Issue Number 5 — November 1983

STOP PRESS DRAGON

TELEPHONE COMPETITION

As mentioned in the editorial the 'mimic a telephone' competition proved to be very popular. After considerable help from a GPO expert we awarded two prizes, one to KEVIN PERRY (aged 11) of Gosport for a most realistic trimphone sound, and one to IAN JONES (aged 8) from Sully for a very good imitation of the sound of a telephone ringing through the earpiece.

Congratulations to you all and the two winners in particular.

```
 1 REM TELEPHONE BY IAN JONES
20 FOR I=1 TO 15
30 PLAY"T250;12E#"
40 NEXT I
50 FOR I=1 TO 60
60 NEXT I
70 FOR D=1 TO 30
75 NEXT D
80 FOR I=1 TO 15
90 PLAY "T200;12E#"
100 NEXT I
130 FOR I=1 TO 1000
140 NEXT I
150 GOTO 20
```

```
 1 REM TELEPHONE BY KEVIN PERRY
10 A$="O5L64DO4L64A#"
20 B$=A$+A$=A$+A$
30 C$=B$+"P3P4P4"
40 PLAY B$
50 PLAY"P20"
60 PLAY B$
70 PLAY"P3P4P4"
80 GOTO 40
```

EDITORIAL

Welcome once again to STOP PRESS. After four issues many of you know what to expect inside. Newcomers however are invited to read on for new programs and articles and other program oriented material.

It is now one year since Dragon Data became an independent company. That year has seen the expansion of high-quality software, the marketing of disk-drives, and the continued development of new machines. Dragon 64 will be on general release shortly following its recent arrival on the US market. All that in one year! To celebrate the occasion Dragon Data have a special software offer - see elsewhere for details.

An apology is in order for the lack of time allowed for the last competition. But a big thank-you to many enthusiastic young readers (aged seven upwards) who sent entries for the telephone competition. Overwhelmed by such a response we called in a GPO engineer to help adjudicate! Congratulations to you all.

A number of letters are received after each issue from readers who have been unable to get a program to work. As is well recognised, type-setting and proof-reading of programs is very demanding. We make every effort to eradicate all mistakes but inevitably human error creeps in. We do stress however that there are a number of situations where it is easy for the reader to make mistakes. Be careful to distinguish between upper case i and the number one, and also between inverted commas surrounding a space and ones surrounding nothing!

The format of STOP PRESS is now under review and we hope plans for its continuation (and indeed expansion) will emerge in the near future. Readers' comments are welcomed - remember that STOP PRESS is produced for you!

This issue sees the first in a series of articles designed to introduce the reader to Dragon software, with a review of the editor-assembler cassette DREAM. Machine-Code Corner takes a look at producing an index with a more detailed look at the MC program next issue. For our young users the DRAW command is explored.

Plans are afoot to link Dragon owners via Micronet 800 to PRESTEL. A bank of 150 programs will be available for instant retrieval and readers are invited to submit programs for inclusion. As soon as possible.

Please address all correspondence to Miss Cathy Hyde, Dragon Data Ltd, Kenfig Industrial Estate, Margam, Port Talbot, SA13 2PE

MACHINE CODE CORNER

The animation of graphics using machine code can be spectacular but it is by no means the only area in which machine code can be usefully applied. An important use of computers is data retrieval, and the arrival of Dragon Disk Drives has enormously extended Dragon's capabilities in this area. Advanced data-retrieval systems are highly complex, but the basic ideas are within our reach.

