  
 810 Zero bytes in end of file block.  
 820 Write end of file block.  
 830 Turn tape motor off.  
 840-910 Check for more tapes to convert, loop if so.  
 1000-1070 Get and check file name.  
 2000 Token and function conversion routine. Converts until end of program marker detected.  
 2010 "S" is used to point to program bytes. "CD" is the value of the code found at each location.  
 2020 Check for end of line (CD = 0) and two bytes following to signal end of program.  
 2030 End of line if CD = 0. Print a dot, call re-location subroutine at 6000, advance S to next byte.  
 2040 If CD < 127 then this byte is text, not token.  
 2050 If CD = FF then the byte following the FF is a function code (e.g. SIN, COS etc), so set flag to indicate different lookup table needed, advance to next byte and go round again.  
 2060 Check for Tandy (TA = 1) function (FF = 1). If byte is a Tandy function convert via Tandy function array (TF), reset function flag, loop round.  
 2070 Check for Tandy token (i.e. TA = 1, FF = 0). If so then convert via Tandy token array (TT), loop round.



2080-2090	To get here,must be a Dragon function or token being converted.Hence use appropriate array depending upon whether FF = 1 or 0.
2100	Poke converted code back into memory at same memory location.
2110	Move to next byte.
2120	Keep looping until end of program at line 2020.
3000-3240	Machine code routine which reads blocks from tape until a header block is found (007C = 0).Having found a header,reads file until end of file detected (007C = FF) Exits on error (0081 = 00 if no error during block read). See listing below.
4000-4570	Check whether source is Tandy or Dragon,choose conversion data accordingly.Set up arrays of tokens and functions.
5000-5040	Come here if tape read error.
6000-6050	Calculate relocated link address,poke it in.

## Machine code listing

READ	JSR [\$A004]	;turn tape on,get into sync
SEARCH	JSR [\$A006]	;get block from tape
	LDA > \$0081	;check i/o result
	CMPA #0	;was it zero?
	BNE ERROR	;exit if non zero
	LDA > \$007C	;check block type
	CMPA #0	;zero = header block
	BNE SEARCH	;keep looking if not header
NBLOCK	LDD > \$007E	;get buffer pointer
	ADDD # \$FF	;increment by 256 bytes
	STD > \$007E	;update pointer
	JSR [\$A006]	;get block
	LDA > \$0081	;check i/o result
	BNE ERROR	;exit on error
	LDA > \$007C	;check block type
	CMPA #1	;1 = data block
	BEQ NBLOCK	;keep reading if still data
ERROR	LDA \$FF21	;get control register of PIA
	ANDA # \$F7	;make CA2 low to turn tape off

The significance of the STA \$FF21 update control register  
 RTS ;and leave

## Variable definitions

Variable name	Function
DT,DF	Dragon token and function arrays.
TT,TF	Tandy token and function arrays.
Z\$	File name read from header block.
N\$	Requested file name.
C\$	String used for replies from user.
EP	Pointer to first free byte after the end of the "Convertor" program and it's variables.
TB	Pointer to start of tape buffer.
RV	Dummy value for machine code calls.
S	Pointer to program bytes.
CD	Value of program byte read from memory.
FF	Flag to signify that a function code will be the next value to convert.
PL	Length of both machine code routines.
U1	Offset of second machine code routine from start of first.
LA	Link address.
RF	Relocation factor.
BC	Byte count when writing to tape.
TA	Source is Tandy if TA = 1.
NF	Number of entries in function table.

## Points of interest

Apart from the details of BASIC program storage, the feature of the Convert program worth noting is the by-passing of the normal tape routines. The routines shown can be adapted to circumvent the lack of any error recovery in the normal tape system. Normally a program will exit to command mode with an error message, but the above program can recover from and trap any errors.

## MONITOR

This elementary monitor will prove useful for trying out simple machine code programs. It can also be used to learn a great deal about the way BASIC programs are stored.

