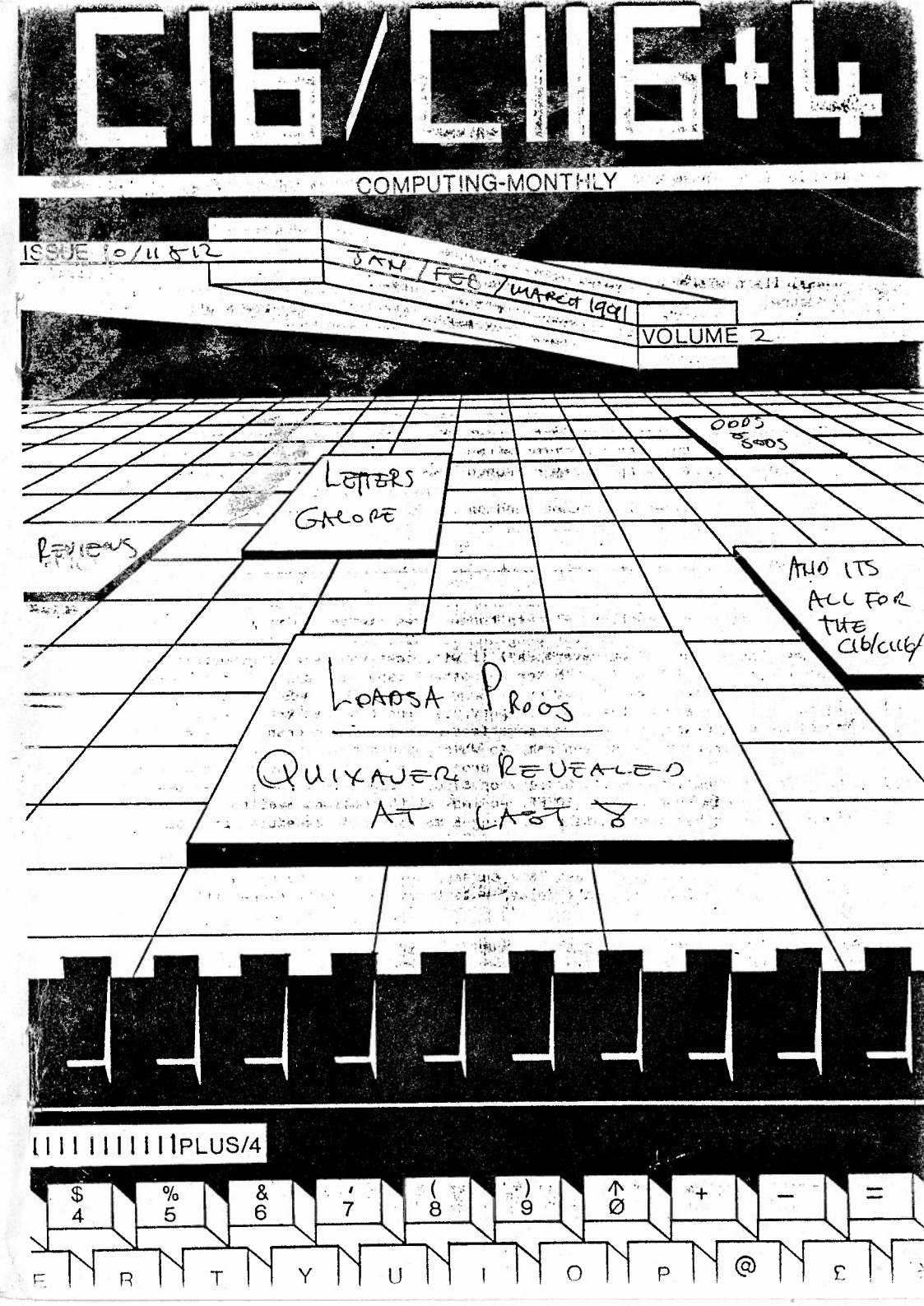

C16/C116+4
COMPUTING-MONTHLY
ISSUE 10/11&12
JAN/FEB/MARCH 1991
VOLUME 2
LETTERS GALORE
REVIEWS
AND ITS ALL FOR THE C16/C116/
LOADSA PROGS
QUIXAWER REVEALED AT LAST
PLUS/4

Ed's Big Bits

Hello dear members,
Well what have we here, another tripple issue, I'll explain why, I did Jans, then it went it Feb, then I added Feb's to Jans, then it went into March then I put March's in with Jan & Febs, so I ended up with a tripple issue.

Well Its now been two years since the inspiration for a C16/+4 club occured, and I'm pleased that I've got this far, but really its a big thankyou to all of you for supporting me, although you know what I'm like now, quite slow, but I get there. Unfortunetly in the last 4-5 months we've lost a few members because they have moved onto different things, well they say that life does'nt stand still, this proves it.

I'm please to introduce 3 new members, these are Lawrence Beazly from Middx, John Lemmermann from Germany, and Karl Schulz also from Germany, we all hope that you enjoy your time with us. All members addresses will be displayed in the new April/May Bi-Monthly Mag, called 'C16/C116/+4 Bi-Monthly Computing', the mag will be Bi-Monthly from April 1991 and each issue will be (hopefully) despatched on the 20th of each second month, ie, 20th May for the April/May 1991 ish.

Whats Inside this month!!!!

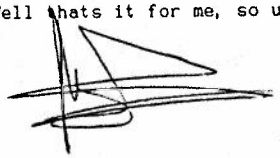
Wel you have 60 pages full of letters, progs, reviews and other various articles for you digestion, also in these pages are the Quixaver instructions/info, but not the program, you will have to buy this off Eric Jones (The Author, of Quixaver), all it will cost you is £2 cheque/po made payable to ERIC JONES, 1 X C15 (or any other) tape, and a 22p (first class) stamp, and what you get in return is a superb Fast Tape saver that does'nt require use of any of the C16/+4 memory, ie, for thos hackers (like I used to be), when Cracking Games you sometimes have on screen graphic loader programs stored in the screen ram, so when you did a Reset + Run/Stop you lost the most importaant part of the program, but this quixaver does'nt use any memory from $1000 - $4000 (c16) or $1000 - $8000 (+4) and it does use the screen ram from $0800 - $0FFF, so look at the options available to you, and also Eric has just finished writing a machine code assembler program, but it uses quixaver, so to use any of Erics other progs you require quixaver, very cunning Eric, I like it, but really, its well worth £2, its better than all the other Fast Tape savers, so getting sending the cheques/po's off today, don't delay, post today, you won't regret it!!!.

Write to:

Eric Jones
The Fold
Bucknell
SHROPSHIRE
SY7 0AA

or Tel for details on 05475-348 after 6pm.

Well thats it for me, so until next ish, Byyyyeeee!!!!!!



AMC

72. St. Neots Road,
Eaton Ford,
St. Neots,
Huntingdon,
Cambs.
PE19 3BD.

Nov 1990

Dear Roy,

I do not know if this simple program is of any interest to the magazine, as it is the first time I have tried preparing anything for someone else.

I have sent you two versions because I am not sure about copyright, and the printout routine come from Your Commodore Jan. 1987.
I am sending you a copy of the printout version on disc along with some more PD programs.

I hope this will be of interest to the readers.

Yours sincerly,

Andy Brett

Andy Brett

AMU

Rubber Banding

This is a simple programe to demonstrate drawing by what is called rubber banding, also how to print lowercase characters on the graphic screen.

The drawing point is moved with the cursor keys to the required position, the line will be drawn if the spacebar is pressed, or the start point moved if the return key is pressed.

The text or 'T' option will print your text starting at the current drawing position. To use the text option press 'T', enter your text, press return.

As well as normal letters, the 'T' command will also print the graphic characters obtained by using the Comodore key and a letter key. If poke740,212 is left out of line 50, only uppercase letters will be printed, but the full range of graphic characters are available.

There are a lot of REM statements to explain the program, these may be omitted to save typing.

The graphic printout routine has been taken from part of the Plus/4 Dumper program in Your Commodore Jan. 1987. If this option is not required omit all of the Rubber Banding Loader and the following lines in Rubber Banding,

line 20
line 190
line 390
line 450 onwards

It is important that the pokes in line 10 of Rubber Banding Loader are entered before the main program is loaded, so if you do not wish to use the loader you must enter the pokes in direct mode before loading the program.

The name in line 20 of the loader program must be the same as the name used to save the main program, if you are using tape change the 8 on the end to a 1

Save the program before you run it, so if anything goes wrong you haven't lost all your work.

RUBBER BANDING LOADER

```
10 PRINT"⊐▣▣GRAPHIC1:GRAPHIC0:POKE44,72:POKE46,72:POKE48,72:POKE50,72:POKE18432,
0:NEW"
20 PRINT"▣▣LOAD"CHR$(34)"RUBBER BAND"CHR$(34)CHR$(44)"8"
30 PRINT"▣▣▣▣RUN"
40 PRINT"⊐";
50 POKE239,4
60 POKE1319,13:POKE1320,3:POKE1321,13:POKE1322,13
```

RUBBER BANDING

```
10 REM SYMBOLS- "⊐" = CURSOR UP, "▣" = CURSOR DOWN, "▮▮" = CURSOR LEFT, "▶▮" = CUR
SOR RIGHT
20 PRINT"⊐":CHAR1,3,12,"PLEASE WAIT LOADING PRINT ROUTINE":GOSUB460
30 A=100:B=160:REM SET DRAWING START POSITION
40 GRAPHIC1,1:GRAPHIC0
50 GOSUB300:GRAPHIC1:PRINTCHR$(14):POKE740,212
60 REM POKE 740,212 SWITCHES TO LOWCASE LETTERS IN GRAPHIC SCREEN
70 REM POKE 740,208 WILL SWITCH BACK TO UPPERCASE LETTERS IN GRAPHIC SCREEN
80 REM 30 TO 130 RUBBER BANDING ROUTINE
90 CA=A:CB=B
100 GETA$:IFA$=" "THENDRAW1,CB,CATOB,A:CA=A:CB=B:REM DRAWS LINE
110 IFA$=CHR$(13)THENCA=A:CB=B:REM MOVES DRAWING POINT TO NEW POSITION
120 SP%=1:IFPEEK(1347)=1THENSP%=10:REM CHECK FOR SHIFT KEY, INCREASE SP% BY 10
IF PRESSED
130 REM SP%=MOVEMENT SPEED
140 IFA$="▮▮"THENB=B-SP%
150 IFA$="▶▮"THENB=B+SP%
160 IFA$="▣"THENA=A+SP%
170 IFA$="⊐"THENA=A-SP%
180 IFA$="T"THENGOSUB250:REM FOR TEXT ROUTINE
190 IFA$="P"THENSYS16645:REM FOR PRINTOUT ROUTINE
200 IFA$=CHR$(20)THENGRAPHIC1,1:REM CLEARS GRAPHIC SCREEN
210 IFA$=CHR$(27)THENGRAPHIC0:END
220 FORZ=1TO2:DRAW1,CB,CATOB,A:DRAW0,CB,CATOB,A:NEXT
230 REM LINE 120 DRAWS AND RUBS OUT LINE I.E. RUBBER BANDING
240 GOTO100
250 GRAPHIC2
260 REM GRAPHIC2 SWITCHES SCREEN TO HIGH-RES+TEXT:GRAPHIC1 SWITCHES IT BACK TO
HIGH RES
270 INPUT"ENTER TEXT";T$:CHAR1,(B/8)+.5,(A/8)+.5,T$
280 PRINT"▣▣▣▣":GRAPHIC1:RETURN
290 REM THE CURSOR DOWN "▣▣▣▣" MOVES INPUT TEXT OUT OF SIGHT FOR NEXT TEXT ENTRY
300 SCNCLR
310 PRINT"▣▣▣▶▶▶▶▶▶▶▶DIRECTIONS, FUNCTIONS"
320 PRINT"▶▶▶▶▶▶▶▶▶AND THE KEYS FOR THEM":PRINT"          ▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬"
330 REM THE UNDERLINE ABOVE IS 21 COMMODORE KEY AND I
340 PRINT"▣▣  USE CURSOR KEYS TO MOVE, SPACEBAR"
350 PRINT"  TO DRAW LINE, RETURN TO MOVE WITHOUT"
360 PRINT"  DRAWING A LINE, SHIFT KEY INCREASES"
370 PRINT,"BANDING SPEED"
380 PRINT,"▣▣T FOR TEXT"
390 PRINT,"▣P FOR PRINTER"
400 PRINT,"▣ESC TO END"
410 PRINT,"▣DEL TO CLEAR SCREEN"
420 PRINT,"▣PRESS ANY KEY TO START"
430 GETKEY Z$:PRINT:PRINT:PRINT
440 RETURN
```

RUBBER BANDING

```
450 :REM MACHINE CODE PRINT ROUTINE
460 SE =  16384 :EE= 17454
470 FORI=SETOEE:READA:POKEI,A:NEXT
480 :
490 DATA 032,132,066,173,019,255,201,209,208,005,160,008,076,017
500 DATA 064,160,007,162,004,169,004,032,186,255,169,000,032,189
510 DATA 255,032,192,255,162,004,032,201,255,169,000,133,034,169
520 DATA 012,133,035,162,000,160,000,152,133,075,056,177,034,133
530 DATA 094,041,127,201,034,208,007,024,165,094,105,005,133,094
540 DATA 165,094,133,093,056,152,072,165,075,168,165,094,041,128
550 DATA 240,009,032,226,064,165,094,041,127,133,094,165,094,233
560 DATA 032,024,016,008,105,096,153,061,068,076,155,064,105,032
570 DATA 056,233,064,024,016,008,105,064,153,061,068,076,155,064
580 DATA 105,064,056,233,096,024,016,008,105,128,153,061,068,076
590 DATA 155,064,105,096,056,233,128,024,016,005,105,192,153,061
600 DATA 068,104,168,200,230,075,165,093,041,128,240,003,032,244
610 DATA 064,192,040,208,135,160,000,185,061,068,032,210,255,200
620 DATA 196,075,208,245,169,013,032,210,255,232,224,025,208,014
630 DATA 169,145,032,210,255,169,004,032,195,255,032,145,066,096
640 DATA 024,165,034,105,040,133,034,165,035,105,000,133,035,076
650 DATA 047,064,138,133,076,165,075,170,169,018,157,061,068,230
660 DATA 075,165,076,170,200,096,138,133,076,165,075,170,169,146
670 DATA 157,061,068,230,075,165,076,170,096,032,132,066,162,004
680 DATA 169,004,160,000,032,186,255,169,000,032,189,255,032,192
690 DATA 255,162,004,032,201,255,169,008,032,210,255,169,000,133
700 DATA 091,133,094,169,158,133,075,169,066,133,076,169,061,133
710 DATA 092,169,068,133,093,165,094,201,028,208,003,076,100,066
720 DATA 230,094,169,000,133,091,160,000,162,000,177,075,149,077
730 DATA 200,232,192,014,208,246,024,165,075,105,014,133,075,165
740 DATA 076,105,000,133,076,169,000,133,091,162,000,169,000,157
750 DATA 053,068,232,224,008,208,246,160,000,177,077,141,046,068
760 DATA 177,079,141,047,068,177,081,141,048,068,177,083,141,049
770 DATA 068,177,085,141,050,068,177,087,141,051,068,177,089,141
780 DATA 052,068,169,001,133,002,160,000,162,000,169,000,157,053
790 DATA 068,232,224,008,208,248,162,000,024,185,046,068,042,153
800 DATA 046,068,144,003,076,193,065,232,224,008,208,239,076,207
810 DATA 065,024,189,053,068,101,002,157,053,068,232,224,008,208
820 DATA 222,200,006,002,192,007,208,212,162,000,024,189,053,068
830 DATA 105,128,157,053,068,232,224,008,208,242,160,000,185,053
840 DATA 068,145,092,200,192,008,208,246,024,165,092,105,008,133
850 DATA 092,165,093,105,000,133,093,165,091,201,039,240,003,076
860 DATA 074,066,169,010,160,000,145,092,200,169,010,145,092,165
870 DATA 094,072,165,075,072,165,076,072,160,000,162,004,185,061
```

RUBBER BANDING

```
880 DATA 068,032,210,255,200,192,000,208,245,185,061,069,032,210
890 DATA 255,200,192,065,208,245,104,133,076,104,133,075,104,133
900 DATA 094,169,061,133,092,169,068,133,093,076,057,065,230,091
910 DATA 162,000,024,181,077,105,008,149,077,232,181,077,105,000
920 DATA 149,077,232,224,014,208,237,076,103,065,162,004,169,013
930 DATA 032,210,255,169,015,032,210,255,169,004,032,195,255,032
940 DATA 145,066,169,013,032,210,255,032,210,255,032,210,255,096
950 DATA 162,000,181,000,157,127,069,232,224,000,208,246,096,162
960 DATA 000,189,127,069,149,000,232,224,000,208,246,096,000,032
970 DATA 001,032,002,032,003,032,004,032,005,032,006,032,007,032
980 DATA 064,033,065,033,066,033,067,033,068,033,069,033,070,033
990 DATA 071,033,128,034,129,034,130,034,131,034,132,034,133,034
1000 DATA 134,034,135,034,192,035,193,035,194,035,195,035,196,035
1010 DATA 197,035,198,035,199,035,000,037,001,037,002,037,003,037
1020 DATA 004,037,005,037,006,037,007,037,064,038,065,038,066,038
1030 DATA 067,038,068,038,069,038,070,038,071,038,128,039,129,039
1040 DATA 130,039,131,039,132,039,133,039,134,039,135,039,192,040
1050 DATA 193,040,194,040,195,040,196,040,197,040,198,040,199,040
1060 DATA 000,042,001,042,002,042,003,042,004,042,005,042,006,042
1070 DATA 007,042,064,043,065,043,066,043,067,043,068,043,069,043
1080 DATA 070,043,071,043,128,044,129,044,130,044,131,044,132,044
1090 DATA 133,044,134,044,135,044,192,045,193,045,194,045,195,045
1100 DATA 196,045,197,045,198,045,199,045,000,047,001,047,002,047
1110 DATA 003,047,004,047,005,047,006,047,007,047,064,048,065,048
1120 DATA 066,048,067,048,068,048,069,048,070,048,071,048,128,049
1130 DATA 129,049,130,049,131,049,132,049,133,049,134,049,135,049
1140 DATA 192,050,193,050,194,050,195,050,196,050,197,050,198,050
1150 DATA 199,050,000,052,001,052,002,052,003,052,004,052,005,052
1160 DATA 006,052,007,052,064,053,065,053,066,053,067,053,068,053
1170 DATA 069,053,070,053,071,053,128,054,129,054,130,054,131,054
1180 DATA 132,054,133,054,134,054,135,054,192,055,193,055,194,055
1190 DATA 195,055,196,055,197,055,198,055,199,055,000,057,001,057
1200 DATA 002,057,003,057,004,057,005,057,006,057,007,057,064,058
1210 DATA 065,058,066,058,067,058,068,058,069,058,070,058,071,058
1220 DATA 128,059,129,059,130,059,131,059,132,059,133,059,134,059
1230 DATA 135,059,192,060,193,060,194,060,195,060,196,060,197,060
1240 DATA 198,060,199,060,000,062,001,062,002,062,003,062,004,062
1250 DATA 005,062,006,062,007,062,000
1260 RETURN
```

Rubber Banding

This is a simple programe to demonstrate drawing by what is called rubber banding, also how to print lowercase characters on the graphic screen.

The drawing point is moved with the cursor keys to the required position, the line will be drawn if the spacebar is pressed, or the start point moved if the return key is pressed.

The text or 'T' option will print your text starting at the current drawing position. To use the text option press 'T', enter your text, press return.

As well as normal letters, the 'T' command will also print the graphic characters obtained by using the Comodore key and a letter key. If poke740,212 is left out of line 50, only uppercase letters will be printed, but the full range of graphic characters are available.

There are a lot of REM statements to explain the program, these may be omitted to save typing.

Save the program before you run it, so if anything goes wrong you haven't lost all your work.

