YOUR £1.50

# COMMODORE

# SERIOUS USERS GUIDE 1988

THE ULTIMATE PROGRAM COLLECTION FOR C64 AND C128 OWNERS

INCLUDING:

DISK OPERATING SYSTEM
PROGRAM COMPACTER
128 FONT EDITOR
MESSAGE CONSTRUCTION KIT
GTX COMPILER
DISK CATALOGUE
MOUSE MANAGER

TECHNICAL INFORMATION FOR THE C64 AND C128

| LIFESAVERS 1 | C64 | DISK/TAPE DEFAULT | 1/1 |
|---|---|---|---|

This handy routine allows you to set the default device number totape or disk for all loading and saving operations. Once it is set, you can then just type in LOAD "filename - you don't even need the final quotes.

If a machine code program is to be saved or loaded then the reset(SYS 720) and normal syntax will have to be used (eg ,8,1).

The program works by redirecting the load and save vectors at 816and 818 to this new routine at 679. To set the default to disk,use SYS 679 and to reset it to tape use SYS 708.

```
10 REM***************************
20 REM*     DEVICE SET          *
30 REM***************************
40 REM * SYS 679 :- DEFAULT TO
DISK *
50 REM * SYS 708 :- DEFAULT TO
TAPE *
60 REM * SYS 720 :- RESET TO
NORMAL *
70 REM***************************
80 FOR L=679 TO 758:READ A:POKE
L,A:D=D+A:NEXT
90 IF D<>8969 THEN PRINT "DATA
ERROR":END
100 DATA 169,8,141,230,2,141,241,
2,169,229,141,48,3,169,240,141
110 DATA 50,3,169,2,141,49,3,169,
2,141,51,3,96,32,175,2
120 DATA 169,1,141,230,2,141,
241,2,96,169,165,141,48,3,169,244
130 DATA 141,49,3,169,237,141,50,
3,169,245,141,51,3,96,169,8
140 DATA 133,186,169,0,133,10,76,
165,244,169,8,133,186,76,237,245
```

| LIFESAVERS 2 | C64 | LISTING PAUSE | 1/1 |
|---|---|---|---|

The problem with the Commodore 64 operating system is that alisted program shoots up the screen too quickly to read. This leaves the user having to laboriously list lines in groups of about ten.

This program solves the problem by allowing the user to pause the scrolling program by pressing the spacebar. The routine is placed high in the 49152 block of memory at 53200 and therefore leaves plenty of scope for using programs which call up other code routines in this area.

Type in the program and save it. When you think you may wish to use it, load it in before doing anything else with the computer and RUN. It will automatically execute itself so all you then have to do is to remember to press the spacebar.

```
10 C=0:REM LIST STOP BY STEPHEN
ELMER
20 FOR I=53200 TO 53245:READ
A:C=C+A:POKE I,A:NEXT
30 IF C<>6879 THEN PRINT"ERROR IN
DATA":END
40 SYS 53200
50 DATA 169,219,141,38,3,169,207,
141
60 DATA 39,3,96,72,165,204,201,1
70 DATA 208,24,165,197,201,60,
208,18
80 DATA 201,60,208,250,165,197,
201,60
90 DATA 240,250,104,76,202,241
```

YOUR
# COMMODORE SERIOUS USERS GUIDE 1988

Editor:
Stuart Cooke
Deputy Editor:
Eric Doyle
Typesetting:
Magazine Typesetters
Design:
Wakeworth Design
Artist:
Alan Batchelor

Within these pages you will find information to help any serious C64 or C128 user to get the best from their computer. Utilities for programmers and other computer users are backed-up by an informative technical section and an extensive hints and tips guide.

If your forte is Basic programming, the GTX Compiler can convert a program to run at over 30 times its normal speed.

A program also needs style if it is going to impress anyone and the 128 Font Editor and the Message Construction Kit for the C64 provide easy routes to adding a 'designer look' to your routines.

Instead of attaching a printer or disk drive to the serial port, using it to link through to another C64, C128, Plus 4 or C16 can add a new dimension to games playing. With the Bus Route 64 program you possess a key which opens up the world of interactive, two player games.

On the technical side there are memory maps of the C128, C64 and 1541 disk drive with detailed tables of many more vital statistics in a quick reference format.

The Your Commodore Serious Users Guide is more than a magazine, it's a reference guide that deserves a place beside every Commodore 64 or 128.

# Contents

**Listings** 4
A guide to help you to enter the Serious User programs.

**Mouse Manager** 6
Whip your NEOS mouse into shape.

**Sticky Cursors** 9
Joystick control is at hand.

**64 Tips for the 64** 10
A comprehensive list of programming hints

**A Bit More** 17
Get into hi-res without losing your memory.

**Diskos** 27
Window control sees you through drive difficulties.

**Mailing List 128** 32
Keeps you in touch with everyone.

**Message Construction Kit** 36
Liven up your listings with a superior scroll.

**Disk Cat** 44
Keep track of your programs.

**Program Compactor** 48
Squeeze more programs into less disk space.

**Super Index** 52
Put all your favourite articles at your fingertips.

**Technical Information** 56
Full memory maps for the C64, C128 and 1541 disk drive.

**128 Font Editor** 68
A utility to give character to your programs.

**GTX Compiler** 73
Gives the go-faster stripe to BASIC routines.

**Bus Route 64** 81
Conduct a conversation with another computer.

Your Commodore is a monthly magazine appearing on the first Friday of each month. Argus Specialist Publications Limited Editorial & Advertisement Office, Your Commodore, No 1 Golden Square, London W1R 3AB. Telephone: 01-437 0626. Telex: 8811896. Subscription rates upon application to Your Commodore Subscriptions Department, Infonet Ltd, 5 River Park Estate, Berkhamsted, Herts, HP4 1HL. U.S.A. Subscription Agent: Wise Owl Worldwide Publications, 4314 West 238th Street, Torrance CA 90505 U.S.A. Distribution SM Distribution, 6 Leigham Court Road, London SW16 2PG. Printed by Chase Web, Plymouth.

# Listings

*Get it right first time with our deluxe program system for the C64.*

You may have noticed that our listings are free of those horrible little black blobs which send you searching around the keyboard for a suitable graphic symbol. You may also have noticed the funny numbers by the side of each line of the listing. Fret no more, it's all part of our easy entry aid.

Instead of those nasty graphics and rows of countless spaces in PRINT statements and strings we use a special coding system. The code, or mnemonic, is always contained in square brackets and you'll soon learn to decipher their meanings.

For example, [SA] would mean type in a Shifted A, or an ace of spades in layman's terms, and [SA10] would mean a row of ten of these symbols.

[S+2] means hold down the shift key and press the plus key twice. It doesn't take a great leap of logic to realise that [C+2] means exactly the same thing except that the Commodore key (bottom left of the keyboard) is held down instead of the shift key.

If more than two spaces appear in a statement then this will be printed as [SPC4] or, exceptionally, [SSPC4]. Translated into English this means press the spacebar four times or in the latter case hold the shift key down while you do it.

A string of special characters could appear as:

[CTRL N, DOWN2,LEFT5,BLUE, F3,C3]

This would be achieved by holding down the CTRL key as you press N, press the cursor key down twice, the cursor left key five times, press the key marked BLUE while holding down the CTRL key, press the F3 key and, finally hold the Commodore key down while pressing the number two key (C2 would of course make the computer print in brown).

Always remember that you should only have a row of graphics characters on your screen with no square brackets and no commas, unless something like this appears:

[SS],[C*]

In this case the two characters should have a comma between them.

On rare occasions [REV T] will appear in a listing. This is a delete symbol and is created by entering the line up to this mnemonic. Then type a closing quotation mark (SHIFT & 2) and delete it. This gets the computer out of quotes mode. Hold down CTRL and press the number nine key (RVSON), type the relevant number of reversed T's and then hold down CTRL and press zero (RVSOFF). Next type another quotation mark and delete it again. Now finish the line and press RETURN.

A list of these special cases is given in the table but remember that only one of these mnemonics will appear outside of a PRINT string: the symbol for pi. This may appear when its value is needed in a calculation so this may look something like:

:CC=2*[PI]*R:

Ignore the square brackets and just type in a shifted upward pointing arrow (ie. the pi symbol).

PROGRAM: SYNTAX CHECKER

```
5 REM SYTAX CHECKER - ERIC DOYLE

10 BL=10    :LN=70    :SA=49152
20 FOR L=0 TO BL:CX=0:FOR D=0 TO
 15
30 READ A:IF A>255THENPRINT"NUMB
ER TO LARGE";LN+(L*10):STOP
40 CX=CX+A:POKE SA+L*16+D,A:NEXT
 D
50 READ A:IF A><CX THENPRINT"ERR
OR IN LINE";LN+(L*10):STOP
60 NEXT L:SYS 49152:NEW
70 DATA 173,5,3,201,165,208,31,1
20,169,9,141,32,208,141,33,208,1
847
80 DATA 169,7,141,134,2,169,13,3
2,210,255,169,64,141,4,3,169,168
2
90 DATA 192,141,5,3,88,96,120,16
9,124,141,4,3,169,165,141,5,1566

100 DATA 3,169,14,141,134,2,141,
32,208,169,6,141,33,208,88,96,15
85
110 DATA 32,124,165,72,138,72,15
2,72,162,0,165,20,133,254,165,21
,1747
120 DATA 24,101,254,133,254,189,
0,2,240,18,69,254,133,254,232,18
9,2346
130 DATA 0,2,240,8,24,101,254,13
3,254,232,208,233,169,1,141,134,
2134
140 DATA 2,165,254,74,74,74,74,3
2,156,192,32,210,255,165,254,41,
2054
150 DATA 15,32,156,192,32,210,25
5,169,13,32,210,255,169,13,32,21
0,1995
160 DATA 255,169,7,141,134,2,104
,168,104,170,104,96,24,105,48,20
1,1832
170 DATA 58,16,1,96,24,105,7,96,
0,0,0,0,0,0,0,0.403
```

**by Eric Doyle**

| Mnemonic | Symbol | Keypress |
|---|---|---|
| [RIGHT] | | CRSR left/right |
| [LEFT] | | SHIFT & CRSR left/right |
| [DOWN] | | CRSR up/down |
| [UP] | | SHIFT & CRSR up/down |
| [F1] | | f1 key |
| [F2] | | SHIFT & f1 key |
| [F3] | | f3 key |
| [F4] | | SHIFT & f3 key |
| [F5] | | f5 key |
| [F6] | | SHIFT & f5 key |
| [F7] | | f7 key |
| [F8] | | SHIFT & f7 key |
| [HOME] | | CLR/HOME |
| [CLR] | | SHIFT & CLR/HOME |
| [RVSON] | | CTRL & 9 |
| [RVSOFF] | | CTRL & 0 |

| Mnemonic | Symbol | Keypress |
|---|---|---|
| [BLACK] | | CTRL & 1 |
| [WHITE] | | CTRL & 2 |
| [RED] | | CTRL & 3 |
| [CYAN] | | CTRL & 4 |
| [PURPLE] | | CTRL & 5 |
| [GREEN] | | CTRL & 6 |
| [BLUE] | | CTRL & 7 |
| [YELLOW] | | CTRL & 8 |
| [POUND] | | £ |
| [LARROW] | | ← |
| [UPARROW] | | ↑ |
| [PI] | | SHIFT & ↑ |
| [INST] | | SHIFT & INST/DEL |
| [REV T] | | see text |
| [Cletter] | | CBM + letter |
| [Sletter] | | SHIFT + letter |

## Checksum Program

The hexadecimal numbers appearing in a column to the left of the listing should not be typed in with the program. These are merely checksum values and are there to help you get each line right. Don't worry if you don't understand the hexadecimal system, as long as you can compare two characters on the screen with the corresponding two characters in the magazine you can use our line checking program.

Type in the Checksum Program, make sure that you've not made any mistakes and save it to tape or disk immediately because it will be used with most of the present and future listings appearing in Your Commodore.

At the start of each programming session, load Checksum and run it. The screen will turn brown with yellow characters and each time you type in a line and press the RETURN key a number will appear on the screen in white. This should be the same as the corresponding value in the magazine.

If the two values don't relate to one another, you have not copied the line exactly as printed so go back and check each character carefully. When you find the error simply correct it and press RETURN again.

If you want to turn off the checker simply type SYS49152 and the screen will return to the familiar blue colours. You can then do whatever it was you wanted to do and if this doesn't use the area where Checksum lies you can go back to it with the same SYS command.

No system is foolproof but the chances of two errors cancelling one another out are so remote that we believe our listings are more reliable than any other magazine in the world. So get typing!

# Mouse

*Type-in this listing for C64 mice and see how they run!*

***by W I Sellers***

The Neos mouse included in the Commodore Connoisseur's Collection is extremely good at its job. It has two operating modes and can act as a joystick emulator to make it widely compatible with commercial software, or in its standard mode, which allows much smoother control. The main drawback is the weak documentation explaining how the mouse can be incorporated in your own programs. The following routine enables you to read the position of the mouse from Basic.

## How It Works

The mouse transmits its movement as an X and Y displacement. The program is a machine-code routine which receives the movement and assigns the values automatically to two Basic variables, DX and DY. The status of the left-hand button is returned in the variable FB. These variable names are created within the routine, so they don't need to be previously assigned.

To use the mouse, switch off the normal function of the keyboard and call the mouse set-up routine at decimal 50003. The mouse-read routine at decimal 50000 must be called each time its position needs to be updated. DX and DY contain the distance moved in the X and Y direction since the last time the read routine was called.

For machine-code users, the subroutine that performs the mouse read is at $C3A3 to $C42C with the X and Y displacements and button status stored from $C42D to $C42F. The rest of the program is concerned with assigning the values to the Basic variables.

## In Use

The program is in the form of a simple demonstration with the mouse used to move a sprite round the screen. Lines 470–920 form a subroutine that creates the machine-code routine and it's this bit that needs to be incorporated into other programs.

You need to insert the mouse after the program has been started because the keyboard produces nonsense with the mouse in place. Save a copy of the program before running it in case there are any mistakes that might lead to a crash. The RUN/STOP key is disabled by the program so that the program can only be stopped by RUN/STOP and RESTORE being pressed simultaneously.

```
LINE# LOC. OBJECT LABELS          LINE

   10 0000                        ;PAGE ZERO LABELS
   20 0000                        ;
   30 0000          NAME          =$45
   40 0000          ADDRESS       =$47
   50 0000          SIGN          =$66
   60 0000                        ;
   70 0000                        ;BASIC ROM ROUTINES
   80 0000                        ;
   90 0000          FINDVAR       =$B0E7
  100 0000          CONVERTYTOFLP =$B3A2
  110 0000          FLPTOMEM      =$BBD4
  120 0000                        ;
  130 0000                        ;
  140 0000                        ;IN/OUT ADDRESSES
  150 0000                        ;
  160 0000          PADDLEX       =$D419
  170 0000          PORT2         =$DC00
  180 0000          DDR2          =$DC02
  190 0000          ICR1          =$DC0D
  200 0000          ICR2          =$DD0D
  210 0000          CRA1          =$DC0E
  220 0000          CRB1          =$DC0F
  230 0000          CRA2          =$DD0E
  240 0000          CRB2          =$DD0F
  250 0000                        ;
  260 0000                        ;START
  270 0000                        ;
  280 C350                        .ORG    50000
  290 C350                        ;
  300 C350                        ;
  310 C350                        ;JUMP TABLE
  320 C350                        ;
  330 C350 4C58C3 READ            JMP START
  340 C353 4C30C4 SETUP           JMP INITIALIZE
  350 C356                        ;
  360 C356                        ;MAIN ROUTINE TO READ MOUSE
  370 C356                        ;AND PUT RESULTS IN 3 VARIABLES
  380 C356                        ;DX, DY AND FB
  390 C356                        ;
  400 C356 20A3C3 START           JSR MOUSEREAD
  410 C359                        ;
  420 C359                        ;DEAL WITH DELX
  430 C359                        ;
  440 C359 AD2DC4                 LDA DELX
  450 C35C A244                   LDX #'D'
  460 C35E A058                   LDY #'X'
  470 C360 2078C3                 JSR STOREACC
  480 C363                        ;
  490 C363                        ;DEAL WITH DELY
  500 C363                        ;
  510 C363 AD2EC4                 LDA DELY
  520 C366 A244                   LDX #'D'
  530 C368 A059                   LDY #'Y'
```

```
LINE# LOC. OBJECT LABELS        LINE

  540 C36A 2078C3               JSR STOREACC
  550 C36D                      ;
  560 C36D                      ;DEAL WITH BUTTON
  570 C36D                      ;
  580 C36D AD2FC4               LDA BUTTON
  590 C370 A246                 LDX #'F'
  600 C372 A042                 LDY #'B'
  610 C374 2078C3               JSR STOREACC
  620 C377                      ;
  630 C377 60                   RTS
  640 C378                      ;
  650 C378                      ;ROUTINE TO STORE ACCUMULATOR
  660 C378                      ;AS BASIC VARIABLE IN XY
  670 C378                      ;
  680 C378                      ;STORE REGISTERS
  690 C378                      ;
  700 C378 48     STOREACC      PHA
  710 C379 8645                 STX NAME
  720 C37B 8446                 STY NAME+1
  730 C37D                      ;
  740 C37D                      ;PUSH DUMMY VALUE ON STACK
  750 C37D                      ;
  760 C37D A9FF                 LDA #$FF
  770 C37F 48                   PHA
  780 C380                      ;
  790 C380                      ;FIND VARIABLE LOCATION
  800 C380                      ;
  810 C380 20E7B0               JSR FINDVAR
  820 C383                      ;
  830 C383                      ;GET RID OF DUMMY
  840 C383                      ;
  850 C383 68                   PLA
  860 C384                      ;
  870 C384                      ;CONVERT A (2'S COMPLEMENT) TO
  880 C384                      ;FLOATING POINT
  890 C384                      ;
  900 C384 68                   PLA
  910 C385                      ;
  920 C385                      ;CHECK IF POSITIVE
  930 C385                      ;
 ---940 C385 100A               BPL SA1
  950 C387                      ;
  960 C387                      ;CONVERT NEGATIVE NUMBER
  970 C387                      ;
  980 C387 49FF                 EOR #%11111111
  990 C389 A8                   TAY
 1000 C38A C8                   INY
 1010 C38B 20A2B3               JSR CONVERTYTOFLP
 1020 C38E 4C9BC3               JMP SA2
 1030 C391                      ;
 1040 C391                      ;CONVERT POSITIVE NUMBER
 1050 C391                      ;
 1060 C391 A8     SA1           TAY
 1070 C392 20A2B3               JSR CONVERTYTOFLP
 1080 C395 A566                 LDA SIGN
 1090 C397 49FF                 EOR #%11111111
 1100 C399 8566                 STA SIGN
 1110 C39B                      ;
 1120 C39B                      ;STORE FLOATING POINT ACCUMULATOR
 1130 C39B                      ;
 1140 C39B A647   SA2           LDX ADDRESS
 1150 C39D A448                 LDY ADDRESS+1
 1160 C39F 20D4BB               JSR FLPTOMEM
 1170 C3A2                      ;
 1180 C3A2                      ;RETURN
 1190 C3A2                      ;
 1200 C3A2 60                   RTS
 1210 C3A3                      ;
 1220 C3A3                      ;ROUTINE TO READ MOUSE POSITION
 1230 C3A3                      ;
 1240 C3A3                      ;DISABLE INTERRUPTS
 1250 C3A3                      ;
 1260 C3A3 78     MOUSEREAD     SEI
 1270 C3A4                      ;
 1280 C3A4                      ;STORE IN/OUT REGISTERS
 1290 C3A4                      ;
 1300 C3A4 AD00DC               LDA PORT2
 1310 C3A7 48                   PHA
 1320 C3A8 AD02DC               LDA DDR2
 1330 C3AB 48                   PHA
 1340 C3AC                      ;
 1350 C3AC                      ;SET BIT 4 TO OUTPUT
 1360 C3AC                      ;
 1370 C3AC A910                 LDA #$10
 1380 C3AE 8D02DC               STA DDR2
 1390 C3B1                      ;
 1400 C3B1                      ;CLEAR BIT 4
 1410 C3B1                      ;
 1420 C3B1 AD00DC               LDA PORT2
 1430 C3B4 29EF                 AND #$EF
 1440 C3B6 8D00DC               STA PORT2
 1450 C3B9                      ;
 1460 C3B9                      ;PAUSE 80 CYCLES
 1470 C3B9                      ;
 1480 C3B9 A008                 LDY #$08
 1490 C3BB 2026C4               JSR PAUSE
 1500 C3BE                      ;
 1510 C3BE                      ;GET HIGH NYBBLE
 1520 C3BE                      ;
 1530 C3BE AD00DC               LDA PORT2
 1540 C3C1 0A                   ASL A

LINE# LOC. OBJECT LABELS        LINE

 1550 C3C2 0A                   ASL A
 1560 C3C3 0A                   ASL A
 1570 C3C4 0A                   ASL A
 1580 C3C5 8D2DC4               STA DELX
 1590 C3C8                      ;
 1600 C3C8                      ;SET BIT 4
 1610 C3C8                      ;
 1620 C3C8 AD00DC               LDA PORT2
 1630 C3CB 0910                 ORA #$10
 1640 C3CD 8D00DC               STA PORT2
 1650 C3D0                      ;
 1660 C3D0                      ;PAUSE 50 CYCLES
 1670 C3D0                      ;
 1680 C3D0 A005                 LDY #$05
 1690 C3D2 2026C4               JSR PAUSE
 1700 C3D5                      ;
 1710 C3D5                      ;GET LOW NYBBLE
 1720 C3D5                      ;
 1730 C3D5 AD00DC               LDA PORT2
 1740 C3D8 290F                 AND #$0F
 1750 C3DA 0D2DC4               ORA DELX
 1760 C3DD 8D2DC4               STA DELX
 1770 C3E0                      ;
 1780 C3E0                      ;CLEAR BIT 4
 1790 C3E0                      ;
 1800 C3E0 AD00DC               LDA PORT2
 1810 C3E3 29EF                 AND #$EF
 1820 C3E5 8D00DC               STA PORT2
 1830 C3E8                      ;
 1840 C3E8                      ;PAUSE 50 CYCLES
 1850 C3E8                      ;
 1860 C3E8 A005                 LDY #$05
 1870 C3EA 2026C4               JSR PAUSE
 1880 C3ED                      ;
 1890 C3ED                      ;GET HIGH NYBBLE
 1900 C3ED                      ;
 1910 C3ED AD00DC               LDA PORT2
 1920 C3F0 0A                   ASL A
 1930 C3F1 0A                   ASL A
 1940 C3F2 0A                   ASL A
 1950 C3F3 0A                   ASL A
 1960 C3F4 8D2EC4               STA DELY
 1970 C3F7                      ;
 1980 C3F7                      ;SET BIT 4
 1990 C3F7                      ;
 2000 C3F7 AD00DC               LDA PORT2
 2010 C3FA 0910                 ORA #$10
 2020 C3FC 8D00DC               STA PORT2
 2030 C3FF                      ;
 2040 C3FF                      ;PAUSE 50 CYCLES
 2050 C3FF                      ;
 2060 C3FF A005                 LDY #$05
 2070 C401 2026C4               JSR PAUSE
 2080 C404                      ;
 2090 C404                      ;GET LOW NYBBLE
 2100 C404                      ;
 2110 C404 AD00DC               LDA PORT2
 2120 C407 290F                 AND #$0F
 2130 C409 0D2EC4               ORA DELY
 2140 C40C 8D2EC4               STA DELY
 2150 C40F                      ;
 2160 C40F                      ;READ BUTTON
 2170 C40F                      ;
 2180 C40F AD19D4               LDA PADDLEX
 2190 C412 C9FF                 CMP #$FF
 2200 C414 D001                 BNE MR1
 2210 C416                      ;
 2220 C416                      ;$FF IN A IF BUTTON PRESSED
 2230 C416                      ;
 2240 C416                      ;SKIP NEXT INSTRUCTION
 2250 C416                      ;
 2260 C416 2C                   .BYTE   $2C
 2270 C417                      ;
 2280 C417 A900   MR1           LDA #$00
 2290 C419 8D2FC4               STA BUTTON
 2300 C41C                      ;
 2310 C41C                      ;RECOVER IN/OUT REGISTERS
 2320 C41C                      ;
 2330 C41C 68                   PLA
 2340 C41D 8D02DC               STA DDR2
 2350 C420 68                   PLA
 2360 C421 8D00DC               STA PORT2
 2370 C424                      ;
 2380 C424                      ;ENABLE INTERRUPTS AND RETURN
 2390 C424                      ;
 2400 C424 58                   CLI
 2410 C425 60                   RTS
 2420 C426                      ;
 2430 C426                      ;ROUTINE TO PAUSE Y * 10 CYCLES
 2440 C426                      ;
 2450 C426 EA     PAUSE         NOP
 2460 C427 EA                   NOP
 2470 C428 EA                   NOP
 2480 C429 88                   DEY
 2490 C42A D0FA                 BNE PAUSE
 2500 C42C 60                   RTS
 2510 C42D 00     DELX          .BYTE   0
 2520 C42E 00     DELY          .BYTE   0
 2530 C42F 00     BUTTON        .BYTE   0
 2540 C430                      ;
 2550 C430                      ;INITIALIZE MOUSE READER
```

```
LINE# LOC. OBJECT LABELS          LINE

 2560 C430                        ;
 2570 C430                        ;DISABLE INTERRUPTS
 2580 C430                        ;
 2590 C430 78      INITIALIZE     SEI
 2600 C431                        ;
 2610 C431                        ;CLEAR INTERRUPT MASKS ON CIA'S
 2620 C431                        ;
 2630 C431 A97F                   LDA #$7F
 2640 C433 8D0DDC                 STA ICR1
 2650 C436 8D0DDD                 STA ICR2
 2660 C439                        ;
 2670 C439                        ;SET PORT TO OUTPUT
 2680 C439                        ;
 2690 C439 A9FF                   LDA #$FF
 2700 C43B 8D02DC                 STA DDR2
 2710 C43E A97F                   LDA #$7F

LINE# LOC. OBJECT LABELS          LINE

 2720 C440 8D00DC                 STA PORT2
 2730 C443                        ;
 2740 C443                        ;SET TIMERS TO DEFAULT VALUES
 2750 C443                        ;
 2760 C443 A908                   LDA #$08
 2770 C445 8D0EDC                 STA CRA1
 2780 C448 8D0FDC                 STA CRB1
 2790 C44B 8D0ED0                 STA CRA2
 2800 C44E 8D0FD0                 STA CRB2
 2810 C451                        ;
 2820 C451                        ;ENABLE INTERRUPTS AND RETURN
 2830 C451                        ;
 2840 C451 58                     CLI
 2850 C452 60                     RTS
 2860 C453                        ;
 2870 C453                        .END
```