It is extremely simple and has only two commands:-

- M - Memory examine and change
- G - Go (i.e. execute a machine code program)

These are used as follows:-

*Memory* Use this command to display the contents of memory locations in hexadecimal notation. Upon entering the Monitor the prompt (a > sign) will appear at the left of the screen. Type "M" followed by a four digit hexadecimal number which corresponds to the location which is to be examined. For example, the following could be typed in:-

```
> M 0019
> 0019 26 & (press < ENTER > )
> 1A 1 . (press < ENTER > again)
> 1B 2A * (press the full stop key)
>
```

The numbers 0019, 1A and 1B are memory locations and 26, 1 and 2A their contents. After the contents of the location have been printed as a hexadecimal number, the Monitor also prints the contents as a character (such as & or \*). The character printed will be the ASCII character for the given code. Non printable codes such as those below 32 (space) and those above 127 (graphics) are printed as a full stop sign.

After the contents of memory have been printed, the user can move to the higher memory location and print its contents, simply by pressing the < ENTER > key.

If the memory is to be changed then a two digit hexadecimal number should be entered before pressing the < ENTER > key. If the memory location does not alter to the correct value (for example if it is attempted to write to ROM) then an error message "MEMORY UNCHANGED" will appear.



The significance of the above example is that the locations 0019 and 001A (hex) are a pointer to the start of the BASIC program. To examine the program, write down the contents of locations 0019 and 001A with the contents of 0019 first. In the case of the numbers quoted above, the result would be:-

Location:	0019	001A	
Contents:	26	01	= 2601 (hex)

Now, to look at the BASIC program (actually "Monitor" itself), type:-

```
> M 2601
> 2601 26 & < ENTER>
> 2602 7 . < ENTER>
> 2603 0 . < ENTER>
> 2604 A . < ENTER>
> 2605 9E & < ENTER> and so on.
```

If you have read the description of "Convertor" you will recognise the above as the start of a program line. Locations 2601 and 2602 are the link address, locations 2603 and 2604 are the first line number (10) and so on.

*Go* This command allows a previously entered machine code program to be executed.

To try it out, type the following program into memory at location 7000 (hex) as described above:-

```
7000 86
7001 41
7002 AD
7003 9F
7004 A0
7005 02
7006 7E
7007 70
7008 00
```

Now type G7000 and you should see the screen fill with "A"s. This is because the above sequence of bytes corresponds to the following machine code program:-

7000	LDA #\$41	;41 hex = ASCII code for letter "A"
7002	JSR [\$A002]	;jump to subroutine pointed to by contents of A002 and A003
7006	JMP 7000	;back to the start

Unfortunately, the only way of stopping this program is to switch off, or press RESET.

The Monitor could be used in conjunction with one of the books on machine code programming on the Dragon as a way of exploring the machine further.

## Program listing

```
10 CLS
20 PRINT"MONITOR V 1.0"
30 PRINT
40 PRINT"> ";
50 C$=INKEY$:IF C$="" THEN GOSUB 3000:GOTO 50
60 PRINT C$;" ";
70 IF C$="M" THEN 110
80 IF C$="G" THEN 280
90 PRINT"? "
100 GOTO 40
110 GOSUB 2000
120 IF EF=1 THEN PRINT"ERROR IN HEX INPUT":GOTO 40
130 PRINT"> ";H$;" ";
140 H$="&H"+H$
150 PRINT TAB(6);HEX$(PEEK(VAL(H$)));" ";
160 Q$=CHR$(PEEK(VAL(H$)))
170 IF ASC(Q$)>31 AND ASC(Q$)<127 THEN PRINT TAB(10);Q$;" "
    ;;ELSE PRINT TAB(10);"." ";
180 GOSUB 1000
190 IF EF=0 THEN 240
200 IF C$="." THEN 30
210 IF C$<>CHR$(13) THEN 120
```

```

220 H$ = HEX$(VAL(H$) + 1)
230 GOTO 130
240 POKE VAL(H$), VAL("&H" + Z$)
250 IF PEEK(VAL(H$)) <> VAL("&H" + Z$) THEN PRINT " MEMORY
UNCHANGED":GOTO 40
260 PRINT
270 GOTO 220
280 GOSUB 2000
290 IF EF = 1 THEN 120
300 H$ = "&H" + H$
310 DEFUSR0 = VAL(H$)
320 RV = USR0(0)
330 PRINT RV
340 GOTO 40
1000 REM GET TWO DIGIT HEX NO.
1010 REM SET EF TO 1 ON ERROR
1020 Z$ = "":I = 1
1030 C$ = INKEY$:IF C$ = "" THEN GOSUB 3000:GOTO 1030
1040 EF = 0
1050 IF ASC(C$) < 48 OR ASC(C$) > 70 THEN EF = 1
1060 IF ASC(C$) > 57 AND ASC(C$) < 65 THEN EF = 1
1070 Z$ = Z$ + C$:PRINT C$;
1080 I = I + 1
1090 IF I > 2 OR EF = 1 THEN RETURN
1100 GOTO 1030
2000 REM GET FOUR DIGIT HEX NO.
2010 H$ = ""
2020 GOSUB 1000:H$ = H$ + Z$
2030 IF EF = 0 THEN GOSUB 1000
2040 H$ = H$ + Z$
2050 PRINT
2060 RETURN:REM WITH EF SET TO ERROR STATUS
3000 REM FLASH CURSOR
3010 PRINT CHR$(175);
3020 FOR J = 1 TO 50 : NEXT J
3030 PRINT CHR$(8);
3040 RETURN