RUBBER BANDING

```
10 REM SYMBOLS- "□" = CURSOR UP, "■" = CURSOR DOWN, "■" = CURSOR LEFT, "■" = CUR
SOR RIGHT
30 A=100:B=160:REM SET DRAWING START POSITION
40 GRAPHIC1,1:GRAPHIC0
50 GOSUB300:GRAPHIC1:PRINTCHR$(14):POKE740,212
60 REM POKE 740,212 SWITCHES TO LOWCASE LETTERS IN GRAPHIC SCREEN
70 REM POKE 740,208 WILL SWITCH BACK TO UPPERCASE LETTERS IN GRAPHIC SCREEN
80 REM 30 TO 130 RUBBER BANDING ROUTINE
90 CA=A:CB=B
100 GETA$:IFA$="" "THENDRAW1,CB,CATOB,A:CA=A:CB=B:REM DRAWS LINE
110 IFA$=CHR$(13)THENCA=A:CB=B:REM MOVES DRAWING POINT TO NEW POSITION
120 SP%=1:IFPEEK(1347)=1THENSP%=10:REM CHECK FOR SHIFT KEY, INCREASE SP% BY 10
IF PRESSED
130 REM SP%=MOVEMENT SPEED
140 IFA$="■"THENB=B-SP%
150 IFA$="■"THENB=B+SP%
160 IFA$="■"THENA=A+SP%
170 IFA$="□"THENA=A-SP%
180 IFA$="T"THENGOSUB250:REM FOR TEXT ROUTINE
200 IFA$=CHR$(20)THENGRAPHIC1,1:REM CLEARS GRAPHIC SCREEN
210 IFA$=CHR$(27)THENGRAPHIC0:END
220 FORZ=1TO2:DRAW1,CB,CATOB,A:DRAW0,CB,CATOB,A:NEXT
230 REM LINE 120 DRAWS AND RUBS OUT LINE I.E. RUBBER BANDING
240 GOTO100
250 GRAPHIC2
260 REM GRAPHIC2 SWITCHES SCREEN TO HIGH-RES+TEXT:GRAPHIC1 SWITCHES IT BACK TO
HIGH RES
270 INPUT"ENTER TEXT";T$:CHAR1,(B/8)+.5,(A/8)+.5,T$
280 PRINT"■■■■■":GRAPHIC1:RETURN
290 REM THE CURSOR DOWN "■■■■■" MOVES INPUT TEXT OUT OF SIGHT FOR NEXT TEXT ENTRY
300 SCNCLR
310 PRINT"■■■■■■■■■■■DIRECTIONS, FUNCTIONS"
320 PRINT"■■■■■■■■■■AND THE KEYS FOR THEM":PRINT"                    "
330 REM THE UNDERLINE ABOVE IS 21 COMMODORE KEY AND I
340 PRINT"■■  USE CURSOR KEYS TO MOVE, SPACEBAR"
350 PRINT"  TO DRAW LINE, RETURN TO MOVE WITHOUT"
360 PRINT"  DRAWING A LINE, SHIFT KEY INCREASES"
370 PRINT,"BANDING SPEED"
380 PRINT,"■■T FOR TEXT"
400 PRINT,"■ESC TO END"
410 PRINT,"■DEL TO CLEAR SCREEN
420 PRINT,"■PRESS ANY KEY TO START"
430 GETKEY Z$:PRINT:PRINT:PRINT
440 RETURN
```

```
********** LUNAR LANDER **********
********** part 1 by PETER CRACK **********
*************************************************************************
* Hello again, club members.
* I have sent this programme in code order so all the addresses will follow
* on, one from the other, but check them as you enter them anyway.
* This programme can only be run on the +4 or C16 with a 64K expansion
* as I have made use of locations $9000 to $BFFF for data storage.
* Now for the scene setting.....
* You are a ground controller on a moon station (any moon around any planet)
* and your job is to take a REMOTE CONTROLLED INTERSTELLAR CARGO SHIP(RCICS)
* and guide it safely to one of nine landing places under your command, each
* landing place or platform will only hold one RCICS, the RCICS will appear
* at the top left hand corner of the main screen and the landing places will
* appear as the flat bits on the terrain outline, your first job is to
* deccelerate the RCICS to a manageable speed, when the RCICS, under your
* guidance, gets close to a landing place the screen will change to a
* greater magnification and three possible landing places will be displayed
* (the larger flat bits), at this point a word of warning!!! the company for
* whom you work are, bu interstellar standards not very rich, so the
* equipment you are using is a bit old and not very accurate, do NOT take
* the RCICS too close to the limits of your controlled area (sides of the
* screen), also when moving from one screen to another the, as you thought,
* perfect position of the RCICS may not be so good after all, ah such is
* life!!!!!. Anyway having moved to the second magnification choose your
* landing place and guide the RCICS over it, remember to keep the speed
* down!!, when the RCICS is over this landing place the screen will change
* to the third and last magnification and only one landing place will be
* presented (the very long flat bit if empty or an evenly spaced hump if
* not), guide the RCICS to this landing place and allow it to slowly sink
* down on its landing gear, this has to be done nine times to end the
* sequence. On screen at all times are four sets of numbers (I did say it
* was old equipment), starting with the top left hand group you have ROTATE
* and a number,this is the rotation speed, this must be zero at landing,
* beneath this is FUEL and a number, this is the number of fuel units you
* have left ....(Clever eh)!! At top right there is VERT and a number.This
* is your vertical speed,(up and down the screen).And it must be less than
* ten at landing. Beneath this is HORI and again,a number.This must be zero
* at landing, all but the FUEL line have two '**' chars if the number is
* zero. There is also two letters following that these are to tell you
* which direction the RCICS is moving, DO=down, UP=up, LE=left and RI= right
* and thats the scene set.
* Now the controls, this is by joystick, I cannot remember which port this
* should be in but run the programme, when it is all in, and push the stick
* forward. If nothing happens, try the other port. To rotate the RCICS, push
* the stick left or right. To run the main engine, push the stick up or
* forward. Try not to overcontrol the RCICS, but keep an eye on the vertical
* speed. If it goes over one hundred you have lost it.
* If you crash then you lose five hundred fuel units, if you land O.K. then
* you gain five hundred units. If there is more than one player, then the
* winner could be the one with the most sucessful landings or if both have
* made nine landings the one with the most fuel left will be the winner.
* Keep a side record of this because the machine will not !!
* To slow down the RCICS, rotate it until it points in the opposite
* direction to that inwhich it is moving and fire the main rocket motor
* until it has reached the desired speed. So, to sum up, when you start
* nine landding places are shown on the screen, these are in three groups
* of three. On screen two you are shown three landing places, select one and
* move to it. On screen three, you are shown the final landing place.
* When on screen one, movement off the top or sides of the screen results in
* loss of the RCICS and five hundred fuel points. When on screen two or
* three, movement off the top or sides results in the next screen being
* displayed. Think of the display as a pyramid with screen one at the top
* with all nine landing places displayed. Beneath are three more screens
* each with three landing places. Beaneath these are nine screens each with
* one landing place. So movement is possible between all screens at each
* level, but try to remeber where you are as the equipment the company has
* provided has not got a clue!!.....GOOD LANDINGS.
```