PROGRAM: MOUSE MANAGER

```
05  10 REM MOUSE CODE ←THIS IS E
    SSENTIAL IF THE SAVE ROUTINE
     IS USED!!!!!
CC  20 REM LINES 10-70 CAN BE OM
    ITTED IF YOU DON'T WANT TO S
    AVE THE CODE
02  30 FORA=0TO62:READX:NEXT:GOS
    UB 550
7A  40 POKE193,80:POKE194,195:PO
    KE174,83:POKE175,196
DA  50 POKE 187,7:POKE188,8
17  60 POKE 183,10:POKE 186,8:RE
    M POKE 186,1 FOR CASSETTE
63  70 POKE 185,0:SYS62954
1A  80 REM MOUSE MANAGER DEMO PR
    OGRAM
52  90 REM (C)OPYRIGHT 1987 W.I.
    SELLERS
7A  100 REM CLEAR SCREEN
A1  110 PRINT"[CLR]POKING IN DAT
    A...."
86  120 REM POKE IN SPRITE DATA
11  130 RESTORE:FORS=832TO894:RE
    ADA:POKE53280,A:POKES,A:NEXT

9B  140 REM POKE IN MACHINE CODE

95  150 GOSUB550
1C  160 PRINT"[CLR]INSERT MOUSE
    IN JOYSTICK PORT2"
51  170 PRINT"[DOWN]AND PRESS <S
    PACE> WHEN READY"
6E  180 GETA$:IFA$<>" "THEN180
9F  190 PRINT"[CLR]";:POKE53281,
    0
C1  200 REM INITIALIZE MOUSE REA
    DER
04  210 SYS50003
52  220 REM INITIALIZE SPRITE 0
45  230 POKE2040,13
FE  240 V=53248
E7  250 POKEV+21,1
B9  260 REM INITIALIZE POSITION
C1  270 X=160:Y=100:C=0
FD  280 REM READ MOUSE
CC  290 SYS50000:PRINTDX,DY
6C  300 REM DX=X CHANGE DY=Y CHA
    NGE
3B  310 REM FB=1 IF PRESSED
70  320 X=X+DX:Y=Y+DY
38  330 REM CHECK X AND Y LIMITS
08  340 IFX<0THENX=0
10  350 IFX>511THENX=511
FA  360 IFY<0THENY=0
62  370 IFY>255THENY=255
44  380 IFC>255THENC=0
B6  390 REM CALCULATE X VALUES
70  400 X2=INT(X/256):X1=X-256*X
    2
2C  410 C=C+FB:IFC=256THENC=0
96  420 POKEV,X1:POKEV+1,Y:POKEV
    +16,X2
1F  430 IF(CAND15)=0THENC=C+1
C7  440 POKEV+39,C
75  450 GOTO290
45  460 REM SPRITE DATA
F8  470 DATA0,0,0,0,0,0,0,0
D6  480 DATA0,0,0,0,0,0,0,8
E1  490 DATA0,8,28,0,28,60,0,30
12  500 DATA126,34,63,127,182,25
    5,127,255
58  510 DATA255,124,73,31,120,12
    7,15,112
05  520 DATA62,7,48,20,6,16,0,4
4B  530 DATA8,0,8,0,0,0,0,0
2D  540 DATA0,0,0,0,0,0,0,0
95  550 REM POKE IN MOUSE READER
     ROUTINE
87  560 REM FROM $C350 TO $C452
91  570 D=50000:L=600
F4  580 T=0
A0  590 FORX=1TO8
17  600 READA:IFA=-1THEN660
FE  610 POKED,A:D=D+1:T=T+A
78  620 POKE53280,A:NEXT
99  630 READA:IFA=-1THEN660
FC  640 IFT<>ATHENPRINT"ERROR IN
     LINE ";L:END
F1  650 L=L+10:GOTO580
DA  660 IFD<>50258+1THENPRINT"AD
    DRESS ERROR":END
2E  670 RETURN
26  680 DATA76,86,195,76,48,196,
    32,163,872
40  690 DATA195,173,45,196,162,6
    8,160,88,1087
90  700 DATA32,120,195,173,46,19
    6,162,68,992
7C  710 DATA160,89,32,120,195,17
    3,47,196,1012
DE  720 DATA162,70,160,66,32,120
    ,195,96,901
24  730 DATA72,134,69,132,70,169
    ,255,72,973
FF  740 DATA32,231,176,104,104,1
    6,10,73,746
C3  750 DATA255,168,200,32,162,1
    79,76,155,1227
7D  760 DATA195,168,32,162,179,1
    65,102,73,1076
25  770 DATA255,133,102,166,71,1
    64,72,32,995
54  780 DATA212,187,96,120,173,0
    ,220,72,1080
B4  790 DATA173,2,220,72,169,16,
    141,2,795
9A  800 DATA220,173,0,220,41,239
    ,141,0,1034
A1  810 DATA220,160,8,32,38,196,
    173,0,827
4D  820 DATA220,10,10,10,10,141,
    45,196,642
46  830 DATA173,0,220,9,16,141,0
    ,220,779
9B  840 DATA160,5,32,38,196,173,
    0,220,824
60  850 DATA41,15,13,45,196,141,
    45,196,692
FA  860 DATA173,0,220,41,239,141
    ,0,220,1034
05  870 DATA160,5,32,38,196,173,
    0,220,824
11  880 DATA10,10,10,10,141,46,1
    96,173,596
C6  890 DATA0,220,9,16,141,0,220
    ,160,766
D3  900 DATA5,32,38,196,173,0,22
    0,41,705
F9  910 DATA15,13,46,196,141,46,
    196,173,826
E8  920 DATA25,212,201,255,208,1
    ,44,169,1115
0E  930 DATA0,141,47,196,104,141
    ,2,220,851
96  940 DATA104,141,0,220,88,96,
    234,234,1117
F8  950 DATA234,136,208,250,96,0
    ,0,0,924
BE  960 DATA120,169,127,141,13,2
    20,141,13,944
CF  970 DATA221,169,255,141,2,22
    0,169,127,1304
3A  980 DATA141,0,220,169,8,141,
    14,220,913
DD  990 DATA141,15,220,141,14,22
    1,141,15,908
9E  1000 DATA221,88,96,-1
```

# Sticky Cursors

*Here's an interrupt which gives a superior method of cursor control*

This is a utility to allow a joystick, plugged into Port 2, to emulate the cursor keys. The program is interrupt-driven and resides in an unused area of memory. It works by checking the status of port 2 every so often and when it finds that the joystick is not centred, the appropriate control code is inserted into the keyboard buffer queue.

The joystick will perform the following functions:

| POSITION | FUNCTION |
|---|---|
| Centred | Nothing |
| Up | Move cursor up a line at a time |
| Down | Move cursor down a line at a time |
| Left | Move cursor left (with wrap around) |
| Right | Move cursor right (with wrap around) |
| Fire and up | Cancel insert and quote mode (C128 only) |
| Fire and down | Clear from cursor to end of screen (C128 only) |
| Fire and left | Clear from cursor to start of line (C128 only) |
| Fire and right | Clear from cursor to end of line (C128 only) |

PROGRAM: STICKY CURSOR - C128

```
10 FOR P = 4864 TO 5122
20 : READ X$
30 : POKE P,DEC(X$)
40 NEXT P
1000 DATA 78,A9,0D,8D,14,03,A9,1
3,8D,15,03,58,60,CE,A1,13,F0,03,
4C,65,FA
1010 DATA A9,03,8D,A1,13,AD,00,D
C,8D,A2,13,29,10,D0,34,AD,A2,13,
29,01,D0
1020 DATA 06,A9,4F,38,4C,8C,13,A
D,A2,13,29,02,D0,06,A9,40,38,4C,
8C,13,AD
1030 DATA A2,13,29,04,D0,06,A9,5
0,38,4C,8C,13,AD,A2,13,29,08,D0,
C0,A9,51
1040 DATA 38,4C,8C,13,AD,A2,13,2
9,01,D0,06,A9,91,18,4C,8C,13,AD,
A2,13,29
1050 DATA 02,D0,06,A9,11,18,4C,8
C,13,AD,A2,13,29,04,D0,06,A9,9D,
18,4C,8C
1060 DATA 13,AD,A2,13,29,08,D0,8
C,A9,1D,18,4C,8C,13,90,04,A2,1B,
86,F0,8D
1070 DATA 4A,03,A5,D0,D0,02,E6,D
0,4C,65,FA,03,03,FF,01,7F,20,D2,
FF,4C,45
1080 DATA 14,20,4E,0A,A9,AA,A0,1
E,20,71,09,20,82,09,20,5D,0A,09,
80,48,AD
1090 DATA 14,20,F0,03,20,38,10,2
0,F6,09,68,4C,E7,0A,38,A5,39,ED,
08,20,85
1100 DATA 3B,A5,3A,ED,09,20,05,3
B,F0,04,A9,05,85,0C,20,4E,0A,A9,
00,A0,1F
1110 DATA 20,71,09,20,1C,13,A5,0
C,C9,05,F0,03,20,37,09,A9,00,A6,
39,A4,3A
1120 DATA 20,D5,FF,90,4F,44,30,0
D
```

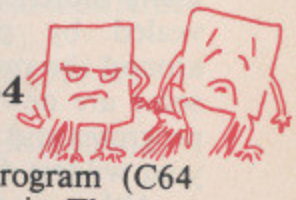
PROGRAM: STICKY CURSOR - C64

```
D9  10 FOR T = 40854 TO 40959
F2  20 READ A
68  30 POKE T,A
9E  40 NEXT T
71  50 SYS 40854
25  100 DATA 120,169,171,141,20,
    3,169,159
AE  110 DATA 141,21,3,88,169,149
    ,133,55
59  120 DATA 169,159,133,56,96,2
    06,254,159
D0  130 DATA 240,3,76,49,234,169
    ,3,141
55  140 DATA 254,159,173,0,220,1
    41,255,159
64  150 DATA 41,16,208,3,76,49,2
    34,173
21  160 DATA 255,159,41,1,208,5,
    169,145
82  170 DATA 76,242,159,173,255,
    159,41,2
75  180 DATA 208,5,169,17,76,242
    ,159,173
E3  190 DATA 255,159,41,4,208,5,
    169,157
D2  200 DATA 76,242,159,173,255,
    159,41,8
C7  210 DATA 208,192,169,29,141,
    119,2,165
C7  220 DATA 198,208,2,230,198,7
    6,49,234
34  230 DATA 1,127
```

## Installing the C128 interrupt

Type in the Basic program (C128 version), save and then run it. The program will then be installed. Save the machine-code from 4864 to 5122 as a binary file.

Type SYS 4864 to run the program and the joystick will emulate the cursor keys. In future when you need the utility it can be loaded directly into memory from the binary file and run by typing SYS 4864.



## Installing the C64 interrupt

Type in the Basic program (C64 version), save and run it. The program will then be installed. Type SYS 40854 to run the program and the joystick will emulate the cursor keys. The extra functions available on the C128 version are not catered for because they are not included in the operating system. The program resides from 40854 to 40959, so the top of Basic memory will be lowered when the program runs.

*by S Grewal*

# 64 Tips for the 64

by Eric Doyle

*A host of useful suggestions to improve your programming prowess*

The Commodore 64 harbours many secrets deep inside its memory banks. Here are 64 of the best but delve around and you may find many more. All the hints were revealed by poking and peeking around the memory. Some are old, some are new but all will improve programs and programming beyond your wildest dreams.

Although these tips are principally for the Commodore 64, the technical section will help you to convert many of them for the C128.

### 1 High and low bytes in decimal

The low byte is derived from the high byte calculation:

HIBYTE = INT(location/256)
LOBYTE = location – (HIBYTE * 256)

### 2 DEC to HEX conversion

```
10 INPUT"NUMBER IN DECIMAL";DE
C:IFDEC>65535THENPRINT"TOO BIG
":GOTO10
20 F=4096:FORA=1TO4
30 D%=DEC/F:DEC=DEC-D%*F
40 D%=D%+48:IFD%>57THEND%=D%+7
50 HX$=HX$+CHR$(D%):F=F/16
60 NEXT
70 PRINT"$"HX$
```

### 3 HEX to DEC conversion

```
10 INPUT"NUMBER IN HEX";HX$
20 L=LEN(HX$)
30 IF LEFT$(HX$,1)="$"THENHX$=
RIGHT$(HX$,L-1):L=L-1
40 IFL>4THEN 10
50 IFL<4THEN HX$="0"+HX$:L=L+1
:GOTO50
60 FORA=1TO4:D=ASC(MID$(HX$,A,
1))-48
70 IFD>9THEND=D-7:IF D>15THENP
RINT"NUMBER ERROR":GOTO10
80 DEC=DEC*16+D:NEXT
90 PRINTDEC
```

### 4 Using ROM routines

Locations 780 to 783 represent the A, X, Y and status registers respectively. By poking suitable values to the relevant location, the registers can be set before ROM routines are called.

### 5 Double byte HEX to DEC

```
10 INPUT"LOW BYTE LOCATION IN
DECIMAL";LOC
20 POKE781,PEEK(LOC):REM LO BY
TE INTO X REGISTER
30 POKE780,PEEK(LOC+1):REM HIG
H BYTE IN A REGISTER
40 SYS48589:REM PRINT A & X AS
 DECIMAL IN ASCII
```

### 6 Sub routine 48589

This is where the LIST routine goes to convert line numbers to decimal. It then prints the value to the screen. Use it for datamaker programs or for any routine where a decimal number has to be printed in ASCII characters. The A-register carries the high byte value and the X-register carries the low byte.

**7 Simulated PRINT@ command**

```
10 PRINT"[CLR]"
20 ROW=12:COLUMN=15
30 POKE 214,ROW
40 POKE 211,COLUMN
50 SYS58732
60 PRINT"HI THERE!"
```

**8 Alternative PRINT@ command**

```
10 PRINT"[CLR]"
20 ROW=12:COLUMN=15
30 POKE 781,ROW :REM X REG
40 POKE 782,COLUMN:REM Y REG
45 POKE 783,PEEK(783)AND 254
50 SYS58634
60 PRINT"HI THERE!"
```

**9 Finding the cursor**

A similar routine can also locate the cursor:

```
10 PRINT"[CLR,DOWN6,RIGHT6]";:
REM 6 ACROSS AND 6 DOWN
20 POKE 783,PEEK(783)OR1
30 SYS58634
40 ROW=PEEK(781)
50 COLUMN=PEEK(782)
60 PRINTROW,COLUMN
```

**10 Code space**

Machine code routines are faster if they access zero page locations but the Basic operating system leaves very few possibilities. Locations $02 and $FA to $FE (250–254) are normally unused but take care when using cartridges because they sometimes use these bytes.

Location $FF can also be used but it is best to use this for transient values.

Also in low memory, $02A7 to $02FF (679–767 can be used for small coded programs and disk users can access the cassette buffer at $0334 to $03FF (820–1023). Cassette users can only use $0334 to $033B (820–827) and $03FC to $03FF (1020–1023).

$C000 to $CFFF (49152–53247) is the coder's paradise, four kilobytes of protected memory which doesn't exist as far as the C64's operating system is concerned.

**11 Room at the top**

Extra protected space can be created by lowering the top of memory:

POKE 55,0:POKE 56,128:CLR

Location 56 holds the high byte and 55 takes the low byte value. In the example this gives decimal 32768 (hex $8000).

**12 Raising Basic**

In a similar way the bottom of Basic can be raised:

POKE 43,1:POKE 44,16:POKE 4096,0:CLR

The location is the new start of Basic+1 or 4097 ($1001). Location 4096 ($1000) must contain a zero otherwise a syntax error will occur when the RUN command is used.

**13 Memory SAVE**

Once a machine code program has been poked into memory this routine will save it:

```
10 REM FILENAME
15 FL=8:LS=0:HS=12*16:LE=6:HE=
12*16
20 POKE 183,FL
30 POKE 187,PEEK(43)+6:POKE188
,PEEK(44)
40 POKE193,LS:POKE194,HS
50 POKE174,LE:POKE175,HE+1
60 POKE186,8:REM 8 FOR DISK
70 POKE185,0:REM SECONDARY ADD
RESS
80 SYS62954
```

The actual filename for the save should be stored where 'filename' appears and FL is the number of characters in the name. LS, HS, LE and HE are the high and low bytes of the code block's start and end locations.

**14 Adding programs together**

Frequently used subroutines can be added to a program in memory. First of all, PEEK locations 43 and 44, noting down the values (usually one and eight). POKE 43, PEEK(45), and POKE 44, PEEK(46), to set the start of Basic to the end of the program in memory and then enter NEW. Next, load the subroutine as normal using ,D (rather than ,D,1), where D is the device number. Finally, the original values can be poked back into 43 and 44.

One word of warning, the line numbers of the appended subroutine must start at a higher value than the program in memory. Failing this, use a renumber routine on the program taking care with GOSUB, GOTO and ON commands.

**15 Directory tricks**

LOAD "$$",8 only loads the header and 'free blocks' lines.

LOAD "$A*",8 will only load programs starting with the letter before the asterisk.

LOAD "$*=S" will load only the SEQ files. Similarly with P, U, R substituted for S, PRG, USR or REL files can be selected.

**16 Scratching around**

To scratch all the files on a disk within a particular category (PRG, SEQ, REL, USR) use the form:

OPEN 1,8,15,"SO:*$=P": CLOSE1

**17 Which device?**

To automatically sense, from within a program, the device which is currently in use, DEV = PEEK(186) will store the number of the last device used as a variable for use in file operations.

**18 Closing files**

To make sure all files are closed use the CLALL call in the Kernal with SYS 65511. This closes all open files but be careful if CMD has been used. Sometimes a mere CLOSE command leaves the screen editor in a confused state. To exit safely use the format:

PRINT#4:CLOSE4

**19 Selecting a bank**

To point the VIC chip at a different block of memory:

POKE 56578, PEEK(56578) OR 3

POKE 56576, (PEEK(56576) AND 252) OR a

Where 'a' is 3 for the normal (default) bank from the start of memory to 16383, 2 for 16384 to 32767, 1 for 32768 to 49151 or 0 for 49152 upwards.

**20 Moving the screen**

Relocate the screen position within a bank with:

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

Where 'b' takes a value from Table 1 and the actual location is the bank location plus the 'b' value.

| TABLE 1 – Bank location values | | | |
|---|---|---|---|
| *b* | *Start Location* | *b* | *Start Location* |
| *0* | *0* | *128* | *8192* |
| *16* | *1024* | *144* | *9216* |
| *32* | *2048* | *160* | *10240* |
| *48* | *3072* | *176* | *11264* |
| *64* | *4096* | *192* | *12288* |
| *80* | *5120* | *208* | *13312* |
| *96* | *6144* | *224* | *14336* |
| *112* | *7168* | *240* | *15360* |

**21 Moving the characters**
To relocate the 2K area from which RAM character information is taken:
POKE 53272, (PEEK(53272) AND 240) OR c

Where 'c' has a value from Table 2.

| TABLE 2 – Character relocation table | |
|---|---|
| *c* | *Start Location* |
| *0* | *0* |
| *2* | *2048* |
| *4* | *4096* |
| *6* | *6144* |
| *8* | *8192* |
| *10* | *10240* |
| *12* | *12288* |
| *14* | *14336* |

**22 Relocating character data**

```
10 POKE 56334,PEEK(56334)AND 2
54
20 POKE1,PEEK(1)AND251
30 FORA=0TO2047
40 POKE NEWLOC+A,PEEK(53248)+A

50 POKE1,PEEK(1)OR4
60 POKE56334,PEEK(56334)OR1
```

This routine cannot be stopped because the keyboard is disabled in line 10. Substitute a value derived from Table 2 for 'newloc'.

**23 GET with cursor**

```
10 GET A$:POKE 204,0:IF
   A$=""THEN 10
```

**24 Display controls in PRINT**

```
POKE 212,1:PRINT"[CLR]
HELLO"
```

**25 Best hi-res location**
Set the VIC chip to Bank 4 and the screen to location 57344 ($E000). The ROM doesn't have to be switched out because the screen 'looks' through it. A routine is needed to switch out the ROM if the screen needs to be PEEKed but POKEs can be performed with the ROM in place.

**26 Easy INPUT**

```
10 INPUT"DO YOU WANT
   TO SAY YES[RIGHT2]Y
   [LEFT3]";A$
```

**27 Position in colour RAM**

```
PRINT PEEK(243) + 256 *
PEEK(224)
```

**28 Position in screen RAM**

```
PRINT PEEK(209) + 256 *
PEEK(210) + PEEK(211)
```

**29 Switch screen off/on**

```
Off: POKE 53265,
     PEEK(53265) AND 239
On:  POKE 53265,
     PEEK(53265) OR 16
```

**30 Supervision**
When using screen routines use base addresses for the screen and colour RAM, CO=55296:SC=1024. This means that a POKE to the video can be colour co-ordinated:

POKE SC+50,character:POKE CO+50,colour

**31 Key detection**
PEEK location 653 to see if the SHIFT (value 1), CBM (2) or CTRL (4) keys have been pressed. If two keys are held down at the same time the value found in 653 is the sum of the key values. For example, a value of five means that the CBM and CTRL keys are down.