```

## Program description

Lines	Purpose
10-40	Clear screen, sign on, put out prompt sign.
50	Wait for key to be pressed, flash cursor.



## Hot programs to feed your Dragon

60	Print key pressed.
70-100	Check for "M" or "G", error if not. If ok then go to correct routine.
110	Come here on "M". Go and get four hex digits as address.
120	Check for error in entry.
130	Print entry.
140	Make it into a hex number string.
150	Print current contents as hex number.
160	Pick up contents as a character.
170	Print character if in range 32 to 127.
180	Get either a new value, < ENTER > or full stop.
190	If no error then poke new value in.
200-210	Check if error caused by full stop (for termination) being typed, or by < ENTER > for next location.
220	Step to next location.
240	Poke new value into memory.
250	Check to see if memory updated.
260-270	Go round again.
280	Get 4 digit address for "GO".
290	Complain if not valid.
300	Form hex string.
310	Define the appropriate USR routine.
320	And call it.
330	Print any return value.
340	And look for next command.
1000-1100	Get two digit hex number, checking for errors in input.
2000-2060	Use above routine twice to get a four digit number.
3000-3040	Put a block cursor at current print position and then remove it by backspacing over it.

## Variable definitions

Variable name	Function
C\$	String used for command entry.
H\$	String used to represent current memory location.
Z\$	String returned from "get two hex digit" routine.
EF	Error flag set by input routines to indicate invalid hex entry.

I 140 IF S<>1 Used to count number of digits entered.  
 RV Dummy return value in USR call.

## COPY

"Copy" is a file to file copy program. It allows text files to be copied from disk to disk, disk to tape and tape to disk.

When the program is run, a menu showing two possible options is displayed. The first stage is to set the device assignment. On entering this option a list of valid device numbers is displayed with the relevant device names. The user is then prompted for the source device, this is the device from which the file is to be copied. After entering the source device the user is prompted for the destination device. That is the device to copy the file to.

The copy process can now be started by entering option (2). The program then prompts the user for the source and then the destination filenames. Once these are entered the file copy commences. For each line of text copied a "." (full stop) is displayed on the screen. The user is informed when the copy is finished and the program returns to the main menu after a short delay.

## Program listing

```

10 REM SIMPLE TEXT FILE TO FILE COPY PROGRAM
15 D=1:S=1
20 CLS
30 PRINT TAB(7)"FILE TO FILE COPY"
40 PRINT TAB(7)"===== "
50 PRINT
60 PRINT
70 PRINT
80 PRINT TAB(5)"0) QUIT."
90 PRINT
100 PRINT TAB(5)"1) SET DEVICE ASSIGNMENT."
110 PRINT
120 PRINT TAB(5)"2) COPY FILES."
130 PRINT
140 PRINT
150 PRINT TAB(5)"please enter option> ";
```