19

```
**********************************************************************
* Here is the explanation.
4000-4006 The rocket and flame sprite are always moved together but when in
*         the crash routine only the rocket routine is used hence this short
*         routine.
4007-40D2 Rocket (RCICS) sprite movement routine.
4007-400C Load 'A' reg. with rotation pointer and compare it with $$2A (*)
*         if it is $$2A then the rotation number is zero (every time any of
*         the movement numbers fall to zero two stars (**) are printed in
*         place of the direction letters) and we branch to $404D else......
4006-4014 load 'A' reg. with the tens column from the rotation number and
*         remove the four leftmost bits (remember these numbers are in ASCII
*         form 0-$$30 and we only want the number) shift it left one place
*         times two and store it in $53F0, this is the delay reload. (all
*         four groups of numbers are increased and decreased in decimal not
*         hex format I.E. 0-9 not 0-F the rotate numbers run from 0-99 and
*         speed of rotation is changed by the value of the number in the
*         tens column the higher it is the faster the rotation).
4017-4028 Load 'Y' reg. with delay counter increase it and compare it to
*         $$0C if it is not store it back and branch to $404D else load
*         delay reload and store it in delay counter.
402B-402D The 'X' reg. holds the first of the two characters following the
*         rotate number and was set in $4007 as it was not a * then it
*         must be a L or R, L in ASCII=$$4C so check for this if not equal
*         branch to $403F else.........
402F-403D Load 'Y' reg. with sprite angle number increase 'Y' reg. compare
*         it with $$0E. $$00 to $$0D are the sprite angle numbers if $$0E
*         has been reached the sprite has gone full circle and we reset it
*         to $$00 then branch to $404A.
403F-4047 Same as above but opposite direction.
404A      Gosub transfer new sprite data to data working area.
* Wether the sprite has been rotated or not this part of the routine is
*         always done.
404D-404E Rom/ram toggle.
4051-4055 Load sprite number into $E5 and gosub get pointers.
4058-405D Check if sprite is to move horizontaly if not branch to $4091 else
405F-4068 Get hundreds column of horizontal speed remove ASCII value to
*         leave just the number and branch to $406A if equal else load
*         'X' reg. with $$02 and branch to $407E.
406A-407C Get tens column of horizontal speed number clear out ASCII value
*         leaving just the number and transfer it to 'X' reg. Load 'Y' reg.
*         with delay counter, increase it and compare it to $$05 branch to
*         $407E if greater else store 'Y' reg. in delay counter and branch
*         to $4091.
407E      If the hundreds column of the horizontal speed is greater than
*         zero then the sprite has to be moved at a fast speed hence the
*         LDX $$02 command in $4066, we now store this value in the delay
*         counter.
4081-408E This part checks to see if we have to move left or right so, load
*         'A' reg. with the first character after the number and compare it
*         with $$4C and branch to $408E if it is and move sprite left one
*         pixel else, move sprite right one pixel and branch to $4091.
4091-4096 Load first character after vertical speed number into 'A' reg. and
*         compare it with $$2A (*) and branch to $40CC if it is else.......
4098-409D Load 'A' reg. with vertical speed number hundreds column remove
*         ASCII value and branch to $40A4 if the result is zero, else......
409F-40A2 Load highest speed in 'X' reg. and branch to $40B9.
40A4-40AA Load vertical speed number tens column in 'A' reg. and clear ASCII
*         value, multyply result by two 'ASL' and transfer it to 'X' reg.
40AB-40B7 Load 'Y' reg. with delay counter, increase it and compare it with
*         $$0A. Branch to $40BA if greater else store it back and branch to
*         $40CE.
40B9      Store 'X' reg. (new delay reload), in delay counter.
40BC-40C7 Now we check to see if we have to move up or down, load 'A' reg.
*         with first character after vertical speed number is it $$44 (a'D')
*         if it is branch to $40C9 and move down one pixel else move up one
*         pixel and branch to $40CC.
********************************************** CONTINUED **********
```

```
40CC-40D2 And at last print sprite in new position, save sprite pointers and
*         return.
* At this point a few words about movement, there are three types and each
* one is done before the sprite is reprinted.
* ROTATION... the delay counter ($53F1) is increased every turn that the
* first character after the rotation speed number is NOT equal to #$2A or *
* until it reaches #$0C, then the sprite is rotated. Only the tens part of
* the number is used to set the speed so from 0 to 9 it rotates at its
* slowest but gradualy gets faster between 10 to 90 .
* HORIZONTAL..speed ranges from 0 to 999 but only 10 to 50 alters the speed.
* Above 50 the speed is set to its fastest below ten its slowest, the delay
* counter ($53F2) is increased each turn that the first character after the
* speed number is NOT #$2A or a * until it reaches #$05, when this is
* reached either the hundreds or the tens column number is used to reload
* the counter with the next value. So if the horizontal speed is over 50
* then the sprite moves every turn.
* VERTICAL... Again speed ranges from 0 to 999. Speed changes are made
* between 10 and 100, over 100 the sprite is moved at its fastest below 10
* at its slowest. The delay counter is increased to #$0A so a full ten speed
* changes are used.
* The rotation speed gradually reduced to zero and held there unless altered
* by joystick action, the horizontal speed is treated in the same way. The
* vertical speed is increased downwards by gravity unless countered by
* joystick action.
40D3-41DA This routine sets the flame sprite at the correct offset position
*         to the rocket sprite. As the rocket is moved or rotated the flame
*         sprite, wether visible or not, must always be adjacent to the
*         rockets motor.
40D3-40E9 For both size one and two sprites the flame sprite definitions are
*         placed in the same page as the rocket sprite (to save space)
*         flame one is from #$60 to #$8F, flame two is from #$90 to #$AF and
*         'NO' flame is from #$C0 to #$EF the latter area is filled with
*         zeros so that when the joystick is released the last flame sprite
*         is erased. Now load 'Y' reg. with 'NO' flame sprite offset, load
*         'A' reg. with $E8 (joystick return), see if it has been pushed
*         forward ($E8=#$01), if not branch to $40E8 else..... load 'Y' reg
*         with #$60 (first flame sprite offset) and increase $53F4 (this
*         registers first bit called bit zreo or the LSB, will alternate
*         between zreo or one depending wether the byte is odd or even),
*         shift all bits right one place putting the LSB into the carry flag
*         test the carry flag (carry set=1, carry flag clear=0), if set
*         branch to $40E8 else load 'Y' reg. with #$90, second flame sprite
*         offset (in this way the flame sprite can be made to flicker),
*         transfer 'Y' to 'A' reg. and save a copy on stack.
40EA-40F7 Load 'A' reg. with rocket sprite picture number mix in sprite
*         sprite group number (remember there are three sizes of rocket
*         sprite), to obtain sprite page number, store this value in the
*         sprite get pointers subroutine, load &A' reg. with sprite number
*         store it in $E5 and gosub get pointers.
40FA-40FC Pull offset back off stack transfer it to 'Y' reg. and gosub
*         transfer data from original page to working page areas.
40FF      Gosub move sprite to correct position within byte.
4102-4103 ROM/RAM toggle.
4106-4115 Load 'Y' reg. with rocket picture number, load 'A' reg. with
*         rocket position down the screen. Clear carry flag and add offset
*         indexed by 'Y' register, set carry flag and subtract the present
*         position of the flame sprite. What is left is the amount to be
*         moved up if negative or down if positive by the flame sprite to
*         obtain correct position, if zero branch to $412B if plus branch to
*         $4127 else.................
4117-4125 Transfer 'A' to 'X' and back again and save a copy on stack, gosub
*         move up one pixel, pull 'A' reg. off stack transfer it to 'X' reg
*         increase it by one branch back to $4118 if not zero else branch to
*         $412B.
4127-4128 Same as above but move down.
********************************************** CONTINUED ******************
```

```
**************************************************************************
412B-4141 This is the move left/right routine, a little bit more complicated
*         as this time two registers have to be used. The screen is 320
*         pixels across and this number is greater than can be held in one
*         eight bit byte. Load 'Y' reg with rocket picture number, load 'A'
*         reg. with offset, for some reason I have decided that instead of
*         using $$FF as minus one to use $$81, probably by the time i got
*         to here in the programme I did not trust my shaky maths and the
*         simple rule that any number with bit seven set means move left was
*         easier. So now we check bit seven with branch if minus to $4143
*         else... Clear carry flag, add rocket position low byte and store
*         it in $61, load rocket position high byte, add zero with carry
*         flag and store it $60. If by adding offset and rocket position low
*         the resulting number was rolled over $$FF the carry flag would be
*         set to one and this value would be added to position high and
*         stored in $60. Finally force branch to $4156.
4143-4154 Clear bit seven in offset byte and store it in $61, load 'A' reg.
*         with rocket position low byte, set carry flag and subtract $61,
*         then store result back into $61. Load 'A' reg. with rocket
*         position high byte and subtract with carry flag zero and store
*         result in $60.
4156-415C At this point $60 and $61 contain the amount that the flame sprite
*         has to move but, further problems arise if the rocket sprite was
*         rotated in its last move and at the same time is very near to the
*         point on the screen where its position moves from 00FF to 0100,
*         about two thirds across the screen, and to stay in correct
*         position the flame sprite will have to move from say, 00F8 to 0108
*         So first compare $DD to $60 that is the present flame sprite
*         position with the new one, if they are the same branch to $4187
*         if the result of the comparison is minus branch to $416F else....
415E-416D At this point we know the new flame position is to the left of its
*         present position also its present position is greater than 0100
*         so, load 'X' reg. with $DE, present position low byte branch if
*         zero to $4165 else move sprite left until its position is 0100.
*         Now load 'X' reg. with $61, this is the next amount we have to
*         move left, branch to $416C if zero or gosub move left and then
*         branch to $419B.
416F-4185 At this point we know that the flame sprite present position is
*         to the left of 0100 and it has to move right and it has to move
*         right to a position equal to or greater than 0100 (dec 256), this
*         routine is the reverse of the previous one.
4187-4195 At this point we know that neither the rocket nor the flame sprite
*         is near 0100. So load 'A' reg with $61 flame sprite new position
*         and subtract $DE flame sprite present position, branch to $419B if
*         result is zero. If the carry is still set branch to $4197 else...
*         If the carry was cleared, this means that the subtraction rolled
*         the 'A' reg. over $$FF, or in other words the value in $DE was
*         greater than the value in $61, transfer the result to the 'X' reg.
*         and move left that number of times and branch to 419B.
4197-4198 Transfer 'A' to 'X' reg. and move right that number of times.
419B-419E Print sprite and gosub save pointers.
41A1-41A4 ROM/RAM toggle.
41A5-41B1 Switch off rocket motor (voice two noise), load 'Y' reg. with
*         'beep' delay counter, decrease it by one and branch if not equal
*         to $41C5.
41B3-41C4 Switch on voice one 'beep', load 'A' reg. with delay quiet period
*         between 'beeps'), store it in $4EDF and return from gosub.
41C5-41DA Make sure voice one is off, store 'Y' reg. in 'beep' delay $4EDF.
*         Load 'A' reg. with $4EDF branch if plus to $41DA else reload
*         'beep' delay with $$40 and return from gosub.
* There are two sounds in this game, one is the rocket motor this is
* switched on when the joystick is moved forward ($E8=$$01) and is switched
* off again when the flame sprite is printed. The delay between switching
* off and on again should the joystick be held forward is hardly noticeable
* but it gives a better sound. The second one is the sound made by the
* (cheap) radar equipment installed by the company you are working for and
* in the final approach gradually increases in frequency.
********************************************** CONTINUED **********************
```

```
*********************************************************************************
41DB-41E1 End of game routine,load 'X' reg. with $$F0 and transfer it to
*         stack pointer to keep it tidy. gosub switch off hi-res and return
*         to lo-res then break programme.
41E2-41EE Print using BASIC CHAR routine, 'Y' reg. holds position down the
*         screen, 'X' reg. holds position across the screen, 'A' reg. holds
*         lenth of strin to be printed, $22 and $23 hold start position of
*         string in memory, JSR$BA2B is char command hi-res start point from
*         machine code.
41F0-4264 This is the get sprite pointers routine and as it is the same as
*         in 'BLOOPING BUG' I will not go over it again.
4265-4272 When moving from one screen to another this routine reprints the
*         rocket sprite $60, $61 and $62 have allready been set to the new
*         position.
4274-4275 RAM/ROM toggle.
4278-427C Store rocket sprite number in $E5 and get pointers.
427F-4285 Load 'X' reg. with $60 (this is the high byte of the position
*         across the screen), branch if equal to $4288 else load 'X' reg.
*         with $$FF an gosub move the sprite right this number of times.
4288-428A Load 'X' reg. with $61 (this is the low byte of the position
*         across the screen) and gosub move the sprite right this number of
*         times.
428D-4291 Load 'X' reg. with $62 (this is the position down the screen)
*         branch if equal to $4294 else move the sprite down the screen this
*         number of times.
4294-429E Gosub transfer $D2,$D3 to $D4,$D5 these are the pointers to the
*         top left hand byte of the sprite, again load $E5 with sprite
*         number and gosub save sprite pointers.
*         The routine just explained in effect moves the sprite without
*         printing it from the top left hand (home) position to werever it
*         is required on the new screen.
42A0-42D8 This routine is used to print the four status messages on the
*         screen. This is stored in one long string starting at $53B0, this
*         address is transfered to $22,$23 in lo-hi format, 'A','Y' and 'X'
*         registers hold the string lenth, position down and position across
*         respectively and $22 is updated each time to point to the next
*         part of the print string in memory. Gosub $41E2 has allready been
*         covered.
42D9-42F9 These are three move routines 'X' reg. holds the number of times
*         to perform the loop, the PHA and PLA commands are needed to keep
*         the 'X' reg intact as it is also used in the subroutines called by
*         these. The first is move left, the second move right ,the third
*         move down.
9000-9C00 To enter this memory dump type in M07F8 and change the 00 at $07F8
*         to 80 then key in F9000 9FFF 00 and press return then key in M9000
*         and press return, move the cursor over the first memory location
*         off you go. This is the data area which holds twelve of the first
*         sixteen small rocket sprites and their associated flame sprites.
*         Each rocket sprite definition starts at the begining of a page.
*         location 00 and runs on to 41, the first flame definition starts
*         at 60 and runs to 71, the second flame starts at 90 and runs to
*         A1, below this on each page is zeros and the erase flame sprite
*         runs from C0 to D1, at the foot of each page at F8 is a string
*         of figures these have no meaning in this programme and are only
*         used in the sprite creator programme so need not be entered.
*  And thats it for this month it only leaves me to say I hope you all
*  had a nice chrismass and to wish you all a happy new year.
*
*                              PETER CRACK.............
*
*
*********************************************************************************
```

```
. 4000  20 07 40 JSR $4007
. 4003  20 D3 40 JSR $40D3
. 4006  60       RTS
. 4007  AE BA 53 LDX $53BA
. 400A  E0 2A    CPX #$2A
. 400C  F0 3F    BEQ $404D
. 400E  AD B8 53 LDA $53B8
. 4011  29 0F    AND #$0F
. 4013  0A       ASL
. 4014  8D F0 53 STA $53F0
. 4017  AC F1 53 LDY $53F1
. 401A  C8       INY
. 401B  C0 0C    CPY #$0C
. 401D  B0 06    BCS $4025
. 401F  8C F1 53 STY $53F1
. 4022  38       SEC
. 4023  B0 28    BCS $404D
. 4025  AD F0 53 LDA $53F0
. 4028  8D F1 53 STA $53F1
. 402B  E0 4C    CPX #$4C
. 402D  D0 10    BNE $403F
. 402F  AC E7 53 LDY $53E7
. 4032  C8       INY
. 4033  C0 0E    CPY #$0E
. 4035  90 02    BCC $4039
. 4037  A0 00    LDY #$00
. 4039  8C E7 53 STY $53E7
. 403C  38       SEC
. 403D  B0 0B    BCS $404A
. 403F  AC E7 53 LDY $53E7
. 4042  88       DEY
. 4043  10 02    BPL $4047
. 4045  A0 0D    LDY #$0D
. 4047  8C E7 53 STY $53E7
. 404A  20 29 4A JSR $4A29
. 404D  78       SEI
. 404E  8D 3F FF STA $FF3F
. 4051  A9 00    LDA #$00
. 4053  85 E5    STA $E5
. 4055  20 F0 41 JSR $41F0
. 4058  AD DE 53 LDA $53DE
. 405B  C9 2A    CMP #$2A
. 405D  F0 32    BEQ $4091
. 405F  AD DA 53 LDA $53DA
. 4062  29 0F    AND #$0F
. 4064  F0 04    BEQ $406A
. 4066  A2 02    LDX #$02
. 4068  D0 14    BNE $407E
. 406A  AD DB 53 LDA $53DB
. 406D  29 0F    AND #$0F
. 406F  AA       TAX
. 4070  AC F2 53 LDY $53F2
. 4073  C8       INY
. 4074  C0 05    CPY #$05
. 4076  B0 06    BCS $407E
. 4078  8C F2 53 STY $53F2
. 407B  38       SEC
. 407C  B0 13    BCS $4091
. 407E  8E F2 53 STX $53F2
. 4081  AD DE 53 LDA $53DE
. 4084  C9 4C    CMP #$4C
. 4086  F0 06    BEQ $408E
. 4088  20 00 44 JSR $4400
. 408B  38       SEC
. 408C  B0 03    BCS $4091
. 408E  20 50 44 JSR $4450
```

```
. 4091  AD D2 53 LDA $53D2
. 4094  C9 2A    CMP #$2A
. 4096  F0 34    BEQ $40CC
. 4098  AD CE 53 LDA $53CE
. 409B  29 0F    AND #$0F
. 409D  F0 05    BEQ $40A4
. 409F  A2 09    LDX #$09
. 40A1  38       SEC
. 40A2  B0 15    BCS $40B9
. 40A4  AD CF 53 LDA $53CF
. 40A7  29 0F    AND #$0F
. 40A9  0A       ASL
. 40AA  AA       TAX
. 40AB  AC F3 53 LDY $53F3
. 40AE  C8       INY
. 40AF  C0 0A    CPY #$0A
. 40B1  B0 06    BCS $40B9
. 40B3  8C F3 53 STY $53F3
. 40B6  38       SEC
. 40B7  B0 13    BCS $40CC
. 40B9  8E F3 53 STX $53F3
. 40BC  AD D2 53 LDA $53D2
. 40BF  C9 44    CMP #$44
. 40C1  F0 06    BEQ $40C9
. 40C3  20 A0 44 JSR $44A0
. 40C6  38       SEC
. 40C7  B0 03    BCS $40CC
. 40C9  20 D8 44 JSR $44D8
. 40CC  20 00 45 JSR $4500
. 40CF  20 00 43 JSR $4300
. 40D2  60       RTS
. 40D3  A0 C0    LDY #$C0
. 40D5  A5 E8    LDA $E8
. 40D7  C9 01    CMP #$01
. 40D9  D0 0D    BNE $40E8
. 40DB  A0 60    LDY #$60
. 40DD  EE F4 53 INC $53F4
. 40E0  AD F4 53 LDA $53F4
. 40E3  4A       LSR
. 40E4  B0 02    BCS $40E8
. 40E6  A0 90    LDY #$90
. 40E8  98       TYA
. 40E9  48       PHA
. 40EA  AD E7 53 LDA $53E7
. 40ED  0D F6 53 ORA $53F6
. 40F0  8D 46 43 STA $4346
. 40F3  A9 01    LDA #$01
. 40F5  85 E5    STA $E5
. 40F7  20 F0 41 JSR $41F0
. 40FA  68       PLA
. 40FB  A8       TAY
. 40FC  20 42 43 JSR $4342
. 40FF  20 4C 4A JSR $4A4C
. 4102  78       SEI
. 4103  8D 3F FF STA $FF3F
. 4106  AC E7 53 LDY $53E7
. 4109  AD 0B 46 LDA $460B
. 410C  18       CLC
. 410D  79 E0 68 ADC $68E0,Y
. 4110  38       SEC
. 4111  E5 DF    SBC $DF
. 4113  F0 16    BEQ $412B
. 4115  10 10    BPL $4127
. 4117  AA       TAX
. 4118  8A       TXA
. 4119  48       PHA
```

```
. 411A  20 A0 44 JSR $44A0
. 411D  68       PLA
. 411E  AA       TAX
. 411F  E8       INX
. 4120  D0 F6    BNE $4118
. 4122  EA       NOP
. 4123  EA       NOP
. 4124  38       SEC
. 4125  B0 04    BCS $412B
. 4127  AA       TAX
. 4128  20 EF 42 JSR $42EF
. 412B  AC E7 53 LDY $53E7
. 412E  B9 F0 68 LDA $68F0,Y
. 4131  30 10    BMI $4143
. 4133  18       CLC
. 4134  6D 0A 46 ADC $460A
. 4137  85 61    STA $61
. 4139  AD 09 46 LDA $4609
. 413C  69 00    ADC #$00
. 413E  85 60    STA $60
. 4140  38       SEC
. 4141  B0 13    BCS $4156
. 4143  29 7F    AND #$7F
. 4145  85 61    STA $61
. 4147  AD 0A 46 LDA $460A
. 414A  38       SEC
. 414B  E5 61    SBC $61
. 414D  85 61    STA $61
. 414F  AD 09 46 LDA $4609
. 4152  E9 00    SBC #$00
. 4154  85 60    STA $60
. 4156  A5 DD    LDA $DD
. 4158  C5 60    CMP $60
. 415A  F0 2B    BEQ $4187
. 415C  30 11    BMI $416F
. 415E  A6 DE    LDX $DE
. 4160  F0 03    BEQ $4165
. 4162  20 E4 42 JSR $42E4
. 4165  A6 61    LDX $61
. 4167  F0 03    BEQ $416C
. 4169  20 37 43 JSR $4337
. 416C  38       SEC
. 416D  B0 2C    BCS $419B
. 416F  A6 DE    LDX $DE
. 4171  F0 0A    BEQ $417D
. 4173  8A       TXA
. 4174  48       PHA
. 4175  20 00 44 JSR $4400
. 4178  68       PLA
. 4179  AA       TAX
. 417A  E8       INX
. 417B  D0 F6    BNE $4173
. 417D  A6 61    LDX $61
. 417F  F0 03    BEQ $4184
. 4181  20 D9 42 JSR $42D9
. 4184  38       SEC
. 4185  B0 14    BCS $419B
. 4187  A5 61    LDA $61
. 4189  38       SEC
. 418A  E5 DE    SBC $DE
. 418C  F0 0D    BEQ $419B
. 418E  B0 07    BCS $4197
. 4190  AA       TAX
. 4191  20 37 43 JSR $4337
. 4194  38       SEC
. 4195  B0 04    BCS $419B
. 4197  AA       TAX
. 4198  20 D9 42 JSR $42D9
. 419B  20 00 45 JSR $4500
. 419E  20 00 43 JSR $4300
. 41A1  8D 3E FF STA $FF3E
. 41A4  58       CLI
. 41A5  AD 11 FF LDA $FF11
. 41A8  29 1F    AND #$1F
. 41AA  8D 11 FF STA $FF11
. 41AD  AC DF 4E LDY $4EDF
. 41B0  88       DEY
. 41B1  D0 12    BNE $41C5
. 41B3  AD 11 FF LDA $FF11
. 41B6  09 10    ORA #$10
. 41B8  8D 11 FF STA $FF11
. 41BB  AE D2 6B LDX $6BD2
. 41BE  BD D3 6B LDA $6BD3,X
. 41C1  8D DF 4E STA $4EDF
. 41C4  60       RTS
. 41C5  AD 11 FF LDA $FF11
. 41C8  29 EF    AND #$EF
. 41CA  8D 11 FF STA $FF11
. 41CD  8C DF 4E STY $4EDF
. 41D0  AD DF 4E LDA $4EDF
. 41D3  10 05    BPL $41DA
. 41D5  A9 40    LDA #$40
. 41D7  8D DF 4E STA $4EDF
. 41DA  60       RTS
. 41DB  A2 F0    LDX #$F0
. 41DD  9A       TXS
. 41DE  20 C9 C7 JSR $C7C9
. 41E1  00       BRK
. 41E2  8C DA 02 STY $02DA
. 41E5  8E DB 02 STX $02DB
. 41E8  8D EA 02 STA $02EA
. 41EB  20 2B BA JSR $BA2B
. 41EE  60       RTS
. 41EF  EA       NOP
. 41F0  EA       NOP
. 41F1  48       PHA
. 41F2  0A       ASL
. 41F3  0A       ASL
. 41F4  0A       ASL
. 41F5  0A       ASL
. 41F6  A8       TAY
. 41F7  18       CLC
. 41F8  69 0E    ADC #$0E
. 41FA  85 E4    STA $E4
. 41FC  A2 00    LDX #$00
. 41FE  B9 00 46 LDA $4600,Y
. 4201  95 D4    STA $D4,X
. 4203  E8       INX
. 4204  C8       INY
. 4205  C4 E4    CPY $E4
. 4207  D0 F5    BNE $41FE
. 4209  B9 00 46 LDA $4600,Y
. 420C  99 00 46 STA $4600,Y
. 420F  C8       INY
. 4210  B9 00 46 LDA $4600,Y
. 4213  99 00 46 STA $4600,Y
. 4216  A5 E1    LDA $E1
. 4218  EA       NOP
. 4219  8D 43 44 STA $4443
. 421C  8D 8F 44 STA $448F
. 421F  68       PLA
. 4220  0A       ASL
```

```
. 4221  0A       ASL
. 4222  0A       ASL
. 4223  A8       TAY
. 4224  B9 00 47 LDA $4700,Y
. 4227  8D 19 45 STA $4519
. 422A  8D 1C 44 STA $441C
. 422D  8D 21 44 STA $4421
. 4230  8D 6A 44 STA $446A
. 4233  8D 6F 44 STA $446F
. 4236  8D 48 45 STA $4548
. 4239  B9 01 47 LDA $4701,Y
. 423C  8D 4B 45 STA $454B
. 423F  8D 53 45 STA $4553
. 4242  8D 46 44 STA $4446
. 4245  8D 92 44 STA $4492
. 4248  B9 03 47 LDA $4703,Y
. 424B  85 E2    STA $E2
. 424D  B9 04 47 LDA $4704,Y
. 4250  85 E3    STA $E3
. 4252  A5 D4    LDA $D4
. 4254  85 D2    STA $D2
. 4256  A5 D5    LDA $D5
. 4258  85 D3    STA $D3
. 425A  B9 05 47 LDA $4705,Y
. 425D  85 E6    STA $E6
. 425F  B9 06 47 LDA $4706,Y
. 4262  85 E7    STA $E7
. 4264  60       RTS
. 4265  20 74 42 JSR $4274
. 4268  20 F0 41 JSR $41F0
. 426B  20 32 45 JSR $4532
. 426E  8D 3E FF STA $FF3E
. 4271  58       CLI
. 4272  60       RTS
. 4273  EA       NOP
. 4274  78       SEI
. 4275  8D 3F FF STA $FF3F
. 4278  A9 00    LDA #$00
. 427A  85 E5    STA $E5
. 427C  20 F0 41 JSR $41F0
. 427F  A6 60    LDX $60
. 4281  F0 05    BEQ $4288
. 4283  A2 FF    LDX #$FF
. 4285  20 D9 42 JSR $42D9
. 4288  A6 61    LDX $61
. 428A  20 D9 42 JSR $42D9
. 428D  A6 62    LDX $62
. 428F  F0 03    BEQ $4294
. 4291  20 EF 42 JSR $42EF
. 4294  20 AE 45 JSR $45AE
. 4297  A9 00    LDA #$00
. 4299  85 E5    STA $E5
. 429B  20 00 43 JSR $4300
. 429E  60       RTS
. 429F  EA       NOP
. 42A0  A9 53    LDA #$53
. 42A2  85 23    STA $23
. 42A4  A9 B0    LDA #$B0
. 42A6  85 22    STA $22
. 42A8  A9 0C    LDA #$0C
. 42AA  A0 0D    LDY #$0D
. 42AC  A2 17    LDX #$17
. 42AE  20 E2 41 JSR $41E2
. 42B1  A9 BC    LDA #$BC
. 42B3  85 22    STA $22
. 42B5  A9 0C    LDA #$0C
. 42B7  A2 18    LDX #$18
. 42B9  A0 0D    LDY #$0D
. 42BB  20 E2 41 JSR $41E2
. 42BE  A9 C8    LDA #$C8
. 42C0  85 22    STA $22
. 42C2  A9 0C    LDA #$0C
. 42C4  A2 17    LDX #$17
. 42C6  A0 1B    LDY #$1B
. 42C8  20 E2 41 JSR $41E2
. 42CB  A9 D4    LDA #$D4
. 42CD  85 22    STA $22
. 42CF  A9 0C    LDA #$0C
. 42D1  A2 18    LDX #$18
. 42D3  A0 1B    LDY #$1B
. 42D5  20 E2 41 JSR $41E2
. 42D8  60       RTS
. 42D9  8A       TXA
. 42DA  48       PHA
. 42DB  20 00 44 JSR $4400
. 42DE  68       PLA
. 42DF  AA       TAX
. 42E0  CA       DEX
. 42E1  D0 F6    BNE $42D9
. 42E3  60       RTS
. 42E4  8A       TXA
. 42E5  48       PHA
. 42E6  20 50 44 JSR $4450
. 42E9  68       PLA
. 42EA  AA       TAX
. 42EB  CA       DEX
. 42EC  D0 F6    BNE $42E4
. 42EE  60       RTS
. 42EF  8A       TXA
. 42F0  48       PHA
. 42F1  20 D8 44 JSR $44D8
. 42F4  68       PLA
. 42F5  AA       TAX
. 42F6  CA       DEX
. 42F7  D0 F6    BNE $42EF
. 42F9  60       RTS
. 42FA  00       BRK
. 42FB  00       BRK
. 42FC  00       BRK
. 42FD  00       BRK
. 42FE  00       BRK
. 42FF  00       BRK

>9000 00 03 C0 00 07 60 00 06 :..@..`..
>9008 20 00 07 20 00 0E 30 00 : .. ..0.
>9010 0E 30 00 0D 10 00 0F 10 :.0......
>9018 00 0C 10 00 0E 10 00 05 :........
>9020 20 00 06 20 00 0F 90 00 : .. ....
>9028 0F F0 00 0D B0 00 09 90 :.p..0...
>9030 00 00 F9 00 00 00 FF 00 :..y.....
>9038 00 00 DB 00 00 00 99 00 :..[.....
>9040 00 00 00 00 00 00 00 00 :........
>9048 00 00 00 00 00 00 00 00 :........
>9050 00 00 00 00 00 00 00 00 :........
>9058 00 00 00 00 00 00 00 00 :........
>9060 00 24 00 18 00 42 00 00 :.$...B..
>9068 00 24 00 10 00 08 00 00 :.$......
>9070 00 00 00 00 00 00 00 00 :........
>9078 00 00 00 00 00 00 00 00 :........
>9080 00 00 00 00 00 00 00 00 :........
>9088 00 00 00 00 00 00 00 00 :........
>9090 00 18 00 00 00 24 00 18 :.....$..
```

```
>9098 00 10 00 08 00 00 00 08 :........
>90A0 00 00 00 00 00 00 00 00 :........
>90A8 00 00 00 00 00 00 00 00 :........
>90B0 00 00 00 00 00 00 00 00 :........
>90B8 00 00 00 00 00 00 00 00 :........
>90C0 00 00 00 00 00 00 00 00 :........
>90C8 00 00 00 00 00 00 00 00 :........
>90D0 00 00 00 00 00 00 00 00 :........
>90D8 00 00 00 00 00 00 00 00 :........
>90E0 00 00 00 00 00 00 00 00 :........
>90E8 00 00 00 00 00 00 00 00 :........
>90F0 00 00 00 00 00 00 00 00 :........
>90F8 25 0F 07 03 08 03 02 B0 :%.......0
>9100 00 00 00 00 0F 00 00 1F :........
>9108 C0 00 1F 60 00 1C 10 00 :@..'....
>9110 1C 10 00 18 08 00 10 08 :........
>9118 00 08 08 00 08 08 00 04 :........
>9120 04 00 02 0E 00 01 0B 00 :........
>9128 01 C8 00 01 B0 00 00 80 :.H..0...
>9130 00 00 13 80 00 00 15 80 :........
>9138 00 00 1C 00 00 00 0C 00 :........
>9140 00 00 00 00 00 00 00 00 :........
>9148 00 00 00 00 00 00 00 00 :........
>9150 00 00 00 00 00 00 00 00 :........
>9158 00 00 00 00 00 00 00 00 :........
>9160 00 60 00 00 00 50 00 08 :.'...P..
>9168 00 10 00 08 00 00 00 04 :........
>9170 00 00 00 00 00 00 00 00 :........
>9178 00 00 00 00 00 00 00 00 :........
>9180 00 00 00 00 00 00 00 00 :........
>9188 00 00 00 00 00 00 00 00 :........
>9190 00 00 00 90 00 08 00 80 :........
>9198 00 24 00 10 00 04 00 00 :.$......
>91A0 00 00 00 00 00 00 00 00 :........
>91A8 00 00 00 00 00 00 00 00 :........
>91B0 00 00 00 00 00 00 00 00 :........
>91B8 00 00 00 00 00 00 00 00 :........
>91C0 00 00 00 00 00 00 00 00 :........
>91C8 00 00 00 00 00 00 00 00 :........
>91D0 00 00 00 00 00 00 00 00 :........
>91D8 00 00 00 00 00 00 00 00 :........
>91E0 00 00 00 00 00 00 00 00 :........
>91E8 00 00 00 00 00 00 00 00 :........
>91F0 00 00 00 00 00 00 00 00 :........
>91F8 25 0F 07 03 08 03 02 B1 :%.......1
>9200 00 00 00 00 3C 00 00 7F :....<...
>9208 00 00 7F 80 00 61 C0 00 :.....a@.
>9210 60 60 00 20 20 00 20 30 :''.  . 0
>9218 00 10 1C 00 08 06 00 06 :........
>9220 06 00 01 88 00 00 98 00 :........
>9228 00 E0 00 00 40 00 00 00 :.'..@...
>9230 00 00 07 40 00 00 07 00 :...@....
>9238 00 00 03 00 00 00 00 00 :........
>9240 00 00 00 00 00 00 00 00 :........
>9248 00 00 00 00 00 00 00 00 :........
>9250 00 00 00 00 00 00 00 00 :........
>9258 00 00 00 00 00 00 00 00 :........
>9260 00 C0 00 00 00 50 00 08 :.@...P..
>9268 00 10 00 04 00 00 00 01 :........
>9270 00 00 00 00 00 00 00 00 :........
>9278 00 00 00 00 00 00 00 00 :........
>9280 00 00 00 00 00 00 00 00 :........
>9288 00 00 00 00 00 00 00 00 :........
>9290 00 00 00 A0 00 08 00 90 :... ....
>9298 00 44 00 10 00 04 00 00 :.D......
>92A0 00 00 00 00 00 00 00 00 :........
```

```
>92A8 00 00 00 00 00 00 00 00 :........
>92B0 00 00 00 00 00 00 00 00 :........
>92B8 00 00 00 00 00 00 00 00 :........
>92C0 00 00 00 00 00 00 00 00 :........
>92C8 00 00 00 00 00 00 00 00 :........
>92D0 00 00 00 00 00 00 00 00 :........
>92D8 00 00 00 00 00 00 00 00 :........
>92E0 00 00 00 00 00 00 00 00 :........
>92E8 00 00 00 00 00 00 00 00 :........
>92F0 00 00 00 00 00 00 00 00 :........
>92F8 25 0F 07 03 08 03 02 B2 :%.......2
>9300 00 00 00 00 00 00 00 00 :........
>9308 00 00 3E 00 00 7F 80 00 :..>.....
>9310 7F C0 00 70 70 00 40 0F :@.pp.@.
>9318 00 20 03 00 20 04 00 18 :. .. ....
>9320 06 00 07 8E 00 00 48 00 :......H.
>9328 00 28 00 00 18 00 00 00 :.(.......
>9330 00 00 12 80 00 00 01 C0 :........@
>9338 00 00 00 00 00 00 00 00 :........
>9340 00 00 00 00 00 00 00 00 :........
>9348 00 00 00 00 00 00 00 00 :........
>9350 00 00 00 00 00 00 00 00 :........
>9358 00 00 00 00 00 00 00 00 :........
>9360 00 00 00 C0 00 90 00 08 :...@....
>9368 00 10 00 04 00 01 00 00 :........
>9370 00 00 00 00 00 00 00 00 :........
>9378 00 00 00 00 00 00 00 00 :........
>9380 00 00 00 00 00 00 00 00 :........
>9388 00 00 00 00 00 00 00 00 :........
>9390 00 00 00 00 00 40 00 A4 :.....@.$
>9398 00 08 00 22 00 00 00 00 :...".....
>93A0 00 00 00 00 00 00 00 00 :........
>93A8 00 00 00 00 00 00 00 00 :........
>93B0 00 00 00 00 00 00 00 00 :........
>93B8 00 00 00 00 00 00 00 00 :........
>93C0 00 00 00 00 00 00 00 00 :........
>93C8 00 00 00 00 00 00 00 00 :........
>93D0 00 00 00 00 00 00 00 00 :........
>93D8 00 00 00 00 00 00 00 00 :........
>93E0 00 00 00 00 00 00 00 00 :........
>93E8 00 00 00 00 00 00 00 00 :........
>93F0 00 00 00 00 00 00 00 00 :........
>93F8 25 0F 07 03 08 03 02 B3 :%.......3
>9400 00 00 00 00 00 00 00 00 :........
>9408 00 00 00 00 00 0F CF 00 :.......O.
>9410 7F 7E 00 FE 2C 00 E0 0F :~.~..'.
>9418 00 C0 07 00 C0 04 00 78 :.@..@..x
>9420 7E 00 0F CF 00 00 00 00 :~..O....
>9428 00 00 00 00 00 00 00 00 :........
>9430 00 00 00 00 00 00 00 00 :........
>9438 00 00 00 00 00 00 00 00 :........
>9440 00 00 00 00 00 00 00 00 :........
>9448 00 00 00 00 00 00 00 00 :........
>9450 00 00 00 00 00 00 00 00 :........
>9458 00 00 00 00 00 00 00 00 :........
>9460 00 00 00 00 00 20 00 8A :..... ..
>9468 00 91 00 24 00 00 00 00 :...$....
>9470 00 00 00 00 00 00 00 00 :........
>9478 00 00 00 00 00 00 00 00 :........
>9480 00 00 00 00 00 00 00 00 :........
>9488 00 00 00 00 00 00 00 00 :........
>9490 00 00 00 20 00 48 00 10 :... .H..
>9498 00 02 00 48 00 20 00 00 :...H. ..
>94A0 00 00 00 00 00 00 00 00 :........
>94A8 00 00 00 00 00 00 00 00 :........
>94B0 00 00 00 00 00 00 00 00 :........
```