The most important component is an index which can be searched efficiently, and efficiency means machine code. For the present, we shall use BASIC techniques to create an index, and a machine code routine to operate it. The following program allows you to enter words and page numbers (or other reference numbers) into an index.

```
10 CLEAR15000
20 DIMA$(500),B$(500):A$(0)=STRING$(10,0):N=0
30 INPUT"READ FILE";X$:IFLEFT$(X$,1)><"Y"THEN70
40 INPUT"FILENAME";N$
50 OPEN"I",#-1,N$:INPUT#-1,N
60 FORI=1TON:INPUT#-1,A$(I),B$(I):NEXT:CLOSE#-1
70 INPUT"ENTRY";X$:IFX$=""THEN180
80 INPUT"PAGE";P
90 I=0
100 J=1
110 IFASC(MID$(X$,J,1))<ASC(MID$(A$(I),J,1))THEN360
120 IFASC(MID$(X$,J,1))>ASC(MID$(A$(I),J,1))THEN330
130 IFX$=A$(I)THEN170
140 IFLEN(X$)=J THEN360
150 IFLEN(A$(I))=J THEN330
160 J=J+1:GOTO110
170 B$(I)=B$(I)+STR$(P):GOTO70
180 INPUT"SAVE";X$:IFLEFT$(X$,1)><"Y"THEN220
190 INPUT"FILENAME";N$
200 OPEN"O",#-1,N$:PRINT#-1,N
210 FORI=1TON:PRINT#-1,A$(I),B$(I):NEXT:CLOSE#-1
220 INPUT"PRINT";X$:IFLEFT$(X$,1)><"Y"THEN260
230 FORI=1TON:PRINTA$(I);"";B$(I):IFINT(I/8)><I/8THEN250
240 X$=INKEY$:IFX$=""THEN240
250 NEXT
260 INPUT"DELETE ENTRY";X$:IFLEFT$(X$,1)><"Y"THEN70
270 INPUT"ENTRY TO DELETE";X$:I=1
280 IFX$=A$(I)THEN300ELSEI=I+1:IFI>N      THENPRINT"NOT
    FOUND":GOTO70
290 GOTO280
300 IFI=N THEN320
310 FORJ=I TON-1:A$(J)=A$(J+1):B$(J)=B$(J+1):NEXT
320 N=N-1:GOTO70
330 IFI<N THENI=I+1:GOTO100
340 I=N+1
350 A$(I)=X$:B$(I)=STR$(P):N=N+1:GOTO70
360 FORK=N TOI STEP-1:A$(K+1)=
    A$(K):B$(K+1)=B(K):NEXT:
    GOTO350
```

Lines 30 to 60 input an index you have previously saved (first time around, answer NO to the "READ FILE" prompt). Lines 70 to 170 add a new reference – if it matches a previous one, the page number will be added to the list. Lines 180 to 210 save the index on tape. Lines 220 to 250 list the index on the screen – if you have a printer, change the PRINT in line 230 to PRINT#-2, for a hard copy. Lines 260 to 350 allow the deletion of an entry (just in case). In each section except the "ENTRY"section (adding a new reference) the first prompt requires a "YES" or "NO" reply.

If you have a Dragon Disk Drive you may like to substitute the following lines:

```
 50 FREADN$;N
 60 FORI=1TON:FREADN$;A$(I),B$(I):NEXT
200 FWRITEN$;N
210 FORI=1TON:FWRITEN$;A$(I)+",",B$(I):NEXT
```

For small indexes, this method of storing the information is good enough, and the index can be interrogated by a BASIC program such as the following:

```
10 PMODE0:PCLEAR1:CLEAR20000
20 DIMA$(500),B$(500):A$(0)=CHR$(0)
30 INPUT"FILENAME";N$
40 OPEN"I",#-1,N$:INPUT#-1,N
50 FORI=1TON:INPUT#-1,A$(I),B$(I):NEXT:CLOSE#-1
60 INPUT"WORD";X$:X=ASC(X$):Y=
   X-64:IFX>ASC(A$(N))THEN120
70 I=INT (Y*N/26)
80 IFASC(A$(I))>X THENY=Y-1:GOTO70
90 IFASC(A$(I))<X THENI=I+1:GOTO90
100 IFX$=A$(I) THEN130
110 IFASC(A$(I))=X THENI=I+1:GOTO100
120 PRINT"NOT FOUND":GOTO60
130 PRINTA$(I);" ";B$(I):GOTO60
```

After you have given the name of the data file which contains the index, the index will be loaded. The prompt WORD? requires a reference word, which the computer will "look up" for you. Note that lines 70 to 90 perform a simple "search pattern" to avoid searching the whole index. Disk owners may substitute:

```
40 FREADN$,N
50 FORI=1TON:FREADN$;A$(I),B$(I):NEXT
```

For large indexes, these methods are rather slow, both in creating indexes and in interrogating them. Machine code would help in both of these exercises, but we shall concentrate here on the interrogation side, since an index is usually created only once, but may be interrogated thousands of times.

The first job is to create a file of data which is easy to search using a machine code program. There are many ways of doing this, and the most common methods use the ASCII codes to

represent the letters, "setting" the most significant bit of the last letter of each word. In other words, 128 is added to the code for the last letter to indicate "end of word". We shall use this method, inserting the reference pages after each word, and ending with a zero for "end of reference". Finally, two zeroes in succession indicate "end of file".