*Hot programs to feed your Dragon*

```
160 A$ = INKEY$
170 IF A$ < "0" OR A$ > "2" THEN 160
180 IF A$ = "1" THEN 1000
190 IF A$ = "2" THEN 2000
200 END
1000 REM SET UP FILE SOURCE DEVICES
1010 CLS
1020 PRINT TAB(5) "DATA DEVICE ASSIGNMENTS"
1030 PRINT TAB(5) "===== "
1040 PRINT
1050 PRINT "VALID DEVICE NUMBERS ARE:-"
1060 PRINT
1070 PRINT TAB(7) "1: DISK DRIVE."
1080 PRINT TAB(6) "-1: TAPE DRIVE."
1090 PRINT TAB(6) "-2: PRINTER."
1100 PRINT TAB(7) "0: CONSOLE."
1110 PRINT
1120 INPUT "ENTER SOURCE"; S
1130 INPUT "ENTER DESTINATION"; D
1140 IF S = 1 OR S = -1 OR S = 0 THEN 1200
1150 PRINT "invalid assignment"
1160 FOR I = 1 TO 2000
1170 NEXT I
1180 GOTO 1000
1200 IF D = -1 AND S = -1 THEN 1150
1210 GOTO 20
2000 REM OPEN DATA PATHS AND COPY
2010 CLS
2020 PRINT TAB(7) "DATA PATH OPENER"
2030 PRINT TAB(7) "===== "
2040 PRINT
2050 PRINT
2060 IF S = 0 THEN 2200
2070 PRINT "ENTER SOURCE FILENAME"
2080 LINE INPUT ">"; S$
2090 PRINT "IS "; S$; " CORRECT?";
2100 A$ = INKEY$
2110 IF A$ = "" THEN 2100
2120 IF A$ < > "Y" AND A$ < > "N" THEN 2100 ELSE PRINT A$
2130 IF A$ = "N" THEN 2000
```



```

2140 IF S<>-1 THEN 2170
2150 PRINT"TYPE SPACE WHEN TAPE READY"
2160 IF INKEY$<>" " THEN 2160
2170 OPEN "I",#S,S$
2180 PRINT S$" OPENED."
2190 GOTO 2220
2200 OPEN "I",#S,"D"
2210 PRINT"SOURCE OPENED."
2220 IF D=0 OR D=-2 THEN 2340
2230 PRINT"ENTER DESTINATION FILENAME"
2240 LINE INPUT">";D$
2250 PRINT"IS ";D$;" CORRECT?";
2260 A$=INKEY$
2270 IF A$="" THEN 2260
2280 IF A$<"N" AND A$<"Y" THEN 2260 ELSE PRINT A$
2285 IF A$="N" THEN 2230
2290 IF D<>-1 THEN 2310
2300 PRINT"HIT SPACE WHEN TAPE READY"
2305 IF INKEY$<>" " THEN 2315
2310 IF S=1 AND D=1 THEN D=2
2315 OPEN "O",#D,D$
2320 PRINT D$;" OPENED."
2330 GOTO 2360
2340 OPEN "O",#D,"D"
2350 PRINT"DESTINATION OPENED."
2360 PRINT"STARTING COPY ";
2370 IF EOF(S) THEN 2500
2380 LINE INPUT #S,L$
2390 PRINT #D,L$
2400 PRINT".";
2410 GOTO 2370
2500 CLOSE #D
2510 CLOSE #S
2515 PRINT
2520 PRINT"COPY COMPLETED."
2530 FOR I=1 TO 2000
2540 NEXT I
2560 GOTO 20

```

### Program description

Lines	Purpose
10-15	Set up default source and destination device assignments.

20	Clear text screen.
30-140	Display menu of all possible options.
150-170	Get valid option.
180-190	Execute selected option.
200	Finish program option must have been selected, terminate execution.
1000	Sets up requested source and destination file device assignments.
1010	Clear text screen.
1020-1110	Display all possible device options.
1120	Get required source device number.
1130	Get required destination device number.
1140	Validate source device option.
1150-1180	Invalid device selected, display message, wait and return to start of option select sequence.
1200-1210	Validate destination device option, return to main menu if valid else re-start option select sequence.
2000	Opens files to selected source and destination file devices.
2010	Clear text screen.
2020-2050	Display headings.
2060	If source is console.
2070-2190	Get source filename and open file.
2200-2210	Open source file as console.
2220	Check if destination device requires filename.
2230-2330	Get destination device filename and open file.
2340-2350	Open destination with no file name.
2360	Display copy started message.
2370	Check if end of source file.
2380-2390	If not get a line from the source file and send it to the destination file.
2400	Display "full stop" on the console to indicate one line has been copied.
2410	Repeat untill end of source file.
2500-2510	Close both source and destination files.
2515-2560	Display completion message, wait and return to main menu.