18

```
>94B8 00 00 00 00 00 00 00 00 :........
>94C0 00 00 00 00 00 00 00 00 :........
>94C8 00 00 00 00 00 00 00 00 :........
>94D0 00 00 00 00 00 00 00 00 :........
>94D8 00 00 00 00 00 00 00 00 :........
>94E0 00 00 00 00 00 00 00 00 :........
>94E8 00 00 00 00 00 00 00 00 :........
>94F0 00 00 00 00 00 00 00 00 :........
>94F8 25 0F 07 03 08 03 02 B4 :%.......4
>9500 00 00 08 00 00 18 00 00 :........
>9508 38 00 03 FC 00 0F FE 00 :8..|..~.
>9510 1F 32 00 3C 04 00 38 07 :.2.<..8.
>9518 00 70 0E 00 70 10 00 70 :.p..p..p
>9520 20 00 60 C0 00 3F 00 00 : .'@.?..
>9528 00 00 00 00 00 00 00 00 :........
>9530 00 07 F0 00 00 00 00 00 :..p.....
>9538 00 00 00 00 00 00 00 00 :........
>9540 00 00 00 00 00 00 00 00 :........
>9548 00 00 00 00 00 00 00 00 :........
>9550 00 00 00 00 00 00 00 00 :........
>9558 00 00 00 00 00 00 00 00 :........
>9560 00 00 00 00 00 09 00 22 :.......".
>9568 00 00 00 84 00 A0 00 00 :..... ..
>9570 00 00 00 00 00 00 00 00 :........
>9578 00 00 00 00 00 00 00 00 :........
>9580 00 00 00 00 00 00 00 00 :........
>9588 00 00 00 00 00 00 00 00 :........
>9590 00 00 00 00 00 02 00 48 :.......H
>9598 00 10 00 48 00 00 00 00 :...H....
>95A0 00 00 00 00 00 00 00 00 :........
>95A8 00 00 00 00 00 00 00 00 :........
>95B0 00 00 00 00 00 00 00 00 :........
>95B8 00 00 00 00 00 00 00 00 :........
>95C0 00 00 00 00 00 00 00 00 :........
>95C8 00 00 00 00 00 00 00 00 :........
>95D0 00 00 00 00 00 00 00 00 :........
>95D8 00 00 00 00 00 00 00 00 :........
>95E0 00 00 00 00 00 00 00 00 :........
>95E8 00 00 00 00 00 00 00 00 :........
>95F0 00 00 00 00 00 00 00 00 :........
>95F8 25 0F 07 03 08 03 02 B5 :%.......5
>9600 00 00 00 00 00 60 00 00 :.....'..
>9608 E0 00 00 F8 00 01 F8 00 :'..x..x.
>9610 07 CC 00 0F C2 00 1F 0C :.L..B...
>9618 00 38 10 00 30 20 00 60 :.8..0 .'
>9620 20 00 40 40 00 40 80 00 : .@@.@..
>9628 43 00 00 3C 00 00 00 00 :C..<....
>9630 00 00 00 00 00 00 00 00 :........
>9638 00 00 00 00 00 00 00 00 :........
>9640 00 00 00 00 00 00 00 00 :........
>9648 00 00 00 00 00 00 00 00 :........
>9650 00 00 00 00 00 00 00 00 :........
>9658 00 00 00 00 00 00 00 00 :........
>9660 00 01 00 00 00 08 00 20 :........
>9668 00 00 00 08 00 A0 00 C0 :..... .@
>9670 00 00 00 00 00 00 00 00 :........
>9678 00 00 00 00 00 00 00 00 :........
>9680 00 00 00 00 00 00 00 00 :........
>9688 00 00 00 00 00 00 00 00 :........
>9690 00 00 00 01 00 02 00 08 :........
>9698 00 04 00 00 00 40 00 00 :.....@..
>96A0 00 00 00 00 00 00 00 00 :........
>96A8 00 00 00 00 00 00 00 00 :........
>96B0 00 00 00 00 00 00 00 00 :........
>96B8 00 00 00 00 00 00 00 00 :........
>96C0 00 00 00 00 00 00 00 00 :........
>96C8 00 00 00 00 00 00 00 00 :........
>96D0 00 00 00 00 00 00 00 00 :........
>96D8 00 00 00 00 00 00 00 00 :........
>96E0 00 00 00 00 00 00 00 00 :........
>96E8 00 00 00 00 00 00 00 00 :........
>96F0 00 00 00 00 00 00 00 00 :........
>96F8 25 0F 07 03 08 03 02 B6 :%.......6
>9700 00 00 09 90 00 0D B0 00 0F :.....0..
>9708 F0 00 0F B0 00 07 20 00 :p..0.. .
>9710 07 20 00 0F 30 00 0E 10 :. ..0...
>9718 00 0E 10 00 0E 10 00 0C :........
>9720 10 00 0C 30 00 04 20 00 :...0.. .
>9728 04 20 00 06 60 00 03 C0 :. ..'..@
>9730 00 00 84 00 00 00 84 00 :........
>9738 00 00 84 00 00 00 78 00 :......x.
>9740 00 00 00 00 00 00 00 00 :........
>9748 00 00 00 00 00 00 00 00 :........
>9750 00 00 00 00 00 00 00 00 :........
>9758 00 00 00 00 00 00 00 00 :........
>9760 00 10 00 00 00 10 00 08 :........
>9768 00 18 00 24 00 00 00 18 :...$....
>9770 00 00 00 00 00 00 00 00 :........
>9778 00 00 00 00 00 00 00 00 :........
>9780 00 00 00 00 00 00 00 00 :........
>9788 00 00 00 00 00 00 00 00 :........
>9790 00 00 00 10 00 08 00 24 :.......$
>9798 00 00 00 42 00 18 00 24 :...B...$
>97A0 00 00 00 00 00 00 00 00 :........
>97A8 00 00 00 00 00 00 00 00 :........
>97B0 00 00 00 00 00 00 00 00 :........
>97B8 00 00 00 00 00 00 00 00 :........
>97C0 00 00 00 00 00 00 00 00 :........
>97C8 00 00 00 00 00 00 00 00 :........
>97D0 00 00 00 00 00 00 00 00 :........
>97D8 00 00 00 00 00 00 00 00 :........
>97E0 00 00 00 00 00 00 00 00 :........
>97E8 00 00 00 00 00 00 00 00 :........
>97F0 00 00 00 00 00 00 00 00 :........
>97F8 25 0F 07 03 08 03 02 B7 :%.......7
>9800 00 00 00 00 00 02 00 00 07 :........
>9808 00 00 1F 00 00 1D 80 00 :........
>9810 78 E0 00 40 F0 00 30 38 :x'.@p.08
>9818 00 0E 04 00 06 04 00 06 :........
>9820 02 00 02 02 00 01 02 00 :........
>9828 00 C2 00 00 3C 00 00 00 :.B..<...
>9830 00 00 60 20 00 00 18 40 :..' ...@
>9838 00 00 07 80 00 00 00 00 :........
>9840 00 00 00 00 00 00 00 00 :........
>9848 00 00 00 00 00 00 00 00 :........
>9850 00 00 00 00 00 00 00 00 :........
>9858 00 00 00 00 00 00 00 00 :........
>9860 00 80 00 00 00 20 00 08 :..... ..
>9868 00 10 00 0A 00 00 00 03 :........
>9870 00 00 00 00 00 00 00 00 :........
>9878 00 00 00 00 00 00 00 00 :........
>9880 00 00 00 00 00 00 00 00 :........
>9888 00 00 00 00 00 00 00 00 :........
>9890 00 00 00 20 00 08 00 22 :... ..."
>9898 00 09 00 10 00 05 00 00 :........
>98A0 00 00 00 00 00 00 00 00 :........
>98A8 00 00 00 00 00 00 00 00 :........
>98B0 00 00 00 00 00 00 00 00 :........
>98B8 00 00 00 00 00 00 00 00 :........
>98C0 00 00 00 00 00 00 00 00 :........
>98C8 00 00 00 00 00 00 00 00 :........
>98D0 00 00 00 00 00 00 00 00 :........
```

```
>98D8 00 00 00 00 00 00 00 00 :........
>98E0 00 00 00 00 00 00 00 00 :........
>98E8 00 00 00 00 00 00 00 00 :........
>98F0 00 00 00 00 00 00 00 00 :........
>98F8 25 0F 07 03 08 03 02 B8 :%.......8
>9900 00 00 00 00 00 18 00 00 1C :........
>9908 00 00 1A 00 00 79 E0 00 :.....y`.
>9910 71 F8 00 23 E4 00 E3 84 :qx.#d.c.
>9918 00 F0 02 00 0E 02 00 02 :.p......
>9920 02 00 01 82 00 00 7C 00 :......|.
>9928 00 00 00 00 00 00 00 00 :........
>9930 00 00 70 60 00 00 0F 80 :..p`....
>9938 00 00 00 00 00 00 00 00 :........
>9940 00 00 00 00 00 00 00 00 :........
>9948 00 00 00 00 00 00 00 00 :........
>9950 00 00 00 00 00 00 00 00 :........
>9958 00 00 00 00 00 00 00 00 :........
>9960 00 00 00 80 00 20 00 08 :..... ..
>9968 00 10 00 09 00 03 00 00 :........
>9970 00 00 00 00 00 00 00 00 :........
>9978 00 00 00 00 00 00 00 00 :........
>9980 00 00 00 00 00 00 00 00 :........
>9988 00 00 00 00 00 00 00 00 :........
>9990 00 00 00 00 00 44 00 10 :.....D..
>9998 00 25 00 02 00 00 00 00 :.%......
>99A0 00 00 00 00 00 00 00 00 :........
>99A8 00 00 00 00 00 00 00 00 :........
>99B0 00 00 00 00 00 00 00 00 :........
>99B8 00 00 00 00 00 00 00 00 :........
>99C0 00 00 00 00 00 00 00 00 :........
>99C8 00 00 00 00 00 00 00 00 :........
>99D0 00 00 00 00 00 00 00 00 :........
>99D8 00 00 00 00 00 00 00 00 :........
>99E0 00 00 00 00 00 00 00 00 :........
>99E8 00 00 00 00 00 00 00 00 :........
>99F0 00 00 00 00 00 00 00 00 :........
>99F8 25 0F 07 03 08 03 02 B9 :%......9
>9A00 00 00 00 00 00 00 00 00 :........
>9A08 00 00 00 00 00 F3 F0 00 :.....sp.
>9A10 7F FE 00 3F E3 00 FE 81 :~.?c.~.
>9A18 00 F8 01 00 20 03 00 6C :.x.. ..l
>9A20 3E 00 F3 F0 00 00 00 00 :>.sp....
>9A28 00 00 00 00 00 00 00 00 :........
>9A30 00 00 00 00 00 00 00 00 :........
>9A38 00 00 00 00 00 00 00 00 :........
>9A40 00 00 00 00 00 00 00 00 :........
>9A48 00 00 00 00 00 00 00 00 :........
>9A50 00 00 00 00 00 00 00 00 :........
>9A58 00 00 00 00 00 00 00 00 :........
>9A60 00 00 00 04 00 12 00 40 :.......@
```

```
>9A68 00 08 00 12 00 04 00 00 :........
>9A70 00 00 00 00 00 00 00 00 :........
>9A78 00 00 00 00 00 00 00 00 :........
>9A80 00 00 00 00 00 00 00 00 :........
>9A88 00 00 00 00 00 00 00 00 :........
>9A90 00 00 00 00 00 24 00 89 :.....$..
>9A98 00 51 00 04 00 00 00 00 :.Q......
>9AA0 00 00 00 00 00 00 00 00 :........
>9AA8 00 00 00 00 00 00 00 00 :........
>9AB0 00 00 00 00 00 00 00 00 :........
>9AB8 00 00 00 00 00 00 00 00 :........
>9AC0 00 00 00 00 00 00 00 00 :........
>9AC8 00 00 00 00 00 00 00 00 :........
>9AD0 00 00 00 00 00 00 00 00 :........
>9AD8 00 00 00 00 00 00 00 00 :........
>9AE0 00 00 00 00 00 00 00 00 :........
>9AE8 00 00 00 00 00 00 00 00 :........
>9AF0 00 00 00 00 00 00 00 00 :........
>9AF8 25 0F 07 03 08 03 02 BA :%......:
>9B00 00 00 00 00 00 00 00 00 :........
>9B08 00 00 00 FC 00 03 FE 00 :...|..~.
>9B10 07 86 00 0E 02 00 7C 02 :......|.
>9B18 00 F8 04 00 30 0C 00 60 :.x..0..`
>9B20 08 00 60 30 00 3E C0 00 :..`0.>@.
>9B28 0C 00 00 18 00 00 10 00 :........
>9B30 00 00 D8 00 00 03 80 00 :..X.....
>9B38 00 00 00 00 00 00 00 00 :........
>9B40 00 00 00 00 00 00 00 00 :........
>9B48 00 00 00 00 00 00 00 00 :........
>9B50 00 00 00 00 00 00 00 00 :........
>9B58 00 00 00 00 00 00 00 00 :........
>9B60 00 00 00 05 00 21 00 00 :.....!..
>9B68 00 44 00 90 00 00 00 00 :.D......
>9B70 00 00 00 00 00 00 00 00 :........
>9B78 00 00 00 00 00 00 00 00 :........
>9B80 00 00 00 00 00 00 00 00 :........
>9B88 00 00 00 00 00 00 00 00 :........
>9B90 00 00 00 00 00 12 00 08 :........
>9B98 00 12 00 40 00 00 00 00 :...@....
>9BA0 00 00 00 00 00 00 00 00 :........
>9BA8 00 00 00 00 00 00 00 00 :........
>9BB0 00 00 00 00 00 00 00 00 :........
>9BB8 00 00 00 00 00 00 00 00 :........
>9BC0 00 00 00 00 00 00 00 00 :........
>9BC8 00 00 00 00 00 00 00 00 :........
>9BD0 00 00 00 00 00 00 00 00 :........
>9BD8 00 00 00 00 00 00 00 00 :........
>9BE0 00 00 00 00 00 00 00 00 :........
>9BE8 00 00 00 00 00 00 00 00 :........
>9BF0 00 00 00 00 00 00 00 00 :........
>9BF8 25 0F 07 03 08 03 02 BB :%......;
```

CONT
NEXT MONTH!

SOUND EFFECTS PROGRAM

```
10 SCNCLR
20 COLOR0,1:COLOR1,2:COLOR4,1
30 CHAR1,8,9,"SOUND EFFECTS PROGRAM"
40 CHAR1,8,10,"======================"
50 CHAR1,8,12,"ENTER YOUR SELECTION(1-4)-->"
55 CHAR1,8,13," OR 5 TO QUIT"
60 VOL8:INPUT X
65 IF X=1THEN90
70 IF X=2THEN135
75 IF X=3THEN160
80 IF X=4THEN180
85 IF X=5THEN END
90 REM ***RED ALERT***
95 SCNCLR:CHAR1,8,10,"■RED ALERT■
100 FORN=1TO5
110 SOUND3,1000,30
115 SOUND1,917,15
120 SOUND1,952,15
125 NEXT N
130 GOTO10
135 REM ***TELEPHONE***
140 SCNCLR:CHAR1,8,10,"■TELEPHONE■
145 FORA=1TO5:FORB=1TO2:FORC=1TO10
146 SOUND1,800,1:SOUND1,900,1
148 NEXT C
150 FORD=1TO100:NEXT D,B
155 FORD=1TO600:NEXT D,A
158 GOTO10
160 REM ***LASER***
165 SCNCLR:CHAR1,8,10,"■LASER■
166 FORN=1TO5
168 FORS=1000TO940 STEP-5
169 SOUND3,8,5
170 NEXT S,N
175 GOTO10
180 REM ***RACE CAR***
185 SCNCLR:CHAR1,8,10,"■RACE CAR■
186 FORN=1TO300
187 SOUND1,N,0
188 NEXT
189 FORN=1TO1000
190 SOUND1,300,0
200 NEXT
210 GOTO10

READY.
```

From
A.I.S Reppatu
Australia

**** PLUS-4 "SYS" & "G" PROGRAM STARTUP NUMBERS. ****

COMPILED BY

*** A.I.J-REDPATH, P.O BOX 26, AVOCA 3467, VICTORIA, AUSTRALIA.***

RSR = MEANS HOLD DOWN RUN/STOP AND PRESS RESET TO GET INTO "MONITOR" MODE.
(R) = PRESS RETURN AFTER EACH ENTERY.
SYS NUMBERS = STARTUP NUMBER AND IS TYPED IN AT THE READY PROMPT = SYS (R)
SYS NUMBERS = WHEN IN MONITOR. TYPE X. (R) BACK TO BASIC. TYPE = SYS (R)
G NUMBERS = STARTUP NUMBER AND IS TYPED IN WHILE STILL IN MONITOR = G (R)

WELL GET TO IT AND GOOD LUCK WITH YOUR SYS AND G NUMBERS.

ROCKMAN.	SYS4128.	ROKMAN-MONSTERS.	SYS7367.
ROBIN TO THE RESCUE.	SYS4112.OR 9984.OR G1010.		
KIKSTART.	SYS8192.OR G2008.OR G2000.	TERRA COGNITA.	SYS8192.
G-MAN.	SYS10240.	MONKEY MAGIC G1020 OR	SYS10608.
P.O.D.	SYS7936.OR G29CE.	GUN LAW. G1008 OR	SYS4120.OR G100D.
GHOST "N" GOBLIN.	G10F7.	POWERBALL.	SYS5751.
PUNCHY. SYS4112 OR	G1010.	SKYHAWK.	G1050.
BOOTY.	SYS4120.	SPEED KING.	SYS15794.
BIG MAC.	SYS7000.	RETURN OF ROCKMAN.	G1010.
MR PUNIVERSE.	G1AF4.	BUBBLE TROUBLE.	SYS5632.
TORPEDO ALLEY.	SYS4120.	STARFORCE NOVA.	SYS4448.
STREET OLYMPICS.	SYS15780.	DROID ONE.	G18CE.
TRAILBLAZER.	G2460.OR. G24C0.	BRIDGEHEAD (+4).	G2000.
AUTOZONE.	SYS6912.OR. G1B00.	AURIGA.	G1468.
BMX RACERS. RUN (R) OR	SYS8330.	MANIC MINER.	SYS10624.
COMANDO.	SYS4109.	XZAP.	SYS4346.
KANE. G1000 OR	SYS15860.	DANGER ZONE.	SYS5816.
SURVIVORS.	SYS11776.	AIRWOLF.	SYS7633.
WINNIE THE WITCH.	SYS4576.	KAKTUS.	SYS6240.
3D QUASERS.	SYS10240.	KNOCK OUT.	SYS8272.
THORST.	G0FF0.	FIRE ANT. G3F90.OR.	G1AF4.
FUTURE KNIGHT.	G2A3F.	BOUNDER.	SYS10608.
SPICY HAROLD.	SYS4096.	SQUIRM.	SYS9220.
DIRTY DEN.	SYS4112.	JET SET WILLY.	SYS10752.
BOMBJACK.	G3117.	SHARK.	SYS8192.
DORK'S DILEMA.	SYS6507.	PHANTOM. G0FE8 OR	G140.
SPACE PILOT.	G100D.	JOEY.	SYS4128.
CRAZY GOLF.	SYS12288.	XARGON WARS.	G1C80.
TYCOON TEX.	G1800.	KUNG FU KID.	G2000.
SPACE SWEEP.	SYS8192.	MAYHEM.	SYS4492.
ICICLE WORKS.	G1000.	ACTION FORCE.	SYS52992.
TUTTI FRUTTI.	SYS8192.	INVASION FORCE.	SYS8960.
SABOTEUR +4.	SYS30236,0.	AARDVARK.	SYS9458.
BANDITS AT ZERO.	SYS13284.	TAZZ.	SYS8005.
EXORCIST.	G3B00.	BERKS 3	G0FF0.
PLANET SEARCH.	G3182.	PIN POINT.	SYS4108.
KNOCK OUT.	SYS8272.	CHIP FACTORY.	SYS4111.
FORMULA 1 SIMULATOR.	SYS14988.	PROSPECTOR PETE.	SYS6144.
GULLWING FALCON.	G1000.	CHESS.	SYS65418.
GOLD RUSH	G1000	NETRUN (2000)	G1000
STREET OLYMPICS	SYS15780	BMAN & HIS DROID	SYS5273
OBLIDO	SYS4200		

CHEATS FOR THE PLUS/4 & C16

SQIJ:

Enter the monitor mode and type:
1929 60
X
RUN

PHEENIX:

Enter the monitor mode and type:
3532 60
G3ECa

ROCKMAN:

POKE 7409,173(Gets rid of monsters)
SYS 7367
G1010

COMMANDO:

POKE 11495,185
POKE 12707,185(enables you to shoot the enemy when they duck)
SYS 4109

BIG MACK:

POKE 12691,255(GIVES YOU 255 LIVES)
SYS 7000

SPACE PILOT:

1302 EA EA (FOR INFINITE LIVES)
G100D

GUN LAW:

A 1A3C NOP(RETURN TWICE)
G1018
SYS 4120(ELIMINATES ALL ENEMY)

FRANK BRUNO's BOXING:

FLING LONG CHOP:LBDEEZ
ANDRA PUNCHEREDOV:UATWIW

CHEATS FOR THE PLUS/4 & C16

TREASURE ISLAND:

LOAD,RESET AND TYPE:
POKE 1162,128(128 LIVES)
SYS 4109(STARTS THE GAME)

KICKSTART:

RUN/STOP RESET THEN TYPE THIS FOR ENDLESS MEN:
A28A3 NOP(RETURN TWICE)
G2003
SYS 8792

JACK ATTACK:

HOLD DOWN COMMODORE CTRL THEN TAP RETURN
PRESS FIRE AND SELECT ANY SCREEN BETWEEN ! AND %(.

VIDIEO MEANIES:

SYS 8330

ROCKMAN:

SYS4119

PUNCHY:

RUN/STOP RESET THEN TYPE THIS FOR ENDLESS MEN:
A 108A NOP
G1010

-/ (0

Mr T. M. Sexton
17,Grove Road North
Portsmouth
HANTS

0705-823470

Prestel 705823470

19/1/90

Dear Roy,

Please find enclosed a Program I have written entitled 'The Dialing Database' it is basically a telephone directory with individual records kept in a Relitive File (This makes access a lot quicker), in addition once you have the number (and if your telephone exchange has Tone Dialing) by Pressing (*) and Placing your telephone handset next to the speaker on your T.V. set the Program will automatically dial the number for you, when the number dialed is answered Press (T) and you have a stopwatch (in seconds, minutes and even hours for when the wife gets hold of the Phone), on completion of the call Press (C) and enter the call charge rate and then the distance rate and you will be told the cost of the call including V.A.T. (only use this function if you have a strong heart or a fat wallet (This counts me out on both counts).

Please feel free to Publish this Program in the magazine Roy although I will say its going to take a lot of typing in as it is 81 Blocks long and involves a lot of Title Pages and cursor up,down,left,right commands and you know how hard they can be to copy, If members are interested however ask them to send me £2 for the disc and postage (sorry as the Program uses relative files it can only be used with a Disc Drive) and Ill send them a working copy of the Prog.

Ill close now and look foreward to seeing you soon, all the best.

Tony Sexton

P.S. There is an instruction option in the Program if you dont understand this quick description.

If any members have a copy of the Script/Plus Cartridge manual for sale or a working copy of NUFONT or WORDPRO could you get them to drop me a line or Phone me at the above address, Thanks again Roy.

35 Burleigh Way,
Cuffley,
Herts,
EN6 4LG.

112 Cliff Road,
Hornsea,
N.Humberside,
HU18 1JE.

Dear Roy,

After our conversation on the phone a few days ago I realised that I had not sent off the money for the magazine so I enclose the 3 pounds for the triple issue.

I am a member of the Independent Commodore Products User Group (ICPUG) and I was wondering if you would like to have access to the clubs large Plus/4 public domain library.If you send me a blank disk I can get their catalogue disk.

What did you think of my Amiga P.D catalogue? I will send you the December issue as soon as we have had more printed.Sorry but we cannot distribute XXX p.d because of the recent clamping down on public domain companies selling software of this nature (N.B.S were recently taken to court over a certain under 18 who had been buying XXX from them).

On page 6 of the Aug/Sept issue there was a news article written by Matthew Newton-Lewis.I would like to correct some of the information printed-The new commodore games console the C64GS is basically a 64 in a new case with no keyboard.All the cartridges produces for the C64GS will work on existing 64's.
The A5000 mentioned is not infact made by Commodore but by Solid State Leisure.It is a processor/memory expansion board for the A500/A2000/A1500 that gives these computers a 68020 processor,68881/2 math coprocessor and up to 4Mb of super fast ram.The board gives a 500-600% speed increase over standard 68000 based Amigas.In simple terms it takes the standard Amigas nearly up to the level of the 68030 based A3000.
You may be interested to know that Commodore UK have released the new Amiga 1500 which is a rebadged A2000 with two 3.5 inch floppies instead of one 3.5 inch and one hard drive.

Well thats about it for now so I hope to hear from you soon.
Yours Sincerely

Daniel Stokes.

Daniel Stokes.