The following BASIC program will convert your data file to the required "binary" file:

```
10 PCLEAR1:CLEAR500,10000:A=10000:B=A
20 INPUT"FILENAME";N$
30 OPEN"I",#-1,N$:INPUT#-1,N
40 FORI=1TON:INPUT#-1,A$,B$
50 FORJ=1TOLEN(A$)
60 POKEB,ASC(MID$(A$,J,1)):B=B+1:NEXTJ
70 POKEB-1,128+PEEK(B-1)
80 J=1:K=2
90 IFLEN(B$)<K THEN140
100 IFMID$(B$,K,1)=" "THEN120
110 K=K+1:GOTO90
120 POKEB,VAL(MID$(B$,J,K-J)):B=B+1
130 J=K:K=J+2:GOTO90
140 POKEB,VAL(MID$(B$,J,K-J))
150 POKEB+1,0:B=B+2:IFB>32000THENPRINT"FILE TOO
    LARGE":STOP
160 NEXTI:POKEB,0:CLOSE#-1
170 INPUT"FILENAME";N$
180 CSAVEMN$,A,B,0
```

The disk version will have:

```
30 FREADN$;N
40 FORI=1TON:FREADN$;A$,B$
180 SAVEN$,A,B+1,0
```

If you want to see a list of words already stored in your Dragon, have a look at ROM addresses, starting at 32819.

This final BASIC program POKEs in 88 bytes of machine code, loads the binary file containing your index, then uses the machine code to interrogate the index.

```
10 DATA10,9E,1B,AE,24,E6,22,F7,7F,FF,10,8E,27,10,CE
20 DATA26,AC,6F,C4,BF,7F,FD,A6,A0,81,80,22,2A,A1,84
30 DATA25,1D,22,14,5A,27,12,30,01,A6,A0,81,80,22,19
40 DATAA1,80,25,06,22,03,5A,26,F1,39,BE,7F,FD,F6,7F
50 DATAFF,A6,A0,26,FC,A6,A0,26,D3,39,80,80,A1,84,25
60 DATAEB,22,E8,5A,26,E6,A6,A0,A7,C0,26,FA,39
70 PCLEAR1:CLEAR500,9900:X$=" "
80 FORI=32000TO32087:READX$:POKEI,VAL("&H"+X$):NEXT
90 INPUT"FILENAME";X$
100 CLOADMX$
110 INPUT"WORD";X$:X=ASC(X$)
120 IFX<65ORX>90THEN190
130 EXEC32000
140 A=9900
150 X=PEEK(A):IFX=0THEN190
160 PRINTX;:A=A+1
170 X=PEEK(A):IFX><0THEN160
180 PRINT:GOTO110
190 PRINT:PRINT"NOT FOUND":GOTO110
```

The disk version will have:

```
100 LOADX$+".BIN"
```

The detail of the machine code will be investigated in a future article. Its purpose is to search the index for the word input, and to "POKE" the references into the memories starting at address 9900. The address following the final reference is then cleared to zero. If the word is not found, memory 9900 is set to zero. These results are then interpreted by lines 140 to 180 of the BASIC program. Note that the use of one byte per reference restricts us to the range 1 to 255, but this can be extended by using 2-byte references.

The response of this program is almost instantaneous, even with several thousand words in the index, whereas the BASIC version gets unacceptably slow after the first few hundred.

DRAGON PUZZLE

```
10 CLS:PRINT"DRAGON PUZZLE 3"
20 PRINT@173,"----"              droop, bunting.
30 PRINT@266,"-----"             reverie.
40 PRINT@227,"----- ------"      assault-pointed missile.
50 PRINT@331,"-----"             game played on squares.
60 PRINT@106,"-----"             search.
70 PRINT@299,"--------"          mad.
80 PRINT@140,"--------"          Greek god of the sea.
90 PRINT@203, "----------"       bodies moving through space.
110 PRINT@400,""
```

Clue

```
100 FORI=109TO360STEP32:P=
    PEEK(I+1024):PRINT@I,CHR$(P+32);:NEXT
```

ISSUE NUMBER 5

Many of our readers noticed that the latest issue of Stop Press was incorrectly numbered. It was, however, issue number 4 and this is issue number 5. We apologise for the confusion this may have caused.

DOODLING

Are you good at doodling? It's fascinating watching a picture build up isn't it? Perhaps you prefer to know exactly what you want before you start, planning the whole thing before you put pen to paper. Whichever you prefer, you can build up pictures on your Dragon as easily as doodling.