## **Variable definitions**

<i>Variable name</i>	<i>Function</i>
A\$	String which holds reply to questions.



D\$	String which holds destination filename.
S\$	String which holds source filename.
L\$	String which contains line of text read from source file.
D	Destination file device number.
S	Source file device number.

## APPEND

One obvious omission from the standard operating system found on the Dragon and the Color Computer is some means of merging tape files.

It would be convenient to have commonly used subroutines saved on tape as a library. To be able to append these routines to a main program would obviously save a great deal of typing.

The program listed below allows files containing sub-programs to be merged with a main program. Both main and subprograms must have been saved on tape already, and there must be no overlap of line numbers between files to be concatenated.

The program will ask for the name of the first sub-file to be read in. This will be searched for and, if found, loaded into memory. The program will be relocated as described in the Converter program. A dot will appear on the screen as each line is relocated.

When this process is finished the user will be asked for the name of the destination file to which the composite program will be written. The tape to receive the final program should be inserted, and the tape drive set to record mode.

Having written the first file to tape the user will be asked to supply another file name containing the first data to be concatenated. The destination tape will have to be removed, making sure the position on the tape is not disturbed. The tape with the next source file should now be inserted. This will be read in and the process described above repeated until the user replies "N" to the question "MORE TAPES TO APPEND?".

Having written all the sub-programs to tape, the complete program may be loaded via the normal "CLOAD" command.

## Program Listing

```

10 REM MUST SET ALL VARIABLES TO ZERO FIRST TO ALLOCATE
   SPACE
20 Z$ = "":N$ = "":C$ = "":I = 0:J = 0:SN = 1:EP = 0:TB = 0:RV = 0:
   S = 0:D = 0:CD = 0:OF = 0:BS = 0:PL = 58:U1 = 49:LA = 0:RF = 0:

```



```
BC=0
30 EP = PEEK(&H001F)*256 + PEEK(&H0020)
40 DEFUSR0 = EP:DEFUSR1 = EP + U1:DEFUSR2 = PEEK(&HA00C)
  *256 + PEEK(&HA00D):DEFUSR3 = PEEK(&HA008)*256 +
  PEEK(&HA009)
50 FOR I = EP TO EP + PL-1
60 READ CD
70 POKE I,CD
80 NEXT I
90 PRINT"ENTER SOURCE FILE NUMBER";SN
100 GOSUB 1000 : REM GET FILENAME
110 PRINT"INSERT TAPE WITH SOURCE FILE"
120 PRINT"PRESS PLAY KEY"
130 PRINT"TYPE < SPACE> WHEN READY"
140 IF INKEY$<" "> THEN 140
150 TB = EP + PL : REM TAPE BUFFER AFTER MACHINE CODE
160 POKE &H007E,TB/256:POKE&H007F,TB-FIX(TB/256)*256
170 RV = USR0(0) : REM READ HEADER + FILE
180 IF PEEK(&H0081)<" "> 0 THEN 5000:REM ERROR IF &H0081<" "> 0
190 IF PEEK(&H007C)<" "> 255 THEN 150
200 I = 0:Z$ = ""
210 C$ = CHR$(PEEK(I + TB))
220 IF C$>" " AND I<8 THEN Z$ = Z$ + C$
230 I = I + 1
240 IF I<8 AND C$<" "> THEN 210
250 REM Z$ NOW HAS FILE HEADER
260 REM NOW COMPARE WITH NAME N$
270 PRINT"FOUND ";Z$
280 IF Z$ = N$ THEN 310
290 PRINT"SEARCHING"
300 GOTO 150 : REM GET NEXT BLOCK
310 TB = EP + PL + 255 : REM SKIP HEADER BLOCK
320 S = TB + 4 : REM POINT TO PROGRAM AFTER LINE NUMBER
330 LA = PEEK(TB)*256 + PEEK(TB + 1)
340 TB = TB + 3 : REM SKIP LINE NUMBER
350 TB = TB + 1 : REM START LOOKING FOR END OF FIRST LINE
360 IF PEEK(TB)<" "> 0 THEN 350 : REM ZERO DENOTES END OF LINE
370 TB = TB + 1 : REM SKIP ZERO BYTE
380 R = TB-LA + OF : REM RELOCATION FACTOR =
  THIS ADDRESS-LOAD ADDRESS + OFFSET
390 TB = EP + PL + 255 : REM RESTORE TB
```

```

400 LA = LA + RF : REM DO FIRST LINE
410 POKE TB,LA/256
420 POKE TB + 1,LA-FIX(LA/256)*256
430 GOSUB 2000 : REM RELOCATE PROGRAM
440 IF SN> 1 THEN 470
450 PRINT:PRINT"ENTER DESTINATION FILE NAME"
460 GOSUB 1000 : REM GET FILE NAME
470 PRINT"INSERT TAPE FOR OUTPUT DATA"
480 PRINT"PRESS RECORD AND PLAY KEYS"
490 PRINT"TYPE < SPACE> WHEN READY"
500 IF INKEY$<" " THEN 500
510 TB = EP + PL : REM RESET TAPE BUFFER POINTER
520 IF SN> 1 THEN 650
530 I = 0
540 POKE (TB + I),ASC(MID$(N$,I + 1,1)) : REM POKE NEW
FILENAME INTO BUFFER
550 I = I + 1 : REM POINT TO NEXT BUFFER POSITION
560 IF I < LEN(N$) AND I < 8 THEN 540
570 IF I = 8 THEN 590 : REM DONT SPACE PACK
580 FOR J = I TO 7 : POKE (TB + J),32 : NEXT J : REM SPACE PACK
BUFFER
590 POKE &H007E,TB/256
600 POKE &H007F,TB-FIX(TB/256)*256
610 RV = USR2(0)
620 POKE&H007C,0
630 POKE &H007D,15
640 RV = USR3(0)
650 RV = USR2(0)
660 TB = TB + 255:POKE&H007E,TB/256:POKE&H007F,TB-FIX
(TB/256)*256
670 POKE &H007C,1
680 BC = S-TB + 1:OF = BC
690 IF BC> 255 THEN BC = 255
700 POKE &H007D,BC
710 RV = USR3(0)
720 TB = TB + BC:POKE&H007E,TB/256:POKE&H007F,TB-FIX
(TB/256)*256
730 IF BC = 255 THEN 680
740 RV = USR1(0) : REM TAPE OFF
750 PRINT"MORE FILES TO APPEND?(Y/N)"
760 C$ = INKEY$