********** RS232 TRANSFER PROGRAMME **********
******** By Peter Crack with lots of help from YORK ELECTRONIC RESEARCH. ***
****

This routine can be used by both the C16 and the +4 although I have only
* run it on the +4, to transfer data from disc to printer via an RS232
* outlet, where the printer sends X on and X off signals, sometimes called
* DC1 and DC3, the values are usually hex 11 and hex 13 or dec 17 and dec 19
* respectively, BUT check this with your printer as mine uses hex 11 and hex
* 93 or dec 17 and dec 147, this has caused me great problems as the things
* just sit there if anything is not exactly right.
* To use this routine first write the letter article or file using the 3+1
* wordprocessor as normal and save it to disc as a sequential file (I think
* they all are anyway), note the name you have given to it, switch off the
* computer to clear it then switch it on again, load this programme,switch
* on the printer, check the baud rates (both the computers and the printers
* have to be the same) put your file disc into the drive and run this
* programme. All (I hope!) should now be printed. This programme is not
* all written by me as I have adapted a basic and M/C programme which came
* with the RS232 adaptor. It may also work using a printer on device 4 but
* I could not test it so I hope that a club member may do so, also I do not
* think that any of the printer control codes which can be used in the 3+1
* wordprocessor can be used as these would be changed in the character table
* set in 2300 to 237F, still it's worth a go,
* now for a line by line explanation.

2200-2202 Clear low res screen.

2205-2213 Prints using low res 'char' command title and set-up, $23 and $22
* contain start address of print string, 'A'register contains length
* of print string, 'X' register contains the row number and the 'Y'
* register contains the column number for the start of printing.

2216-2220 As above but prints input prompt.

2223-222F This moves the cursor first to the start of the next line down
* by printing carriage return, then one character right, then one
* line down, not very elegant I admit, if the final position is
* known then the kernal 'PLOT' routine I have used later is better.

2232-2240 This is the same as BASIC 'INPUT' command but the relevant
* characters, in this case, are stored in $4000 on, be careful with
* this command, in basic it stores all the characters on the screen
* and you can see what is happening using it this way it stores them
* where you say, in this case $4000 on and if you input 255
* characters by,say leaning on the keyboard while reading some
* instructions then a whole page will be entered overwriting
* anything that may have been there, not very likely I know but
* possible,so set aside a complete page if you can, when return is
* pressed that character is placed at the end of the string or at
* the 88th position, if you have gone mad, as with all BASIC
* 'FOR NEXT' loops the loop counter is increased at the end of the
* action and because we do not want to include the carriage return
* in the file name we reduce the 'Y' register by one and store it in
* $4000 for use later.

2243-225A Next we check the length of the file name just entered, if it is
* longer then dec 16 or hex $$10 then we move the cursor to the
* next line and print an error message, this (JSR$FFF0) is a more
* elegant way of moving the cursor X and Y contain the new row and
* column numbers setting the carry flag tells the computer to move
* the cursor while clearing the carry flag would return the cursor
* row and column positions in the X and Y registers respectively,
* else we branch to $2266.

225C-2264 Wait for the 'esc' key to be pressed and when it is branch to
* $2000 and start again, I use this key for two reasons first I am
* left handed and it is easy for me to use second it is not a key
* that would normally be used during a programme, this is important
* as the keyboard is not 'de-bounced' in other words hold a key too
* long and it will be repeated.

2266-226C SETLFS kernal routine 'A' register=file number (0-255 dec or $$00

********** CONTINUED **********

```
**********************************************************
*         -$$FF hex), 'X' register holds device number $$08 for disc,
*         'Y' register holds channel number $$00=load, $$01=save, $$0F=
*         command channel, so $$02 to $$0E can be used.
226F-2276 SETNAM kernal routine 'A' register holds length of filename,
*         'X' register holds start address of filename string low byte
*         'Y'  ''     ''    ''     '' ''     ''   ''  high byte
2279      Do open file kernal routine.
227C-2280 SETLFS for RS232 as above but 'Y' register (channel number) has no
*         meaning (probably is only one) so it must be set to $$FF.
2285-228B SETNAM 'A' register holds lenth of filename (always two).
*         'X' and 'Y' registers as previous SETNAM routine, first char of
*         filename is always control register value, lower four bits set
*         baud rate, second is command register and for RS232 is always set
*         to $$05, see pages 207 to 212 of computer user manual for details.
228E      Open routine.
2291-2297 These two registers hold the X on and X off values the are used by
*         the printer (in this case) to start and stop transmission of data
*         if the printer is using this system,and apparently not all do, the
*         printer will transmit back to the computer a character to start
*         the computer sending and one to stop, the trick however is to find
*         out what they are they should be hex 11 and hex 13 but in my case
*         they were hex 11 and hex 93 with the +4 the best way is to send
*         data at a faster rate then the printer can deal with, while at the
*         same time monitoring the value in dec 2005 or hex $07D5 this is
*         the temporary position for the last received byte, to do this from
*         BASIC,OPEN the printer as normal but at the highest baud rate your
*         printer will support, send any data but no carriage return
*         characters, this will save paper, also AFTER the last OPEN command
*         (this applies to M/C also, put hex 11 and hex 17 into $FC and $FD
*         to start with, you can change them to the returned values in a
*         later run,then include the command PRINT,PEEK(2005) in the print
*         loop, this will change from one value to another when the printer
*         buffer over flows, make a note of these two values and when you
*         enter this programme change $2292 to the first value and $2296 to
*         the value obtained when the buffer over flows, if you check the
*         routine from $EA95 to $EAF0 in ROM,this is the receive routine,you
*         can see where these two values are checked, and lastly remember
*         that $FC and $FD must be set after all OPEN routines have been
*         called as these routines set both $FC and $FD to zero.
229B-22A7 Set up and print using 'CHAR' routine 'SENDING.....'
22AA-22B1 Set disc channel to input and RS232 to output 'X' register holds
*         channel number.
22B4      BASIC 'GET' command, but using kernal routine get a byte from the
*         input channel.
22B7-22B9 When using 3+1 wordprocessor $$9F means return key has been
*         entered this is no good to the RS232 so.....
22BB-22BD Substitute a carriage return character and send it.
22C0      See further down.
22C3-22C9 Because the +4 character set does not conform to standard ASCII
*         we transfer the received character to the 'Y' register and use it
*         as an offset to obtain the correct character value in the table at
*         $2300 on, and send that instead. (this table is explained further
*         on).
22CC-22CD Force branch back to next 'GET' command.
22D0-22E9 Send subroutine.
22D0      transfer character to be sent to 'X' register for safekeeping.
22D1-22D4 Check to see if shift key has been pressed and branch to ABORT
*         routine if so, not strictly needed but handy if the printer tries
*         to unwind meters of paper. (this has happened to me).
22D8-22DD Check to see if the last character has been sent, bit 1 of this
*         register will be set to zero if the output buffer is full.
22DF-22E8 Load 'A' register with RS232 status byte and push it onto the
*         stack, transfer 'X' to 'A' registers, (this is the character to
*         be transmitted) and send it using the kernal print routine, pull
********************************************** CONTINUED ********************
```

```
*********************************************************************
*         the status byte back off stack and check to see if it is equal
*         if it has any value other than zero something went wrong,we do not
*         know what but branch to end transmission if so. else return.
22EA-22FF Close all files, clear all channels, and reset to screen and
*         keyboard as normal (default) I/O channels, pull the last return
*         address off the stack (keeps the pointer tidy) and print'ALL sent'
*         not strictly true if it has aborted with a fault but I have run
*         out of space, and jump to $2491.
2300-237A This is the ASCII set table which I have found to be of most use
*         character codes $$00 to $$1F are used for comms commands. so when
*         a lower case 'a' character is sent from a 3+1 file it has a value
*         of $$01 but in ASCII it should be $$61 this table puts this right.
23A0-23AC This routine stores the values in the correct registers for the
*         'char' low res print routine, and joins the BASIC 'char' routine
*         $BA0D, for high res it is the same but joins at $BA2B.
23B0-2478 Screen message strings.
2450-2451 This is the filename for the RS232 device, the first byte $$1C is
*         the control register value, the three rightmost bits set parity
*         checks and are only of use if the transmission line are subjet to
*         interference such as telephone lines, these are set to zero, bit
*         four must be set to 1, this is the receiver clock source bit
*         (whatever that means), bits 321 and 0 set the baud rate (see the
*         +4 user manual for an explanation of setting the rates, baud not
*         POLL TAX that is), the second byte is the command register and
*         must always be set to $$05 (thats what York Electronic Research
*         say, and it works!!!!!).
2480-248E This routine is required because a 3+1 W/P line is only 77 chars
*         long but no matter where you put the carriage return character
*         each line is stored in memory as 80 characters, so having sent a
*         C/R character we now discard all characters of less than hex 21 or
*         dec 33 and only begin sending again when a character of greater
*         value is received, this is why I always start a line with either
*         a number, a capital letter or a multiply sign. The status byte is
*         also checked because this changes in value when the last character
*         of a file is received from the disc, if it is the last character
*         then jump to $22EA else return to main print routine.
2491-249B And finally close all files , 'X' register holds file numbers, and
*         end programme.
*         O.K. well thats it, as always any problems just tell me or write
*         in to the MAGAZINE.
*                                   PETER CRACK.............
```

```
. 2200  A9 93     LDA #$93
. 2202  20 D2 FF  JSR $FFD2
. 2205  A9 23     LDA #$23
. 2207  85 23     STA $23
. 2209  A9 B0     LDA #$B0
. 220B  85 22     STA $22
. 220D  A9 40     LDA #$40
. 220F  A0 01     LDY #$01
. 2211  A2 01     LDX #$01
. 2213  20 A0 23  JSR $23A0
. 2216  A9 F0     LDA #$F0
. 2218  85 22     STA $22
. 221A  A9 48     LDA #$48
. 221C  A0 01     LDY #$01
. 221E  A2 04     LDX #$04
. 2220  20 A0 23  JSR $23A0
. 2223  A9 0D     LDA #$0D
. 2225  20 D2 FF  JSR $FFD2
. 2228  A9 1D     LDA #$1D
. 222A  20 D2 FF  JSR $FFD2
. 222D  A9 11     LDA #$11
. 222F  20 D2 FF  JSR $FFD2
. 2232  A0 00     LDY #$00
. 2234  20 CF FF  JSR $FFCF
. 2237  99 01 40  STA $4001,Y
. 223A  C8        INY
. 223B  C9 0D     CMP #$0D
. 223D  D0 F5     BNE $2234
. 223F  88        DEY
. 2240  8C 00 40  STY $4000
. 2243  C0 11     CPY #$11
. 2245  90 1F     BCC $2266
. 2247  A2 0A     LDX #$0A
. 2249  A0 00     LDY #$00
. 224B  18        CLC
. 224C  20 F0 FF  JSR $FFF0
. 224F  A0 00     LDY #$00
. 2251  B9 38 24  LDA $2438,Y
. 2254  20 D2 FF  JSR $FFD2
. 2257  C8        INY
. 2258  C0 17     CPY #$17
. 225A  D0 F5     BNE $2251
. 225C  20 E4 FF  JSR $FFE4
. 225F  C9 1B     CMP #$1B
. 2261  D0 F9     BNE $225C
. 2263  38        SEC
. 2264  B0 9A     BCS $2200
. 2266  A9 08     LDA #$08
. 2268  A2 08     LDX #$08
. 226A  A0 08     LDY #$08
. 226C  20 BA FF  JSR $FFBA
. 226F  AD 00 40  LDA $4000
. 2272  A2 01     LDX #$01
. 2274  A0 40     LDY #$40
. 2276  20 BD FF  JSR $FFBD
. 2279  20 C0 FF  JSR $FFC0
. 227C  A9 02     LDA #$02
. 227E  A2 02     LDX #$02
. 2280  A0 FF     LDY #$FF
. 2282  20 BA FF  JSR $FFBA
. 2285  A9 02     LDA #$02
. 2287  A2 50     LDX #$50
. 2289  A0 24     LDY #$24
. 228B  20 BD FF  JSR $FFBD
. 228E  20 C0 FF  JSR $FFC0
. 2291  A9 11     LDA #$11
```

```
. 2293  85 FC     STA $FC
. 2295  A9 93     LDA #$93
. 2297  85 FD     STA $FD
. 2299  A9 58     LDA #$58
. 229B  85 22     STA $22
. 229D  A9 24     LDA #$24
. 229F  85 23     STA $23
. 22A1  A9 0F     LDA #$0F
. 22A3  A2 0C     LDX #$0C
. 22A5  A0 01     LDY #$01
. 22A7  20 A0 23  JSR $23A0
. 22AA  A2 08     LDX #$08
. 22AC  20 C6 FF  JSR $FFC6
. 22AF  A2 02     LDX #$02
. 22B1  20 C9 FF  JSR $FFC9
. 22B4  20 CF FF  JSR $FFCF
. 22B7  C9 9F     CMP #$9F
. 22B9  D0 08     BNE $22C3
. 22BB  A9 0D     LDA #$0D
. 22BD  20 D0 22  JSR $22D0
. 22C0  20 B0 24  JSR $2480
. 22C3  A8        TAY
. 22C4  B9 00 23  LDA $2300,Y
. 22C7  EA        NOP
. 22C8  EA        NOP
. 22C9  20 D0 22  JSR $22D0
. 22CC  38        SEC
. 22CD  B0 E5     BCS $22B4
. 22CF  EA        NOP
. 22D0  AA        TAX
. 22D1  AD 43 05  LDA $0543
. 22D4  29 01     AND #$01
. 22D6  D0 12     BNE $22EA
. 22D8  AD 10 FD  LDA $FD10
. 22DB  29 02     AND #$02
. 22DD  F0 F2     BEQ $22D1
. 22DF  A5 90     LDA $90
. 22E1  48        PHA
. 22E2  8A        TXA
. 22E3  20 D2 FF  JSR $FFD2
. 22E6  68        PLA
. 22E7  D0 01     BNE $22EA
. 22E9  60        RTS
. 22EA  20 CC FF  JSR $FFCC
. 22ED  68        PLA
. 22EE  68        PLA
. 22EF  A9 68     LDA #$68
. 22F1  85 22     STA $22
. 22F3  A9 10     LDA #$10
. 22F5  A2 0E     LDX #$0E
. 22F7  A0 01     LDY #$01
. 22F9  20 A0 23  JSR $23A0
. 22FC  4C 91 24  JMP $2491
. 22FF  EA        NOP

. 23A0  8D EA 02  STA $02EA
. 23A3  8E DB 02  STX $02DB
. 23A6  8C DA 02  STY $02DA
. 23A9  20 0D BA  JSR $BA0D
. 23AC  60        RTS
. 23AD  EA        NOP

. 2480  20 CF FF  JSR $FFCF
. 2483  AA        TAX
. 2484  A5 90     LDA $90
. 2486  D0 06     BNE $248E
```

```
. 2488  8A       TXA
. 2489  C9 21    CMP #$21
. 248B  90 F3    BCC $2480
. 248D  60       RTS
. 248E  4C EA 22 JMP $22EA
. 2491  A9 08    LDA #$08
. 2493  20 C3 FF JSR $FFC3
. 2496  A9 02    LDA #$02
. 2498  20 C3 FF JSR $FFC3
. 249B  00       BRK
. 249C  4D 4D 4D EOR $4D4D
```

```
>2300 20 61 62 63 64 65 66 67 : abcdefg
>2308 68 69 6A 6B 6C 6D 6E 6F :hijklmno
>2310 70 71 72 73 74 75 76 77 :pqrstuvw
>2318 78 79 7A 20 20 20 20 20 :xyz
>2320 20 21 27 23 24 25 26 27 : !'#$%&'
>2328 28 29 2A 2B 2C 2D 2E 2F :()*+,-./
>2330 30 31 32 33 34 35 36 37 :01234567
>2338 38 39 3A 3B 3C 3D 3E 3F :89:;<=>?
>2340 40 41 42 43 44 45 46 47 :@ABCDEFG
>2348 48 49 4A 4B 4C 4D 4E 4F :HIJKLMNO
>2350 50 51 52 53 54 55 56 57 :PQRSTUVW
>2358 58 59 5A 5B 5C 5D 5E 5F :XYZ[\]↑_
>2360 60 61 62 63 64 65 66 67 :`abcdefg
>2368 68 69 6A 6B 6C 6D 6E 6F :hijklmno
>2370 70 71 72 73 74 75 76 77 :pqrstuvw
```

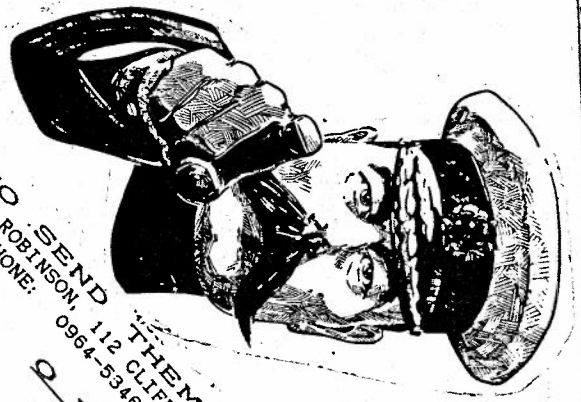
```
>2378 78 79 7A 20 20 20 20 20 :xyz
>2380 20 20 20 20 20 20 20 20 :

>2388 52 53 32 33 32 20 54 52 :RS232 TR
>2390 41 4E 53 46 45 52 20 50 :ANSFER P
>2398 52 4F 47 52 41 4D 4D 45 :ROGRAMME
>23A0 20 41 54 20 34 38 30 30 : AT 4800
>23A8 20 42 41 55 44 20 20 20 : BAUD
>23B0 20 20 20 20 20 20 58 20 :      X
>23B8 4F 4E 3D 31 31 20 20 58 :ON=11  X
>23C0 20 4F 46 46 3D 39 33 2E : OFF=93.
>23C8 50 4C 45 41 53 45 20 45 :PLEASE E
>23D0 4E 54 45 52 20 46 49 4C :NTER FIL
>23D8 45 20 4E 41 4D 45 2C 20 :E NAME,
>23E0 20 20 20 20 20 20 20 20 :
>23E8 20 20 20 20 20 20 20 20 :
>23F0 53 49 58 54 45 45 4E 20 :SIXTEEN
>23F8 43 48 41 52 41 43 54 45 :CHARACTE
>2400 52 53 20 4D 41 58 49 4D :RS MAXIM
>2408 55 4D 20 20 20 20 20 20 :UM
>2410 20 46 49 4C 45 20 4E 41 : FILE NA
>2418 4D 45 20 54 4F 4F 20 4C :ME TOO L
>2420 4F 4E 47 2E 20 20 20 20 :ONG.
>2428 1C 05 FF 00 00 00 00 00 :........
>2430 53 45 4E 44 49 4E 47 2E :SENDING.
>2438 2E 2E 2E 2E 2E 2E 2E 2E :........
>2440 2E 2E 2E 2E 2E 2E 2E 2E :........
>2448 41 4C 4C 20 53 45 4E 54 :ALL SENT
```

END

IN FACT ANYTHING TO DO WITH THE C16/C116/+4 IS MOST WELCOME.

REVIEWS PROGRAMS



SO SEND THEM TO:—
ROY ROBINSON, 112 CLIFF ROAD, HORNSEA N. HUMBERSIDE, HU18 1JE.
TELEPHONE: 0964-534611 after 6pm but not on Friday Night.

OR ELSE!!!

THIS MAGAZINE NEEDS YOUR ARTICLES!!

```
**************************************************************************
***************************** TRANSFER DATA ******************************
********** FROM DISC TO PRINTER VIA R.S.232 OR DEVICE 4 **********************
***************************** BY PETER CRACK *****************************
**************************************************************************
*     This programme complements the first transfer programme by printing
* out the memory or dissassembly of any M/C programme you wish to write also
* it prints it out in 84 column and A4 size page format, so you no longer
* have to work out how many lines each listing will take up.
* To use this programme you have to have a printer and a disc drive,I feel
* pretty sure it will work using a device 4 printer but I cannot test this
* so I will leave that up to you. Having written yor programme save a copy
* onto disc as normal then list your programme using the 'D' and 'M'
* commands as normal now make a list of all the areas you wish to either
* dissassemble 'D' or memory dump 'M' commands, (writing down the start and
* end addresses as you go (exactly as you would to list them to the screen)
* now enter X and press return, in imediate mode that is to say without
* line numbers enter the following, OPEN,8,8,8,'filename,s,w' and press
* return (filename can be anything short), now enter CMD8 and press return
* (this will send all screen output to the disc file) now enter MONITOR
* and press return enter the list of locations as you have written them
* pressing return after each complete command when all done enter X and
* press return,and finaly enter CLOSE8 and press return, this last sends all
* the characters remaining and closes the file, switch off the computer
* count to ten and pray, switch the computer on and load this programme,
* make sure the printer is ready,place the file disc into the drive and run
* the programme from MONITOR with the command G1300, when prompted enter
* the name of the file you have just created and watch it being printed
* GOOD LUCK!!!!!          here is the listing and how it runs.
1300-130C Clear the screen and set $D0 to $E7 to $$00.
130E-131C Set registers for print message routine and print message $22 and
*         $23 hold address of message lo-hi format,'A' reg. holds length of
*         message string,'Y' reg. holds position across the screen and 'X'
*         reg. holds position down the screen.
131F-132D Input routine for getting filename (be careful because the
*         JSR$FFCF command will over write up to 255 bytes so always leave
*         a page clear for input or as I have done store the input above the
*         the main programme), in this case the file name will be stored at
*         $1601 on,this programme is not error trapped so do not attempt to
*         enter more then 16 characters as the programme will crash and you
*         will have to start again. The length of the file name is stored in
*         $1600.
1330-1336 Load 'X' register with device value, load 'A' register with file
*         number, load 'Y' register with channel number and gosub SETLFS.
1339-1340 Load 'A' reg. with filename length,'X' and 'Y' registers hold file
*         name string start position in lo-hi format and gosub SETNAM.
1343      Gosub OPEN routine.
1346-134C As first SETLFS but for R.S.232 this will have to be changed for
*         device 4 printers.
134F-1355 This sets filename for R.S.232 and may have to be changed for
*         device 4 printers, in this case it is the same as the first SETNAM
*         routine but the first byte sets the control register (baud rates
*         and parity checking)  the second the comand register (always set
*         to $$05) of the 6551 ACIA chip.
1358      Gosub OPEN.
135B-1363 Set Xoff and Xon codes to match your printer, these may be
*         different from the ones I have used so please check.
1364-137C When OPENING the sequential file from BASIC we put in one or two
*         commands which produced a screen reply these of course were sent
*         to the disc drive and are now at the front of your file by
*         looking at this file I have found that nine lines have been sent
*         so, by stripping off nine $$0D characters I can position the read
*         head over the first character we wish to print out, so clear $E0
*         and set disc channel to input, get a character from channel,is
*         $$0D no, then get another, else increase $$E0 and check to see if
*         is equal to $$09 if yes then reset I/O channels else do it again.
************************************************** CONTINUED **********
```

```
*************************************************************************
1380-13B4 Prepares $2000 to $35B0 to receive print lines in 84 column format
1380-138C Set $D0 and $D1 to $2000 in lo-hi format,load 'X' register with
*         the number of lines we wish to have printed $$41=dec66, clear 'Y'
*         register and load 'A' register with the space character value.
138E-1391 Print 83 spaces.
1393-1397 Make sure 83 spaces have been printed and then print a carriage
*         return character.
1399-13A2 Increase $D0 and $D1 by $$54 to set them to the begining of the
*         next line.
13A4-13A5 Decrease 'X' register and see if it is still positive if yes
*         branch to $138A and print another line else.......
13A7-13B4 At this point we have prepared 66 lines of 84 space characters now
*         we print 4 cariage return characters to make the printer feed four
*         empty lines giving us a clear page break and a end of page flag
*         $$03 this is to tell the send routine that the end of a page has
*         been reached.
13B6-13C4 Another print message routine same as the one at $130E but of
*         course a different message string.
13C7-13CD Set $D9 and $DA to $35B0 start of data input area.
13CF-13D1 Set disc channel to input.
13D4-1400 Input from disc routine, this routine will input up to 132 lines
*         of data.
13D4-13D7 Get one character from input channel and transfer it to 'X' reg.
13D8-13DA Get status byte and store it in $12BF,this will be non zero at EOF
13DD-13E0 Transfer 'X' to 'A' registers and store input character in data
*         area offset by 'Y' register.
13E2-13E5 Retreive status byte and check to see if it is zero if no then
*         branch to $1402 (either something has gone wrong with the data
*         transmission or the end of the file has been reached) else.......
13E7-13EA Check to see if last input character was a carriage return if not
*         branch to $13F4 else.....................
13EC-13F2 Increase $D5 ($D5 was cleared at the start of the programme and
*         will be cleared again in the $12A0 subroutine so it is always set
*         to zero at the start of this routine) and check to see if it has
*         reached $$84 or dec 132 if yes then branch to $1402 else.........
13F4-1400 Increase $D9 by one and $DA by one if $D9 is rolled over $$FF and
*         branch back to $13D4 for another input character.
1402-1407 At this point $D5 contains $$84 or dec132 that is to say a full
*         page of A4 size in 40 column or, if the end of the file has been
*         reached as many lines as were left, so to change this into 84
*         column we first transfer the full number to $D8 then divide it by
*         two (LSR) and store this value in $D6 ($D6 now contains the number
*         of lines to be printed down the left hand side of the page and $D8
*         the number of lines down the right hand side).
1409-144C This routine takes the first group of lines from $35B0 onwards and
*         puts them into the left hand end of each of the A4 page lines,
*         starting at $2000, the pointers and flags used are as follows 'X'
*         register is used as the total number of lines counter, $D5 the
*         number of characters placed on each line before a carriage return
*         is issued, $D0,$D1 start position of left hand column in each line
*         $D2,$D3 start position of right hand column in each line, $D9,$DA
*         start of input data area from which lines of characters are taken,
*         $E0 temp for storing 'Y' register during routine.
1409-1421 Set flags and registers to start values.
1423      Get first character, 'Y' register is offset to point to position
*         within page.
1425-1427 During a memory screen dump a $$12 value character is used to set
*         the rightmost eight characters to reverse field this does not work
*         on my printer so discard this character and branch to next one.
1429-142B Is this character a carriage return if not branch to $143D else...
142D-1436 Save 'X' register on stack, remember this is the line counter and
*         must not be corrupted throughout the routine, load 'X' reg. with
*         correct offset for the increase line subroutine, (explained later)
*         gosub increase line pointers, retreive 'X' register and increase
*         it by one.
********************************************************* CONTINUED *********
```