We've already looked at the CIRCLE command and we know that some of you are experts at using it because we have seen the excellent entries to the Dragon Logo competition. This time let's look at the DRAW command and doodle with that. It's useful to start with a tiny program which will enable you to type in DRAW commands and see the results. Here it is.

```
10 CLS:PCLEAR4:PMODE3:PCLS
30 PRINT" ENTER INSTRUCTION ";
40 LINE INPUT D$
50 SCREEN1,1:DRAW D$
60 IF INKEY$="" THEN 60 ELSE 30
```

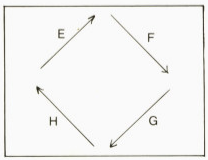
The first line sets up the high resolution graphics screen and clears it. It also clears the Text screen. The next two lines tell you what to input and take in the instructions for your drawing. LINEINPUT is used because it will accept commas. Line 50 reveals the screen and does the DRAWing. The last line holds the screen in view until you press a key. Then you are ready for a new input.

Type the program and then RUN it. Read on for an explanation of its use.

It's as though your Dragon were holding a pen at the back of the screen. When you start it is in the middle of the screen and you can tell your Dragon how to move the pen.

The easiest instructions you can give are to draw up, down, left and right. The first letters of the direction are used. So U20 means up 20 pixels (the points on the screen). L35 means left 35 pixels. INPUT U20 and you will see a line on the screen. Now press any key to return to the print screen. This time INPUT L35. You can put several instructions in at once separating them with a semi-colon (;). The semicolon isn't always necessary but if in doubt put it in. Now try D5;R15;U3L6 .

The next set of instructions are not so easy to remember. They use the letters E,F,G and H and I think a diagram is the best way of explaining them.



Try E50;F10 and G20H4;

Well thats fine for doodling without taking your pen off the paper. If we want to lift the pen up and make it move without drawing we simply put a B (for blank) in front. So BD80;L5 will take the pen down 80 pixels without making a mark and then draw leftwards for five pixels. We can also make the pen draw in one direction and go back to its starting position before the next drawing. Try this one:- L10;NE10;NR10;NF10;L10, an impression of a pigeon standing on one leg! (The N stands for 'no up date of pen position').

Of course sometimes these eight directions aren't enough and we want to draw a line at a different angle. Well if we want to move the 'pen' to the point 123 pixels from the left hand side of the screen and 47 pixels down we just say M123,47; - try it. Put B in front if you want to move the pen without drawing.

You can change the pen colour too. C7; changes the pen to magenta. C5; draws in the background colour and is useful for rubbing out. First try C7G50 then try C5E50. You had better type in C8; to get back to the orange pen.

Now you are ready to doodle using the program. Use BREAK and RUN when you want to start afresh with a clean screen.

You have learned how to write strings which form instructions to the DRAW command. You may like to use such a string in a program. The short strings can be added together to make one long one. I have got a string which draws a wriggly worm. I'll call it W$. We'll put it into our program as line 20. Just type in:-

```
20 W$="F15E15F15E15F5NL4G20H15G15H15U10"
```

Now we can explore some more commands which you will find useful. We have a string called W$. We can command it to be drawn by entering the string XW$; you see you have drawn the worm. (The semi-colon is essential here.) Now try XW$;R10XW$; Now you have the worm twice. You can change the size of the worm with another instruction.

S8XW$; draws a large worm,
S1XW$; draws a small one.

If you want the worm to stand on its head try A1XW$; if you try A2XW$; you will find the worm has turned! Experiment by putting other numbers after S or A.

Finally by using the background colour to wipe out our worm and moving it to the right we can make it creep across the screen as you press the space bar. We need to rewrite lines 30, 40 and 50. and modify line 60. This is our new program.

```
10 CLS:PCLEAR4:PMODE3:PCLS
20 W$="F15E15F15E15F5NL 4G20H15G15H15U10"
30 DRAW "BL150"
40 D$="C5;XW$;R5;C7;XW$;"
50 SCREEN1,1:DRAW D$
60 IF INKEY$="" THEN 60 ELSE 40
```

You will find all the details about the DRAW command in the programming book which came with your Dragon.

PLAY HOUSE

This program is for the very young! If you have a child in the family who is just beginning to recognise the first letters of words then why not try it? After typing the program in and RUNning, the keys H (for House), W (for Window), D (for Door) C (for Chimney), P (for Path, T (for Tree) and A (for Again) are the only keys that are active and pressing each key draws the object it stands for. However, the position it draws it in will only be the correct one if the sequence H D W W W W C P T is used. This allows for some fun if you press C instead of W and have a house with a chimney where a window should be!

As you can see, the program is quite short and uses the DRAW command. You might think that is a little strange when it comes to windows (where we could have used the LINE command). However, use of DRAW protects the program from crashing if the picture goes outside the screen.