**32 Clearing the key buffer**
Before detecting a keypress, make sure that the key buffer is empty by POKE 198,0.

**33 Filling the key buffer**
Loading from inside a program has its drawbacks. A better way is to place the loading sequence in the key buffer so that the next program loads as though the commands have been typed in.

The buffer queue is located at 631 and continues through the following nine bytes. This means that only ten characters can be stored here as ASCII codes. A routine such as this would suffice:

```
10 PRINT"[CLR][SO]"+CHR$(34)+
"ZULU"+CHR$(34)+",8,1[DOWN4]":
PRINT"RUN"
20 POKE631,19:POKE632,13:POKE6
33,13
30 POKE198,3
```

This would appear at the end of the first program and would load a program called ZULU. To save on buffer space, the commands are written to a cleared screen. They have to be correctly spaced to allow for the on-screen messages that the LOAD routine displays.

The buffer is poked with a HOME command to place the cursor on the top screen line and this is followed by two returns. Location 198 has to be told that these three characters are waiting in the buffer.

When the program ends the first return automatically executes the LOAD command and then the loaded program automatically runs.

A modified version of this could load a series of machine code routines and execute a SYS command at the end. With a ten character buffer there could be 9 returns stored; that means nine commands executed after the first program ends.

**34 List protect 1**
To protect a program from prying eyes use POKE 775, 200. This prevents listing and can be restored by poking the original value of 167 back again.

**35 List protect 2**
Another protection method is to use a REM statement which contains a shifted L. When the program is listed it produces a syntax error after the statement. This is easily overridden by listing the program and pressing the return key on the listed line. Using this method in conjunction with the next method can improve this command.

**36 List protect 3**
A REM statement with a few delete symbols (reversed T) can create havoc with the LIST command.

To insert a delete symbol, type REM and then two pairs of inverted commas, delete the second pair of quotes. After the remaining set of quotes, press the insert iey (shifted INST/DEL key) once for each character of the current line which you want to hide. Next press the delete key the same number of times. When the program is LISTed and deletes will each erase a character which lies to their left. For example:

```
10 SYS49152:REM"[DEL13]"
   REM[SL]
```

would pull the second REM over the SYS command, making the line look and behave like a REM with a shifted L.

**37 Key repeat**

The only keys which will repeat when held down are the cursor keys and spacebar. Poking a value of 128 to location 650 makes all keys repeat but poking 64 instead will prevent any of the keys from repeating.

**38 Repeat delay**

If all keys are set to repeat, the gap between the first character appearing and the row of repeats can be altered by poking different values to location 651. The maximum delay is given when the poked value is 255, causing a delay of around four seconds.

**39 UnNEW**

Accidentally NEWing a program before it is saved does not mean that the program is lost forever. The easiest way to recover is to reset the line link vectors, rechain the program and restore the end of program pointer.

One way to do this is to store a program at the top of memory. Enter the following line:

```
POKE 43,1:POKE
44,159:POKE 46,159:POKE
40704,0:NEW
```

This sets the start of Basic at 40704 ($9F00). The following UNNEW program can not be typed in and saved with ",dev,1" where dev is 1 or 8 for tape or disk.

```
10 FORA=2050TO40703
20 GOSUB60
30 NEXT:IFA=40704THENEND
35 A=INT(B/256):B=B-A*256+3
40 POKE251,B:POKE252,A
50 POKE44,8:POKE2050,255:SYS42
291:POKE45,PEEK(251):POKE46,PE
EK(252):CLR:END
60 IFPEEK(A)=0ANDPEEK(A+1)=0AN
DPEEK(A+2)=0THENB=A:A=49999
70 RETURN
```

The program will always load at 40704 as long as the ',1' is added.

When a program is to be restored after NEW, enter the line of pokes mentioned at the beginning of this section and load the UNNEW program.

The program can now be RUN but it may take a while before the READY prompt appears again. When it does, the program will have been restored.

**40 Extending Basic**

When a Basic program is decoded by the operating system, it calls in the Basic lines for analysis in the input buffer starting at $0200 (512). Part of the decoding routine lies in RAM and this is where an extra routine can be wedged in.

This routine is stored at $0079 and looks like this:

```
$0073 INC $7A
$0075 BNE $0079
$0077 INC $7B
$0079 LDA $0201
$007C CMP #$3A
$007E BCS $008A
$0080 CMP #$20
$0082 BEQ $0073
$0084 SEC
$0085 SBC #$30
$0087 SEC
$0088 SBC #$D0
$008A RTS
```

A suitable break point can be created by moving CMP #$20 to $007C and BEQ $0073 to $007E. Location $0080 can now point to the new decoding routine (eg. JMP $C000).

The routine grabs a byte from the buffer and attempts to decode it. By using a prefix on all of the new commands, decoding becomes easier. The 'at' symbol located next to the asterisk on the keyboard is a good prefix so the new routine could look like this:

```
$C000 CMP #$40
$C002 BEQ $C012
$C004 CMP #$38
$C006 BCC $C00B
$C008 JMP $C011
$C00B SEC
$C00C SBC #$30
$C00E SEC
$C00F SBC #$D0
$C011 RTS
$C102 DECODE ROUTINE
```

The decode routine can call extra bytes from the buffer with JSR $0073 until a colon is detected. After execution of the command the new routine should hand back control to the normal operating system with JMP $0073.

**41 Setting up interrupts**

Once written, an interrupt routine can be called by:

```
SEI
LDA #$1F
STA $DC0D
STA $DD0D
LDA $DC0D
LDA $DD0D
LDA #$low byte
STA $0314
LDA #$high byte
STA $0315
LDA #$01
STA $D01A
CLI
RTS
```

The high and low bytes refer to the location of the new interrupt routine. Somewhere in the new routine three extra lines should be included. The first two are:

```
LDA #$01
STA $D019
```

This can be located anywhere in the new code but the third extra line should be used instead of RTS:

```
JMP $EA31
```

This checks the keyboard to see if an input has occurred. If the keyboard check isn't necessary the routine should end with:

```
PLA
TAY
PLA
TAX
PLA
RTI
```

**42 WAIT problems**

The WAIT command is probably the least used and most misunderstood Basic term. Its use is princi-

pally for I/O actions because it has to use a memory location that can be changed externally.

The most common applications are to test for a datasette or keyboard keypress:

WAIT 1,32,32
WAIT 197,8

The first key is the datasette sensor which waits until bit 5 of location 1 is set by pressing any of the motor keys.

The second command peeks into the keypress register. Any key decode value which sets bit 3 will terminate the pause, so 'A' (value 10) will be accepted but 'R' (17) would have no effect.

Another use is to detect the pressing of the CTRL, CBM or SHIFT keys through location 653. WAIT 653,1 will detect the SHIFT key, WAIT 653,2 finds the CBM key and WAIT 653,4 is the CTRL command. Alternatively, WAIT 653,6 will detect either the CTRL or CBM keys.

**43 Wait a jiffy**
One internal use of WAIT is to create a time delay when used with the TI$ register.

TI$="000000":WAIT 161,1 will give a 4 second pause.

**44 Unofficial pause**
The pause routine called by the tape loading routines can be used as an alternative to the usual 'press any key' pause. SYS 58592 followed by POKE198,0 will give an infinite pause and forget which key was pressed. The key must be one of those to the left of the keyboard.

**45 Colour change**
Character colours can be changed through control codes in PRINT statements but a better way for certain functions is to poke the colour value direct to location 646.

**46 Flash data**
It can be very boring waiting for a program to read in data. Sometimes it can take so long that you start to wonder if the computer has hung up. One way to reassure the user is to add an extra POKE command to flash the border. The nice thing about this location is that it will accept any value up to 255 without a murmur even though there are only sixteen colours to choose from.

For numerical data POKE 53280,A will create a border flash if A is the data value which has just been read in. If the data is greater than 255, a small division routine to keep the numbers down to acceptable values can be included.

With string data a modified version can be used taking ASC(A$) as the value.

**47 Screen shake**
One less serious function of location 53270 can be shown with this line:

```
FOR A = 0 TO 255:POKE
53270,A:NEXT
```

The screen shakes as though a violent earthquake was stirring inside the 64. For added effect, poking A to 53281 will really blow your mind.

**48 Reset prevention**
The RUN/STOP with RESTORE reset can be prevented by POKE 808,225. The function can be re-enabled by POKE 808,237.

**49 Cold restart**
When a program has been altering the start of Basic and various other parameters, the easiest way to get the C64 back to normal is by using SYS 64738 instead of the END command.

**50 Scrolling screens**
An upwards screen scroll can be forced at any time by SYS 59626.

**51 Making spaces**
SYS 59749 will open up a screen line beneath the current cursor position, effectively scrolling the screen downwards.

Combining this routine with the normal scroll, a routine can be devised to give quite stunning effects.

```
10 PRINT"[HOME,YELLOW,SF40,C7]
":POKE214,255
20 FORA=0TO23:SYS59749:NEXT
30 FORA=0TO1200:NEXT
40 FORA=0TO25:SYS59626:NEXT
50 FORA=1024 TO2023:POKEA,A-IN
T(A/256)*256:NEXT
60 GOTO10
```

**52 Screen flipping**
Location 648 directs the operating system to the current screen area.

```
10 PRINT"[CLR]THIS IS SCREEN 1
"
20 PRINT"PRESS ANY KEY"
30 WAIT198,1
40 POKE56578,PEEK(56578)OR3
50 GOSUB160
60 PRINT"[CLR]THIS IS SCREEN 2
[DOWN3]"
70 INPUT"TYPE ANYTHING IN";A$
80 GOSUB200
90 PRINT"[HOME,DOWN3]BACK AGAI
N"
100 PRINTA$
110 FORB=1TO10
120 FORA=0TO1000:NEXT:GOSUB160

130 FORA=1TO1000:NEXT
140 GOSUB200
150 NEXT:END
160 POKE648,132
170 POKE56576,(PEEK(56576)AND2
52)OR1
180 POKE53272,(PEEK(53272)AND1
5)OR16
190 RETURN
200 POKE648,4
210 POKE56576,(PEEK(56576)AND2
52)OR3
220 RETURN
```

**53 Open files**
With complex programs, it's best to keep track of the number of open files. PEEK(152) will reveal the current number of active files.

**54 Motor kill**
To stop the datasette motor at any time, SYS $FCCA.

**55 String grabbing**
Strings can be pulled out of memory by using the ROM routine at 43806. The string must end with a quotation mark or a zero and the start address is stored in the accumulator (780) and Y register (782) in low byte/high byte format:

```
10 REM "WHO STOLE THE BYTES? ^
^^"
20 POKE780,115:POKE782,228:SYS
43806
30 POKE780,96:POKE782,228:SYS4
3806
40 PRINT:PRINT
50 POKE780,8:POKE782,8:SYS4380
6
```

**56 Last filename**
The name of the last file to be loaded into the computer can be found from a ROM routine which the

computer uses to print the 'searching for' message: SYS 62913.

**57 Faster Basic**
Maintaining the screen is a time consuming job for the 64. Complex calculations can be speeded up by five percent if the screen is blanked out first:

POKE 56325, PEEK(56325) AND 239

To get the screen back again the command becomes:

POKE 56325, PEEK(56325) OR 16

**58 Memory configuration**
Switching ROMs in and out of the 64 memory is controlled by the first three bits of location 1.

Bit 0 controls the Basic ROM at 40960 ($A000)

Bit 1 switches the Kernal ROM at 57344 ($E000)

Bit 2 is responsible for the character ROM at 53248 ($D000)

**59 One-line clear**
A partial screen clear can be achieved by using part of the in-built screen clear command. The row to be cleared is poked into the X register (780). Remember that the numbering starts with line 0.

```
10 FOR A=1024 TO 2023:
   POKE A, 1: NEXT
20 FOR A=5 TO 20: POKE
   781, A
30 SYS 59903
40 NEXT
```

**60 Line snatching**
The screen routines can also be used for line grabbing.

Location 59855 is the start of a routine which will copy a line from anywhere in memory and place it on the screen. The only thing to remember is that the data must be written in screen poke values and be 40 characters long.

The low/high byte information for the start of the line is poked into locations 172/173.

POKE 172, 0:POKE 173, 8:SYS 59855

The line is printed at the current cursor position which should be set before the routine is called (see tip 7).

Used with imagination, this routine can be used to create special effects such as screen rotation or a one line scroll.

**61 Wot, no query**
Sometimes a question mark seems odd when an input is required. Not all inputs are questions so why should there be a question mark?

POKE 19,1 before using INPUT will cure the problem.

**62 Tape headers**
The cassette buffer is a misnomer, it is actually the tape header buffer. When the header is FOUND switch off the datasette and press RUN/STOP. Run the following program to find out what's in the buffer.

```
10 CB=828
20 S=PEEK(CB)
30 LA=PEEK(CB+1)+PEEK(CB+2)*25
6
40 EA=PEEK(CB+3)+PEEK(CB+4)*25
6
50 FORA=CB+5TOCB+20
60 F$=F$+CHR$(PEEK(A)):NEXT
70 FORA=CB+21TO1019:IFPEEK(A)=
20THEN90
80 MC=1
90 NEXT
100 PRINT"[CLR,DOWN,SPC15]TAPE
 INFO"
110 PRINT"[SPC15,SE9]"
120 PRINT"[DOWN3]SECONDARY ADD
RESS =";S
130 PRINT"[DOWN]START ADDRESS[
SPC5]=";LA
140 PRINT"[DOWN]END ADDRESS[SP
C7]=";EA
150 PRINT"[DOWN]FILENAME[SPC10
]=";:POKE212,1:PRINTF$
160 IFMCTHENPRINT"[DOWN]MACHIN
E CODE IN RESIDENCE"
```

**63 Controlled Print**
It's no secret that control codes can be used for changing the print colour but have you tried non-colour control characters. After typing PRINT, open quotes, hold down CTRL and a letter and see what happens when the program runs.

CTRL M can't be used because it does an automatic shifted return and erases the rest of the line that you're trying to enter. CTRL N automatically switches the computer into lower case.

**64 FAC facts**
Floating point is difficult to understand but can easily be used by machine code routines.

The conversion of the numbers is performed by the routine at $B391. It takes a number stored in the Y/A registers in low/high byte format and stores it in a special register known as FAC#1 (Floating point ACcumulator). If two numbers are to be operated on they must be transferred in the correct order to FAC#1 and FAC#2.

To transfer a number from FAC#1 to FAC#2 the routine at $BC0F is used. This is shown in the example routine which divides 1000 by 4 in hex.

```
  LDA #$03
  LDY #$E8
  JSR $B391
  JSR $BC0F
  LDA #$00
  LDY #$04
  JSR $B391
  JSR $BB12
  JSR $BDDD
  LDY #$05
*LDA $0100,Y
  STA $00F9,Y
  DEY
  BNE *
  RTS
```

To convert the result in FAC#1 into a usable form, $BDDD returns the result in ASCII to $0100 from which it can be stored and used, in this case from location $FA.

Other useful routines are available:

| | |
|---|---|
| $BBFC | transfer FAC#2 to FAC#1 |
| $B86F | add FAC#1 to FAC#2 |
| $B853 | subtract FAC#1 from FAC#2 the result is stored in FAC#1 |
| $BA30 | multiply FAC#1 by FAC#2 and store the result in FAC#1 |
| $BB12 | divide FAC#2 by FAC#1 and store the result in FAC#1 |
| $BF7B | raise FAC#1 to the power FAC#2 and store the result in FAC#1 |
| $BF71 | calculate the square root of FAC#1 and store the result in FAC#1 |

If you discover an interesting routine for the Commodore 64, why not let everyone in on the secret and send it to: Tips for the Serious User, Your Commodore, ASP Ltd, 1 Golden Square, London W1R 3AB.

| LIFESAVERS 3 | C128 | TEXT SCREEN DUMP | 1/1 |
|---|---|---|---|

This utility will dump the current text screen from inside a Basic program. It reads the location where the screen is stored and translates them into ASCII code values ready for printing.

Place the program inside a listing and the command GOTO 10 will set the process in motion.

```
10 REM TEXT SCREEN DUMP
20 REM BY ANDREW GORRIE
30 FAST : P$=CHR$(16) : A$="" :
OPEN4,4,7
40 AA=1024:FOR P=0 TO 24
50 FOR N=AA TO AA+39 : GOSUB 60 :
NEXT : AA=AA+40 : PRINT#4,P$ "15"
A$: NEXT P :SLOW : END
60 IF PEEK(N)>63 AND PEEK(N)<91
THEN Q=32
70 IF PEEK(N)>128 THEN Q=-128
80 IF PEEK(N)<26 THEN Q=64
90 A$=A$+CHR$(PEEK(N)+Q) : Q=0 :
RETURN
```

| LIFESAVERS 4 | C64 | RESTORE BORDER CHANGE | 1/1 |
|---|---|---|---|

Running this program means that the border can be changed by pressing the RESTORE key. The colour cycles through to a new value for each press of the key.

The program works by resetting the STOP routine vector at 808-9.By altering these you can point the restore key to any routine you like. The border change routine has been chosen to highlight a side effect of this method. When a program is loaded or saved,the border changes as the ROM routines call on the STOP routine while performing their tasks.

Pressing RUN/STOP with restore will deactivate the border routine and POKE 808,0:POKE 809,192 will activate it.

```
10 REM RESTORE KEY BORDER CHANGE
20 C=0:FOR I=0 TO 5: READ A:POKE
49152+I,A:C=C+A:NEXT
30 IF C<>1037 THEN PRINT"ERROR IN
LINE 40":END
40 DATA 238,32,208,76,237,246
50 POKE 808,0:POKE 809,192
60 PRINT"PRESS RESTORE"
```

# A Bit More

*Turn your C64 into a hi-res machine with a bitmap, 16 sprites, a redefinable character set, 8K of storage plus 21 commands – but retaining 36K for Basic!*

This utility combines two ideas in the one program. Firstly, the memory map is reconfigured so that the screen, character set, and bitmap are in Bank 3 (from 49152 to 65535) and, secondly, an arsenal of 21 machine-code commands is loaded into the area of memory once used by the screen (from 1020 to 3794), the start of Basic being moved up to 3795 to protect it. If you have a look at the memory map provided you will see that almost the whole of the memory is now usable. The exceptions being the I/O area from 53248 to 56344 and the area from 0 to 3494, which still leaves some 57945 bytes free.

The program is in two parts. The first, called MCLOAD, is stored from 680 to 762 and forms a machine-code loader which sets the start of Basic memory to 3795 and then loads in the main program, called MCFILE, and calls the reconfigure command (SYS1020), finally ending with NEW to set the Basic pointers correctly.

MCFILE is the program containing the machine-code commands and consists of a jump table (from 1020 to 1083) for the command routines, followed by the routines themselves between 1086 and 3794.

The following is a detailed list of all the new commands available.

**RECONFIGURE – SYS1020**

This is the command that sets up the new memory configuration and can be used at any time. The screen, sprite pointers, sprites and bitmap maintain their relative positions but move up into Bank 3.

The screen is at 1024+49152=50176, the first sprite pointer is at 2040+49152=51192, sprites 0–15 are at 0*64+49152= 49152 to 15*64+49152=50112 and the bitmap is at 8192+49152= 57344.

The character set is now in RAM at 51200 to 53248 and consists of the first 256 characters of the normal set but it can be redefined at any time.

**RASTER ON – SYS1023**

The raster interrupt routine is turned on with this command and can be used for both split-screen graphics and sound as detailed later. It is not recommended that you leave the raster interrupt routine running when using the LOAD and SAVE or the MEMORY MOVE and MEMORY FILL commands.

**RASTER OFF – SYS1026**

This command is obviously to turn off the interrupt routine.

**SPLIT-SCREEN – SYS1029, <0 or 1–200>**

This command is used in conjunction with the RASTER ON command to set up a split-screen where the top part is bitmapped while the rest is the normal screen display. It requires one parameter which sets the split to a screen line between 1 and 200. A value of zero turns the split-screen off.

**BITMAP ON – SYS1032**

This turns on the full screen bitmap. If a full bitmap and an interrupt are required at the same time for running the sound routine, then the SPLIT-SCREEN command SYS1029,200 should be used rather than this one. If at any time this command does not appear to be working, check that the raster routine has been switched off first because they won't work together.

**BITMAP OFF – SYS1035**

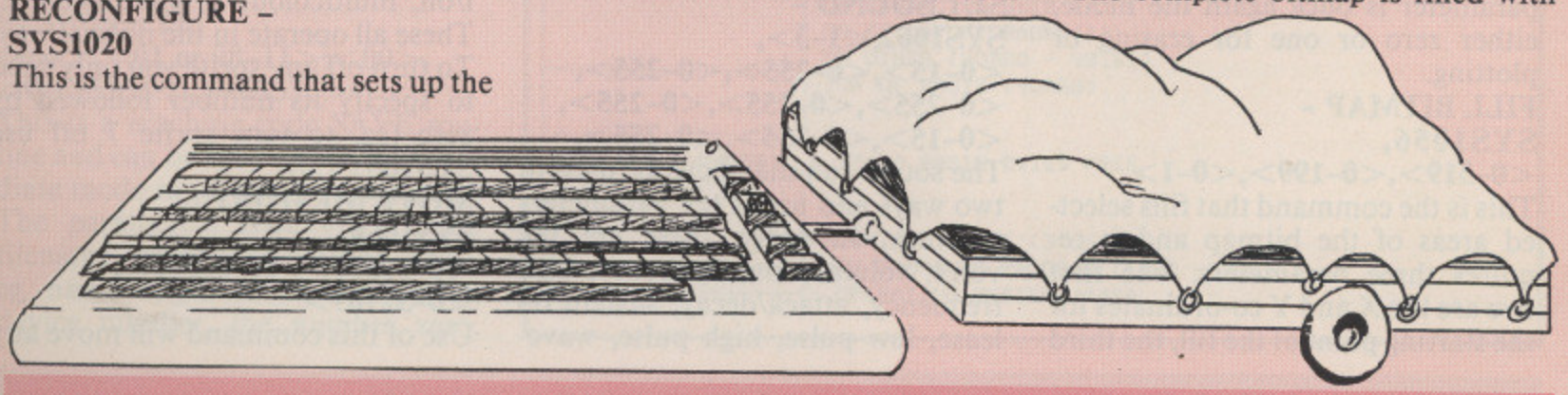
Turns off the bitmap. The same rules apply to these interrupts as those detailed in the BITMAP ON command.

**CLEAR BITMAP – SYS1038**

This command clears the complete bitmap. If you want to clear only parts of a bitmap see the MEMORY FILL command and Example 3 in the DEMO program.

**COLOUR BITMAP – SYS1041**

The complete bitmap is filled with



*by Malcolm W Gallon*

the current colour stored in 254, this is set with the SET CURRENT COLOUR command.

**SET CURRENT COLOUR – SYS1044,<0–15>,<0–15>**

Use this command to set the display colour. It requires two parameters, the first is the plot or drawing colour using the standard colour codes and the second is the background colour.

**PLOT POINT ON BITMAP – SYS1047,<0–319>,<0–199>, <0–1–2>**

Points on the bitmap screen will be plotted in the current colour. Three parameters are required – the first is the X co-ordinate between 0 and 319, the second is the Y co-ordinate between 0 and 199, the third is the mode which can be zero to unplot or erase a point, one to plot a point or two to test if a pixel is on or off. If the pixel is off then location 2 will contain a zero, if the pixel is on then it will contain the number 100.