1437-1439 Check to see if 'X' register has reached the number of lines
* required for the right hand side of the page if yes branch to
* $144E else branch to $1447
143D-1445 Store the get 'Y' register in $EØ and load 'Y' register with $D5
* this is the character per line offset for the store part of the
* routine, store the character in its correct place, increase $D5,
* retreive the original value of 'Y' register from $EØ.
1447-144C Increase 'Y' register by one (this is the offset value for the get
* part of the routine at $1423), and check to see if it has rolled
* over #$FF or dec255, if it has then increase $DA and branch back
* to $1423 else just branch to $1423.
144E-1478 At this point we have completed the left hand side of the page, so
* we increase 'Y' reg by one to move the get routine to the next
* character and use $D2,$D3 to position the character at the right
* hand side of the line (remember $D2 and $D3 were set to $2Ø2C at
* start) this routine runs exactly as the first except that the 'X'
* register now starts from the value set in $D6 and goes on to the
* value set in $D8 and at $145B the 'X' register is set to #$Ø2 so
* that $D2 and $D3 are updated to the next line.
147A-1486 At this point the whole page has been completed and just in case
* it is not a complete A4 size we put four carriage returns and an
* end of page marker (#$Ø3) at the end.
1488-1495 Close all channels and print onto the screen a new message.
1498-14A4 Set pointers for start of send routine, $EØ is the offset for the
* get command at $14A9, $DØ and $D1 are set to the start of the
* assembled A4 page, load 'X' register with the R.S.232 channel
* number, do set channel to output.
14A7-14A9 Set 'Y' register to current value in $EØ and get the first
* character to be sent to the printer.
14AB-14AD Is it the end of page marker? (#$Ø3) if yes then branch to $14D1
* else.....
14AF-14B4 Increase 'Y' register to point to next get character, and check
* to see if it has rolled over #$FF if yes then increase $D1 else
* branch to $14B4 and store 'Y' reg. in $EØ.
14B6-14C3 First put the get character into the 'X' register for safe keeping
14B6-14C3 then check $Ø543 to see if the shift key has been pressed if yes
* branch to $14D9 and end transmission, else check to see if the
* output buffer is empty (has the last char been transmitted?) if no
* then branch back to $14B7 else.....
14C5-14CF Load 'A' register with status byte and push it onto the stack,
* transfer our get character from the 'X' reg to the 'A' reg and
* gosub transmit, lastly we pull the status byte back off the stack
* and check to see if it is zero, if it is not then something has
* wrong with the transmission so branch to $14D9 and end programme.
* else branch back to $14A9 and get the next character.(at this
* point you may have noticed that as we do not go back far enough
* in the programme to reload the 'Y' register with $EØ that $EØ is
* redundant and you would be right,this is something that I have
* only just noticed but as the programme runs O.K. as it is I am
* going to leave it in.
14CF-14D6 Now check the status byte for the input routine (set at $13DA)
* if it is zero then another page has still to be sent from the disc
* so jump to $137C else $12BF contains the end of file marker and we
* branch to $14D9.
14D9-14E6 Reset all I/O channels,close disc file , close printer file and
* end programme.
1180-129F Screen print messages.
12BD-12BC Control and command register values for R.S.232 output, these two
* bytes make up the filename.
12A0-12AF This routine adds #$54 dec84 to the line pointers,by changing the
* value of the 'X' register either $DØ,$D1 or $D2,$D3 can be updated
* and clear the character position within the line counter ($D5).
12B0-12BC This routine places the 'CHAR' print pointers in their correct
* positions and gosub print.
* Thats it, this routine should work on both C16 and +4 computers
* if you have any problems or hints please phone me or write to the
* magazine.....PETER CRACK. phone Ø81-367-3152.....

```
. 1300  A9 93     LDA #$93
. 1302  20 D2 FF  JSR $FFD2
. 1305  A2 17     LDX #$17
. 1307  A9 00     LDA #$00
. 1309  95 D0     STA $D0,X
. 130B  CA        DEX
. 130C  10 FB     BPL $1309
. 130E  A9 11     LDA #$11
. 1310  85 23     STA $23
. 1312  A9 80     LDA #$80
. 1314  85 22     STA $22
. 1316  A9 E9     LDA #$E9
. 1318  A0 01     LDY #$01
. 131A  A2 01     LDX #$01
. 131C  20 B0 12  JSR $12B0
. 131F  A2 00     LDX #$00
. 1321  20 CF FF  JSR $FFCF
. 1324  E8        INX
. 1325  9D 00 16  STA $1600,X
. 1328  C9 0D     CMP #$0D
. 132A  D0 F5     BNE $1321
. 132C  CA        DEX
. 132D  8E 00 16  STX $1600
. 1330  A2 08     LDX #$08
. 1332  A9 01     LDA #$01
. 1334  A0 03     LDY #$03
. 1336  20 BA FF  JSR $FFBA
. 1339  AD 00 16  LDA $1600
. 133C  A2 01     LDX #$01
. 133E  A0 16     LDY #$16
. 1340  20 BD FF  JSR $FFBD
. 1343  20 C0 FF  JSR $FFC0
. 1346  A2 02     LDX #$02
. 1348  A9 02     LDA #$02
. 134A  A0 FF     LDY #$FF
. 134C  20 BA FF  JSR $FFBA
. 134F  A9 02     LDA #$02
. 1351  A2 BD     LDX #$BD
. 1353  A0 12     LDY #$12
. 1355  20 BD FF  JSR $FFBD
. 1358  20 C0 FF  JSR $FFC0
. 135B  A9 11     LDA #$11
. 135D  85 FC     STA $FC
. 135F  A9 93     LDA #$93
. 1361  85 FD     STA $FD
. 1363  EA        NOP
. 1364  A9 00     LDA #$00
. 1366  85 E0     STA $E0
. 1368  A2 01     LDX #$01
. 136A  20 C6 FF  JSR $FFC6
. 136D  20 CF FF  JSR $FFCF
. 1370  C9 0D     CMP #$0D
. 1372  D0 F9     BNE $136D
. 1374  E6 E0     INC $E0
. 1376  A5 E0     LDA $E0
. 1378  C9 09     CMP #$09
. 137A  D0 F1     BNE $136D
. 137C  20 CC FF  JSR $FFCC
. 137F  EA        NOP
. 1380  A9 20     LDA #$20
. 1382  85 D1     STA $D1
. 1384  A9 00     LDA #$00
. 1386  85 D0     STA $D0
. 1388  A2 41     LDX #$41
. 138A  A0 00     LDY #$00
. 138C  A9 20     LDA #$20
. 138E  91 D0     STA ($D0),Y
. 1390  C8        INY
. 1391  C0 53     CPY #$53
. 1393  D0 F9     BNE $138E
. 1395  A9 0D     LDA #$0D
. 1397  91 D0     STA ($D0),Y
. 1399  A5 D0     LDA $D0
. 139B  18        CLC
. 139C  69 54     ADC #$54
. 139E  90 02     BCC $13A2
. 13A0  E6 D1     INC $D1
. 13A2  85 D0     STA $D0
. 13A4  CA        DEX
. 13A5  10 E3     BPL $138A
. 13A7  A0 00     LDY #$00
. 13A9  A9 0D     LDA #$0D
. 13AB  91 D0     STA ($D0),Y
. 13AD  C8        INY
. 13AE  C0 04     CPY #$04
. 13B0  90 F9     BCC $13AB
. 13B2  A9 03     LDA #$03
. 13B4  91 D0     STA ($D0),Y
. 13B6  A9 12     LDA #$12
. 13B8  85 23     STA $23
. 13BA  A9 70     LDA #$70
. 13BC  85 22     STA $22
. 13BE  A2 0E     LDX #$0E
. 13C0  A0 01     LDY #$01
. 13C2  A9 17     LDA #$17
. 13C4  20 B0 12  JSR $12B0
. 13C7  A9 B0     LDA #$B0
. 13C9  85 D9     STA $D9
. 13CB  A9 35     LDA #$35
. 13CD  85 DA     STA $DA
. 13CF  A2 01     LDX #$01
. 13D1  20 C6 FF  JSR $FFC6
. 13D4  20 CF FF  JSR $FFCF
. 13D7  AA        TAX
. 13D8  A5 90     LDA $90
. 13DA  8D BF 12  STA $12BF
. 13DD  8A        TXA
. 13DE  A0 00     LDY #$00
. 13E0  91 D9     STA ($D9),Y
. 13E2  AD BF 12  LDA $12BF
. 13E5  D0 1B     BNE $1402
. 13E7  8A        TXA
. 13E8  C9 0D     CMP #$0D
. 13EA  D0 08     BNE $13F4
. 13EC  E6 D5     INC $D5
. 13EE  A5 D5     LDA $D5
. 13F0  C9 84     CMP #$84
. 13F2  F0 0E     BEQ $1402
. 13F4  A5 D9     LDA $D9
. 13F6  18        CLC
. 13F7  69 01     ADC #$01
. 13F9  90 02     BCC $13FD
. 13FB  E6 DA     INC $DA
. 13FD  85 D9     STA $D9
. 13FF  38        SEC
. 1400  B0 D2     BCS $13D4
. 1402  A5 D5     LDA $D5
. 1404  85 D8     STA $D8
. 1406  4A        LSR
. 1407  85 D6     STA $D6
. 1409  A9 00     LDA #$00
. 140B  85 D0     STA $D0
```

```
. 140D  85 D5     STA $D5
. 140F  A8        TAY
. 1410  AA        TAX
. 1411  A9 20     LDA #$20
. 1413  85 D1     STA $D1
. 1415  85 D3     STA $D3
. 1417  A9 2C     LDA #$2C
. 1419  85 D2     STA $D2
. 141B  A9 35     LDA #$35
. 141D  85 DA     STA $DA
. 141F  A9 B0     LDA #$B0
. 1421  85 D9     STA $D9
. 1423  B1 D9     LDA ($D9),Y
. 1425  C9 12     CMP #$12
. 1427  F0 1E     BEQ $1447
. 1429  C9 0D     CMP #$0D
. 142B  D0 10     BNE $143D
. 142D  8A        TXA
. 142E  48        PHA
. 142F  A2 00     LDX #$00
. 1431  20 A0 12  JSR $12A0
. 1434  68        PLA
. 1435  AA        TAX
. 1436  E8        INX
. 1437  E4 D6     CPX $D6
. 1439  F0 13     BEQ $144E
. 143B  D0 0A     BNE $1447
. 143D  84 E0     STY $E0
. 143F  A4 D5     LDY $D5
. 1441  91 D0     STA ($D0),Y
. 1443  E6 D5     INC $D5
. 1445  A4 E0     LDY $E0
. 1447  C8        INY
. 1448  D0 D9     BNE $1423
. 144A  E6 DA     INC $DA
. 144C  D0 D5     BNE $1423
. 144E  C8        INY
. 144F  B1 D9     LDA ($D9),Y
. 1451  C9 12     CMP #$12
. 1453  F0 1E     BEQ $1473
. 1455  C9 0D     CMP #$0D
. 1457  D0 10     BNE $1469
. 1459  8A        TXA
. 145A  48        PHA
. 145B  A2 02     LDX #$02
. 145D  20 A0 12  JSR $12A0
. 1460  68        PLA
. 1461  AA        TAX
. 1462  E8        INX
. 1463  E4 D8     CPX $D8
. 1465  F0 13     BEQ $147A
. 1467  D0 0A     BNE $1473
. 1469  84 E0     STY $E0
. 146B  A4 D5     LDY $D5
. 146D  91 D2     STA ($D2),Y
. 146F  E6 D5     INC $D5
. 1471  A4 E0     LDY $E0
. 1473  C8        INY
. 1474  D0 D9     BNE $144F
. 1476  E6 DA     INC $DA
. 1478  D0 D5     BNE $144F
. 147A  A0 00     LDY #$00
. 147C  A9 0D     LDA #$0D
. 147E  91 D2     STA ($D2),Y
. 1480  C8        INY
. 1481  C0 03     CPY #$03
. 1483  D0 F9     BNE $147E
. 1485  98        TYA
. 1486  91 D2     STA ($D2),Y
. 1488  20 CC FF  JSR $FFCC
. 148B  A9 88     LDA #$88
. 148D  85 22     STA $22
. 148F  A2 0E     LDX #$0E
. 1491  A0 01     LDY #$01
. 1493  A9 17     LDA #$17
. 1495  20 B0 12  JSR $12B0
. 1498  A9 00     LDA #$00
. 149A  85 E0     STA $E0
. 149C  85 D0     STA $D0
. 149E  A9 20     LDA #$20
. 14A0  85 D1     STA $D1
. 14A2  A2 02     LDX #$02
. 14A4  20 C9 FF  JSR $FFC9
. 14A7  A4 E0     LDY $E0
. 14A9  B1 D0     LDA ($D0),Y
. 14AB  C9 03     CMP #$03
. 14AD  F0 22     BEQ $14D1
. 14AF  C8        INY
. 14B0  D0 02     BNE $14B4
. 14B2  E6 D1     INC $D1
. 14B4  84 E0     STY $E0
. 14B6  AA        TAX
. 14B7  AD 43 05  LDA $0543
. 14BA  29 01     AND #$01
. 14BC  D0 1B     BNE $14D9
. 14BE  AD 10 FD  LDA $FD10
. 14C1  29 02     AND #$02
. 14C3  F0 F2     BEQ $14B7
. 14C5  A5 90     LDA $90
. 14C7  48        PHA
. 14C8  8A        TXA
. 14C9  20 D2 FF  JSR $FFD2
. 14CC  68        PLA
. 14CD  D0 0A     BNE $14D9
. 14CF  F0 D8     BEQ $14A9
. 14D1  AD BF 12  LDA $12BF
. 14D4  D0 03     BNE $14D9
. 14D6  4C 7C 13  JMP $137C
. 14D9  20 CC FF  JSR $FFCC
. 14DC  A9 01     LDA #$01
. 14DE  20 C3 FF  JSR $FFC3
. 14E1  A9 02     LDA #$02
. 14E3  20 C3 FF  JSR $FFC3
. 14E6  00        BRK
. 14E7  EF        ???
. 14E8  10 EF     BPL $14D9

. 12A0  B5 D0     LDA $D0,X
. 12A2  18        CLC
. 12A3  69 54     ADC #$54
. 12A5  90 02     BCC $12A9
. 12A7  F6 D1     INC $D1,X
. 12A9  95 D0     STA $D0,X
. 12AB  A9 00     LDA #$00
. 12AD  85 D5     STA $D5
. 12AF  60        RTS
. 12B0  8D EA 02  STA $02EA
. 12B3  8E DB 02  STX $02DB
. 12B6  8C DA 02  STY $02DA
. 12B9  20 0D BA  JSR $BA0D
. 12BC  60        RTS
```

```
>1180 43 52 45 41 54 45 20 41 :CREATE A
>1188 34 20 53 49 5A 45 20 44 :4 SIZE D
>1190 4F 55 42 4C 45 20 43 4F :OUBLE CO
>1198 4C 55 4D 4E 20 50 41 47 :LUMN PAG
>11A0 45 53 20 46 4F 52 20 20 :ES FOR
>11A8 54 52 41 4E 53 4D 49 53 :TRANSMIS
>11B0 49 4F 4E 20 54 4F 20 52 :ION TO R
>11B8 2E 53 2E 32 33 32 2E 20 :.S.232.
>11C0 50 52 49 4E 54 45 52 53 :PRINTERS
>11C8 20 31 33 43 50 49 20 20 : 13CPI
>11D0 36 4C 50 49 2E 20 34 38 :6LPI. 48
>11D8 30 30 20 42 41 55 44 2E :00 BAUD.
>11E0 20 58 4F 4E 3D 23 24 31 : XON=#$1
>11E8 31 2E 20 58 4F 46 46 3D :1. XOFF=
>11F0 23 24 39 33 2E 0D 0D 20 :#$93...
>11F8 50 4C 45 41 53 45 20 45 :PLEASE E
>1200 4E 53 55 52 45 20 42 4F :NSURE BO
>1208 54 48 20 44 49 53 43 20 :TH DISC
>1210 41 4E 44 20 50 52 49 4E :AND PRIN
>1218 54 45 52 53 20 20 20 20 :TERS

>1220 41 52 45 20 52 45 41 44 :ARE READ
>1228 59 2E 0D 0D 20 45 4E 54 :Y... ENT
>1230 45 52 20 4E 41 4D 45 20 :ER NAME
>1238 4F 46 20 46 49 4C 45 2E :OF FILE.
>1240 2E 2E 2E 2E 54 4F 20 42 :....TO B
>1248 45 20 53 45 4E 54 2E 0D :E SENT..
>1250 0D 20 41 4E 44 20 50 52 :. AND PR
>1258 45 53 53 20 52 45 54 55 :ESS RETU
>1260 52 4E 20 4B 45 59 0D 0D :RN KEY..
>1268 20 20 20 20 20 20 20 20 :
>1270 82 12 50 52 45 50 41 52 :..PREPAR
>1278 49 4E 47 20 44 41 54 41 :ING DATA
>1280 20 42 4C 4F 43 4B 92 84 : BLOCK..
>1288 82 12 53 45 4E 44 49 4E :..SENDIN
>1290 47 2E 2E 2E 44 41 54 41 :G...DATA
>1298 20 42 4C 4F 43 4B 92 84 : BLOCK..
>12A0 B5 D0 18 69 54 90 02 F6 :5P.iT..v
>12A8 D1 95 D0 A9 00 85 D5 60 :Q.P)..U'
>12B0 8D EA 02 8E DB 02 8C DA :..j..[..Z
>12B8 02 20 0D BA 60 1C 05 40 :. .:'..@
```

END

QUIXAVER for the COMMODORE C-16

and also QUIKADON and QUIKREST

QUIXAVER is a tape turbo for the C-16.

It doesn't take up any of the C-16's precious BASIC memory, and can be used in all graphics modes as well as text mode.

As well as BASIC programs, it can handle screens of text or graphics and blocks of machine code or data.

It can easily handle multi-part files.

It can load BASIC programs correctly even if the start-location of BASIC has been changed since they were saved.

It can save BASIC programs complete with all their variables and reload them for further use.

It can load one BASIC program to replace another while preserving all the old program's variables for the new program to use.

It provides an auto-run facility for both BASIC and machine-code programs.

As well as its normal QUIXAVE, it provides a shortened form — FLIXAVE — that saves time, tape, and temper when several files have to be loaded one after another; and FLIXAVE can also be used for TRIXAVE.

It fast-loads with the screen on, and progress is shown by a flying cursor.

It works in the ordinary C-16, or in one with expanded memory (up to the full 64K), though it does clash with some commercial programs.

It can easily be disabled when not required.

It loads and initialises itself automatically whenever anything saved with it is loaded, so it rarely needs to be loaded separately.

All QUIXAVEd files will load 'from cold' using the normal commands with no preliminaries, whether or not QUIXAVER is present.

* * *

QUIKADON will load a BASIC program and append it (add it on) to one already in the machine, using a single 'SYS' call.

QUIKREST will restore a BASIC program after a 'NEW' or 'RESET', using just a normal 'LOAD'.