The same idea could be used in many different situations using other words - all that is required is the same program structure but with lines 100 to 160 replaced by other appropriate sub-routines to draw the objects.

```
10 PCLEAR4:C$="HWDCPTA":DIMP(10,10):
   PMODE3,1:SCREEN1,0:PCLS1
20 FOR I=1 TO 9:FOR J=1 TO 2:READ P(I,J):NEXT J,I
30 DATA 40,145,100,145,50,90,130,90,50,130,130,130,80,50,
   98,146,230,190
40 I=0
50 R$=INKEY$:IF R$="" " THEN 50
60 R=INSTR(1,C$,R$):IF R=0 THEN 50
70 I=I+1:X=P(I,1):Y=P(I,2):ON R GOSUB 100,110,120,130,140
   ,150,170
80 IF I=9 THEN I=0
90 GOTO50
100 GOSUB180:COLOR4:DRAWX$+"U80R140D80L140U80E30
   R80F30":PAINT(X+2, Y-2),4,4:RETURN
110 GOSUB180:COLOR2:DRAWX$+"U20R40D20L40U20R40
   D10L20U20":RETURN
120 GOSUB180:COLOR2:DRAWX$+"U30R20D30L20
   ":PAINT(X+2,Y-2),3,2:RETURN
130 GOSUB180:COLOR3:DRAWX$+"U25R4U3R3D3R3D25L10
   ":PAINT(X+2,Y-15),4,3:RETURN
140 GOSUB180:COLOR2:DRAWX$+"F45R30
   H45L30":PAINT(X+4,Y+2),2,2:RETURN
150 GOSUB180: COLOR3: FORK=1 TO 15: A1$=STR$(20+RND
   (40)):A2$= STR$ (RND(30)): A3$=STR$ (RND(30))
160 DRAWX$+"U"+A1$+"E"+A2$+"G"+A2$+"H"
   +A3$:NEXT:RETURN
170 RUN
180 X$="BM"+STR$(X)+","+STR$(Y):RETURN
```

Note that in line 50, before the **JOYSTK** function is called, there is an **EXEC32786.** This is necessary because **JOYSTK(0)** is not always executed at this point. The joystick values are only updated when **JOYSTK(0)** is executed. When other values (e.g. **JOYSTK(2)**) are executed, the number returned is the one which was sampled last time **JOYSTK(0)** or **EXEC32786** were performed.

RACE

Here is a joystick game for two players. After the race track is set up, and you have said how many laps you want to race, the first to press "fire" is away as car number 1. Each turn, the cars travel in a straight line for a distance which depends on how far the joystick is pushed over. If this results in a collision with the other car, or with the side of the track, you go back to where you started that turn. The race takes place in an anti-clockwise direction.

```
10 REM RACE
20 REM A.D.MAYER, 1983
30 DATA 34,62,3,65,70,5,91,95,12,203,209,
   2,225,229,2,234,237,7,239,242,7,290,
   292,5,387,396,3,406,415,3,432,438,2
40 CLS:INPUT"HOW MANY LAPS";N:IFN>99THEN40
50 CLS6:FORI=0TO10:READX,Y,Z:FORJ=1TOZ
60 PRINT@X,STRING$(Y-X,191);:X=X+32:Y=Y+32:NEXTJ,I
70 FORI=15TO143STEP32:PRINT@I,CHR$(143);:NEXTI
80 M(1)=1:M(2)=1:PRINT@178,N;"LAPS";:GOSUB420
90 C(1)=49:C(2)=50:P(1)=1070:P(2)=1134
100 POKEP(1),C(1):POKEP(2),C(2)
110 P=PEEK(65280)
120 IFP=126ORP=254ORP=125ORP=253THEN130ELSE110
130 P=P-125:IFP>1THENP=P-128
140 T(1)=P:T(2)=1-P:C=1
150 EXEC32786:X1=JOYSTK(2-2*T(C)):Y1=JOYSTK(3-2*T(C))
160 X=FIX((X1-31.5)/6):Y=FIX((Y1-31.5)/6)
170 R=INT(SQR(X*X+Y*Y)):H=SGN(X):V=SGN(Y)
180 IFABS(H)+ABS(V)<2THEN220
190 TH=H*(X1-31.5)/(V*(Y1-31.5))
200 IFTH<.4142THENH=0:GOTO220
210 IFTH>2.4142THENV=0
220 X=1:X1=P(C):Y1=X1:M=M(C)
230 IFX>R THEN360
240 Z=Y1+H+32*V:IFPEEK(Z)=191THEN310
250 IFPEEK(Z)=143THEN330
260 SOUNDRND(400)+T(C)*215,4
270 FORI=0TO20:POKEY1,128:POKEY1,191:NEXT
280 GOSUB430:POKEX1,C(C):Y1=X1:M(C)=M
290 GOSUB420:GOTO360
300 POKEX1,C(C):GOTO360
310 POKEZ,C(C):POKEY1,191:X=X+1:GOSUB430
320 Y1=Z:GOTO230
330 IFZ=1039ORZ=1167THEN260
340 IFH<0THENM(C)=M(C)-H:IFM(C)>N
   THEN370ELSEGOSUB420
350 GOTO310
360 P(C)=Y1:C=3-C:GOTO150
370 POKEZ,C(C):POKEY1,191
380 FORI=0TO10:PLAY"O3T200CDEFG":NEXT
390 PRINT@0,C;"IS THE WINNER - ANOTHER RACE?";
400 X$=INKEY$:IFX$="Y" THENRESTORE:GOTO40
410 IFX$="N" THENSTOPELSE400
420 PRINT@243,"LAP";M(1);:PRINT@307,"LAP";M(2);:RETURN
430 IFY1=1071ORY1=1103ORY1= 1135
   THENPOKEY1,143ELSERETURN
440 IFH>0 THENM(C)=M(C)-H:GOSUB420
450 RETURN
```