**DRAW LINE – SYS1050, <0–319>,<0–199>,<0–319>, <0–199>,<0–1>**

To draw lines on a bitmap, this command can be used with its five parameters. The first two are the starting X and Y co-ordinates of the line, the next two are the end of line X and Y co-ordinates, the last is the mode, which can be either zero to erase or one to plot.

**DRAW CIRCLE – SYS1053, <0–319>,<0–199>,<0–129>, <0–1>**

Here is the command which draws circles from four parameters. The first two are the X and Y co-ordinates of the centre of the circle, the next is the radius of the circle. The smallest circle that can be drawn has a radius of one and the largest has radius 129. The last parameter is once again the mode, either zero or one for erasing or plotting.

**FILL BITMAP – SYS1056, <0–319>,<0–199>,<0–1>**

This is the command that fills selected areas of the bitmap and it requires three parameters. The first two are the X and Y co-ordinates for the starting point of the fill, the third is again the mode of either zero or one. The starting point of the fill is important as the routine is not the smartest around. Try the following example and you will see what I mean.

Set up a bitmap using the following commands in either program or immediate mode.

| | |
|---|---|
| SYS1044,2,10 | – set colours to red on light red |
| SYS1038:SYS1041 | – clear and colour the bitmap |
| SYS1032 | – turn on the bitmap |
| SYS1053,160,100,50,1 | – draw circle |

Next try SYS1056,140,100,1 and you will see that the circle is not properly filled. Erase the above with SYS1056,140,100,0 and draw the circle again (same as before) and then use SYS1056,160,100,1. The circle will now be filled correctly. Have a look at Example 6 in the demo program and you will see some of the limitations of the fill commands. The two fill routines are reasonably fast and compact in code so you can't expect miracles.

**FILL WITH CHARACTER – SYS1059,<0–319>,<0–199>, <0–1>,<0–255>**

This routine works in the same way as the ordinary fill routine but is has an extra parameter. This is a character from the character set, numbered 0 to 255, and is used to specify the fill pattern. You can fill or erase any area of the screen or even the whole screen with this character and, since the character set is redefinable, there are hundreds of possible fill patterns available. The limitations of the fill routine also apply to this command.

**SET SOUND – SYS1062,<1–3>, <0–15>,<0–255>,<0–255>, <0–255>,<0–255>,<0–255>, <0–15>,<0–255>,<0–255>,**

The sound command can be used in two ways and needs ten parameters to work. In order, these are the voice, volume, low frequency, high frequency, attack/decay, sustain/release, low pulse, high pulse, waveform and the duration of a note. All but one of these operate in the normal way, the exception being the duration.

If the interrupt is not turned on, then the duration is simply a set-up command and the particular voice used will have to be gated off in the normal way. If, however, the interrupt is on, then the duration parameter comes into effect. The length of the note is calculated by:

Duration = time in seconds × 50

eg. a five second note gives a parameter of 5 × 50 = 250

All three voices can be used at once under interrupt control and you can have split-screen graphics and interrupt controlled sound at the same time.

To check if a voice has finished processing, or requires more data, a PEEK to the voice control register is needed. For the three voices, the registers are located at 1228, 1129 and 1130 respectively. If the Voice 1 register at 1128 contains 100 then the voice is still on. If it contains zero then the voice has been gated off. The other two voice registers operate in the same way.

**SET SPRITE – SYS1065,<0–7>, <0–1>,<0–255>,<0–15>, <0–320>,<0–255>,<0–1>, <0–1>,<0–1>,<0–15>, <0–15>**

This command allows sprites to be set up with only one instruction but it has eleven parameters which need to be specified. These are: sprite number, sprite off/on, sprite pointer (remember data should be somewhere in bank 3 ie. for a pointer set to 13 the data should be at 13*64+49152=49984 onwards), sprite colour, X position, Y position, X expand (0=ordinary, 1=expanded), Y expand, multicolour off/on, multicolour 1, multicolour 2. These all operate in the normal way. To turn off any sprite you only need to specify its number followed by zero (eg. to turn sprite 7 off use SYS1065,7,0).

**MOVE MEMORY – SYS1068, <0–65535>,<0–65535>, <0–32767>**

Use of this command will move any

block of memory to any addresss as specified by the three parameters. The first is the block's destination address, the second is the starting address of the block to be moved, and the third is its length which is limited to a maximum of 32K.

MOVE MEMORY has access to all of the RAM, including that under the Basic and Kernal ROMS, except for the RAM under the VIC, SID, and the I/O ROMs.

As was mentioned earlier, it is not advisable to use this command with the interrupt running. If you need to have a split-screen and to move or fill large areas of memory as well, then simply switch off the interrupt before calling this routine and switch it back on immediately afterwards. This will cause the screen to flicker or upset sound timings but it is simply a consequence of not being able to have interrupts running while maintaining access to the RAM under ROMs at the same time.

**FILL MEMORY –**
**SYS1071,**
**<0–65535>,<0–32767>,**
**<0–255>**

You can fill any area of memory up to a block size of 32K with a number between 0 and 255 using this command. The parameters required form the starting address of the memory block to be filled, the length of the block (up to 32K), and the number with which the block is filled.

This command uses part of the MOVE MEMORY routine and the interrupt restrictions also apply. There are no restrictions concerning which part of memory you are filling so be careful that you don't overwrite something important, such as the operating system and program areas.

**SAVE MEMORY –**
**SYS1074,<file**
**name>,<01 or 08–11>,<00>,**
**<0–65535>,<0–65535>**

This is a machine code SAVE routine and can be used either in immediate mode or as part of a program. The parameters required are the filename (usual restrictions apply eg. name length of 16 characters), device number, the number zero, the starting address of the block to be saved, the end address+1 of the block to be saved.

**LOAD MEMORY –**
**SYS1077,**
**<file name>,<01 or 08–11>,**
**<00>,<0–65535>**

The LOAD command can be used in either immediate mode or within a program and requires the following parameters. Firstly the filename, then the device, the number zero (this is essential), and finally the load address.

**SET CURSOR POSITION –**
**SYS1080,<0–24>,<0–39>**

This is the last command, and it is used to print text to a specific row and column on the screen, eg. SYS1080,10,14 followed by PRINT<text message> will print the message on row 10, column 14.

These are all the new commands that are available and if you have a look at the DEMO program provided it should give you some idea of how to make use of them. The best way to find out what can and cannot be achieved is to experiment as much as possible and see what happens.

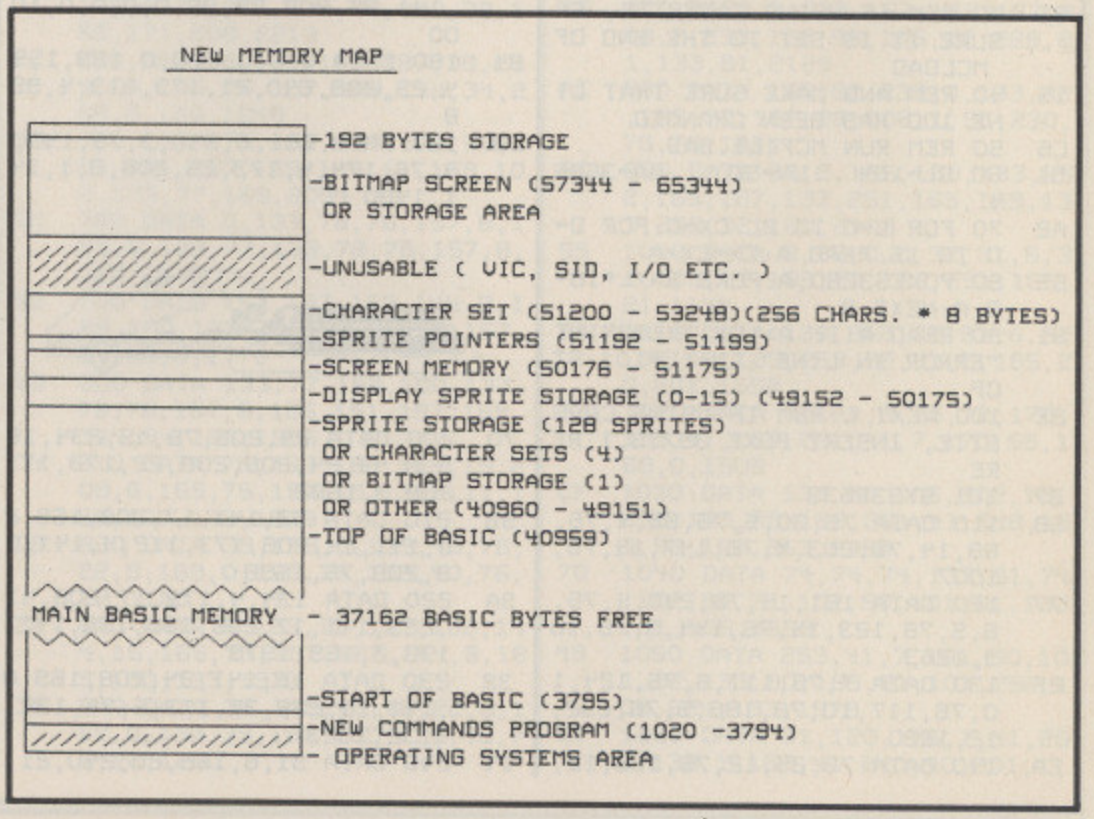
If for any reason you press the RUN/STOP and RESTORE keys, you should put the disk that contains the MCFILE program into the drive, and type SYS686. This will get everything back to normal without losing any Basic program that may be in memory. Do not use SYS680 as this contains the NEW command and any Basic program will be lost.

## Split loyalties

The problems that prevent using the memory move or fill commands with a raster interrupt running relate to the operating system ROMs.

To set the scene, the raster interrupt comes into operation at the specified screen line, either turning on or off the bitmap as necessary. The routine then exits via the normal interrupt routine handler at $EA31 to ensure correct Basic operation with keyscan. So far so good.

The MEMORY MOVE command requires access to the RAM under the Basic and Kernal ROMs, so these are both switched out. Interrupts cannot now be allowed to occur since the Basic interpreter is no longer in memory. If the MEMORY MOVE command is running, the interrupt flag is set so, if the routine runs for longer than 1/50th of a second (the time it takes



for the raster to scan a complete screen), the raster interrupt will be held up. When the MEMORY MOVE command is finished the interrupt flag is cleared and the raster interrupt can now occur. Unfortunately, the raster display will be frozen during the MEMORY MOVE interrupt giving a full high resolution screen or a full normal screen. Either way the result is a mess!

The only way round this problem is to avoid using the two commands together or simply switch the raster off before using the MEMORY MOVE command and switch it back on again afterwards using the routines provided.

All the above also applies to the MEMORY FILL command because memory has been saved by using some of the subroutines of the MEMORY MOVE command.

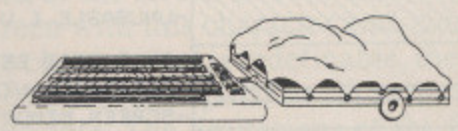
Finally, if you want to load the program without the menu system, type LOAD"MCLOAD",8,1 and then SYS680. To get the demo program type LOAD"DEMO",8 then RUN.

PROGRAM: MCLOAD

```
3B   10 REM MCLOAD
0A   20 REM THIS PROGRAM LOADS AN
     D SAVES THE MCLOAD FILE
26   30 REM TYPE THIS IN AND SAVE
      IT TO TAPE OR DISK AS "MCLO
     AD.BAS"
8B   40 REM INSERT YOUR MASTER TA
     PE OR DISK AND RUN MCLOAD.BA
     S
67   50 REM THE SCREEN WILL FLASH
      AND THEN THE SAVE WILL OCCU
     R AUTOMATICALLY
06   60 :
7C   70 :
25   80 BL=4:LN=50:SA=680
CF   90 FOR L=0 TO BL:CX=0:FOR D=
     0 TO 15
50   100 READ A:CX=CX+A:POKE SA+L
     *16+D,A:POKE53280,A:NEXT D
55   110 READ A:IF A><CX THENPRIN
     T"ERROR IN LINE";LN+(L*10):S
     TOP
FB   120 NEXT L:READA:POKE760,A:R
     EADA:POKE761,A
CE   130 DATA 32,174,2,76,219,2,1
     69,2,162,8,160,1,32,186,255,
     169,1649
E5   140 DATA 6,162,244,160,2,32,
     189,255,169,0,32,213,255,169
     ,211,133,2232
BB   150 DATA 43,169,14,133,44,16
     9,0,141,210,14,32,30,5,169,1
     47,32,1352
26   160 DATA 210,255,96,169,4,13
     3,198,169,78,141,119,2,169,6
     9,141,120,2073
4D   170 DATA 2,169,87,141,121,2,
     169,13,141,122,2,96,77,67,70
     ,73,1352
F5   180 DATA 76,69,0,0,0,0,0,0,0
     ,0,0,0,0,0,0,0,145
E9   190 POKE193,168:POKE194,2:PO
     KE174,251:POKE175,2
74   200 POKE187,7:POKE188,8:POKE
     183,6:POKE186,8:REM POKE 186
     ,1 FOR CASSETTE
27   210 POKE185,0:SYS62954
```

PROGRAM: MCFILE

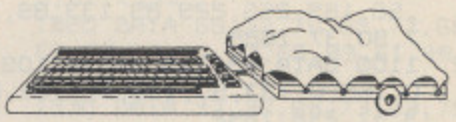
```
35   10 REM TYPE IN THIS PROGRAM
     AND SAVE ON A SPARE TAPE OR
     DISK AS "MCFILE.BAS"
65   20 REM BEFORE RUNNING PLACE
     THE DISK OR CASSETTE WITH MC
     LOAD INTO THE SAVE DEVICE
81   30 REM IF USING CASSETTE, EN
     SURE IT IS SET TO THE END OF
      MCLOAD
15   40 REM AND MAKE SURE THAT LI
     NE 100 HAS BEEN CHANGED
C6   50 REM RUN MCFILE.BAS
51   60 BL=184  :LN=50   :SA=3686
     4
A2   70 FOR L=0 TO BL:CX=0:FOR D=
     0 TO 15:READ A:CX=CX+A
65   80 POKE53280,A:POKE SA+L*16+
     D,A:NEXT D
51   90 READ A:IF A><CX THENPRINT
     "ERROR IN LINE";LN+(L*10):ST
     OP
2E   100 NEXT L:REM IF USING CASS
     ETTE, INSERT POKE 39719,1 HE
     RE
B7   101 SYS39639
58   110 DATA 76,30,5,76,62,4,76,
     68,14,76,203,4,76,147,14,76,
     1007
77   120 DATA 161,14,76,240,4,76,
     6,5,76,123,14,76,144,9,76,16
     3,1263
2F   130 DATA 7,76,111,6,76,124,1
     0,76,117,10,76,186,5,76,252,
     12,1220
EA   140 DATA 76,25,12,76,186,12,
     76,14,14,76,48,14,76,175,14,
     96,990
0B   150 DATA 96,96,120,173,14,22
     0,41,254,141,14,220,173,17,2
     08,41,127,1955
0F   160 DATA 141,17,208,169,0,14
     1,18,208,169,114,141,20,3,16
     9,4,141,1663
36   170 DATA 21,3,173,26,208,9,1
     ,141,26,208,88,96,0,0,0,0,10
     00
84   180 DATA 0,0,0,0,0,0,169,1,4
     4,25,208,240,21,173,113,4,99
     8
C8   190 DATA 201,0,240,3,76,145,
     4,76,174,4,173,25,208,9,1,14
     1,1480
```



```
70   200 DATA 25,208,76,49,234,16
     9,8,44,24,208,208,22,173,17,
     208,9,1682
5B   210 DATA 32,141,17,208,169,2
     6,141,24,208,173,112,4,141,1
     8,208,76,1698
9A   220 DATA 134,4,173,17,208,41
     ,223,141,17,208,169,196,141,
     136,2,169,1979
3B   230 DATA 18,141,24,208,169,0
     ,141,18,208,32,113,5,76,134,
     4,32,1323
14   240 DATA 91,6,165,20,240,21,
     201,201,144,3,76,72,178,24,1
     65,20,1627
69   250 DATA 105,50,141,112,4,16
     9,100,141,113,4,96,169,0,141
     ,112,4,1461
7A   260 DATA 141,113,4,96,162,32
     ,169,224,133,252,169,0,133,2
     51,168,145,2192
BB   270 DATA 251,200,208,251,230
     ,252,202,208,246,96,162,4,16
     9,196,133,252,3060
89   280 DATA 169,0,133,251,168,1
     65,254,145,251,200,208,251,2
     30,252,202,208,3087
D2   290 DATA 246,96,120,173,2,22
     1,9,3,141,2,221,173,0,221,41
     ,252,1921
D9   300 DATA 9,0,141,0,221,169,1
     8,141,24,208,169,196,141,136
     ,2,169,1744
40   310 DATA 0,133,251,169,200,1
     33,252,169,0,133,253,169,208
     ,133,254,165,2622
15   320 DATA 1,41,251,133,1,160,
     0,177,253,145,251,230,253,23
     0,251,165,2542
A0   330 DATA 251,208,244,230,252
     ,230,254,165,254,201,216,208
     ,234,165,1,9,3122
61   340 DATA 4,133,1,88,96,173,1
     04,4,240,19,173,107,4,208,11
     ,169,1534
C6   350 DATA 0,141,4,212,141,104
     ,4,76,137,5,206,107,4,173,10
     5,4,1423
BB   360 DATA 240,19,173,108,4,20
     8,11,169,0,141,11,212,141,10
     5,4,76,1622
```