* * *

THE ORIGIN OF QUIXAVER

The origin of QUIXAVER goes back to a C-16 tape turbo by Nick Hampshire, published under the title 'Break the Speed Limit' in 'Your Commodore' for February 1986. This did all that was claimed for it. But it took up a fair amount of space at the top of the BASIC RAM; it couldn't be used in graphics modes; and it wouldn't load a BASIC program correctly if the start of BASIC wasn't in the same place as when the program was saved. So I decided to have a go at modifying it, and QUIXAVER is the eventual result. I had a great deal of help from Beresford's 'C-16 Machine Language for the Absolute Beginner', Gerrard and Bergin's 'The Complete Commodore 16 ROM Disassembly', and Zaks' 'Programming the 6502'; but it took a very long time, because I had only the C-16's simple built-in MONITOR to do it with and I'd never done any machine-code programming before.

I condensed the program by using existing subroutines from the ROM wherever I could. (This would be risky with a current-model machine because the ROM might later be changed, but that doesn't seem likely to happen with the C-16!) I found a way of calling up the 'save' routine without clearing the cassette buffer, and this allowed the first part of QUIXAVER to live there all the time; then I moved most of the remainder into an area of RAM said to be 'reserved for extra ROMs', and what was too much for that spilled over into the lower end of the 'BASIC pseudo-stack', where — so far — it doesn't seem to clash with anything I've tried to do. I don't know just what the pseudo-stack does, but I suspect that it handles the return addresses for subroutines, just like the processor stack, and that unless the subroutines are very deeply nested the stacking won't extend far enough down to corrupt QUIXAVER. (Any information would be welcome!) I also added some extra facilities and made the program easier to use. It doesn't look a bit like the original now!

I did the modifications just a little step at a time, and after each step — and there were a good many hundred in all — I used the program as it then stood to save and reload itself, to check that it would still work; and I've used it in its various editions for all my tape saves for over three years. Throughout all these hundreds of saves and loads there has never yet been a load error except when I've done something silly. In fact I had to introduce a dummy error when I wanted to test the error-trap!

The fundamental method of recording bits on the tape has remained unchanged throughout, and its reliability is a tribute to the soundness of Nick Hampshire's original scheme.

WHAT A QUIXAVE CONSISTS OF

A QUIXAVE begins with a normal slow-save of just two bytes. On loading, these two bytes will replace the normal 'output vector' with one that makes the system jump from the end of the slow-load to the start of QUIXAVER's fast-load routine, which was concealed in the header of the slow-save and will have been read into the cassette buffer.

After the header-and-two-byte slow-save comes the fast-save. This begins (after a run-in) with the remainder of QUIXAVER, followed by five bytes giving the start-address, length, and device-number of the file being saved, then the file itself, and finally a checkbyte.

40

The whole fast-save is made in one unbroken stream of bits, eight for each byte in turn. Each bit is recorded on the tape as one whole cycle of a square-wave, the cycle being about twice as long for a '1' as for a '0'. When the tape is being read during the fast-load, a count-down timer is restarted from a fixed initial value at the beginning of each cycle. The initial value is chosen so that the time taken to count down to zero is half-way between the duration of a short cycle and that of a long cycle; so at the end of a short cycle the MSB of the timer will still be '0', but by the end of a long cycle the count will have gone down through zero and the MSB will be '1'. Successive bits taken from the timer MSB at the end of each cycle are assembled into eight-bit bytes and stored in the appropriate memory locations.

The run-in at the beginning of the fast-save allows the fast-loader to set itself in synchronism with the signals from the tape, so that the stream of bits will be divided into bytes correctly. The run-in has 512 bytes, each one (after the first) being $01, and the fast-loader can pick up synchronism if it starts reading anywhere before the last two bytes of the run-in. As a precaution against false starts, it checks that the first byte after a string of $01's has the right value for the first byte of the fast-save (which is always the same); if it hasn't, the fast-loader realises that what appeared to be a run-in wasn't really one at all, and starts all over again. (Obviously, if you start the tape in the wrong place there must always be some possibility of an unfortunate combination of bytes in a file mimicking a genuine run-in and causing a false start, whatever security scheme is used; but the scheme used here has to find a specific sequence of 17 successive bits before it starts the fast-load, and this won't happen by accident very often. I've never had any false-start troubles, even with an earlier version that looked for only a nine-bit sequence.)

The last byte read by the fast-loader is the checkbyte, which it compares with one that it calculates during the fast-load. If they agree, all is well; the BASIC program (if any) in the machine is rechained; the BASIC pointers are reset; QUIXAVER is re-initialised; the processor stack is adjusted; and the system returns to wherever the load was called from — with certain exceptions. If the checkbytes don't agree, 'LOAD ERROR' is signalled.

SELECTING QUIXAVER'S OPTIONS

The way QUIXAVER saves and loads a file is controlled entirely by the device number used for the save. This may be any integer in the range 0..255, except that the C-16's system won't let it accept device number 3 in MONITOR mode. The use of this wide range of device numbers with QUIXAVER doesn't make any difference to the way device numbers are used with other peripherals such as printers, though trying to save to a disc drive with QUIXAVER present would certainly cause problems!

All normal QUIXAVEs are made with odd device numbers. Even device numbers produce FLIXAVEs, which are just like QUIXAVEs but with all the slow-save part omitted. This is possible because the slow-save part of QUIXAVER is always identically the same apart from the actual filename; so once any QUIXAVEd file whatsoever has been loaded, the fast-load of any other file can go ahead without repeating the slow-load. A FLIXAVE ignores any filename, so a null filename is used when FLIXAVing.

If the device number of the save was in the range 0-127 ($00-$7F), the file will load at the current start of BASIC, wherever that may then be. This range of device numbers is normally usable only for BASIC program files, because QUIXAVER always rechains from the current start of BASIC at the end of every load, no matter what sort of file it has just loaded.

If the device number was in the range 128-255 ($80-$FF), the file will load back to the memory locations it was saved from. This range of device numbers may be used for a memory-block file or for a BASIC program file. But because the system will end the load by rechaining, the file just loaded will almost certainly be corrupted if it includes the current start of BASIC or the following byte — unless of course it actually is a BASIC program that was saved from the same start of BASIC.

If a BASIC program is QUIXAVED with an odd device number in the range 65-127 ($41-$7F), or if a machine-code program is QUIXAVED with an odd device number in the range 193-255 ($C1-$FF), it will auto-run when loaded in direct BASIC mode with the normal 'LOAD' command. Auto-run doesn't operate when loading in any other way. (A machine-code program auto-runs from its lowest address.)

The device number of the save may be used as an ident, and this is particularly useful when files have been saved without filenames (as with FLIXAVEs). The device number loads back into memory at $0717. In BASIC it may be read with 'PEEK(1815)'.

QUIXAVER AS A SUBROUTINE

However it is called, either for saving or for loading, QUIXAVER always behaves as a subroutine and returns for the next instruction (if there is one), except when a program auto-runs after a direct BASIC 'LOAD' or when 'LOAD' is used within a BASIC program. Special considerations apply in this last instance — see below.

HOW TO MAKE QUIXAVES AND FLIXAVES

QUIXAVEs and FLIXAVEs of BASIC programs are made in exactly the normal way in direct BASIC mode or within a BASIC program, using any device number in the appropriate range. If no device number is entered, the default value ('1') is assumed, and a QUIXAVE results. For a FLIXAVE (even device number), use a null filename ('""'). The save behaves as a subroutine and returns for the next instruction, if there is one.

QUIXAVEs and FLIXAVEs of blocks of memory — whether text screens, graphics screens, machine-code programs, or anything else — are made in exactly the normal way in MONITOR or within a machine-code program, using any device number in the appropriate range. Don't forget that the device number in MONITOR is always interpreted as a hex number (as are the «start address» and «end+1 address») although no '$' is used. In a machine-code program the save behaves as a subroutine and returns for the next instruction, but in MONITOR it just returns to the prompt because MONITOR can't handle more than one instruction at a time.

42

SAVING A BLOCK IN MACHINE-CODE

The following machine-code example saves the text screen. Colour memory is $800-$BE7 inclusive (decimal 2048-3047), character memory $C00-$FE7 inclusive (3072-4071). If required, the values in the various TED colour registers may be copied to some of the 24 spare bytes ($BE8-$BFF, 3048-3071) between the top of colour memory and the bottom of screen memory, so that they are saved with the screen and can be copied back to the TED registers later on when the saved screen has been loaded.

```
        LDA #$«device number»
        STA $AE
*       LDA #$«length of filename»                ;  will go to $AB
*       LDX #$«low byte of address of filename»   ;  will go to $AF
*       LDY #$«high byte of address of filename»  ;  will go to $B0
*       JSR $FFBD                                 ;  send them there
        LDA #$00      ;  low byte of start address
        STA $73       ;  store in convenient zero-page location
        LDA #$08      ;  high byte of start address
        STA $74       ;  store in next zero-page location
        LDA #$73      ;  zero-page address of start-address pointer
        LDX #$E8      ;  low byte of end+1 address ;  will go to $9D
        LDY #$0F      ;  high byte of end+1 address;  will go to $9E
        JSR $FFD8                                 ;  perform save
        (continue . . .)
```

For a null filename, the four lines marked '*' may be replaced by the two lines

```
        LDA #$00
        STA $AB
```

The zero-page locations ($73, $74) used here for the start address are described as holding the line-increment value for 'AUTO', and using them shouldn't cause any problems; but any available zero-page address-pair could be used instead.

SAVING A BLOCK IN A BASIC PROGRAM

To QUIXAVE or FLIXAVE a block of memory from within a BASIC program, the 'start of BASIC' and 'end+1 of BASIC, start of variables' pointers are altered to point to the start and end+1 of the block; then the save is made, with a device number in the range 128-255 so that it will load back to where it was saved from. Finally the pointers are reset to their original values, which have been temporarily stored in array variables.

The following BASIC example saves the graphics screen. Colour memory is $1800_$1BE7 (decimal 6144_7143) and $1C00_$1FEF (7168_8167), bit-map memory $2000_$3F3F (8192_16191), all inclusive. If required, the values in the various TED colour registers may be copied by peeking and poking to some of the 48 spare bytes between these sections of memory, so that they are saved with the screen and can be restored to the TED registers later on when the saved screen has been loaded.

```
[ ... DIM S%(4),S$(0) ]

 900 REM : Store BASIC pointers and provisional device number
 905 FOR N = 0 TO 3 : S%(N) = PEEK(43 + N) : NEXT : S%(4) = 129
 910 REM : Get and store wanted device number (128_191)
 915 INPUT "SCREENSAVE:  DEVICE 129";S%(4)
 920 REM : Store provisional (null) filename
 925 S$(0) = ""
 930 REM : Set up opening quotes for filename entry
 935 POKE 1319,34:POKE 239,1
 940 REM : Get and store wanted filename
 945 INPUT "SET TAPE;  FILENAME";S$(0)
 950 REM : Get confirmation
 955 A$ = "" : INPUT "OK";A$ : IF A$ <> "OK" THEN 1000 [?]
 960 REM : Set pointers to 7168_16192
 965 POKE 45,64 : POKE 46,63
 970 POKE 43,0 : POKE 44,24
 975 REM : Save
 980 SAVE S$(0),S%(4)
 985 REM : Restore pointers (a FOR/NEXT loop won't work here)
 990 POKE 43,S%(0) : POKE 44,S%(1)
 995 POKE 45,S%(2) : POKE 46,S%(3)
1000 (continue . . .)
```

QUIXAVING QUIXAVER

To QUIXAVE QUIXAVER itself, without anything else, go into MONITOR and use

```
S "QUIXAVER",99,347,348 «RETURN»
```

Since you can't QUIXAVE nothing-at-all, this QUIXAVEs just one byte of QUIXAVER itself (after automatically saving the whole of QUIXAVER!), so when it's loaded it won't alter anything else in the machine. Any other 'harmless' byte and any other odd device number in the range ($)81_($)BF could equally well be used.

HOW TO LOAD QUIXAVES AND FLIXAVES

The way QUIXAVER loads a file is controlled entirely by the device number used for the save; files of any type may be loaded in any mode. No preliminaries are needed for loading QUIXAVEs, whether or not QUIXAVER is already present.

When the message 'FOUND «filename»' appears on the screen, QUIXAVER puts an extra character immediately after it to show that the file is a QUIXAVEd one. This character is at present '$' (in either character set), but may be replaced by any printable character to 'personalise' your system; use POKE 839, «ASCII value of character». (This character may also be used as an ident; in BASIC it may be read with 'PEEK(839)'.)

In direct BASIC mode, QUIXAVEd files are loaded by the normal 'LOAD' command, with the default device number '1'. No secondary address is needed, no matter what type of file is being loaded; if one is given it will be ignored.

For the use of 'LOAD' within a BASIC program see below.

In MONITOR, QUIXAVEd files of any type (including BASIC program files) are loaded by the normal 'L' command with the default device number '1'.

Within a machine-code program QUIXAVEd files of any type (including BASIC program files) are loaded in the normal way with device number and secondary address both '1', as in the following example:

```
      LDA #$01
      STA $AE                                   ;  device number
      STA $AD                                   ;  secondary address
*     LDA #$«length of filename»                ;  will go to $AB
*     LDX #$«low byte of address of filename»   ;  will go to $AF
*     LDY #$«high byte of address of filename»  ;  will go to $B0
*     JSR $FFBD                                 ;  send them there
§     LDA $9A
§     PHA                                       ;  remember mode
      LDA #$00
§     STA $9A                                   ;  'program' mode
      JSR $FFD5                                 ;  perform load
§     PLA
§     STA $9A                                   ;  restore mode
      (continue . . .)
```

The five lines marked '§' prevent the slow-load from stopping to display every filename it comes across as it looks for the one it wants. If you want the display omit these lines.

For a null filename omit the four lines marked '*' and replace them by one line 'STA $AB' inserted after 'LDA #$00'. But for a null filename a FLIKLOAD is easier if you don't need the display.

The normal load commands won't load a FLIXAVE; instead, a FLIKLOAD is called by using 'SYS843' in BASIC (direct or within a BASIC program), 'G 348' in MONITOR, or 'JSR $034B' within a machine-code program. QUIXAVER — or at least the part of it in the slow-saved tape header — must be present, but no other preliminaries are required. These commands will load the next program on the tape whether it was FLIXAVEd or QUIXAVEd. Note that the last two use different addresses; this is to enable QUIXAVER to return correctly in each mode.

BASIC 'SYS843' may be made to load files one after another until a certain one is found. This is done by using the device number of the QUIXAVE or FLIXAVE as an ident and giving each file a different device number within the allowable range. The required file is loaded by setting up a loop such as

```
DO : SYS843 : LOOP UNTIL PEEK(1815) = «required number»
```

— but take care that the files loaded and then discarded on the way to loading the wanted one don't corrupt anything valuable.

LOADING BASIC PROGRAMS

When any QUIXAVEd or FLIXAVEd file has been loaded, the BASIC program then in the machine is rechained — whether it has just been loaded, or whether it is one that was already there. But QUIXAVER never clears the existing variables. This means that when a BASIC program has been loaded you mustn't start it with a 'GOTO' unless you use 'CLR' first; but you can of course start it with 'RUN' or 'RUN «line number»', because these do an automatic 'CLR'.

IF YOU SHIFT THE START OF BASIC

If you intend to load a BASIC program at a new start-of-BASIC, then as well as altering the pointer ($2B, $2C) you have to make sure that the byte just below the new start-of-BASIC is zero. And to avoid trouble with QUIXAVER trying to 'rechain' whatever happens to have been left in memory at the new start-of-BASIC, you shouldn't load anything else between altering the pointer and loading the BASIC program.

RETAINING BASIC VARIABLES

There is one important exception to the need for a 'CLR' after a load: if a new BASIC program replaces a previous one of identically the same length, all the old program's variables will be available unaltered for the new program to use if it's started with a 'GOTO'.

BRINGING PROGRAMS TO THE SAME LENGTH

Two programs may be brought to the same length by padding out the shorter one, and this can be done in several ways. One is to add spaces within existing BASIC lines; another is to add 'REMs'; but probably the easiest is to add lines containing nothing but rows of as many colons as may be needed. You can put up to 80 colons in a line, but remember that the number of bytes that each such line adds is 5 more than the number of colons in it.

To find out how long a program is, go into MONITOR and enter '>2B «RETURN»' to display the BASIC pointers (in hex). The first two are the low and high bytes of the 'start of BASIC program' pointer ($2B, $2C), and the next two are the low and high bytes of the 'start of variables' pointer ($2D, $2E). Do this for both programs (you don't have to run them); the difference between their 'start of variables' pointers will tell you how many bytes to add to the shorter one. It's wise to check again after adding what you think is the right number!

LOADING FROM WITHIN A BASIC PROGRAM

When 'LOAD' is used within a BASIC program, the C-16's slow-load system doesn't jump to the output vector at the end of the slow-load. Instead, it starts to run the current BASIC program from its first line, but without clearing the existing variables. This means that 'LOAD' within a BASIC program doesn't call up the fast-load, so it must somehow be followed by 'SYS843' after the jump back to the first line.

When the system jumps back, the 'output vector' has not yet been restored to normal; it still points to the beginning of the fast-load routine. So any BASIC instruction that ends by jumping to the output vector will call up the fast-load after doing whatever it normally does. One such instruction is 'INPUT', and there may be others. The program has to be arranged so that a 'SYS843' does the fast-load 'officially' before any such instruction is met; the output vector is automatically restored at the end of the fast-load.

When only blocks of memory are to be loaded, leaving the existing BASIC program intact, all that is needed is to start the program with a line such as

```
10 IF QX THEN SYS843 : ON QX GOTO 100,200,300
11 (continue . . .)
```

and then to follow this with appropriate lines such as

```
 99 QX = 1 : LOAD "FIRST ITEM"
100 (continue . . .)

199 QX = 2 : LOAD "SECOND ITEM"
200 (continue . . .)

299 QX = 3 : LOAD "THIRD ITEM"
300 (continue . . .)
```

When the program is started, QX = 0; and so lines 11-99 are run. Then the slow-saved part of FIRST ITEM is loaded, the program restarts with QX = 1, the fast-saved part of FIRST ITEM is loaded, and lines 100-199 are run. Next QX = 2, SECOND ITEM is loaded, lines 200-299 are run; and so on. In this way various graphics screens or machine-code programs can be loaded in turn for the BASIC program to use.

If the files to be loaded all follow one another on the tape, the filenames may of course be omitted. But in that case it's simpler to FLIKLOAD the files with 'SYS843', which doesn't need any special treatment at all; after each FLIKLOAD the program just carries on with its next instruction. And then of course all the files may be FLIXAVEd, saving about 24 seconds on the loading of each.

REPLACING A BASIC PROGRAM BY ANOTHER

When a BASIC program 'LOADs' another BASIC program that will replace it, after the slow-load the system jumps back to the beginning of the old program, goes on until it gets to a 'SYS843', and then does the fast-load. But when it returns after the fast-load to get the next instruction, the new program has replaced the old. So there must be a suitable instruction in the new program for the system to 'return' to, at exactly the right place in RAM. The best way to ensure this is to start each of the programs with the line

```
10 IF QX THEN SYS843 : CLR
11 (continue . . .)
```

and then to end the old program with the line

```
123 QX = 1 : LOAD "NEW PROGRAM"
```

If the new program is FLIXAVEd, replace 'LOAD "NEW PROGRAM"' with 'GOTO 10' (which of course will also work with a QUIXAVEd program if it's next on the tape).

(If one of the programs also has to call up the loading of a file that isn't a BASIC program, it's easiest just to use 'SYS843' for that wherever you want it; but if you need to find a specific filename and so have to use 'LOAD', the 'SYS843' in the first line will have to be followed by 'ON QX GOTO.....,«line following 'LOAD'»,.....' or something equivalent, and QX will have to be given the appropriate value before each 'LOAD'.)

If the old and new programs are identically the same length 'CLR' may be omitted to allow the new program to take over the existing values of the old program's variables.

SAVING A PROGRAM WITH ITS VARIABLES

To QUIXAVE a BASIC program together with the current values of all its variables, two blocks of memory have to be saved one after the other. The first block comprises all the RAM available to BASIC, and may be extended to include text screen and/or graphics screen and/or machine-code areas if desired. For example, in an unexpanded C-16 'everything' may be QUIXAVED in MONITOR with

S "«filename»",99,800,4000 «RETURN»

or the BASIC RAM, in graphics mode, plus the graphics screen, with

S "«filename»",99,1001,3F40 «RETURN»

or the BASIC RAM alone, in text mode, with

S "«filename»",99,1001,3FF6 «RETURN»

The last of these may be used for most ordinary requirements, in text or graphics modes. The limits of the save will of course be different in an expanded C-16.

The other block of memory comprises the pointers that show where the variables are stored in memory; this block is FLIXAVED immediately after the first block with

S "",88,2F,39 «RETURN»

To load the program complete with all its saved variables (and screens and/or machine-code), first make sure that the start of BASIC is in the same place as it was when the program was saved; and then in direct BASIC mode enter

LOAD "«filename»" : SYS843 : GOTO «suitable line number» «RETURN»

If the program uses any arrays, choose the 'suitable line number' with care to avoid a 'REDIM'D ARRAY' error.

For a neater way of dealing with multi-part saves, see below.

(If the C-16's memory has been expanded and the program to be saved-with-variables is fairly short, it may be worth while to split the saving of the BASIC memory area into two parts — one for program, variables, and arrays, and the other for strings. But with an unexpanded memory it's not worth the effort. QUIXAVing 'unexpanded everything' takes about 95 seconds, depending on the proportion of '1s' to '0s' in memory.)

RELOCATING MACHINE-CODE FILES

A machine-code file that has been loaded in one place and is wanted in another may of course be moved after loading by using the MONITOR 'T' command. If the place-where-it-is overlaps the place-where-it-should-be, you can't make the move directly. You can often make the move in two steps — first to some non-overlapping vacant place, and then from there to the final place — but you can't do this if there's no suitable place vacant. Or of course you can move the file in sections small enough to avoid the overlap, but this method is prone to errors. A better way is to load the file beforehand into an otherwise empty machine and make the move there, and then save the file again from its new position. Loading it is then straightforward.

QUIXAVER allows you to relocate the file in yet another way. For this you begin the file with two zero bytes — you can remove them afterwards — and save it from MONITOR in the usual way but with a device number in the range ($)00_($)3F, as though it were a BASIC program. To do a relocated load, move the start-of-BASIC pointer to where you want the first of the two zero bytes to go, and load the file in the normal way. The 'rechaining' at the end of the load will read the two zero bytes as 'end-of-program' and leave the whole file unaltered. The end+1-of-BASIC pointer will have been changed, so you will normally need to restore both that and the start-of-BASIC pointer to their original values. All this can be done within a BASIC or machine-code program (use array variables for storing the 'return' values in BASIC).

SEQUENTIAL FILES

QUIXAVER won't work in the normal way with sequential files. Sequential data must first be stored in a block of memory, and this can then be QUIXAVED or FLIXAVED with a device number in the range 128_191 so that it will later reload to the place it was saved from for re-use.