LETTERS FROM READERS

Thank you to all readers who wrote to us with comments and programs. Four readers' programs are included with slight modifications to economise on space.

Patterns was the theme of two programs written by P.C.Asbury-Smith ('Random Fantasy') and John Oliver ('Tunnel'). Mr Ian Robertson sent us his bargraph program which accepts monthly frequency data and constructs a barchart. The statement 60 READ M(N) can be replaced by 60 INPUT M(N) and the demo-data removed by deleting line 220.

A program to convert joystick readings to 'keypad' numbers was sent by J Wrennall. The crux of the program lies in a clever use of a user-defined function in line 20.

We are grateful to N Atkinson for drawing attention to an error in the program ISLANDS in the last issue. Lines 530 and 540 need to be amended to the following to eradicate an occasional problem that occurred with the original version.

```
530 IFD(I) <L THEN D(I)=L+1:GOTO550
540 IF D(I)>256-L THEN D(I)=255-L ELSE D(I)=D(I)
    +SGN(JJ)*L
```

Note too that a number of lines were missed out in the introduction to the program which showed how it could be converted to use a joystick. Simply delete lines 510 and 520 and replace lines 500 and 550 with the given lines.

```
 10 Z=16:REM1-9 VALUES FROM STICK:J WRENNALL
 20 DEF   FNA(X)=5+SGN(JOYSTK(0)-32)*(-1*(ABS(JOYSTK(0)-
    32)>X))+SGN(JOYSTK(1)-32)*( 3* (ABS(JOYSTK(1)-32>X))
 30 CLS:PRINT@35,"STICK   STORED"
 40 PRINT@67,"VALUE   VALUE"
 50 PRINT@200,"7 8 9":PRINT@232, "4 5 6":PRINT@264, "1 2 3";
 60 PRINT@289,"NOTE THE CHANGE IN STICK VALUE AS YOU
    MOVE THE JOYSTICK AROUND"
 70 PRINT@352, "AND THE STORING OF THE VALUE AS YOU
    PRESS THE BUTTON"
 80 PRINT@448, "(CHANGE THE 'Z' VALUE IN LINE 10 TO ALTER
    STICK SENSITIVITY.)"
 90 PRINT@485, "press break to quit";
100 PRINT@100,FNA(Z),A
110 FOR D=1 TO 100:NEXT D
120 P=PEEK(65280)
130 IF P=126 OR P=254 THEN A=FNA(Z):SOUND10,1
140 GOTO100
```

```
 10 REM RANDOM FANTASY P.C.ASBURY-SMITH 1983
 20 CLS0
 30 X=31:Y=15:SET(X,Y,RND(8))
 40 FOR N=1 TO 2500
 50 D=2-RND(3):X=X+D:E=2-RND(3):Y=Y+E
 60 IF X>61 THEN X=61 ELSE IF X <0 THEN X=0
 70 IF Y>31 THEN Y=31 ELSE IF Y <0 THEN Y=0
 80 SET(X,Y,RND(8)):NEXT N
 90 GOTO90
```

```
 10 REM DRAGON BARGRAPH IAN ROBERTSON JUNE 1983
 20 DIMM (12),A$(12)
 30 FOR N=1 TO 12
 40 PRINT"ENTER DATA FOR EACH MONTH"
 50 PRINT@64, "MONTH";N
 60 READ M(N):M(N)=M(N)+10
 70 IF M(N)>191 THEN M(N)=191
 80 CLS:NEXT
 90 FOR N=1 TO 12:READ A$(N):NEXT
100 PRINT@98,"TO SEE GRAPH PRESS spacebar"
110 PRINT@224,STRING$(32,"$")
120 PRINT@290,"spacebar AGAIN FOR NEW GRAPH"
130 SCREEN0,1
140 IF INKEY$=" " THEN 140
150 PMODE3,1:PCLS:SCREEN1,1
160 N=17:FOR N1=1 TO12:DRAW"BM"+STR$(N)+",191C7"
    +A$(N1):N=N+20:NEXT
170 C=2:X=7:COLORC:FOR N=1 TO 12:COLORC
180 LINE(X,182)-(X+20,191-M(N)),PSET,BF:COLOR5
190 LINE(X,182)-(X+20,191-M(N)),PSET,B
200 X=X+20:C=C+1:IFC>4 THEN C=2
210 NEXT
220 DATA 123,3,73,187,69,100,57,65,87,32,145,150
230 DATAB4R3U8L2R4, U4NR1U4NR4, BL4U8F4E4D8, BL4U4N
    R6U4R6D8,  BL4U8F4E4D8,  BL4R3U8L2R4,  BL4R3U8L2R4,
    BL4U4NR6U4R6D8
240 DATA BL4R6U4L6U4R6, BL4R6U8L6D8, BL4U8BD1F6BD1U8,
    BL4U8R4F2D5G2L4
250 IF INKEY$<>CHR$(32) THEN 250 ELSE RUN
```

```
 10 CLS:REM TUNNEL JOHN OLIVER
 20 INPUT"TYPE VALUE OF STEP";Z
 30 PMODE4,1:SCREEN1,1:PCLS
 40 A=RND(128):B=RND(96):X=RND(128)+18:Y=RND(96)+96
 50 A=A+Z:B=B+Z:X=X-Z:Y=Y-Z
 60 GOSUB 200:IF F=-1 THEN 90
 70  LINE(A,B)-(X,Y),PSET,B
 80 GOTO50
 90 A=A-Z:B=B-Z:X=X+Z:Y=Y+Z
100 GOSUB 200:IF F=-1 THEN 30
110 LINE(A,B)-(X,Y), PRESET,B
120 GOTO90
200 F=A<0 OR A>255 OR B<0 OR B>255 OR X <0 OR
    X>255 OR Y<0 OR Y>255:RETURN
```