```



*Hot programs to feed your Dragon*

```
770 IF C$ = "" THEN 760
780 IF C$ = "Y" THEN SN = SN + 1:GOTO 90
790 POKE &H007E,TB/256:POKE&H007F,TB-FIX(TB/256)*256:
POKE TB,0:POKE TB + 1,0
800 POKE &H007C,1
810 POKE &H007D,2
820 RV = USR2(0)
830 RV = USR3(0)
840 POKE &H007C,255
850 POKE &H007D,0
860 RV = USR2(0)
870 RV = USR3(0)
880 RV = USR1(0)
890 END
1000 REM GET FILENAME
1010 PRINT"FILENAME> ";
1020 LINE INPUT N$
1030 PRINT"IS ";N$;" CORRECT?(Y/N)"
1040 C$ = INKEY$
1050 IF C$ = "" THEN 1040
1060 IF C$ < > "Y" THEN 1010
1070 RETURN
2000 REM RELOCATE PROGRAM
2010 REM SEARCHES FOR LINK ADDRESS AT START OF EACH
LINE
2020 CD = PEEK(S)
2030 IF CD = 0 AND PEEK(S + 1) = 0 AND PEEK(S + 2) = 0 THEN
RETURN
2040 IF CD = 0 THEN PRINT". ";:GOSUB 6000:S = S + 5:GOTO 2020
2050 S = S + 1
2060 GOTO 2020
3000 REM MACHINE CODE ROUTINE TO READ TAPE
3010 REM LOOKS FOR HEADER AND THEN READS REST OF FILE
3020 REM RETURNS ON ERROR OR END OF FILE
3030 DATA &HAD,&H9F,&HA0,&H04
3040 DATA &HAD,&H9F,&HA0,&H06
3050 DATA &HB6,&H00,&H81
3060 DATA &H81,&H00
3070 DATA &H26,&H22
3080 DATA &HB6,&H00,&H7C
3090 DATA &H81,&H00
```



```

3100 DATA &H26,&HEE
3110 DATA &HFC,&H00,&H7E
3120 DATA &HC3,&H00,&HFF
3130 DATA &HFD,&H00,&H7E
3140 DATA &HAD,&H9F,&HA0,&H06
3150 DATA &HB6,&H00,&H81
3160 DATA &H81,&H00
3170 DATA &H26,&H07
3180 DATA &HB6,&H00,&H7C
3190 DATA &H81,&H01
3200 DATA &H27,&HE5
3210 DATA &HB6,&HFF,&H21
3220 DATA &H84,&HF7
3230 DATA &HB7,&HFF,&H21
3240 DATA &H39
5000 REM I/O ERROR
5010 PRINT"I/O ERROR.REWIND TAPE"
5020 PRINT"PRESS < SPACE> WHEN READY"
5030 IF INKEY<" "> THEN 5030
5040 GOTO 150
6000 LA = PEEK(S + 1)*256 + PEEK(S + 2) : REM PICK UP LOAD
      ADRESS
6010 LA = LA + RF : REM CALCULATE RELOCATED ADDRESS
6020 POKE (S + 1),LA/256
6030 POKE (S + 2),LA-FIX(LA/256)*256
6040 REM POKE RELOCATED ADRESS IN
6050 RETURN

```

## Program description

Note: A large part of this program is identical to the program "Convertor". The latter was described in some detail above, and this will not be repeated here. Only those parts which differ will be described.

Lines	Purpose
10-80	See "Convert".
90	Ask for source files, starting with number 1 (SN = 1 initially).
100-420	See "Convert".

430	IF C\$ = ""	Call routine which relocates program. This will ensure that the concatenated program has the correct link addresses.
440	IF SN = 1	If not the first file (SN = 1) then don't ask for the destination file.
450-460		Get file name for output.
470-510		Set things up for writing.
520		Only write the file header if this is the first file.
530-640		Write header to tape.
650-740		Write remainder of file to tape.
750-780		Check if more files to append. If so then go back, get next filename and repeat the same process. If last file then must close the tape file by writing an end of file marker.
790-890		Close file and exit.

## Variable definitions

As for "Convert" except:-

Name	Function
SN	Counter of files appended. Initially set to 1, for first file.

## CATALOG

How many times have you wished you could find out what was on a tape without having to keep typing "SKIPF"? Even if you can bring yourself to do this, invariably the names gets forgotten as they are wiped off the screen with each new "SKIPF" command.

CATALOG can help. Run it and it will list all files on a tape, along with other information as to file type and load addresses.