The data can be shifted into the chosen area of memory with BASIC PEEKs and POKEs in a FOR-NEXT loop, or with a machine-code routine (which would be a lot faster). But usually the easiest way in BASIC, if all the data is printable, is to print it on to a reserved area of the text screen and then save that area. Another program can then read the data back by putting the cursor in the right place and using 'INPUTXX$', followed by 'POKE 1319,13 : POKE 239,1' to simulate the pressing of «RETURN».

MULTI-PART SAVES

The example that follows shows how to use QUIXAVER for saving multi-part files so that they load with a single 'LOAD'; the method can be considerably extended.

BUILDING BASIC PROGRAMS : QUIKADON

QUIKADON will add on ('append') any QUIXAVEd or FLIXAVEd BASIC program to the end of a program already in the machine. You can use it to build up a new program from previously-saved segments, or to load previously-saved subroutines or lists of DATA statements for an existing program to use. The only restrictions are that the program to be QUIKADONed must have been saved with a device-number in the range 0_127, and that its line-numbers must all be greater than any in the existing program.

QUIKADON doesn't take up any BASIC memory space, it doesn't interfere with BASIC or QUIXAVER, it isn't affected by a «RESET», and it's run with a single 'SYS' call.

QUIKADON's machine-code is split into two parts so that it can be tucked out of the way, but its QUIXAVEd file actually contains three blocks. Block zero runs from $D8 to $E0 inclusive (an area 'used by application software'), and is QUIXAVEd. When loaded with a direct BASIC 'LOAD' it auto-runs, FLIKLOADs the other two blocks, and returns to direct BASIC mode. After that it's not wanted and needn't be kept.

The other two blocks contain QUIKADON itself and are FLIXAVEd. Block one runs from $BE8 to $BFF, between the colour-memory and character-memory of the text screen, and block two runs from $FE8 to $FFF, between the character-memory and the normal start of BASIC.

Once loaded, QUIKADON can be run with 'SYS3048 «RETURN»' in direct BASIC mode. First of all it stores the current start-of-BASIC address, and alters the pointer so that it points to the end of the existing program. Then it FLIKLOADs whatever comes next on the tape. When it's done that, it restores the start-of-BASIC pointer to its original value and performs a 'CLR' to reset the other BASIC pointers. Finally it prints a reminder that a 'RENUMBER' may be necessary, and returns to direct BASIC mode.

If there is no BASIC program in the machine, QUIKADON's 'SYS3048' will just load the new program in the same way as a normal FLIKLOAD's 'SYS843' except that it will perform a 'CLR' at the end of the load.

THE PROGRAM

```
block 0
      . 00D8  20 4B 03 JSR $034B   ;FLIKLOAD block 1
      . 00DB  20 4B 03 JSR $034B   ;FLIKLOAD block 2
      . 00DE  4C DC 8B JMP $8BDC   ;exit to direct BASIC mode

block 1
start . 0BE8  A5 2B    LDA $2B     ;stack start-of-BASIC pointer
      . 0BEA  48       PHA
      . 0BEB  A5 2C    LDA $2C
      . 0BED  48       PHA
      . 0BEE  A5 2D    LDA $2D     ;set start-of-BASIC pointer to
      . 0BF0  E9 01    SBC #$01    ;     end+1-of-BASIC minus two
      . 0BF2  85 2B    STA $2B     ;     (carry is clear when
      . 0BF4  A5 2E    LDA $2E     ;     QUIKADON is called)
      . 0BF6  E9 00    SBC $00
      . 0BF8  85 2C    STA $2C
      . 0BFA  20 4B 03 JSR $034B   ;FLIKLOAD file to be QUIKADONed
      . 0BFD  4C E8 0F JMP $0FE8   ;jump to block 2

block 2
      . 0FE8  68       PLA         ;restore start-of-BASIC pointer
      . 0FE9  85 2C    STA $2C
      . 0FEB  68       PLA
      . 0FEC  85 2B    STA $2B
      . 0FEE  20 9A 8A JSR $8A9A   ;CLR to reset other pointers
      . 0FF1  20 D8 FB JSR $FBD8   ;print text
      >0FF4 0D 12 52 45 4E 55 4D 3F :..RENUM?   ;text
      . 0FFC  00       BRK         ;end of print
      . 0FFD  4C DC 8B JMP $8BDC   ;exit to direct BASIC mode
```

GETTING IT IN AND SAVING IT

Go into MONITOR and enter the three parts in the usual way. Then, with QUIXAVER present, save them one after the other with

```
S "QUIKADON",FF,D8,E1 «RETURN»
S "",88,BE8,C00 «RETURN»
S "",88,FE8,1000 «RETURN»
```

LOADING QUIKADON

Use a single 'LOAD' in direct BASIC mode.

USING QUIKADON

Check that the line-numbers of the program to be QUIKADONed are all higher than any in the existing program (if they aren't, renumber). Set up the tape and enter 'SYS3048 «RETURN»'. Renumber if necessary.

RESTORING BASIC PROGRAMS : QUIKREST

QUIKREST will restore a BASIC program that has been accidentally 'NEWed' or killed by a cold «RESET». All you have to do is to load QUIKREST and it's done, wherever the start of BASIC is, without disturbing anything else in the machine. (But the variables will have been corrupted, although there are methods by which you may be able to recover some of them.) Restart the restored program with 'RUN' or 'RUN «line number»', or use 'CLR' before 'GOTO'.

To prepare QUIKREST, enter in direct BASIC mode with the start of BASIC in its normal place at $1001 (decimal 4097)

```
POKE 4097,1 «RETURN»
```

Then go into MONITOR and enter

```
S "QUIKREST",1,1001,1002 «RETURN»
```

to save this one-byte program — note that the device number is one you would normally use only for BASIC program files.

TRIXAVER

Anything that can be FLIXAVEd in less than about 6 seconds can be TRIXAVEd; a TRIXAVE is a FLIXAVE that's hidden from normal loading. It's concealed between the header of a QUIXAVE and the two-byte slow-save that follows; it overwrites part of the run-in of the slow-save, but that doesn't worry the slow-loader. To make a TRIXAVE, first make the concealing QUIXAVE as usual, and then set up the tape ready to load it. Enter 'LOAD «RETURN»', but as soon as the QUIXAVE is 'found' press «RUN/STOP» and stop the tape. Then make a normal FLIXAVE. Now rewind the tape and you should be able to 'LOAD' the QUIXAVED file normally, with no indication that the TRIXAVE is present. Rewind the tape, and 'SYS843' will load the TRIXAVED file; and then a further 'SYS843' will load the QUIXAVED file. (QUIXAVER — or at least the slow-saved first part of it — must of course be present before you can use 'SYS843'.)

One use of TRIXAVER is to add 'afterthought information' or an index at the beginning of a file — for example, the current list of parts in a multi-part sequential file that may be extended from time to time — without altering any existing parts of the file. The TRIXAVE can of course be read without reading the whole file, and new information may be TRIXAVED, overwriting any existing TRIXAVE, every time the file is extended.

THINGS THAT DISABLE QUIXAVER

QUIXAVER is disabled by a 'warm reset' («RUN/STOP» and «RESTORE»), but can be restored by entering 'G 34E «RETURN»' immediately afterwards.

QUIXAVER is killed dead by a 'cold reset' (plain «RESTORE»), by slow-loading or slow-saving anything, or of course by switching off.

AWFUL WARNING

After a program crash, QUIXAVER may have been corrupted. This can happen without it being immediately obvious; for example, if the corruption is in the cassette buffer QUIXAVEs may appear to be normal but the resultant tapes won't load at all, or won't do the right thing in certain modes. The latter is particularly objectionable because you won't spot the error until you happen to use one of the faulty loading modes — by which time you may have used the faulty version for several saves, each of which you may have used as a source of QUIXAVER for further saves, and so on . . .

So — after every crash, reload QUIXAVER to be on the safe side!

GETTING QUIXAVER IN

The initial keying-in of QUIXAVER in MONITOR is straightforward, but it's wise to save the program before trying it out; otherwise if you've made any mistake the system will almost certainly crash and you'll have to enter the whole thing again. But the saving does pose a problem; you don't want to risk using QUIXAVER yet, so you have to use the old slow save — and the very act of doing that will clear the cassette buffer and wipe out a large part of what you've just laboriously keyed in . . . So after entering QUIXAVER, remain in MONITOR and enter

T 347 712 1347 «RETURN»

to copy what you've just entered to a higher place in memory. Then save it from there by entering

S "QUIXAVER SHIFTED", 1, 1347, 1713 «RETURN»

As QUIXAVER hasn't yet been initialised, this causes an ordinary slow-save of the shifted version of QUIXAVER (including all the miscellaneous bytes that lie between its two working parts — but no matter); and in the process it kills the original unshifted QUIXAVER. So copy the shifted version back to where it came from with

T 1347 1712 347 «RETURN»

Then initialise QUIXAVER with

G 34E «RETURN»

and you're ready to try it.

First, still in MONITOR and with a fresh tape in place, try to save QUIXAVER itself with

S "QUIXAVER", 99, 347, 348 «RETURN»

The usual things should happen, as for an ordinary slow save. If they do, rewind the tape and (still in MONITOR) enter

L «RETURN»

which again should cause the usual things to happen as for an ordinary slow-load. But when the screen shows 'FOUND QUIXAVER', '*' should appear after the filename. Then a little while after the tape is restarted the screen should turn on again; shortly after that you should see a flying cursor for about a second, and then a steady cursor (the usual MONITOR prompt). Now rewind the tape, switch off the C-16 for a few seconds, switch it on again, and this time try 'LOAD' in BASIC (no need to add device number or secondary address). QUIXAVER should load just as it did in MONITOR, ending with the BASIC 'READY' prompt.

If you get this far it's highly probable that all is well, and you can proceed to put QUIXAVER through all its paces. But if all isn't well you must switch off, switch on, go into MONITOR, slow-load 'QUIXAVER SHIFTED', copy it back to where it should be, find the errors, put them right, copy the corrected version up, slow-save it for security, copy it down again, re-initialise, and try again. Once you've got all the errors cleared up, you shouldn't need 'QUIXAVER SHIFTED' again, provided that you keep a QUIXAVEd master copy of QUIXAVER somewhere safe!

GAME REVIEW
REVIEWER :Mark Lennon
GAME REVIEWED :JOE BLADE 2

I can well remember my disbelief when I walked into my local C16/PLUS/4 stockist and saw this game,what can be termed as a big budget release had made it onto the C16.The makers of this game 'Players'were and still are great supporters of our beloved machines.If you're wondering what I mean by 'still are' I will explain.A few months back I wrote to 'Players' asking whether they were going to release any new titles ,I told them about the mag and that us C16/PLUS/4 owners were here to stay,and to my surprise 3 days later I got a reply saying that they were going to continue their support and that they were going to release a few more titles in the next few months,at Present I have heard nothing but thats not to say they wont ,we'll have to wait and see.
Anyway enough waffling on with the review.The idea of Joe Blade 2 is to rome around the dangerous streets beating up muggers and to rescue innocent citizens,to do this you have to walk in to the citizen and you will then be asked to do a sub-game,in this you have to get the numbers 1,2,3,and 4 in the correct order within a time limit ,if you do this you will have saved that citizen and you then go looking for more ,of course its not that easy,and in your way are the muggers who you can dispose of by 'kicking them in the head .The graphics in the game are excellent ,but due to memory Probs are in monochrome,but to tell the truth this does'nt de-tract from the game at all,one area which is lacking though is the sound,not one single noise through the whole of the game,again due to lack of memory.Another area where I can pick is the game is far too easy ,I completed it after 2 days,
But overall the game is great ,superb graphics,excellent playability,and all the usual 'players' finishing,and only 2.99 what a bargain.
This only leaves me to state the obvious,BUY IT! If you already have it in your collection I'm sure you will agree with me,"A superb game".

Marks out of 100%

Graphics - 92%
Sound - 0%
Playability - 80%
VFM - 98%
Overall - 95%

Company - PLAYERS
Price - £2.99

Letters

Dear Roy
Please could you send or put in the mag the following lines for the CRIBBAGE prog, from 1570 - 1620, 2980 - 3020 & 4380 - 4410 as they are not too clear or left out altogether, I'm not 'CRIBBING' (HA, HA, Peter, ED) - mind you, and thanks for the phone call and your interest.

Yours faithfully
Peter Appleby, NOTTS.

P.S Something thats just occurred to me - do you think; that one day all the club members could meet and have a get together, day/weekend, anyway something to divile on perhaps.

Thanks for the letter Peter, right the lines that were unfortunetly cut off of last months mag are below, also the get together, if members this year were to pay and extra £1/£2 this includes monthly subscribers, then maybe I can get the £75 needed to get to the all Formats Computer Fair, please don't send any money yet, I'll draw up something first.

CRIBBAGE MISSING LINES

```
1550 Q$=STR$(PO): IFPO>1THENQ$=Q$+" POINTS"+S$: ELSEQ$=Q$+" POINT "+S$
1560 CHAR, 7, 12, Q$
1570 FORA=1TO2000: NEXT
1580 IFPO>0THENSOUND1, 930, 0: MP=MP+PO: R=36: S=9: P=MP: GOSUB3350
1590 B=29: R=12: S=R: C=7: GOSUB3330: RETURN
1600 B=0: GOSUB3240: T=0: A=1
1610 FORB=0TO3: IFR%(0, B)+1=R%(0, B+1)THENA=A+1: NEXT
1620 IFA=5THENT=5: GOTO1750
1630 IFR%(0, 0)+1=R%(0, 1)ANDR%(0, 1)+1=R%(0, 2)ANDR%(0, 2)+1=R%(0, 3)THENT=T+4

2270 Q$=STR$(PO): IFPO=1THENQ$=Q$+" POINT FOR ME": ELSEQ$=Q$+" POINTS FOR ME"
2280 CHAR, 7, 12, Q$
2290 FORA=1TO1645: NEXT
2300 RETURN
2310 A=5: B=3
2320 DOWHILEB>=0
2330 IFT+V%(1, B)=15ANDM%(B)=0THENA=B: EXIT: ELSEB=B-1

2960 RETURN
2970 IFTP=2THEN3140
2980 B=1: GOSUB3320: L=0: D=1
2990 FORC=BTOTP-1
3000 IFR%(0, C)+1<>R%(0, C+1)ANDR%(0, C)<>R%(0, C+1)THEND=1: GOTO3030
3010 IFR%(0, C)<>R%(0, C+1)THEND=D+1
3020 IFD>2THENL=0
3030 NEXT

3670 C%(0, A)=P%(0, C+1): C%(1, A)=P%(1, C+1)
3680 A=A+1
```

```
3690 NEXT
3700 RETURN

4360 NEXT
4370 B=13: C=0: R=0: S=6: GOSUB3330
4380 RETURN
4390 A=RND(-TI): COLOR4, 6, 5: COLOR0, 6, 5: COLOR1, 1
4400 Q$="OUT FOR CRIB": GRAPHIC1: SCNCLR
4410 CHAR, 1, 1, Q$: CHAR, 28, 1, Q$
4420 CHAR, 1, 23, Q$: CHAR, 28, 23, Q$
```

ANOTHER LETTER!!

Dear Roy
For sometime a COMMODORE 1084s Stereo Monitor has been advertised in my local paper. Could you please tell me would it work on my C16/+4 or indeed will all the monitors work with C16/+4, C64, etc, if not why?

Yours faithfully
Peter Appleby, 18 Abbey Road, Newstead, NOTTS, NG15 OBL.

Ah, tricky one this, really, I don't know, but maybe with the correct cable it would, but remeber you would not get stereo sound, because the +4 does'nt output music via two external points, unlike the AMIGA, but because it maybe an RGB monitor then it should be suitable. Some monitors are Composite Video, which are just monochrome, but thats another game altogether, so if anyone has any help, maybe Eric Jones, you could help, then please write to the above address with info.

Missing Bit!!!!

Well, Well, Well, Peter Crack strikes again, below are lines that Peter so kindly missed out of last issues Part 10/11 (final part) of BLOOPING BUG, he says he's sorry, well, I'll let you off this time Peter, your RS232 Progs make up for the mistake, oh by the way, Peter apologises to all for any inconvenience.

```
A4398  LDA $E5
       CMP #$0F
       BEQ $4360
       NOP
       NOP
       LDA $D8
       BEQ $43D3
       LSR
       BCC $43AC
       LDX #$D8
       STX $4318
       LSR
       BCC $43B4
       LDX #$A0
       STX $4318
       LSR
       BCC $43BC
       LDX #$00
       STX $430A
       LSR
       BCC $43C4
       LDX #$50
       STX $430A
       INC $D6
       JSR $4322
       LDA $D6
       CMP #$02
       BCC $43D3
       LDA #$00
       STA $D6
       LDA $D8
       BEQ $43DF
       LDA #$23
       STA $FF
       NOP
       NOP
       NOP
       RTS
```

THATS ALL FOLKS!!

Letter

10/2/91
Roy,
I am writing (no not printing this letter to you because unfortunetly I have sold my +4 setup. I am discontinuing my membership with club without a +4.
I have just ordered an Amiga (1084S, meg, clock, second disk drive, etc), and in order to raise the received amount I needed to sell my +4.
I already have several contacts from which I can get manygames and lots of PD s/w. I hear that you have an Amiga so I hope we can exchnage programmes.
Keep the money (if any (Thanks, thats £2, ED)) from my membership fees. I have greatly enjoyed being a club member and I hope my contributions have been useful.
Thank you again for all you services and I hope you continue the great job your doing.

Yours faithfully
Matthew Newton-Lewis, W. SUSSEX.

P.S I've forwarded the club address onto the new owner of my +4. I expect he will be in touch soon.

Well what a nice letter, many thanks Matt, sorry to loose you, but I'm glad you enjoyed your time with us, and you contributions were appreciated, many thanks!!!!! I will keep in touch!!

Letters

Dear Ed
I am trying to write an adventure for the C16/+4, could club members please write in with information on any articles or books which may help also any features they have would like to see in it & any pet hates they would like to see left out.

Peter Crack, 88 Burleigh Road, ENFIELD, MIDDX, EN1 1NX. Tel 081-367-3152

The above phone number is available wanting a breakdown of the bit of Blooping Bug that was missed out of the last ish.

Review

Title: Digital Ball (64K)
Publisher: NOVOTRADE (of Hungary)
Price: ?
Reviewer: Andy Tang, LONDON.

Digital Ball (also called Faltenisz) is probably the best breakout game available for the plus/4 (I think Tony Sexton will agree there, ED), it is produced by Novotrade of Hungary where the plus/4 is a popular computer - I got my copy of the game from Ronald de Bruin (thanks Ronald!)

I think you all know what the gameplay is like : it plays very similar to ArthurNoid, apart from different types of bonuses available and the way you aquire them - those of you who have arthurnoid know that bonuses are gained byg collecting the barrels which roll down the screen, but this is not the case in Digital Ball where theres a small extra section of playing area in the left wall/ boundary which stores a bonus, usually a door blocks off this area from the bat; but occasionally the door is removed (for a very short time!) to allow the bat access to the bonus. You have to be fast in grabbing the bonus, if not you could be trappped as the door reappears blocking you off from the main playing area - this normally results with you losing a life, unless you have previously collected a 'BONUS WALL'. Other bonuses include, BOMBS and the very useful EXTRA LIFE. The game has 50 levels.

The main reason why Digital Ball stands out fom the rest od the breakout clones is because of its presentation: A good loading screen - Picture of a dragon like creature, nicely drawn colourful graphics with a space backdrop, good sound effects, and excellent digitized speech!!!!!

Theres also another in built game thrown in free - though this game is not better then Digital Ball, its a nice bonus.

All you plus/4 owners who want a good breakout game, I recommend you get Digital Ball.

Graphics: 8
Sound: 9
Playability: 9
Value: Can't rate - don't know price!
Overall: 9

Note: This game can be purchased directly from Novotrade: (though you have to write a letter asking for a s/w list and prices):-

Address:
Novotrade RT
2C Szamitastechnikai szakuzlet
1136 Budapest
Balz c u. 35
HUNGARY

OR a copy can be obtained by sending a disk/C60 tape + £2 cheque/PO (to cover P&P and copy fee) from Roy Robinson, 112 Cliff Road, HORNSEA, N. HUMBERSIDE, HU18 1JE, all proceeds will go to the Stall costs for a computer show.

Here is a list of all C16/+4 breakout games that are available so far...

Digital Ball Novotrade
ArthurNoid Alternative
Reflex Players
Demolition Anco
Jailbreak Bug Byte, Breakout Melbourne House (programmed in basic)
Video Classics Silverbird

C16/+4 News

Non-UK C16/+4 Software: the shape of things to come ?

All you club members who are considering upgrading to different machines, I ask you to think again because although new s/w is virtually non-existant here, in other European Countries; the C16/+4 are very much alive! In Hungary, Germany and Holland new software is still being produced anything from classic games like 'ELITE' (yes that 3d space trading game is available for the +4), Digitized graphic & music demos, to the latest utilities (ie, GEOS, (are you sure ANDY, ED)).
The good thing is that alot of this software is Public Domain which is finding its way over here in Britain.
You might think PD s/w is of poor quality (how very wrong, ED) but take it from me its NOT! for example:
I have Tir Na Nog a brilliant 64K arcade adventure with excellent animated graphics its virtually the same as the C64 version released a few years ago by Gargoyle Games (this is currently one of my favourite games!).
Another good thing about PD s/w is that its cheap, since you only pay a small fee for copying and P7P, just think about the poor Amiga owners who have to fork out £25 each times they buy a game (I don't, oops, less said the better, ED).

You all can obtain some of this non-uk PD s/w from the clubs PD Service (if I ever get it going, ED), but you first have to check with Roy (ED) Robinson for the arrangements (maybe copying fees, etc?). And remember with 1992 coming up it would be alot easier to but commercial s/w (such as Patric & Digital Ball) for you computers, I'm one for a single European Currency (but Andy please remember theres good points aswell as bad about one currency, ie, One European Bank, NOT FOR ME, and one European Leader, DEFINITLY NOT FOR ME, I'm very British and the Pounds & Pence must stay, maybe with a PARALEL EUROPEAN CURRENCY, ED!).

I hope this news encourages other members to continue to use their C16 or +4 machine(s) for a long time to come.

Please Note:
Alot of Credit for the introduction of non-uk Pd s/w into this club goes to Ronald de Bruin and his Father Gerard de Bruin of HOLLAND for there priceless help, many thanks for the help so far from Roy Robinson & Andy Tang, (Ronald, drop me a line, ROY).
The s/w has come from RONSOFT UNLIMITED,
Many thanks RONALD & GERARD! ! ! !

Here are some other titles to wet your appetite:
* One On One (basketball game)
* Super Cobra (???)
  Godzilla (monster game, quite good because it uses an expanded screen!)
* Pink Panther (speaks for itself)
* Davids Midnight Magic (Pinball Clasic!)

* = Plus/4 only or C16 +64K

The above news was compiled by Andy Tang, LONDON.