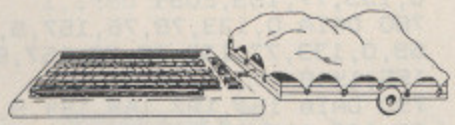
```
61  370 DATA 161,5,206,108,4,173
    ,106,4,240,19,173,109,4,208,
    11,169,1700
54  380 DATA 0,141,18,212,141,10
    6,4,76,185,5,206,109,4,96,32
    ,91,1426
2B  390 DATA 6,165,20,240,4,201,
    4,144,3,76,72,178,165,20,168
    ,136,1602
14  400 DATA 140,111,4,201,3,240
    ,20,201,2,240,8,169,0,141,11
    0,4,1594
1F  410 DATA 76,236,5,169,7,141,
    110,4,76,236,5,169,14,141,11
    0,4,1503
11  420 DATA 32,91,6,32,228,13,1
    41,24,212,32,91,6,32,101,6,1
    74,1221
94  430 DATA 110,4,157,0,212,32,
    91,6,32,101,6,174,110,4,157,
    1,1197
0C  440 DATA 212,32,91,6,32,101,
    6,174,110,4,157,5,212,32,91,
    6,1271
66  450 DATA 32,101,6,174,110,4,
    157,6,212,32,91,6,32,101,6,1
    74,1244
CE  460 DATA 110,4,157,2,212,32,
    91,6,32,228,13,174,110,4,157
    ,3,1335
6A  470 DATA 212,32,91,6,32,101,
    6,174,110,4,157,4,212,32,91,
    6,1270
```

```
50  480 DATA 32,101,6,172,111,4,
    153,107,4,169,100,153,104,4,
    96,32,1348
96  490 DATA 253,174,32,138,173,
    32,247,183,96,165,21,240,3,7
    6,72,178,2083
CF  500 DATA 165,20,96,32,91,6,3
    2,109,14,165,20,133,168,165,
    21,133,1370
34  510 DATA 167,32,91,6,32,97,1
    4,133,169,32,91,6,165,20,240
    ,4,1299
98  520 DATA 201,130,144,3,76,72
    ,178,133,82,32,91,6,32,240,1
    3,133,1566
3E  530 DATA 41,169,0,133,80,165
    ,82,133,81,169,1,133,170,165
    ,82,10,1614
64  540 DATA 56,233,1,133,171,56
    ,233,1,133,75,24,165,168,101
    ,81,170,1801
32  550 DATA 165,167,105,0,168,2
    4,165,169,101,80,176,3,32,14
    1,7,24,1527
68  560 DATA 165,168,101,80,170,
    165,167,105,0,168,56,165,169
    ,229,81,144,2133
16  570 DATA 3,32,141,7,56,165,1
    68,229,81,170,165,167,233,0,
    168,56,1841
62  580 DATA 165,169,229,80,144,
    3,32,141,7,56,165,168,229,80
    ,170,165,2003
7B  590 DATA 167,233,0,168,24,16
    5,169,101,81,176,3,32,141,7,
    230,80,1777
B7  600 DATA 230,170,230,170,56,
    165,75,229,170,133,75,176,22
    ,56,169,255,2381
CC  610 DATA 229,75,133,75,230,7
    5,198,81,198,171,198,171,56,
    165,171,229,2455
95  620 DATA 75,133,75,24,165,16
    8,101,80,170,165,167,105,0,1
    68,24,165,1785
E7  630 DATA 169,101,81,176,3,32
    ,141,7,24,165,168,101,81,170
    ,165,167,1751
1F  640 DATA 105,0,168,56,165,16
    9,229,80,144,3,32,141,7,56,1
    65,168,1688
D1  650 DATA 229,80,170,165,167,
    233,0,168,56,165,169,229,81,
    144,3,32,2091
8E  660 DATA 141,7,56,165,168,22
    9,81,170,165,167,233,0,168,2
    4,165,169,2108
17  670 DATA 101,80,176,3,32,141
    ,7,165,80,197,81,176,3,76,18
    2,6,1506
46  680 DATA 96,201,200,176,17,1
    33,253,192,0,240,4,224,64,17
    6,7,134,2117
55  690 DATA 251,132,252,32,180,
    9,96,32,91,6,32,109,14,165,2
    0,133,1554
69  700 DATA 167,165,21,133,168,
    32,91,6,32,97,14,133,169,32,
    91,6,1357
D1  710 DATA 32,109,14,165,21,13
    3,76,165,20,133,75,32,91,6,3
    2,97,1201
21  720 DATA 14,133,151,32,91,6,
    32,240,13,133,41,56,165,75,2
    29,167,1578
63  730 DATA 133,170,165,76,229,
    168,133,171,176,20,165,170,7
    3,255,133,170,2407
ED  740 DATA 230,170,165,171,73,
    255,133,171,165,170,208,2,23
    0,171,56,165,2535
DA  750 DATA 151,229,169,133,38,
    176,6,73,255,133,38,230,38,1
    65,171,208,2213
22  760 DATA 6,165,170,197,38,14
    4,66,165,76,197,168,144,34,2
    08,6,165,1949
48  770 DATA 75,197,167,144,26,1
    65,151,197,169,144,11,169,10
    0,133,77,169,2094
EC  780 DATA 0,133,78,76,157,8,1
    69,0,133,77,133,78,76,157,8,
    165,1448
8B  790 DATA 151,197,169,144,9,1
    69,100,133,77,133,78,76,157,
    8,169,0,1770
23  800 DATA 133,77,169,100,133,
    78,76,157,8,165,151,197,169,
    144,34,165,1956
D4  810 DATA 76,197,168,144,19,2
    08,6,165,75,197,167,144,11,1
    69,100,133,1979
D8  820 DATA 39,169,0,133,40,76,
    22,9,169,0,133,39,133,40,76,
    22,1100
22  830 DATA 9,165,76,197,168,14
    4,15,165,75,197,167,144,9,16
    9,100,133,1933
30  840 DATA 39,133,40,76,22,9,1
    69,0,133,39,169,100,133,40,7
    6,22,1200
D5  850 DATA 9,165,171,74,165,17
    0,106,133,81,169,0,133,82,13
    3,79,133,1803
01  860 DATA 80,32,126,9,165,78,
    208,11,230,167,165,167,208,1
```



```
    5,230,168,2059
8A  870 DATA 76,201,8,198,167,16
    5,167,201,255,208,2,198,168,
    56,165,81,2316
DD  880 DATA 229,38,133,81,165,8
    2,233,0,133,82,176,38,165,81
    ,73,255,1964
3F  890 DATA 133,81,230,81,165,8
    2,73,255,133,82,165,77,208,5
    ,198,169,2137
0E  900 DATA 76,241,8,230,169,56
    ,165,170,229,81,133,81,165,1
    71,229,82,2286
30  910 DATA 133,82,165,80,197,1
    71,208,7,165,79,197,170,208,
    1,96,230,2189
0B  920 DATA 79,165,79,208,2,230
    ,80,76,173,8,165,38,74,133,8
    1,169,1760
DE  930 DATA 0,133,82,133,79,32,
    126,9,165,40,208,5,230,169,7
    6,47,1534
5C  940 DATA 9,198,169,56,165,81
    ,229,170,133,81,165,82,229,1
    71,133,82,2153
A8  950 DATA 176,52,165,81,73,25
    5,133,81,230,81,165,82,73,25
    5,133,82,2117
D2  960 DATA 165,39,208,13,198,1
    67,165,167,201,255,208,13,19
    8,168,76,101,2342
BE  970 DATA 9,230,167,165;167,2
    08,2,230,168,56,165,38,229,8
    1,133,81,2129
AC  980 DATA 165,82,233,0,133,82
    ,165,79,197,38,208,1,96,230,
    79,76,1864
8C  990 DATA 33,9,165,168,133,25
    2,165,167,133,251,165,169,13
    3,253,32,180,2408
55  1000 DATA 9,96,0,0,32,91,6,3
    2,109,14,165,20,133,251,165,
    21,1144
EB  1010 DATA 133,252,32,91,6,32
    ,97,14,133,253,32,91,6,165,2
    0,201,1558
37  1020 DATA 3,144,3,76,72,178,
    133,41,165,251,41,7,170,56,1
    69,0,1509
CF  1030 DATA 133,90,106,202,16,
    252,133,89,165,251,41,248,13
    3,92,165,253,2369
70  1040 DATA 74,74,74,133,91,74
    ,102,90,74,102,90,101,91,133
    ,91,165,1559
49  1050 DATA 253,41,7,101,90,10
    1,92,133,90,165,91,101,252,1
    05,224,133,1979
D7  1060 DATA 91,120,165,1,41,25
    3,133,1,165,41,240,40,201,1,
```

```
    240,22,1755
25  1070 DATA 160,0,177,90,36,89
    ,208,7,169,0,133,2,76,109,10
    ,169,1435
AC  1080 DATA 100,133,2,76,109,1
    0,160,0,177,90,5,89,145,90,3
    2,50,1268
92  1090 DATA 10,76,109,10,160,0
    ,56,169,255,229,89,133,89,17
    7,90,37,1689
02  1100 DATA 89,145,90,76,109,1
    0,169,0,133,90,133,91,165,25
    3,41,248,1842
B8  1110 DATA 133,92,133,90,10,3
    8,91,10,38,91,24,101,92,133,
    90,165,1331
A9  1120 DATA 91,105,0,133,91,24
    ,169,0,101,90,133,90,24,169,
    196,101,1517
F7  1130 DATA 91,133,91,165,252,
    74,165,251,106,74,74,168,165
    ,254,145,90,2298
6A  1140 DATA 96,165,1,9,2,133,1
    ,88,96,169,100,133,171,76,12
    8,10,1378
7B  1150 DATA 169,0,133,171,32,9
    1,6,32,109,14,165,20,133,167
    ,165,21,1428
51  1160 DATA 133,168,32,91,6,32
    ,97,14,133,169,32,91,6,32,24
    0,13,1289
99  1170 DATA 133,75,201,0,208,7
    ,169,100,133,76,76,173,10,16
    9,0,133,1663
17  1180 DATA 76,165,171,240,52,
    32,91,6,169,0,133,252,165,21
    ,240,3,1816
4D  1190 DATA 76,72,178,165,20,1
    0,38,252,10,38,252,10,38,252
    ,133,251,1795
E4  1200 DATA 24,169,0,101,251,1
    33,251,169,200,101,252,133,2
    52,160,0,177,2373
02  1210 DATA 251,153,94,0,200,1
    92,8,208,246,169,0,133,170,1
    65,167,133,2289
A0  1220 DATA 251,165,168,133,25
    2,165,169,133,253,169,2,133,
    41,32,199,11,2276
C4  1230 DATA 165,2,197,76,208,2
    0,165,75,133,41,32,172,11,19
    8,251,165,1911
F4  1240 DATA 251,201,255,208,2,
    198,252,76,245,10,165,167,13
    3,251,165,168,2747
B4  1250 DATA 133,252,230,251,16
    5,251,208,2,230,252,169,2,13
    3,41,32,199,2550
04  1260 DATA 11,165,2,197,76,20
    8,10,165,75,133,41,32,172,11
    ,76,30,1404
B4  1270 DATA 11,165,170,208,34,
    165,167,133,251,165,168,133,
    252,198,253,169,2642
A5  1280 DATA 2,133,41,32,199,11
    ,165,2,197,76,208,3,76,2,11,
    169,1327
9E  1290 DATA 100,133,170,165,16
    9,133,253,165,167,133,251,16
    5,168,133,252,230,2787
93  1300 DATA 253,169,2,133,41,3
    2,199,11,165,2,197,76,208,3,
    76,2,1569
82  1310 DATA 11,96,165,253,201,
    200,176,8,165,252,240,28,201
    ,1,240,18,2255
7B  1320 DATA 165,75,240,5,169,1
    00,76,151,11,169,0,133,2,169
    ,100,133,1698
2C  1330 DATA 79,96,165,251,201,
    64,176,232,32,180,9,169,0,13
    3,79,96,1962
4C  1340 DATA 165,171,240,206,32
    ,240,11,165,78,240,8,165,75,
    133,41,32,2002
6D  1350 DATA 126,11,96,169,0,13
    3,41,32,126,11,96,32,126,11,
    165,79,1254
FA  1360 DATA 208,4,165,171,208,
    1,96,165,75,208,251,32,240,1
    1,165,2,2002
92  1370 DATA 197,78,208,11,165,
    2,201,100,240,236,169,100,13
    3,2,96,169,2107
BB  1380 DATA 0,133,2,96,165,253
    ,41,7,170,181,94,133,77,165,
    251,41,1809
76  1390 DATA 7,170,56,169,0,106
    ,202,16,252,133,89,165,77,37
    ,89,201,1769
98  1400 DATA 0,240,5,169,100,13
    3,78,96,169,0,133,78,96,32,9
    1,6,1426
DA  1410 DATA 165,20,72,165,21,7
    2,32,91,6,165,20,72,165,21,7
    2,32,1191
85  1420 DATA 91,6,160,3,104,153
    ,97,0,136,16,249,165,20,5,21
    ,240,1466
06  1430 DATA 67,165,98,197,100,
    144,65,208,6,165,97,197,99,1
    44,57,216,2025
8C  1440 DATA 24,165,97,101,20,1
    33,97,165,98,101,21,133,98,2
    4,165,99,1541
EC  1450 DATA 101,20,133,99,165,
    100,101,21,133,100,32,170,12
    ,198,100,198,1683
AF  1460 DATA 98,160,255,177,99,
    145,97,152,208,4,198,100,198
    ,98,136,32,2157
D8  1470 DATA 157,12,208,239,32,
    178,12,96,32,170,12,160,0,17
    7,99,145,1729
DE  1480 DATA 97,200,208,4,230,1
    00,230,98,32,157,12,208,240,
    32,178,12,2038
```



```
F8  1490 DATA 96,165,20,208,2,19
    8,21,198,20,165,20,5,21,96,1
    20,165,1520
DA  1500 DATA 1,41,252,133,1,96,
    165,1,9,3,133,1,88,96,32,91,
    1143
B0  1510 DATA 6,165,20,72,165,21
    ,72,32,91,6,165,20,72,165,21
    ,72,1165
90  1520 DATA 32,91,6,165,21,208
    ,37,166,20,104,133,21,104,13
    3,20,104,1365
41  1530 DATA 133,98,133,100,104
    ,133,97,133,99,230,97,208,2,
    230,98,160,2055
18  1540 DATA 0,138,145,99,32,15
    7,12,240,6,76,132,12,76,72,1
    78,96,1471
37  1550 DATA 32,91,6,165,20,201
    ,8,144,3,76,72,178,141,227,1
    3,32,1409
37  1560 DATA 252,13,32,91,6,32,
    240,13,208,10,173,21,208,45,
    226,13,1583
8B  1570 DATA 141,21,208,96,32,9
    1,6,32,101,6,174,227,13,157,
    248,199,1752
07  1580 DATA 32,91,6,32,228,13,
    174,227,13,157,39,208,32,91,
    6,174,1523
FA  1590 DATA 227,13,138,10,170,
    141,227,13,165,20,157,0,208,
    165,21,240,1915
52  1600 DATA 12,173,16,208,13,2
    25,13,141,16,208,76,98,13,17
    3,16,208,1609
F8  1610 DATA 45,226,13,141,16,2
    08,32,91,6,32,101,6,174,227,
    13,157,1488
24  1620 DATA 1,208,32,91,6,32,2
    40,13,208,12,173,29,208,45,2
    26,13,1537
08  1630 DATA 141,29,208,76,139,
    13,173,29,208,13,225,13,141,
    29,208,32,1677
70  1640 DATA 91,6,32,240,13,208
    ,12,173,23,208,45,226,13,141
    ,23,208,1662
57  1650 DATA 76,168,13,173,23,2
    08,13,225,13,141,23,208,32,9
    1,6,32,1445
44  1660 DATA 240,13,208,12,173,
    28,208,45,226,13,141,28,208,
    76,197,13,1829
77  1670 DATA 173,28,208,13,225,
    13,141,28,208,32,91,6,32,228
    ,13,141,1580
F1  1680 DATA 37,208,32,91,6,32,
    228,13,141,38,208,173,21,208
    ,13,225,1674
1C  1690 DATA 13,141,21,208,96,0
    ,0,0,165,20,201,16,144,3,76,
    72,1176
FA  1700 DATA 178,165,20,96,165,
    20,201,2,144,3,76,72,178,165
    ,20,96,1601
9F  1710 DATA 168,200,56,169,0,4
    2,136,208,252,141,225,13,73,
    255,141,226,2305
2B  1720 DATA 13,96,32,253,174,3
    2,212,225,32,91,6,165,20,72,
    165,21,1609
11  1730 DATA 72,32,91,6,166,20,
    164,21,104,133,21,104,133,20
    ,169,20,1276
A2  1740 DATA 32,216,255,96,32,2
    53,174,32,212,225,32,91,6,16
    9,0,166,1991
28  1750 DATA 20,164,21,32,213,2
    55,96,225,120,169,49,141,20,
    3,169,234,1931
6F  1760 DATA 141,21,3,173,26,20
    8,41,254,141,26,208,173,14,2
    20,9,1,1659
CC  1770 DATA 141,14,220,88,96,1
    65,20,201,200,144,3,76,72,17
    8,165,20,1803
4F  1780 DATA 96,165,21,240,9,16
    5,20,201,64,144,3,76,72,178,
    96,32,1582
```

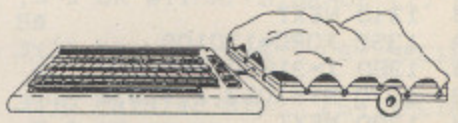
```
1C  1790 DATA 91,6,32,228,13,10,
    10,10,10,133,254,32,91,6,32,
    228,1186
01  1800 DATA 13,24,101,254,133,
    254,96,173,17,208,9,32,141,1
    7,208,169,1849
C6  1810 DATA 26,141,24,208,96,1
    73,17,208,41,223,141,17,208,
    169,18,141,1851
6F  1820 DATA 24,208,96,32,91,6,
    165,21,208,25,165,20,201,25,
    176,19,1482
C0  1830 DATA 72,32,91,6,104,170
    ,165,21,208,9,164,20,192,40,
    176,3,1473
54  1840 DATA 76,240,255,76,72,1
    78,0,169,0,133,250,169,144,1
    33,251,169,2315
25  1850 DATA 252,133,174,133,19
    3,169,3,133,175,133,194,169,
    215,133,252,169,2630
A1  1860 DATA 154,133,253,160,0,
    177,250,145,174,230,250,208,
```

```
    2,230,251,230,2847
EC  1870 DATA 174,208,2,230,175,
    165,250,197,252,208,234,165,
    251,197,253,208,3169
A5  1880 DATA 228,169,114,133,18
    7,169,155,133,188,169,6,133,
    183,169,0,133,2269
DB  1890 DATA 185,169,0,32,144,2
    55,169,8,76,61,155,200,208,2
    45,32,207,2146
99  1900 DATA 255,240,251,201,49
    ,240,4,201,56,48,230,41,15,1
    33,186,76,2226
14  1910 DATA 234,245,147,17,17,
    73,78,80,85,84,32,68,69,86,7
    3,67,1455
47  1920 DATA 69,32,78,85,77,66,
    69,82,13,17,67,65,83,61,49,3
    2,945
FA  1930 DATA 47,32,68,73,83,75,
    61,32,56,32,79,82,32,57,58,4
    5,912
E1  1940 DATA 32,0,77,67,70,73,7
    6,69,0,0,0,0,0,0,0,0,464
5D  1950 DATA 0,0,255,255,255,25
    5,255,0,0,0,0,0,0,0,0,0,1275
```

PROGRAM: DEMO

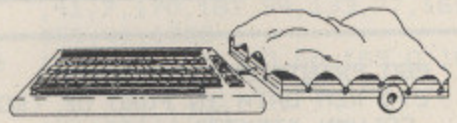
```
4A  10 REM ** EXAMPLE1 **
CA  20 PRINT"[CLR,DOWN] MOVE MEM
    ORY FROM CHARACTER SET"
50  30 PRINT" STARTING AT 51200
    ($C800) TO THE"
56  40 PRINT" BITMAP SCREEN AT 5
    7344 ($E000)"
E2  50 PRINT" IN BLOCKS OF 8. IE
    .(1 CHARACTER)"
2D  60 PRINT" THEN FILL REMAININ
    G SCREEN WITH"
6D  70 PRINT" CHARACTER NO. 127.
    "
AF  80 GOSUB2990:REM GET ANY KEY
     TO CONTINUE
AA  90 SYS1044,2,10:REM SET COLO
    URS RED ON L/RED
1C  100 SYS1038:SYS1041:REM CLEA
    R AND COLOUR BITMAP
```

```
B3  110 SYS1032:REM BITMAP ON
4D  120 READ ROW,COL,R$
66  130 IFROW=256THEN230
D9  140 FOR C=1TOLEN(R$)
80  150 SBM=57344+(ROW*320)+(COL
    *8):REM POSITION ON BITMAP
3C  160 CH$=MID$(R$,C,1):CHAR=AS
    C(CH$)
5B  170 IFCHAR>=64THENCHAR=CHAR-
    64:REM CONVERT ASCII
C1  180 IFCHAR>=96THENCHAR=CHAR-
    64:REM TO CBM DISPLAY
DB  190 CN=51200+(CHAR*8):REM WH
    ICH CHARACTER TO PRINT
4A  200 SYS1068,SBM,CN,8:REM MOV
    E MEMORY FROM CHARACTER SET
    TO BITMAP
B7  210 COL=COL+1:NEXT
E4  220 GOTO120
DD  230 SYS1059,25,100,1,127:SYS
    1059,300,100,1,127:REM FILL
    BITMAP
9B  240 GET A$:IFA$=""THEN240
98  250 SYS1035:REM BITMAP OFF
CD  260 PRINT"[CLR]":GOTO430
32  270 REM DATA IN FORM OF (ROW
    ,COLUMN,"TEXT")
CF  280 DATA 6,9,"[SU,S*20,SI]"
3E  290 DATA 7,9,"[S-,SPC20,S-]"

86  300 DATA 8,9,"[S-] THIS IS A
    N EXAMPLE [S-]"
38  310 DATA 9,9,"[S-,SPC20,S-]"

58  320 DATA 10,9,"[S-]  OF WRIT
    ING TEXT[SPC3,S-]"
07  330 DATA 11,9,"[S-,SPC20,S-]
    "
6D  340 DATA 12,9,"[S-] TO A BIT
    MAP SCREEN [S-]"
31  350 DATA 13,9,"[S-,SPC20,S-]
    "
66  360 DATA 14,9,"[S-]  USING T
    HE MEMORY  [S-]"
DB  370 DATA 15,9,"[S-,SPC20,S-]
    "
B6  380 DATA 16,9,"[S-,SPC4]MOVE
     COMMAND.[SPC3,S-]"
C5  390 DATA 17,9,"[S-,SPC20,S-]
    "
3D  400 DATA 18,9,"[SJ,S*20,SK]"

71  410 DATA 256,256,"":REM END
    OF DATA
20  420 REM ** EXAMPLE2 **
DA  430 PRINT"[DOWN] USING THE A
    BOVE BITMAP"
BE  440 PRINT" HERE IS AN EXAMPL
    E OF USING"
66  450 PRINT" THE SPLIT-SCREEN
    COMMAND TO"
49  460 PRINT" ROLL THE BITMAP U
    P AND DOWN"
DF  470 PRINT" THE SCREEN."
8D  480 PRINT" PRESS ANY KEY TO
    END DEMO."
EA  490 GOSUB2990
59  500 SYS1023:REM RASTER INTER
    RUPT ON
97  510 SYS1041:REM COLOUR BITMA
    P
5C  520 SYS1029,1:REM SET BITMAP
     AT TOP OF SCREEN
C9  530 GET A$:IFA$=""THEN570
32  540 SYS1029,0:REM SPLIT SCRE
    EN OFF
BC  550 SYS1026:REM RASTER OFF
9A  560 PRINT"[CLR]":GOTO650
A6  570 FOR R=1 TO 200
1B  580 SYS1029,R:REM SCREEN SPL
    IT
D2  590 NEXT
0B  600 FORR=200 TO 1STEP-1
F2  610 SYS1029,R
EC  620 NEXT
45  630 GOTO530
84  640 REM ** EXAMPLE3 **
0B  650 PRINT"[DOWN] USE MEMORY
    FILL COMMAND TO COLOUR"
C9  660 PRINT" PARTS OF BITMAP A
    ND THEN USE IT"
AC  670 PRINT" TO CLEAR PART OF
    THE SCREEN,"
9A  680 PRINT" PRESSING ANY KEY
    TO GET NEXT PART"
37  690 PRINT" OF DEMO."
A5  700 GOSUB2990
C4  710 SYS1041:REM COLOUR BITMA
    P
27  720 SYS1032:REM BITMAP ON
2B  730 GET A$:IFA$=""THEN730
EA  740 SYS1044,0,5:REM SET COLO
    URS
D7  750 C=PEEK(254):REM 254 IS T
    HE CURRENT COLOUR
26  760 SCREEN=1024+49152+(5*40)
    :REM SCREEN ADDRESS IN BANK
    3 + 5 LINES DOWN
76  770 NO=15*40:REM NO=NUMBER O
    F LINES * 40 CHARACTERS TO T
    HE LINE
```



```
F0  780 SYS1071,SCREEN,NO,C
AA  790 GET A$:IFA$=""THEN790
C5  800 BITMAP=57344+(320*5):NO=
    320*15:REM CLEAR 15 LINES 5
    LINES DOWN
0E  810 SYS1071,BITMAP,NO,0:REM
    FILL MEMORY WITH (0)
54  820 GET A$:IFA$=""THEN820
D1  830 SYS1035:REM BITMAP OFF
89  840 PRINT"[CLR]"
0A  850 REM ** EXAMPLE4 **
A6  860 PRINT"[DOWN] NOW FOR SOM
    E LINE DRAWING DEMOS"
```

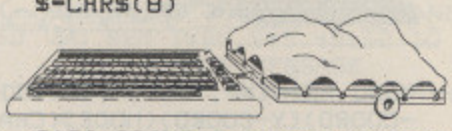
```
4F  870 GOSUB2990:POKE53280,0
C1  880 SYS1044,7,0:SYS1041:SYS1
    038:REM SET COLOUR, COLOUR A
    ND CLEAR BITMAP
72  890 SYS1032:REM BITMAP ON
AF  900 FOR X=0TO319STEP5
D7  910 SYS1050,X,0,X,199,1:REM
    DRAW LINE COMMAND
19  920 NEXT
7C  930 FORY=0TO199STEP5
2F  940 SYS1050,0,Y,319,Y,1:REM
    1=PLOT
3B  950 NEXT
86  960 FORD=0TO319STEP4
01  970 B=319-D
DD  980 SYS1050,D,0,B,199,1
63  990 NEXT
23  1000 FORD=0TO199STEP3
17  1010 B=199-D
A7  1020 SYS1050,0,B,319,D,1
88  1030 NEXT
5E  1040 FORD=0TO199STEP3
A8  1050 B=199-D
7F  1060 SYS1050,0,D,319,B,0:REM
     0=UNPLOT
B0  1070 NEXT
61  1080 FORD=0TO319STEP4
7E  1090 B=319-D
B9  1100 SYS1050,B,0,D,199,0
D8  1110 NEXT
4D  1120 GETA$:IFA$=""THEN1120
E9  1130 SYS1035:PRINT"[CLR]"
40  1140 REM ** EXAMPLE5 **
BE  1150 PRINT"[DOWN] NOW FOR SO
    ME CIRCLES"
73  1160 GOSUB2990
20  1170 SYS1044,4,3:SYS1038:SYS
    1041:SYS1032
E2  1180 FOR R=1TO129
C2  1190 SYS1053,160,100,R,1:REM
     CILCLE COMMAND (1=PLOT)
36  1200 NEXT
30  1210 FOR R=129TO1STEP-2
9B  1220 SYS1053,160,100,R,0:REM
     0=UNPLOT
50  1230 NEXT
14  1240 GETA$:IFA$=""THEN1240
39  1250 POKE53280,6
DD  1260 SYS1044,14,7:SYS1041
F3  1270 SYS1038:REM CLEAR BITMA
    P
A0  1280 FORD=1TO100
21  1290 SYS1053,D,D,D,1
9B  1300 NEXT
92  1310 FORD=1TO100
B8  1320 A=199-D
AC  1330 SYS1053,D,A,D,1
C3  1340 NEXT
7A  1350 FORD=1TO100
C8  1360 X=319-D:Y=199-D
18  1370 SYS1053,X,Y,D,1
EB  1380 NEXT
A2  1390 FORD=1TO100
93  1400 X=319-D
BD  1410 SYS1053,X,D,D,1
13  1420 NEXT
F2  1430 GETA$:IFA$=""THEN1430
66  1440 FORD=100TO1STEP-2
D9  1450 X=319-D
4E  1460 SYS1053,X,D,D,0
41  1470 NEXT
5E  1480 FORD=100TO1STEP-2
56  1490 X=319-D:Y=199-D
5B  1500 SYS1053,X,Y,D,0
69  1510 NEXT
16  1520 FORD=100TO1STEP-2
AA  1530 A=199-D
38  1540 SYS1053,D,A,D,0
96  1550 NEXT
BB  1560 FORD=100TO1STEP-2
0B  1570 SYS1053,D,D,D,0
B0  1580 NEXT
20  1590 GETA$:IFA$=""THEN1590
E1  1600 SYS1035:PRINT"[CLR]"
BB  1610 SYS1035:PRINT"[CLR]"
A1  1620 REM ** EXAMPLE6 **
0E  1630 PRINT"[DOWN] DRAW SOME
    SHAPES ON SCREEN AND"
AE  1640 PRINT" USE THE FILL AND
     PATTERN FILL COMMANDS,"
EA  1650 PRINT" FINALLY ERASING
    EVERYTHING AGAIN"
81  1660 GOSUB2990
```