Pleasant DREAMs

Each issue of Stop Press has included machine code programs both in their mnemonic form and in their binary form. Developing such programs needs the facility of an assembler program. DREAM is such a program available from Dragon Data Ltd. The cassette-based program comes with a comprehensive booklet explaining its ability to edit text (of any sort) and to assemble program text files into machine code. Together with DREAMBUG (also on cassette) or combined, in a cartridge (ALLDREAM) DREAM is ideal for implementing machine code programs. What follows is designed to provide you with an introduction to DREAM and present a working session using it to assemble a program. To do this all key-strokes are shown together with an indication of their results. Where necessary, keys such as ENTER and BREAK are shown in square brackets. In addition SHIFT together with SPACE-BAR (used for tabulating between fields) is presented by [TAB].

To illustrate the use of DREAM we will enter the text version of a program we included in Stop Press 1. First load DREAM and press 'N' for a new text.

```
CLEAR 200,20000 [ENTER]
CLOADM"DREAM"[ENTER]
N
```

Now key in each line of the text.

```
[TAB]              LDA    [TAB]   $7FFF    [ENTER]
[TAB]              LDX    [TAB]   #$400    [ENTER]
LOOP    [TAB]      STA    [TAB]   ,X       [ENTER]
```

(Ooops - missed the '+' out)

```
↑[TAB][TAB]> > +[ENTER]
```

(Move up to previous line, tab to 'X' and add '+')

```
[TAB]              CMPX   [TAB]   $600     [TAB]
[TAB]              BNE    [TAB]   LOOP     [ENTER]
[TAB]              RTS    [ENTER]
```

We now have the text of the program as in the newsletter - complete with mistake - it should have been CMPX #$600, the # denoting an immediate address. To correct this either use the arrow keys to tab to the appropriate place or use the EDITOR as follows.

```
[BREAK] H [ENTER]          returns cursor to home position
[BREAK] C/$6/ #$6/ [ENTER]
```

The editor scans down the text file to find the first occurrence of $6 and replaces it with #$6.

There are a few points to note here. First, you must identify uniquely, that part of the text you wish to alter - it would be no use using the command C/$/#$/ from the home position as it would alter the first $ sign it came to. Secondly, note that C/$6// would delete the first occurrence of $6.