## Program listing

```
10 REM MUST SET ALL VARIABLES TO ZERO FIRST TO ALLOCATE SPACE
20 C$ = "":Z$ = "":I = 0:EP = 0:TB = 0:RV = 0:S = 0:CD = 0:PL = 31
```

```

30 EP = PEEK(&H001F)*256 + PEEK(&H0020)
40 DEFUSR0 = EP
50 FOR I = EP TO EP + PL - 1
60 READ CD
70 POKE I, CD
80 NEXT I
150 TB = EP + PL : REM TAPE BUFFER STARTS AFTER MACHINE
CODE
160 POKE &H007E, TB/256 : POKE &H007F, TB-FIX(TB/256)*256
170 RV = USR0(0) : REM READ HEADER + FILE
180 IF PEEK(&H0081) < > 0 THEN 5000 : REM ERROR IF &H0081 < > 0
190 Z$ = "" : PRINT
200 FOR I = TB TO TB + 7
210 Z$ = Z$ + CHR$(PEEK(I))
220 NEXT I
230 PRINT Z$;
240 FOR I = TB + 8 TO TB + 14
250 PRINT HEX$(PEEK(I));
260 NEXT I
270 GOTO 150
3000 REM MACHINE CODE ROUTINE TO READ IN A FILE
3010 REM LOOKS FOR A HEADER AND LOADS FILE AFTER
HEADER
3020 REM DOES NOT CHECK FILE NAME
3030 REM EXITS ON ERROR OR END OF FILE
3040 DATA &HAD, &H9F, &HA0, &H04
3050 DATA &HAD, &H9F, &HA0, &H06
3060 DATA &HB6, &H00, &H81
3070 DATA &H81, &H00
3080 DATA &H26, &H07
3090 DATA &HB6, &H00, &H7C
3100 DATA &H81, &H00
3110 DATA &H26, &HEE
3120 DATA &HB6, &HFF, &H21
3130 DATA &H84, &HF7
3140 DATA &HB7, &HFF, &H21
3150 DATA &H39
5000 REM I/O ERROR
5010 PRINT:PRINT "I/O ERROR. REWIND TAPE"
5020 PRINT "PRESS < SPACE> WHEN READY"
5030 IF INKEY$ < > " " THEN 5030

```



## **Program description**

Most of this program has been described above. It uses the same machine code routines to read a block from tape, but this time only the header is of interest.

The file name is printed at line 230, character by character. It is read from the tape buffer as in the "Convertor" program. The remaining information (7 bytes) after the name is printed in lines 240 to 260. These 7 bytes comprise three bytes defining file type, two bytes which seem to be a load address and two bytes which may be the relocation factor. The authors have not yet found the exact nature of these bytes.

## **Variable definitions**

As for "Convert" and "Append".

## **Software on cassette**

For anyone who wishes to save the tedium of typing in the programs presented in this book, cassettes are available with these, and a few other, programs. Write for details to:-

RHAMCODE SOFTWARE,  
P.O. BOX 8,  
CARNFORTH,  
LANCS..

Don't forget an S.A.E to help us get details back to you as quickly as possible.

## **CATALOG**

How many times have you wished you could find out what programs you have on your cassette without having to keep typing "SKIPF"? Even if you have a good memory, it is inevitable that names get forgotten. The good news is that you can now use the "SKIPF" command.

CATALOG can help. Run it and it will list all the programs on your cassette. It will also give you the address of each program, so you can find it quickly.

## **Program listing**

10 REM MUST PRINT TO TAPE  
20 PRINT "PRESS <SPACE> WHEN READY"  
30 PRINT "PRESS <SPACE> WHEN READY"  
40 PRINT "PRESS <SPACE> WHEN READY"









## About This Book:-

This book contains, not only the best programs for your Dragon (or Tandy Color Computer) but also a full description of how the programs work, how you can change them, and how you can produce your own programs. The book divides naturally into four sections:-

*ADVANCED GAMES:* These include *3D-Sub Hunt* (complete with Radar screen and Sonar) and arcade-style graphics games.

*SOUND AND GRAPHICS:* *Drawpack* — line drawing with joysticks, *Logo* — a comprehensive implementation of this new language (A copy of *Logo* would cost far more than the price of this book). *Logo* includes Turtle Graphics — a new system of geometry that is much easier for children than Cartesian co-ordinates.

*BUSINESS:* For the business user of the Dragon 32, here is a full *Word Processor*, a *Telephone Call Costing System* and an *Information Retrieval System*. Each of these will turn your Dragon into a useful business tool.

*UTILITIES:* To start with, there is a utility to convert Tandy tapes to Dragon format and vice versa, so you can take full advantage of all available Tandy Color Computer software. Also, there are programs to help with machine code usage, tape file handling and for appending one program to another. These are all *extensions* to the basic Dragon.

We publish many other books for the Dragon and most popular micros. Write today for a catalogue to:-

Sigma Technical Press  
5 Alton Road  
Wilmslow  
Cheshire  
SK9 5DY

ISBN: 0 905104 51 X