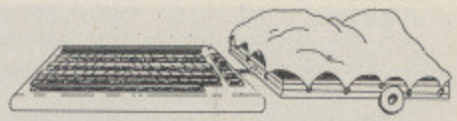
```
82  1670 SYS1044,2,3:SYS1041:SYS
    1038
0E  1680 SYS1032
FF  1690 M=1:REM 1=PLOT 0=UNPLOT

18  1700 SYS1050,10,35,35,10,M:R
    EM DRAW LINE COMMANDS
BE  1710 SYS1050,35,10,60,35,M
B9  1720 SYS1050,60,35,35,60,M
62  1730 SYS1050,35,60,10,35,M
F5  1740 SYS1050,260,10,310,10,M

BE  1750 SYS1050,310,10,310,60,M

8B  1760 SYS1050,310,60,260,60,M

3A  1770 SYS1050,260,60,260,10,M

5B  1780 SYS1050,260,165,285,140
    ,M
9D  1790 SYS1050,285,140,310,165
    ,M
43  1800 SYS1050,310,165,285,190
    ,M
3F  1810 SYS1050,285,190,260,165
    ,M
21  1820 SYS1050,10,140,60,140,M

4D  1830 SYS1050,60,140,60,190,M

57  1840 SYS1050,60,190,10,190,M

19  1850 SYS1050,10,190,10,140,M

A1  1860 SYS1053,110,100,40,M:RE
    M DRAW CIRCLE COMMANDS
75  1870 SYS1053,160,50,40,M
EF  1880 SYS1053,210,100,40,M
F2  1890 SYS1053,160,150,40,M
0C  1900 SYS1053,160,100,90,M
AD  1910 SYS1056,110,100,M:REM F
    ILL AREAS OF BITMAP
DC  1920 SYS1056,160,50,M
FA  1930 SYS1056,210,100,M
D1  1940 SYS1056,160,150,M
24  1950 SYS1059,35,35,M,49:REM
    FILL WITH PATTERN
7E  1960 SYS1059,285,35,M,50:REM
     FILL NO'S REPRESENT
33  1970 SYS1059,285,165,M,51:RE
    M (1,2,3,4) RESPECTIVELY
AE  1980 SYS1059,35,165,M,52:REM
     SEE SCREEN DISPLAY CODES IN
     USER GUIDE
0C  1990 SYS1059,65,100,M,102
23  2000 SYS1059,255,100,M,102
FF  2010 SYS1059,5,35,M,102
E1  2020 SYS1059,315,35,M,102
45  2030 SYS1059,315,165,M,102
8B  2040 SYS1059,5,165,M,102
4F  2050 SYS1059,160,100,M,102
8E  2060 IFM=1THENSYS1068,40960,
    57344,8000:REM STORE BITMAP
    UNDER BASIC ROM
D8  2070 GETA$:IFA$=""THEN2070
5E  2080 IFM=1THENM=0:GOTO1700:R
    EM SET MODE TO 0=UNPLOT
8D  2090 SYS1035:PRINT"[CLR]"
82  2100 REM ** EXAMPLE7 **
A4  2110 PRINT"[DOWN] MOVE THE P
    REVIOUSLY STORED BITMAP"
DA  2120 PRINT" ( SEE LINE 2130
    IN ABOVE DEMO)"
4C  2130 PRINT" FROM THE STORAGE
     SPACE UNDER THE BASIC"
BE  2140 PRINT" ROM BACK INTO TH
    E BITMAP SCREEN"
55  2150 GOSUB2990
BC  2160 SYS1041:SYS1032
49  2170 SYS1068,57344,40960,800
    0:REM MOVE DATA
8A  2180 GETA$:IFA$=""THEN2180
91  2190 SYS1035:PRINT"[CLR]"
9D  2200 REM ** EXAMPLE8 **
E7  2210 PRINT"[DOWN] MOVE BITMA
    P DATA TO MEMORY AT"
B0  2220 PRINT" 24576 ($6000) AN
    D THEN SAVE IT TO DISK"
51  2230 PRINT" THEN CLEAR BITMA
    P AND LOAD THE SAVED"
63  2240 PRINT" DATA DIRECTLY IN
    TO THE BITMAP SCREEN"
F1  2250 PRINT" AT 57344 ($E000)
    ."
D2  2260 PRINT" PLEASE ENSURE TH
    AT THE DRIVE IS READY"
53  2270 PRINT" BEFORE CONTINUIN
    G WITH THIS DEMO."
28  2280 PRINT" IF YOU WANT TO S
    KIP THIS DEMO PRESS <S>"
F9  2290 GOSUB2990
9E  2300 IF A$="S"THEN2390
55  2310 SYS1032:SYS1041
C9  2320 SYS1068,24576,57344,800
    0:REM MOVE BITMAP DATA TO ME
    MORY STARTING AT 24576
88  2330 SYS1038:REM CLEAR BITMA
    P
61  2340 SYS1074,"DATA",08,00,24
    576,32576:REM SAVE MEM. FROM
     24576 TO 24576+8000
2F  2350 SYS1077,"DATA",08,00,57
    344:REM LOAD DATA DIRECTLY I
    NTO BITMAP
EB  2360 GETA$:IFA$=""THEN2360
C4  2370 SYS1035:PRINT"[CLR]"
17  2380 REM ** EXAMPLE9 **
6E  2390 PRINT"[CLR,DOWN] NOW FO
    R A DEMONSTRATION OF THE SOU
    ND"
29  2400 PRINT" AND SPRITE SETUP
     COMMANDS"
CD  2410 GOSUB2990:POKE53280,0:P
```

```
    OKE53281,0
69  2420 SYS1080,19,0:PRINT"[RVS
    ON,SPC28,RVSOFF,SPC10,RVSON]
     [RVSOFF]"
98  2430 SYS1080,20,28:PRINT"[C*
    ,SPC8,S£]":REM PRINT AT ROW-
    21, COLUMN-29
0A  2440 SYS1080,21,28:PRINT" [R
    VSON,CI,RVSOFF,CI6,RVSON,CI,
    RVSOFF] "
8E  2450 SYS1080,23,8:PRINT"[SPC
    26]"
05  2460 SYS1023:FIN=0:X=40:Y=25
    :GOSUB2620
E5  2470 V1=PEEK(1128):REM TEST
    VOICE1
D3  2480 IF V1=0 THENGOSUB2720:R
    EM IF OFF THEN READ NEXT NOT
    E DATA
47  2490 IF MC=2 THENS=D:REM USE
     DELAY TO CONTROL SPRITE MOV
    EMENT
89  2500 X=X+(S/50):Y=Y+(S/75):R
    EM LONGER NOTE PLAYS FASTER
    SPRITE MOVES
12  2510 SYS1065,0,1,0,7,X,Y,1,1
    ,1,1,MC:REM SPRITE SETUP COM
    MAND
D8  2520 IFFIN<>1THEN2470:REM IF
     MORE NOTES
28  2530 FORD=0TO10
80  2540 Y=Y+1
16  2550 SYS1065,0,1,0,7,X,Y,1,1
    ,1,1,0:REMSPRITE SETUP COMMA
    ND
88  2560 NEXT
DC  2570 V3=PEEK(1130):IF V3<>0T
    HEN2570:REM WAIT TILL VOICE
    3 FINISHED PLAYING
DF  2580 SYS1026:REM TURN OFF IN
    TERRUPT
17  2590 GOSUB2990
AE  2600 SYS1065,0,0
52  2610 PRINT"[CLR]":GOTO3040
F5  2620 FORD=0TO63:READ A:POKE4
    9152+D,A:NEXT
8C  2630 RETURN:REM PLACE SPRITE
```

```
    DATA IN SPRITE 0 (49152-)
2D  2640 DATA 000,000,000,000,00
    0,000,000,000
C3  2650 DATA 000,000,000,000,00
    0,060,000,002
4D  2660 DATA 190,128,042,170,16
    8,170,170,170
78  2670 DATA 170,170,170,238,23
    8,238,187,187
98  2680 DATA 187,170,170,170,04
    2,170,168,006
B1  2690 DATA 170,144,005,000,08
    0,020,000,020
57  2700 DATA 016,000,004,084,00
    0,021,000,000
F7  2710 DATA 000,000,000,000,00
    0,000,000,000
07  2720 READ LF,HF,D,MC:REM GET
     NOTE DATA PLUS MULTICOL 2(S
    EE LINE 3542)
6C  2730 IFLF=256THEN2780:REM TE
    ST FOR END OF DATA
5E  2740 SYS1062,1,15,LF,HF,0,24
    0,0,3,65,D:REM VOICE 1
FC  2750 SYS1062,2,15,43,137,128
    ,255,0,0,33,100:REM VOICE2
0E  2760 SYS1062,3,15,149,68,128
    ,255,0,0,129,100:REM VOICE3
D9  2770 GOTO2790
7D  2780 FIN=1:REM SIGNAL NO MOR
    E DATA SEE LINE 3550
7E  2790 RETURN
E1  2800 DATA 53,7,75,2:REM NOTE
     DATA
4F  2810 DATA 23,8,75,1
B7  2820 DATA 108,6,75,2
C5  2830 DATA 0,0,15,1
98  2840 DATA 54,3,75,2
2C  2850 DATA 208,4,100,1
46  2860 DATA 53,7,50,2
ED  2870 DATA 23,8,50,1
30  2880 DATA 108,6,50,2
54  2890 DATA 0,0,10,1
AB  2900 DATA 54,3,50,2
85  2910 DATA 208,4,75,1
7B  2920 DATA 107,14,20,2
8B  2930 DATA 47,16,20,1
72  2940 DATA 216,12,20,2
9A  2950 DATA 0,0,5,1
BB  2960 DATA 108,6,20,2
9D  2970 DATA 159,9,40,1
F9  2980 DATA 256,0,0,0
63  2990 SYS1080,23,8:REM SET CU
    RSOR AT 23 LINE, 8 CHARACTER
    S ALONG
9A  3000 PRINT"[RVSON]PRESS ANY
    KEY TO CONTINUE[RVSOFF]"
BC  3010 GET A$:IF A$=""THEN3010
59  3020 RETURN
9A  3030 REM ** MEM/MAP AND COMM
    ANDS PRINTOUT **
8C  3040 PRINT"[DOWN] THIS PART
    OF THE PROGRAM PRINTS OUT"
76  3050 PRINT" A MEMORY MAP AND
     A LIST OF THE NEW"
E1  3060 PRINT" COMMANDS. SO IF
    YOU HAVE A PRINTER"
BB  3070 PRINT" MAKE SURE IT IS
    READY NOW."
2C  3080 PRINT" IF YOU DON'T HAV
    E A PRINTER THEN YOU"
C2  3090 PRINT" WILL JUST HAVE T
    O REFER TO THE MAGAZINE"
17  3100 GOSUB2990
AC  3110 OPEN2,4:SI$=CHR$(15):BS
    $=CHR$(8)
```

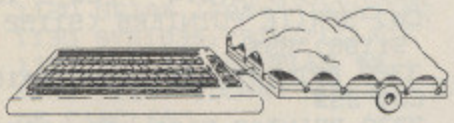


```
EA  3120 PRINT#2,CHR$(14)"[SPC13
    ]NEW MEMORY MAP"
AB  3130 PRINT#2,CHR$(14)"[SPC12
    ,CY16]"SI$
FA  3140 PRINT#2
D5  3150 PRINT#2,SI$"  [CA,S*18,
    CS]"BS$
04  3160 PRINT#2,SI$"  [CQ,S*18,
    CW]-192 BYTES STORAGE"BS$
F7  3170 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
39  3180 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
CE  3190 PRINT#2,SI$"  [S-,SPC18
    ,S-]-BITMAP SCREEN (57344 -
    65344)"BS$
A5  3200 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
06  3210 PRINT#2,SI$"  [S-,SPC18
    ,S-] OR STORAGE AREA"BS$
01  3220 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
```



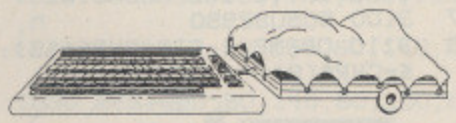
```
DB  3230 PRINT#2,SI$"  [CQ,S*18,
    CW]"BS$
CD  3240 PRINT#2,SI$"  [S-,SN18,
    S-]"BS$
F9  3250 PRINT#2,SI$"  [S-,SN18,
    S-]-UNUSABLE ( VIC, SID, I/O
     ETC. )"BS$
81  3260 PRINT#2,SI$"  [S-,SN18,
    S-]"BS$
73  3270 PRINT#2,SI$"  [CQ,S*18,
    CW]"BS$
4A  3280 PRINT#2,SI$"  [S-,SPC18
    ,S-]-CHARACTER SET (51200 -
    53248)(256 ";
0E  3290 PRINT#2,"CHARS. * 8 BYT
    ES)"BS$
91  3300 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
BC  3310 PRINT#2,SI$"  [CQ,S*18,
    CW]-SPRITE POINTERS (51192 -
     51199)"BS$
71  3320 PRINT#2,SI$"  [CQ,S*18,
    CW]"BS$
72  3330 PRINT#2,SI$"  [S-,SPC18
    ,S-]-SCREEN MEMORY (50176 -
    51175)"BS$
B6  3340 PRINT#2,SI$"  [CQ,S*18,
    CW]"BS$
1A  3350 PRINT#2,SI$"  [S-,SPC18
    ,S-]-DISPLAY SPRITE STORAGE
    (0-15) (49152";
AC  3360 PRINT#2," - 50175)"BS$
F8  3370 PRINT#2,SI$"  [CQ,S*18,
    CW]"BS$
DF  3380 PRINT#2,SI$"  [S-,SPC18
    ,S-]-SPRITE STORAGE (128 SPR
    ITES)"BS$
9C  3390 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
B2  3400 PRINT#2,SI$"  [S-,SPC18
    ,S-] OR CHARACTER SETS (4)"B
    S$
28  3410 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
DF  3420 PRINT#2,SI$"  [S-,SPC18
    ,S-] OR BITMAP STORAGE (1)"B
    S$
04  3430 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
86  3440 PRINT#2,SI$"  [S-,SPC18
    ,S-] OR OTHER (40960 - 49151
    )"BS$
88  3450 PRINT#2,SI$"  [CQ,S*18,
    CW]"BS$
17  3460 PRINT#2,SI$"  [S-,SPC18
    ,S-]-TOP OF BASIC (40959)"BS
    $
EC  3470 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
BE  3480 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
38  3490 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
```

```
A6  3500 PRINT#2,SI$"  [S-,SN,SM
    ,SN,SM,SN,SM,SN,SM,SN,SM,SN,
    SM,SN,SM,SN,SM,SN,SM,S-]"
5A  3510 PRINT#2,SI$"[SPC3]MAIN
    BASIC MEMORY  - 37162 BASIC
    BYTES FREE"
53  3520 PRINT#2,SI$"  [S-,SM,SN
    ,SM,SN,SM,SN,SM,SN,SM,SN,SM,
    SN,SM,SN,SM,SN,SM,SN,S-]"BS$

80  3530 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
C2  3540 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
FD  3550 PRINT#2,SI$"  [S-,SPC18
    ,S-]"BS$
79  3560 PRINT#2,SI$"  [S-,SPC18
    ,S-]-START OF BASIC (3795)"B
    S$
50  3570 PRINT#2,SI$"  [CQ,S*18,
    CW]"BS$
BD  3580 PRINT#2,SI$"  [CQ,S*18,
    CW]-NEW COMMANDS PROGRAM (10
    20 -3794)"BS$
AD  3590 PRINT#2,SI$"  [S-,SN18,
    S-]"BS$
1C  3600 PRINT#2,SI$"  [CZ,S*18,
    CX]- OPERATING SYSTEMS AREA"
    BS$
```

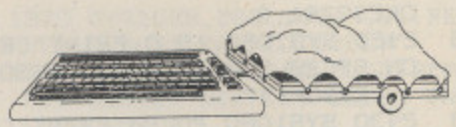


```
8C  3610 PRINT#2,SI$
5E  3620 PRINT#2,"  CURRENT COLO
    UR STORED IN - (254)":PRINT#
    2
83  3630 PRINT#2,"  VOICE CONTRO
    LS (PEEK TO FIND OUT IF VOIC
    E STILL ON)";
9B  3640 PRINT#2,"I.E. 0=OFF, 10
    0=ON"
49  3650 PRINT#2,"[SPC5]V1=1128,
     V2=1129, V3=1130"
96  3660 PRINT#2:PRINT#2,"  PIXE
    L ON OR OFF - PEEK(02) I.E.
    0=OFF, 100=ON"
6B  3670 GOSUB2990
5E  3680 PRINT#2,CHR$(14)"[SPC12
    ]LIST OF COMMANDS"
FF  3690 PRINT#2,CHR$(14)"[SPC11
    ,CY18]"SI$
4C  3700 PRINT#2
56  3710 PRINT#2,"RECONFIGURE -
    SYS1020"
65  3720 PRINT#2,"[CT11]"
57  3730 PRINT#2,"RASTER INTERRU
    PT ON - SYS1023"
11  3740 PRINT#2,"[CT19]"
06  3750 PRINT#2,"RASTER INTERRU
    PT OFF - SYS1026"
5C  3760 PRINT#2,"[CT20]"
2A  3770 PRINT#2,"SPLIT-SCREEN -
     SYS1029,0 OR 1-200"
39  3780 PRINT#2,"[CT12,SPC10](0
    =OFF, 1-200=SCREEN LINE)"
76  3790 PRINT#2
FC  3800 PRINT#2,"BITMAP ON - SY
    S1032"
ED  3810 PRINT#2,"[CT9]"
CB  3820 PRINT#2,"BITMAP OFF - S
    YS1035"
78  3830 PRINT#2,"[CT10]"
BB  3840 PRINT#2,"CLEAR BITMAP -
     SYS1038"
7B  3850 PRINT#2,"[CT12]"
AE  3860 PRINT#2,"COLOUR BITMAP
    - SYS1041"
08  3870 PRINT#2,"[CT13]"
85  3880 PRINT#2,"SET CURRENT CO
    LOUR - SYS1044,0-15,0-15"
BD  3890 PRINT#2,"[CT18,SPC10](D
    RAW)(BACKGROUND)"
05  3900 PRINT#2
59  3910 PRINT#2,"PLOT POINT ON
    BITMAP - SYS1047,0-319,0-199
    ,0-1-2"
F7  3920 PRINT#2,"[CT20,SPC10](X
    -COORD)(Y-COORD)(MODE 0=UNLO
    T,";
```

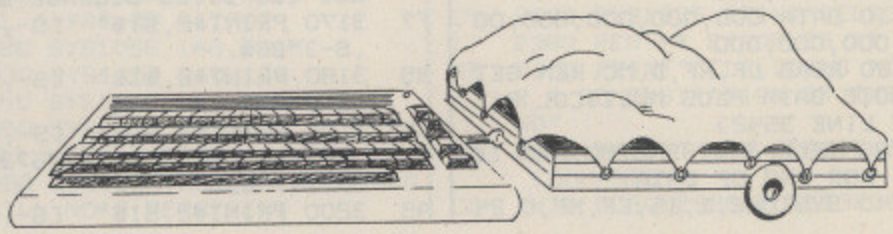
```
C5  3930 PRINT#2,"1=PLOT,2=PEEK)
    "
1D  3940 PRINT#2
03  3950 PRINT#2,"DRAW LINE - SY
    S1050,0-319,0-199,0-319,0-19
    9,0-1"
F0  3960 PRINT#2,"[CT9,SPC10](X1
    )(Y1)(X2)(Y2)(MODE)"
3B  3970 PRINT#2
0C  3980 PRINT#2,"DRAW CIRCLE -
    SYS1053,0-319,0-199,1-129,0-
    1"
E8  3990 PRINT#2,"[CT11,SPC10](C
    X-COORD)(CY-COORD)(RADIUS)(M
    ODE)"
69  4000 PRINT#2
4F  4010 PRINT#2,"FILL BITMAP -
    SYS1056,0-319,0-199,0-1"
61  4020 PRINT#2,"[CT11,SPC10](X
    -COORD)(Y-COORD)(MODE)"
87  4030 PRINT#2
EB  4040 PRINT#2,"FILL WITH CHAR
    ACTER - SYS1059,0-319,0-199,
    0-1,0-255"
```



```
51  4050 PRINT#2,"[CT19,SPC10](X
    -COORD)(Y-COORD)(MODE)(CHAR.
    NO)"
A5  4060 PRINT#2
87  4070 PRINT#2,"SET SOUND - SY
    S1062,1-3,0-15,0-255,0-255,0
    -255,0-255,0-255,";
A7  4080 PRINT#2,"0,15,0-255,0-2
    55"
```



```
3C  4090 PRINT#2,"[CT9,SPC9](VOI
    CE)(VOLUME)(LOW FREQ)(HIGH F
    REQ)(A/D)(S/R)"
E1  4100 PRINT#2,"[SPC18](LOW PU
    LSE)(HIGH PULSE)(WAVEFORM)(D
    URATION)"
B8  4110 PRINT#2
99  4120 PRINT#2,"SET SPRITE - S
    YS1065,0-7,0-1,0-255,0-15,0-
    320,0-255,0-1,0-1";
B0  4130 PRINT#2,",0-1,0-15,0-15
    "
64  4140 PRINT#2,"[CT10,SPC10](S
    PRITE NO)(OFF/ON)(SPRITE POI
    NTER)(COLOUR)";
23  4150 PRINT#2,"(X POS)(Y POS)
    "
83  4160 PRINT#2,"[SPC20](X EXP)
    (Y EXP)(MULTI.COL OFF/ON)(MC
     1)(MC 2)"
F4  4170 PRINT#2
E9  4180 PRINT#2,"MOVE MEMORY -
    SYS1068,0-65535,0-65535,0-32
    767"
51  4190 PRINT#2,"[CT11,SPC10](A
    DDRESS MOVE TO)(ADDRESS MOVE
     FROM)";
2F  4200 PRINT#2,"(LENGTH)"
4C  4210 PRINT#2
61  4220 PRINT#2,"FILL MEMORY -
    SYS1071,0-65535,0-32767,0-25
    5"
E7  4230 PRINT#2,"[CT11,SPC10](S
    TART ADDRESS)(LENGTH)(NUMBER
    )"
6A  4240 PRINT#2
28  4250 PRINT#2,"SAVE MEMORY -
    SYS1074,'[SPC4]',08,00,0-655
    35,0-65535"
94  4260 PRINT#2,"[CT11,SPC10](F
    ILE NAME)(DEVICE)(00)(MEMORY
     START)";
29  4270 PRINT#2,"(MEMORY FINISH
    )"
82  4280 PRINT#2
1D  4290 PRINT#2,"LOAD MEMORY -
    SYS1077,'[SPC4]',08,00,0-655
    35"
C8  4300 PRINT#2,"[CT11,SPC10](F
    ILE NAME)(DEVICE)(00)(LOAD A
    DDRESS)"
B0  4310 PRINT#2
3D  4320 PRINT#2,"SET CURSOR POS
     - SYS1080,0-24,0-39"
EE  4330 PRINT#2,"[CT14,SPC10](R
    OW)(COLUMN)"
90  4340 PRINT#2:CLOSE2:PRINT"[C
    LR]":END
```



# DiskOS

*Accessing the disk drive is child's play through the square window*

DiskOS is an operating system which employs windows to ease communications with a disk drive. Its ingenious routines interrupt without interrupting! After calling up DiskOS, programs can resume as though nothing had happened.

Whenever you need to use one of the functions of DiskOS just press the CBM key with the CTRL key and a menu will appear at the top of the screen. This may be done when running Basic or machine-code programs and should be compatible with most of them because it does not use the IRQ interrupts.

When selecting options use the first capital letter of the menu name or option. After a command is complete, press the spacebar to get back to the opening menu.

The QUIT option on all the menus will return you to the start-up menu at the top of the screen.

## Menu 1 – Info

Press 'I' for Info and the menu will appear. Just one item is contained on this one – a short note about the program.