Finally, in some situations we need to change all occurrences of a pattern of text. To do this use an 'A' at the end of the command. This can be particularly useful when a complicated piece of text occurs frequently - substitute say ZZZ for the piece and after executing the HOME command use the command C/ZZZ/whatever it should be /A.

DREAM's editor facilities include many other features. Although our text is ready to assemble we will demonstrate some more commands.

```
[BREAK] H [LOOP]
[BREAK] F/LOOP/[ENTER]
```

(Finds the first occurrence of 'LOOP'. If you wanted to find all occurrences repeat the last command using
[SHIFT][@] .)

```
[BREAK] D2 [ENTER]
```

(Deletes two lines starting in the current position.)

```
[BREAK] H [ENTER]
↑↑ [BREAK] I2 [ENTER]
```

(Prepares for insertion of two lines in the position of the cursor.).

```
LOOP    [TAB]      STA    [TAB]   ,X+      [ENTER]
[TAB]              CMPX   [TAB]   #$600    [ENTER]
[BREAK] H [ENTER]
```

(Text now restored to previous state.)

Continued on next page

Pleasant DREAMs (continued)

To assemble the program type **[BREAK] A [ENTER].** The resulting screen has as its first line **4E21 B67FFF LDA $7FFF.** This shows that the bytes of code **B6, 7F,FF** have been assembled starting at memory location **4E21** or decimal **20001** which is the default setting used by DREAM. If you wish to place the code elsewhere then use an ORG instruction as the first line of the text program.

As you should see there are no errors. What happens if there are?

[BREAK] Q [ENTER] (returns to text file)
[BREAK] H [ENTER]
[BREAK] C/STA/STZ/[ENTER] (changes STA to STZ which is wrong.)
[BREAK] A [ENTER]

Now the screen shows an error on the incorrect line. Type [ENTER] again to continue assembly. Now to restore the text, type: [BREAK] Q [ENTER] [BREAK] H [ENTER] [BREAK] C/STZ/STA/ [ENTER]
Re-assemble using [BREAK] A [ENTER] [BREAK] Q [ENTER].

It is important to save the text file as soon as it has assembled without errors as running the machine code program could corrupt any of the contents of RAM. If you wish to manipulate a tape, exit to BASIC using **[BREAK] Q [ENTER]** from text mode and you are free to control the tape recorder.

It is a good idea to keep a separate tape for DREAM text files as the SKIPF command does not function with such files. (This is so that text files may be merged from tape.) Positioning the tape can be achieved by using MOTOR ON, AUDIO ON and PLAY. Then listen for the sound (lack of) indicating a clear space on the tape. Then type **EXEC 27776 [ENTER] Y** (to return to the old text) followed by **[BREAK]S SCREEN [ENTER].**

Note that the space between S and the title is necessary. Each line of text disappears as it is loaded until finally the HOME position is repeated.

Finally we will show how merging of files can be achieved by adding an ORG statement to the recorded program.

[BREAK] Q [ENTER]
EXEC 27776 [ENTER]N [TAB] ORG [TAB] $4F00 [ENTER]

Prepare the recorder to play back the stored program and type **[BREAK] L SCREEN [ENTER].**

DREAM will search through the tape until **SCREEN** is found and load it after the current line.

A few readers have experienced difficulties with the saving and loading of DREAM text files. These difficulties can be avoided as follows:-

If the Header information on your tape is being corrupted then POKE &H745B,255.

If the Text File will not load correctly then POKE &H746B,128. If this is successful, then gradually decrease the POKE value from 128 to a minimum of 1, but DO NOT USE POKE &H746B,0 as this will produce an extremely long Header. If unsuccesful, gradually increase this value from 128 to 255.

DREAM has many other facilities (for example text programs may be sent to a printer using the editor command P) but we have used those which are most useful for our simple machine-code programs. If you wish to implement the material in MCC why not try DREAM?

WIN A DRAGON 64

We are looking for the Dragon User of 1983. If you have an unusual application story for your Dragon 32 whether at home, at school or at work then we want to hear from you.

Send a report outlining in less than 100 words on how you use your Dragon computer to:

Cathy Hyde,
'Dragon User of 1983',
Dragon Data Ltd,
Kenfig Industrial Estate,
Margam,
Port Talbot,
West Glamorgan.

Entries must be received by November 30th.
The winner will receive a new Dragon 64 computer.
The judges decision will be final.
The winning application story will appear in a future issue.

Published by: Welbeck Public Relations, 2 Endell Street, London WC2. Printed by: Drydens Printers Ltd., Brent Crescent, North Circular Road, London NW10.