## Menu 2 – Disk

This allows access to the disk commands.

DIR displays the directory which, unlike LOAD"$",8, does not overwrite a Basic program.

ERROR reads the disk status (if the red LED flashes).

INIT reads in the disk information after a disk change.

VAL is the validate command which cleans up the disk, and should always be used after SCRATCHing files.

FORMAT is the same as the usual Basic command OPEN 1,8,15,"NEW:TEST DISK,64": CLOSE 1.

RENAME will change a file name to another name, just type NEW NAME=OLD NAME and press RETURN.

COPY will copy a file from one disk to another and will ask for the "source" disk which holds the original and the 'destination' disk onto which the copy is made.

## Menu 3 – Misc

KILL. This returns the C64 to normal and disables the use of CTRL+CBM. SYS 49152 will restore it.

EXIT will let the computer carry on from where you interrupted it by pressing CTRL+CBM.

## Menu 4 – Screen

COLOURS allows you to change the screen colours and the new setting will be maintained until changed again or SYS 49152 is called to restore the default values.

DUMP#4 will dump the text screen at $0400 to the printer. If you design a screen using the cursor keys in Basic and go to the screen menu you can then print it out with this facility.

PROGRAM: DISKOS.BAS

```
DE   10 REM DISKOS
9D   20 REM TYPE IN THIS PROGRAM
     AND SAVE IT ON A SPARE TAPE
     OR CASSETTE
DD   30 REM WHEN YOU RUN IT MAKE
     SURE YOU HAVE CUED UP THE TA
     PE/DISK FOR SAVING DISKOS
30   40 REM WHEN USING DISKOS, LO
     AD USING ,8,1 AND TYPE NEW
6D   50 REM TYPE SYS 49152 TO GET
      THE DISKOS MESSAGE
4B   60 BL=252  :LN=50   :SA=4915
     2
A2   70 FOR L=0 TO BL:CX=0:FOR D=
     0 TO 15:READ A:CX=CX+A
65   80 POKE53280,A:POKE SA+L*16+
     D,A:NEXT D
51   90 READ A:IF A><CX THENPRINT
     "ERROR IN LINE";LN+(L*10):ST
     OP
4A   100 NEXT L
DF   110 DATA 169,167,160,192,32,
     30,171,169,14,141,32,208,169
     ,6,141,33,1834
74   120 DATA 208,120,169,30,141,
     143,2,169,192,141,144,2,88,9
     6,173,141,1959
74   130 DATA 2,201,6,240,3,76,72
     ,235,120,169,72,141,143,2,16
     9,235,1886
85   140 DATA 141,144,2,88,173,24
     ,208,141,63,3,169,23,141,24,
     208,169,1721
F1   150 DATA 8,32,210,255,56,32,
     240,255,140,60,3,142,61,3,16
     9,0,1666
45   160 DATA 133,255,32,199,192,
     32,115,192,169,1,133,255,32,
     199,192,24,2155
D7   170 DATA 172,60,3,174,61,3,3
     2,240,255,32,17,192,173,63,3
     ,141,1621
98   180 DATA 24,208,96,169,1,133
     ,204,169,5,160,193,32,30,171
     ,32,228,1855
75   190 DATA 255,240,251,201,73,
     208,3,76,107,193,201,68,208,
     3,76,72,2235
52   200 DATA 194,201,77,208,3,76
     ,89,195,201,83,208,3,76,189,
     204,76,2083
EB   210 DATA 126,192,169,0,133,1
     98,96,154,147,17,32,68,73,83
     ,75,79,1642
94   220 DATA 83,32,86,49,46,50,3
     2,40,85,83,69,32,67,84,82,76
     ,996
98   230 DATA 43,67,66,77,41,13,0
     ,169,0,133,251,133,253,169,4
     ,133,1552
C7   240 DATA 252,169,160,133,254
     ,165,255,208,20,160,0,177,25
     1,145,253,200,2802
```

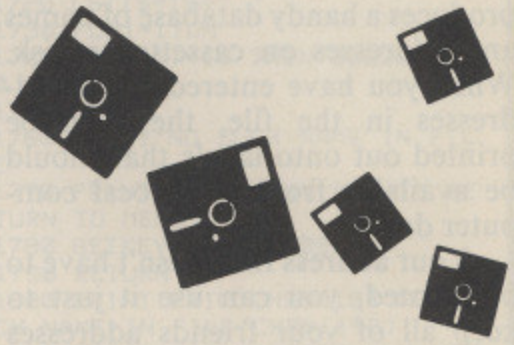
by Gary Mayhew

```
A2  250 DATA 208,249,230,252,230
    ,254,165,252,201,8,208,239,9
    6,160,0,198,2950
11  260 DATA 1,177,253,145,251,2
    00,208,249,230,254,230,252,1
    65,252,201,8,3076
48  270 DATA 208,239,230,1,96,14
    7,147,18,176,192,192,192,192
    ,192,192,178,2592
59  280 DATA 192,192,192,192,192
    ,192,178,192,192,192,192,192
    ,192,178,192,192,3044
04  290 DATA 192,192,192,192,192
    ,192,174,13,18,221,32,201,78
    ,70,79,32,2070
98  300 DATA 221,32,196,73,83,75
    ,32,221,32,205,73,83,67,32,2
    21,32,1678
EE  310 DATA 211,67,82,69,69,78,
    32,221,13,18,173,192,192,192
    ,192,192,1993
45  320 DATA 192,177,192,192,192
    ,192,192,192,177,192,192,192
    ,192,192,192,177,3027
2D  330 DATA 192,192,192,192,192
    ,192,192,192,189,13,0,169,11
    7,160,193,32,2409
81  340 DATA 30,171,76,173,193,1
    9,17,17,18,171,192,192,192,1
    92,192,192,2037
4B  350 DATA 219,13,18,221,193,6
    6,79,85,84,32,221,13,18,173,
    192,192,1819
42  360 DATA 192,192,192,192,189
    ,13,0,32,228,255,240,251,201
    ,89,240,12,2518
92  370 DATA 201,78,208,243,104,
    104,76,115,192,76,151,193,96
    ,32,228,255,2352
E1  380 DATA 240,251,201,65,208,
    247,76,187,193,208,220,169,2
    00,160,193,32,2850
3F  390 DATA 30,171,32,176,197,7
    6,115,192,19,17,17,17,17,17,
    17,17,1127
BC  400 DATA 17,17,32,32,18,176,
    192,192,192,192,192,192,192,
    192,192,192,2212
7E  410 DATA 192,192,192,192,192
    ,192,192,192,192,192,192,192
    ,192,192,192,192,3072
A0  420 DATA 192,192,192,192,192
    ,192,192,174,13,32,32,18,221
    ,32,196,73,2135
7E  430 DATA 83,75,207,211,32,86
    ,49,46,50,32,66,89,32,199,46
    ,205,1508
86  440 DATA 65,89,72,69,87,32,2
    02,65,78,32,39,56,56,32,221,
    13,1208
33  450 DATA 32,32,18,173,192,19
    2,192,192,192,192,192,192,19
    2,192,192,192,2559
6C  460 DATA 192,192,192,192,192
    ,192,192,192,192,192,192,192
    ,192,192,192,192,3072
D4  470 DATA 192,192,192,192,192
    ,189,13,0,169,83,160,194,32,
    30,171,76,2077
21  480 DATA 18,195,96,19,17,17,
    29,29,29,29,29,29,29,18,219,
    192,994
11  490 DATA 192,192,192,192,192
    ,219,13,29,29,29,29,29,29,29
    ,18,221,1634
6D  500 DATA 196,73,82,46,32,32,
    221,13,29,29,29,29,29,29,29,
    18,916
89  510 DATA 221,197,82,82,79,82
    ,32,221,13,29,29,29,29,29,29
    ,29,1212
9C  520 DATA 18,221,201,78,73,84
    ,46,32,221,13,29,29,29,29,29
    ,29,1161
D8  530 DATA 29,18,221,214,65,76
    ,46,32,32,221,13,29,29,29,29
    ,29,1112
07  540 DATA 29,29,18,221,211,67
    ,82,46,32,32,221,13,29,29,29
    ,29,1117
23  550 DATA 29,29,29,18,221,198
    ,79,82,77,65,84,221,13,29,29
    ,29,1232
A8  560 DATA 29,29,29,29,18,221,
    210,69,78,65,77,69,221,13,29
    ,29,1215
5D  570 DATA 29,29,29,29,29,18,2
    21,195,79,80,89,32,32,221,13
    ,29,1154
36  580 DATA 29,29,29,29,29,29,1
    8,221,209,85,73,84,32,32,221
    ,13,1162
74  590 DATA 29,29,29,29,29,29,2
    9,18,173,192,192,192,192,192
    ,192,189,1735
7D  600 DATA 13,0,32,228,255,240
    ,251,201,81,208,3,76,115,192
    ,201,68,2164
C1  610 DATA 208,3,76,17,196,201
    ,69,208,3,76,212,196,201,73,
    208,3,1950
DA  620 DATA 76,8,198,201,86,208
    ,3,76,32,198,201,83,208,3,76
    ,16,1673
68  630 DATA 200,201,70,208,3,76
    ,132,200,201,82,208,3,76,220
    ,200,201,2281
F5  640 DATA 67,208,3,76,48,201,
    76,18,195,169,100,160,195,32
    ,30,171,1749
E9  650 DATA 76,224,195,96,19,17
    ,17,29,29,29,29,29,29,29,29,
    29,905
0A  660 DATA 29,29,29,29,29,18,2
    19,192,192,192,192,192,192,2
    19,13,29,1795
1D  670 DATA 29,29,29,29,29,29,2
    9,29,29,29,29,29,29,18,221,2
    03,819
EA  680 DATA 73,76,76,32,32,221,
    13,29,29,29,29,29,29,29,29,2
    9,784
DF  690 DATA 29,29,29,29,29,18,2
    21,197,88,73,84,32,32,221,13
    ,29,1153
14  700 DATA 29,29,29,29,29,29,2
    9,29,29,29,29,29,29,18,221,2
    09,825
91  710 DATA 85,73,84,32,32,221,
    13,29,29,29,29,29,29,29,29,2
    9,801
5C  720 DATA 29,29,29,29,29,18,1
    73,192,192,192,192,192,192,1
    89,13,0,1690
7A  730 DATA 32,228,255,240,251,
    201,81,208,3,76,115,192,201,
    69,208,6,2366
5C  740 DATA 169,9,32,210,255,96
    ,201,75,208,230,169,14,141,3
    2,208,169,2218
EB  750 DATA 6,141,33,208,169,21
    ,141,24,208,169,9,32,210,255
    ,76,148,1850
2E  760 DATA 227,169,148,160,196
    ,32,30,171,169,36,133,251,16
    9,251,133,187,2462
E8  770 DATA 169,0,133,188,169,1
    ,133,183,169,8,133,186,169,9
    6,133,185,2055
54  780 DATA 32,213,243,165,186,
    32,180,255,165,185,32,150,25
    5,169,0,133,2395
A8  790 DATA 144,160,3,132,251,3
    2,165,255,133,252,173,141,2,
    201,1,240,2285
DD  800 DATA 249,164,144,208,47,
    32,165,255,164,144,208,40,16
    4,251,136,208,2579
D6  810 DATA 226,166,252,32,205,
    189,169,32,32,210,255,32,165
    ,255,166,144,2530
3F  820 DATA 208,18,170,240,6,32
    ,210,255,76,107,196,169,13,3
    2,210,255,2197
A5  830 DATA 160,2,208,191,32,66
    ,246,169,180,160,196,32,30,1
    71,32,176,2051
B7  840 DATA 197,76,115,192,147,
    200,79,76,68,32,211,200,201,
    198,212,32,2236
16  850 DATA 84,79,32,208,65,85,
    83,69,32,204,73,83,84,73,78,
    71,1403
B1  860 DATA 46,17,13,0,17,208,8
    2,69,83,83,32,39,211,80,65,6
    7,1112
B6  870 DATA 69,39,32,70,79,82,3
    2,196,73,83,75,207,211,32,20
    5,69,1554
F3  880 DATA 78,85,46,0,24,160,0
    ,162,21,32,240,255,169,23,16
    0,197,1652
E6  890 DATA 32,30,171,169,15,16
    2,8,160,15,32,186,255,169,0,
    32,189,1625
19  900 DATA 255,32,192,255,162,
    15,32,198,255,32,207,255,201
    ,13,240,5,2349
5A  910 DATA 32,210,255,208,244,
    169,15,32,195,255,32,204,255
    ,32,176,197,2511
4F  920 DATA 76,115,192,32,176,1
    97,96,32,18,176,192,192,192,
    192,192,192,2262
44  930 DATA 192,192,192,192,192
    ,192,192,192,192,192,192,192
    ,192,192,192,192,3072
D0  940 DATA 192,192,192,192,192
    ,192,192,192,192,192,192,192
    ,192,192,174,13,2875
87  950 DATA 32,18,221,211,84,65
    ,84,85,83,58,32,32,32,32,32,
    32,1133
64  960 DATA 32,32,32,32,32,32,32
    ,32,32,32,32,32,32,32,32,32
    ,512
CD  970 DATA 32,32,32,32,32,32,32
    ,32,221,13,32,18,173,192,192,1
    92,192,1449
```

```
B2  980 DATA 192,192,192,192,192
    ,192,192,192,192,192,192,192
    ,192,192,192,192,3072
8B  990 DATA 192,192,192,192,192
    ,192,192,192,192,192,192,192
    ,192,192,192,192,3072
6E  1000 DATA 189,157,157,157,15
    7,157,157,157,157,157,157,15
    7,157,157,157,157,2544
70  1010 DATA 157,157,157,157,15
    7,157,157,157,157,157,157,15
    7,157,157,145,0,2343
2D  1020 DATA 32,228,255,201,32,
    208,249,96,169,8,32,177,255,
    32,183,255,2412
F2  1030 DATA 41,128,208,5,169,1
    11,32,147,255,169,0,162,0,18
    9,222,197,2035
39  1040 DATA 32,168,255,232,201
    ,13,208,245,32,174,255,76,21
    2,196,0,58,2357
4D  1050 DATA 0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0
BB  1060 DATA 0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0
D2  1070 DATA 0,0,0,0,0,0,0,0,0,32
    ,19,198,169,73,141,222,197,1
    051
7B  1080 DATA 76,184,197,162,0,1
    69,13,157,224,197,232,224,39
    ,208,246,96,2424
8F  1090 DATA 32,19,198,169,86,1
    41,222,197,76,184,197,162,0,
    169,100,157,2109
09  1100 DATA 251,6,232,236,65,3
    ,208,245,162,0,142,64,3,32,1
    57,198,2004
EF  1110 DATA 32,228,255,240,248
    ,201,20,208,6,32,133,198,76,
    61,198,201,2337
22  1120 DATA 13,208,9,169,100,1
    74,64,3,157,251,6,96,201,91,
    48,3,1593
0B  1130 DATA 76,61,198,201,32,1
    6,3,76,61,198,201,42,208,3,7
    6,61,1513
45  1140 DATA 198,174,64,3,41,63
    ,157,251,6,238,64,3,174,64,3
    ,236,1739
87  1150 DATA 65,3,208,185,96,17
    4,64,3,224,0,208,1,96,202,16
    9,100,1798
61  1160 DATA 206,64,3,174,64,3,
    157,251,6,157,252,6,96,174,6
    4,3,1680
B5  1170 DATA 169,160,157,251,6,
    96,32,18,176,192,192,192,192
    ,192,192,192,2409
09  1180 DATA 192,192,192,192,19
    2,192,192,192,192,192,192,19
    2,192,192,192,192,3072
6D  1190 DATA 192,192,192,192,19
    2,192,192,192,192,192,192,19
    2,192,174,13,32,2715
1A  1200 DATA 18,221,32,32,32,32
    ,32,32,32,32,32,32,32,32,32,
    32,687
AB  1210 DATA 32,32,32,32,32,32,
    32,32,32,32,32,32,32,32,32,3
    2,512
AF  1220 DATA 32,32,32,32,32,32,
    221,13,32,18,173,192,192,192
    ,192,192,1609
B7  1230 DATA 192,192,192,192,19
    2,192,192,192,192,192,192,19
    2,192,192,192,192,3072
49  1240 DATA 192,192,192,192,19
    2,192,192,192,192,192,192,19
    2,192,192,192,189,3069
FF  1250 DATA 157,157,157,157,15
    7,157,157,157,157,157,157,15
    7,157,157,157,157,2512
89  1260 DATA 157,157,157,157,15
    7,157,157,157,157,157,157,15
    7,157,157,157,157,2512
23  1270 DATA 157,157,157,157,14
    5,0,32,18,176,192,192,192,19
    2,192,192,192,2343
A4  1280 DATA 192,192,192,192,19
    2,192,192,192,192,192,192,19
    2,192,192,192,192,3072
DC  1290 DATA 192,192,192,192,19
    2,192,192,192,192,192,192,19
    2,192,174,13,32,2715
A9  1300 DATA 18,221,32,32,32,32
    ,32,32,32,32,32,32,32,32,32,
    32,687
C4  1310 DATA 32,32,32,32,32,32,
    32,32,32,32,32,32,32,32,32,3
    2,512
77  1320 DATA 32,32,32,32,32,32,
    221,13,32,18,221,32,32,32,32
    ,32,857
90  1330 DATA 32,32,32,32,32,32,
    32,32,32,32,32,32,32,32,32,3
    2,512
B6  1340 DATA 32,32,32,32,32,32,
    32,32,32,32,32,32,32,32,32,2
    21,701
FE  1350 DATA 13,32,18,173,192,1
    92,192,192,192,192,192,192,1
    92,192,192,192,2540
74  1360 DATA 192,192,192,192,19
    2,192,192,192,192,192,192,19
    2,192,192,192,192,3072
1B  1370 DATA 192,192,192,192,19
    2,192,192,192,189,157,157,15
    7,157,157,157,157,2824
FE  1380 DATA 157,157,157,157,15
    7,157,157,157,157,157,157,15
    7,157,157,157,157,2512
A2  1390 DATA 157,157,157,157,15
    7,157,157,157,157,157,157,15
    7,157,145,145,0,2331
F8  1400 DATA 169,17,141,65,3,24
    ,160,0,162,14,32,240,255,169
    ,166,160,1777
2D  1410 DATA 198,32,30,171,169,
    68,160,200,32,30,171,32,19,1
    98,32,43,1585
FA  1420 DATA 198,173,64,3,240,1
    1,169,83,141,222,197,32,103,
    200,76,184,2096
A6  1430 DATA 197,76,115,192,208
    ,76,69,65,83,69,32,197,78,84
    ,69,82,1692
78  1440 DATA 32,198,73,76,69,78
    ,65,77,69,32,212,79,32,211,6
    7,82,1452
1E  1450 DATA 65,84,67,72,46,13,
    0,162,0,189,251,6,201,100,24
    0,14,1510
74  1460 DATA 201,31,16,3,24,105
    ,64,157,224,197,232,76,105,2
    00,169,13,1817
A5  1470 DATA 157,224,197,96,169
    ,20,141,65,3,24,160,0,162,14
    ,32,240,1704
46  1480 DATA 255,169,166,160,19
    8,32,30,171,169,184,160,200,
    32,30,171,32,2159
34  1490 DATA 19,198,32,43,198,1
    73,64,3,240,11,169,78,141,22
    2,197,32,1820
D8  1500 DATA 103,200,76,184,197
    ,76,115,192,197,78,84,69,82,
    58,32,39,1782
0F  1510 DATA 40,196,73,83,75,32
    ,206,65,77,69,41,44,73,68,39
    ,32,1213
5F  1520 DATA 198,79,82,32,198,7
    9,82,77,65,84,13,0,169,35,14
    1,65,1399
5A  1530 DATA 3,24,160,0,162,14,
    32,240,255,169,166,160,198,3
    2,30,171,1816
19  1540 DATA 169,16,160,201,32,
    30,171,32,19,198,32,43,198,1
    73,64,3,1541
B2  1550 DATA 240,11,169,82,141,
    222,197,32,103,200,76,184,19
    7,76,115,192,2237
99  1560 DATA 210,69,78,65,77,69
    ,58,32,39,40,206,69,87,32,20
    6,65,1402
2A  1570 DATA 77,69,41,61,40,207
    ,76,68,206,65,77,69,41,39,13
    ,0,1149
BE  1580 DATA 32,19,202,169,17,1
    41,65,3,24,160,0,162,14,32,2
    40,255,1535
21  1590 DATA 169,166,160,198,32
    ,30,171,169,167,160,201,32,3
    0,171,32,19,1907
02  1600 DATA 198,32,43,198,173,
    64,3,240,75,32,103,200,32,12
    5,203,169,1890
E3  1610 DATA 122,160,202,32,30,
    171,32,207,202,32,215,202,76
    ,243,202,169,2297
A7  1620 DATA 240,160,201,32,30,
    171,32,52,203,169,2,32,195,2
    55,32,204,2010
6E  1630 DATA 255,169,62,160,204
    ,32,30,171,32,207,202,169,32
    ,160,204,32,2121
D9  1640 DATA 30,171,32,110,204,
    32,166,203,32,138,204,32,204
    ,255,32,176,2021
BF  1650 DATA 204,76,212,196,76,
    115,192,208,76,69,65,83,69,3
    2,197,78,1948
58  1660 DATA 84,69,82,32,198,73
    ,76,69,78,65,77,69,32,84,79,
    32,1199
AD  1670 DATA 195,79,80,89,46,13
    ,17,32,32,32,32,32,32,32,32,
    32,807
83  1680 DATA 32,32,32,32,32,32,
    32,32,32,32,32,32,32,32,32,3
    2,512
64  1690 DATA 32,32,32,32,32,32,
    32,32,32,32,32,32,32,32,13,0
    ,461
A8  1700 DATA 145,32,32,32,32,32
    ,32,32,210,69,65,68,73,78,71
    ,32,1035
35  1710 DATA 46,46,46,32,194,76
    ,79,67,75,32,206,79,46,32,48
```

```
    ,48,1152
24  1720 DATA 48,13,0,24,160,0,1
    62,14,32,240,255,169,70,160,
    199,32,1578
A2  1730 DATA 30,171,169,44,160,
    202,32,30,171,76,151,193,157
    ,215,65,82,1948
6F  1740 DATA 78,73,78,71,33,32,
    195,79,80,89,32,215,73,76,76
    ,32,1312
C4  1750 DATA 197,82,65,83,69,32
    ,193,78,89,32,208,82,79,71,8
    2,65,1507
A4  1760 DATA 77,13,29,29,18,201
    ,78,32,205,69,77,79,82,89,32
    ,46,1156
34  1770 DATA 46,46,32,195,79,78
    ,84,73,78,85,69,32,195,79,80
    ,89,1340
20  1780 DATA 32,40,89,47,78,41,
    32,63,13,0,145,145,145,29,29
    ,29,957
03  1790 DATA 18,208,76,65,67,69
    ,32,210,197,193,196,32,196,7
    3,83,75,1790
8E  1800 DATA 32,201,78,32,196,8
    2,73,86,69,32,40,210,69,84,8
    5,82,1451
60  1810 DATA 78,41,17,17,17,13,
    32,32,32,32,32,32,32,32,32,3
    2,503
C4  1820 DATA 32,32,32,32,32,32,
    32,32,32,32,32,32,32,32,32,3
    2,512
ED  1830 DATA 32,32,32,32,32,32,
    32,32,32,32,32,32,32,32,13,0,32
    ,461
1F  1840 DATA 228,255,201,13,208
    ,249,96,169,2,162,8,160,2,32
    ,186,255,2226
D0  1850 DATA 173,65,3,162,146,1
    60,203,32,189,255,32,192,255
    ,162,2,32,2063
53  1860 DATA 198,255,96,169,15,
    162,8,160,15,32,186,255,169,
    0,32,189,1941
67  1870 DATA 255,32,192,255,162
    ,15,32,198,255,32,207,255,20
    1,48,208,23,2370
14  1880 DATA 32,207,255,201,48,
    208,16,169,15,32,195,255,169
    ,2,32,195,2031
74  1890 DATA 255,32,215,202,76,
    111,201,169,15,32,195,255,16
    9,2,32,195,2156
08  1900 DATA 255,76,212,196,169
    ,1,133,252,169,8,133,253,160
    ,0,32,207,2256
8C  1910 DATA 255,145,252,230,25
    2,208,16,162,2,32,105,203,23
    0,253,165,253,2763
74  1920 DATA 197,56,208,3,76,21
    9,203,36,144,80,227,165,252,
    133,254,165,2418
84  1930 DATA 253,133,255,162,2,
    32,105,203,96,254,21,7,189,2
    1,7,201,1941
DA  1940 DATA 58,208,9,169,48,15
    7,21,7,202,76,105,203,96,162
    ,0,189,1710
D7  1950 DATA 224,197,157,146,20
    3,201,13,240,5,232,224,16,20
    8,241,142,65,2514
84  1960 DATA 3,96,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,99
B4  1970 DATA 0,0,0,0,0,0,169,1,
    133,252,169,8,133,253,169,54
    ,1341
3B  1980 DATA 133,1,160,0,56,165
    ,252,229,254,165,253,229,255
    ,176,18,177,2523
00  1990 DATA 252,32,210,255,230
    ,252,208,236,162,2,32,156,20
    4,230,253,208,2922
DC  2000 DATA 227,169,55,133,1,1
    62,2,32,156,204,96,24,160,0,
    162,21,1604
FF  2010 DATA 32,240,255,169,166
    ,160,198,32,30,171,169,255,1
    60,203,32,30,2302
BF  2020 DATA 171,32,138,204,32,
    204,255,32,176,197,104,104,7
    6,115,192,18,2050
01  2030 DATA 57,57,44,198,73,76
    ,69,32,212,79,79,32,204,79,7
    8,71,1440
FB  2040 DATA 32,212,79,32,195,7
    9,80,89,44,49,56,44,48,49,13
    ,0,1101
F0  2050 DATA 32,32,32,32,32,32,
    32,215,82,73,84,73,78,71,32,
    46,978
53  2060 DATA 46,46,32,194,76,79
    ,67,75,32,206,79,46,13,0,145
    ,145,1281
B8  2070 DATA 145,145,145,29,29,
    29,18,208,76,65,67,69,32,215
    ,210,201,1683
0A  2080 DATA 212,197,32,196,73,
    83,75,32,201,78,32,196,82,73
    ,86,69,1717
FB  2090 DATA 32,40,210,69,84,85
    ,82,78,41,17,17,17,13,0,169,
    1,955
66  2100 DATA 162,8,160,1,32,186
    ,255,173,65,3,162,146,160,20
    3,32,189,1937
83  2110 DATA 255,32,192,255,162
    ,1,32,201,255,96,169,2,32,19
    5,255,169,2303
AD  2120 DATA 1,32,195,255,169,1
    5,32,195,255,76,204,255,222,
    21,7,189,2123
D9  2130 DATA 21,7,201,47,208,9,
    169,57,157,21,7,202,76,156,2
    04,96,1638
7C  2140 DATA 162,0,169,0,157,0,
    8,232,224,40,208,246,96,169,
    199,160,2070
F7  2150 DATA 204,32,30,171,76,1
    12,205,19,17,17,29,29,29,29,
    29,29,1057
B1  2160 DATA 29,29,29,29,29,29,
    29,29,29,29,29,29,29,29,29,1
    8,453
D4  2170 DATA 219,192,192,192,19
    2,192,192,192,192,179,13,29,
    29,29,29,29,2092
C5  2180 DATA 29,29,29,29,29,29,
    29,29,29,29,29,29,29,29,29,2
    9,464
A5  2190 DATA 18,221,195,79,76,7
    9,85,82,83,32,221,13,29,29,2
    9,29,1300
F9  2200 DATA 29,29,29,29,29,29,
    29,29,29,29,29,29,29,29,29,2
    9,464
B3  2210 DATA 29,18,221,196,85,7
    7,80,32,35,52,32,221,13,29,2
    9,29,1178
8D  2220 DATA 29,29,29,29,29,29,
    29,29,29,29,29,29,29,29,29,2
    9,464
86  2230 DATA 29,29,18,221,209,8
    5,73,84,32,32,32,32,221,13,2
    9,29,1168
81  2240 DATA 29,29,29,29,29,29,
    29,29,29,29,29,29,29,29,29,2
    9,464
9B  2250 DATA 29,29,29,18,173,19
    2,192,192,192,192,192,192,19
    2,189,13,0,2016
92  2260 DATA 32,228,255,240,251
    ,201,81,208,3,76,115,192,201
    ,68,208,3,2362
83  2270 DATA 76,138,205,201,67,
    208,233,76,242,205,169,1,133
    ,255,32,199,2440
5F  2280 DATA 192,169,4,133,186,
    169,126,133,184,169,0,162,4,
    133,113,134,2011
F9  2290 DATA 114,133,183,133,18
    5,32,192,255,166,184,32,201,
    255,162,25,169,2421
EE  2300 DATA 13,32,210,255,32,2
    25,255,240,46,160,0,177,113,
    133,103,41,2035
9B  2310 DATA 63,6,103,36,103,16
    ,2,9,128,112,2,9,64,32,210,2
    55,1150
62  2320 DATA 200,192,40,208,230
    ,152,24,101,113,133,113,144,
    2,230,114,202,2198
FE  2330 DATA 208,205,169,13,32,
    210,255,32,204,255,162,126,3
    2,195,255,76,2429
A4  2340 DATA 115,192,169,3,160,
    206,32,30,171,169,109,160,20
    6,32,30,171,1955
BD  2350 DATA 76,14,207,19,17,17
    ,17,17,17,17,17,13,32,32,32,
    32,576
A3  2360 DATA 32,32,48,32,46,46,
    46,32,194,76,65,67,75,32,32,
    32,887
23  2370 DATA 56,32,46,46,32,207
    ,82,65,78,71,69,13,32,32,32,
    32,925
35  2380 DATA 32,32,49,32,46,46,
    46,32,215,72,73,84,69,32,32,
    32,924
CF  2390 DATA 57,32,46,46,46,32,
    194,82,79,87,78,13,32,32,32,
    32,920
76  2400 DATA 32,32,50,32,46,46,
    46,46,46,32,210,69,68,32,32,
    32,851
D5  2410 DATA 65,32,46,46,46,32,
    210,69,68,32,50,13,0,32,32,3
    2,805
```

```
CF  2420 DATA 32,32,32,51,32,46,
    46,46,46,32,195,89,65,78,32,
    32,886
18  2430 DATA 32,66,32,46,46,32,
    199,82,69,89,32,49,13,32,32,
    32,883
B0  2440 DATA 32,32,32,52,32,46,
    46,32,208,85,82,80,76,69,32,
    32,968
9B  2450 DATA 32,67,32,46,46,32,
    199,82,69,89,32,50,13,32,32,
    32,885
21  2460 DATA 32,32,32,53,32,46,
    46,46,32,199,82,69,69,78,32,
    32,912
F3  2470 DATA 32,68,32,46,32,199
    ,82,69,69,78,32,50,13,32,32,
    32,898
E5  2480 DATA 32,32,32,54,32,46,
    46,46,46,32,194,76,85,69,32,
    32,886
77  2490 DATA 32,69,32,46,46,32,
    194,76,85,69,32,50,13,32,32,
    32,872
E0  2500 DATA 32,32,32,55,32,46,
    46,32,217,69,76,76,79,87,32,
    32,975
61  2510 DATA 32,70,32,46,46,32,
    199,82,69,89,32,51,13,0,169,
    56,1018
E5  2520 DATA 160,207,32,30,171,
    32,170,207,141,32,208,169,94
    ,160,207,32,2052
1D  2530 DATA 30,171,32,170,207,
    141,33,208,169,132,160,207,3
    2,30,171,32,1925
1F  2540 DATA 170,207,141,134,2,
    76,115,192,13,17,32,32,212,8
    9,80,69,1581
45  2550 DATA 32,195,79,68,69,32
    ,198,79,82,32,194,79,82,68,6
    9,82,1440
A5  2560 DATA 32,195,79,76,79,85
    ,82,46,46,46,46,46,32,0,13,1
    7,920
53  2570 DATA 32,32,212,89,80,69
    ,32,195,79,68,69,32,198,79,8
    2,32,1380
2B  2580 DATA 194,65,67,75,71,78
    ,68,32,195,79,76,79,85,82,46
    ,46,1338
34  2590 DATA 46,46,32,0,13,17,3
    2,32,212,89,80,69,32,195,79,
    68,1042
C5  2600 DATA 69,32,198,79,82,32
    ,212,69,88,84,32,195,79,76,7
    9,85,1491
1C  2610 DATA 82,46,46,46,46,46,
    46,46,32,0,32,228,255,240,25
    1,201,1643
97  2620 DATA 71,16,247,201,48,4
    8,243,201,65,16,4,32,210,255
    ,96,32,1785
1E  2630 DATA 210,255,56,233,55,
    96,0,0,0,0,0,0,0,0,0,0,905
79  2640 POKE193,0:POKE194,192:P
    OKE174,198:POKE175,207
26  2650 POKE187,7:POKE188,8:POK
    E183,6:POKE186,8:REM SUBSTIT
    UTE POKE 186,1 FOR TAPE
9B  2660 POKE185,0
86  2670 SYS62954
```



# DON'T MISS OUT

Fill in your name and address and give this form to your newsagent.

Please order me a copy of YOUR COMMODORE and reserve/deliver me a copy every month.

NAME .....

ADDRESS .....

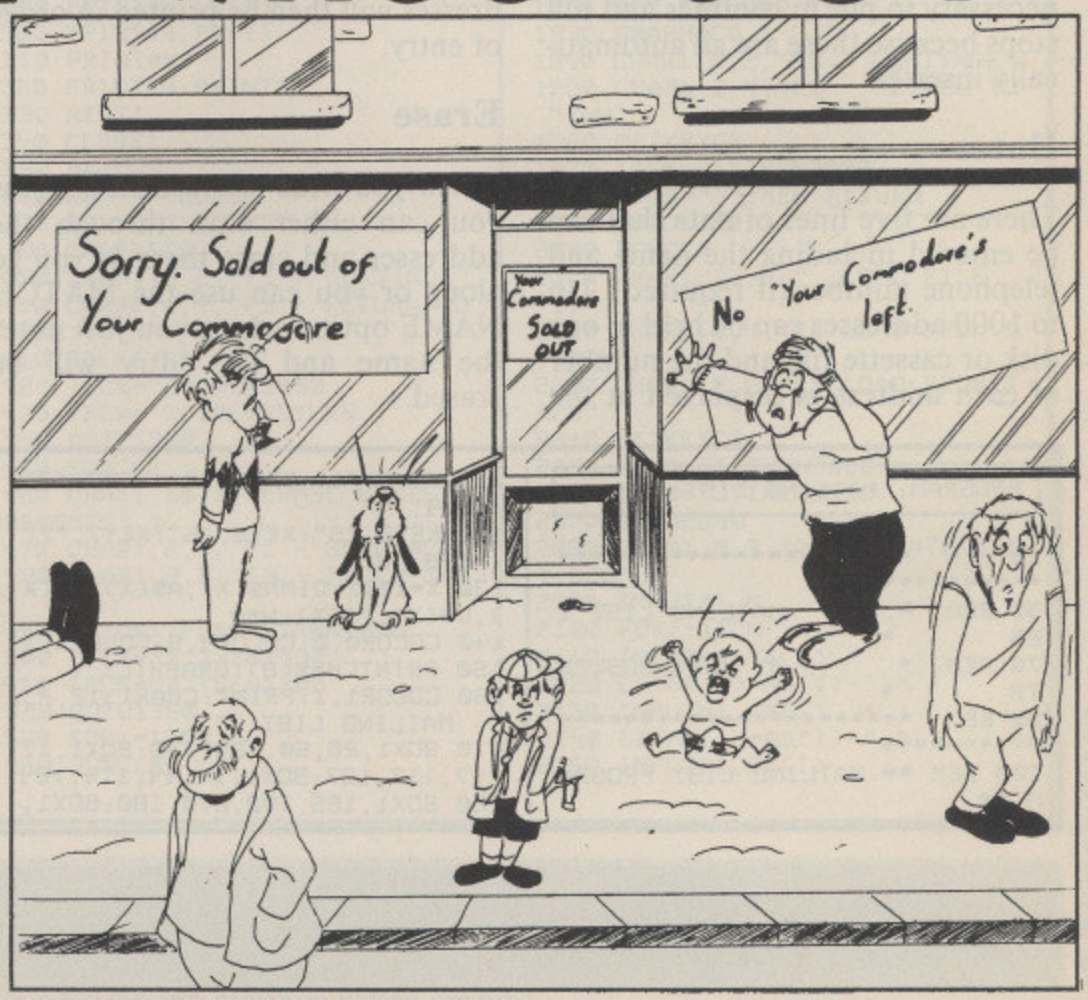
.....

.....

.....

*Newsagent:* This magazine is made available to your wholesaler through:
S.M. Distribution Ltd
6 Leigham Court Road
Streatham
LONDON
SW16 2PG

Tel: 01-677 8111



# Mailing List 128

*Address your Christmas cards in plenty of time with our useful address database*

Mailing List 128 uses the 40 column screen in 128 mode and produces a handy database of names and addresses on cassette or disk. When you have entered all the addresses in the file, they can be printed out onto labels that should be available from your local computer dealer.

Your address file doesn't have to be printed, you can use it just to keep all of your friends addresses together in one place. When you enter the addresses you will be asked for a telephone number which can be entered if you're using them for reference but if you wish to have your addresses printed out, you can either enter part of the address in this space or leave it blank. When you're entering addresses it isn't necessary to put in commas and full stops because these are all automatically inserted.

## Enter

There are five lines of data that can be entered including the name and telephone number (if required). Up to 1000 addresses can be held in one disk or cassette file and the number of each address is displayed at the top of the screen along with the maximum number of addresses.

## Read

All of the addresses can be viewed on the screen in order of entry or, if your looking for a particular person's address, you can just enter a name and the relevant address will be shown.

As you scan through the file, the current address number and total number of addresses in memory are shown at the top of the screen.

## Print

When you are ready to print your addresses, position the printer head about 5mm from the top of the first label and press F1. All of the addresses will then be printed in order of entry.

## Erase

When you wish to erase an address you can either scan through the addresses and erase them as you go along or you can use the MATCH NAME option where you just enter the name and the entry will be erased.

## Load and save

All your addresses are saved into a sequential file named 128 MAIL LIST. If you are using a disk drive, the disk status is shown in the form of 00,ERROR MESSAGE,00. All of the error types are explained in the Technical Information section of this Guide.

## Change

If an address or telephone number changes, the file can be updated by scanning through the addresses and changing them as they appear on the screen by using the MATCH option.

You can easily change any line of the address by entering the line number and typing in the new information. As you change the address its new form is shown at the top of the screen.

## Exit

When this option is selected you are asked whether you have saved your new addresses. If not, type N and you will be taken back to the main menu.

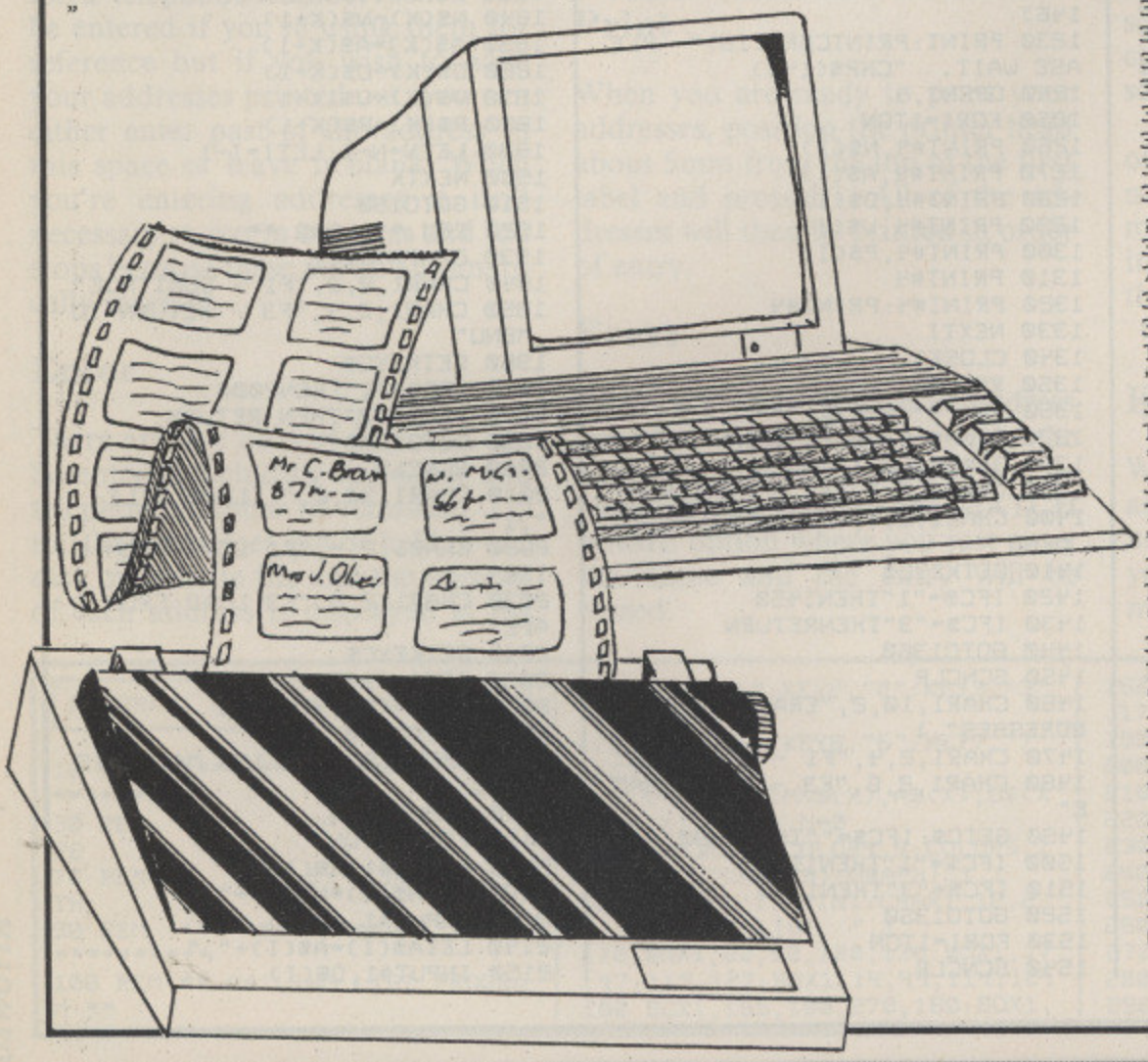
PROGRAM: MAILING LIST 128

```
10 REM  ************************
**********
30 REM  *        MAILING LIST 1
28       *
70 REM  *        BY JOHN BOSWOR
TH       *
90 REM  ************************
**********
100 REM ** MAILING LIST PROGRA
M **
110 KEY1,"1":KEY2,"2":KEY3,"3"
:KEY4,"4"
120 KEY5,"5":KEY6,"6":KEY7,"7"
:KEY8,"8"
130 X=1001:DIMN$(X),A$(X),D$(X
),U$(X),P$(X):N=0
140 COLOR0,2:COLOR4,5:COLOR5,1
150 PRINTCHR$(8):GRAPHIC1,1
160 COLOR1,1:PRINT:CHAR1,12,2,
"  MAILING LIST  ",1
170 BOX1,20,50,120,130:BOX1,17
,47,117,127:BOX1,14,44,114,124
180 BOX1,165,100,270,180:BOX1,
162,97,267,177:BOX1,159,94,264
,174:BOX1,5,5,314,195
190 CHAR1,16,4,"  128   ",1
200 CHAR1,3,8,"F1 - ENTER"
210 CHAR1,3,10,"F2 - READ"
220 CHAR1,3,12,"F3 - LOAD"
230 CHAR1,3,14,"F4 - SAVE"
240 CHAR1,21,14,"F5 - PRINT"
250 CHAR1,21,16,"F6 - CHANGE"
260 CHAR1,21,18,"F7 - ERASE"
270 CHAR1,21,20,"F8 - EXIT"
280 CHAR1,25,6,"BY"
290 CHAR1,20,8,"JOHN BOSWORTH"
```

*by J. Bosworth*

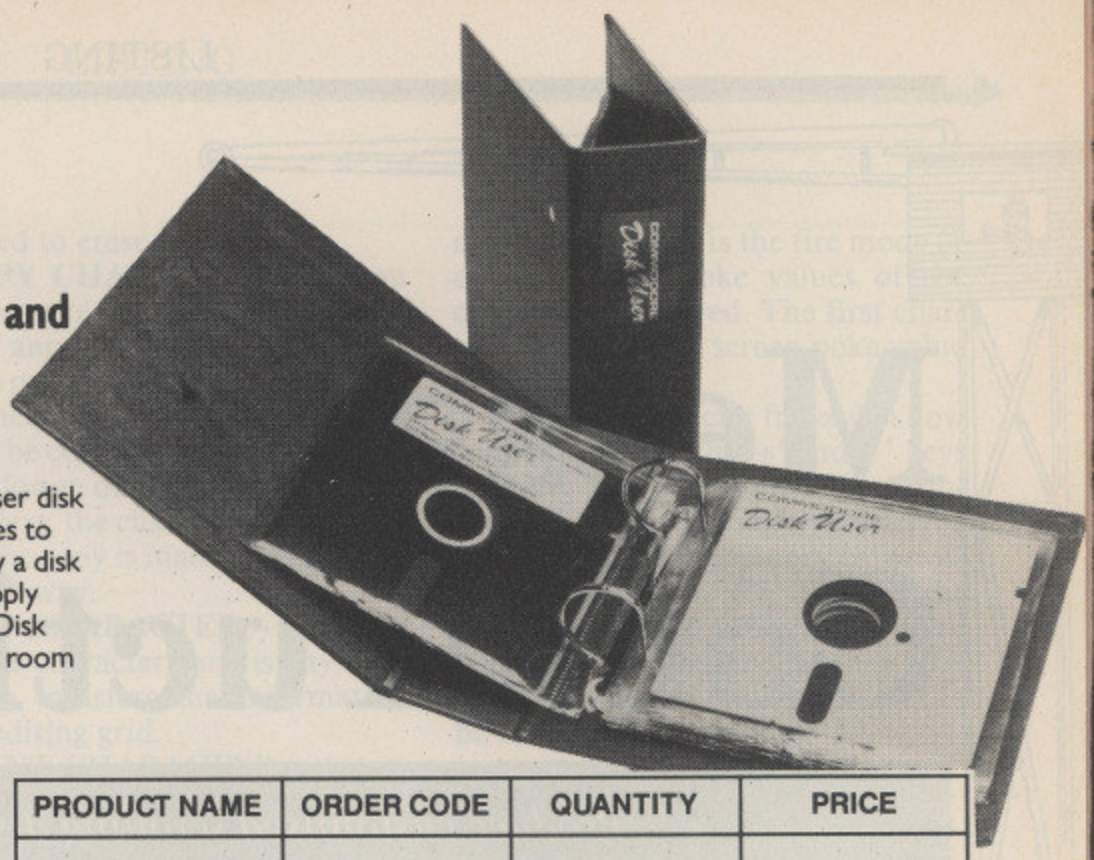
```
300 GETKEYC$
310 IFC$="1"THENGOSUB400
320 IFC$="2"THENGOSUB720
330 IFC$="3"THENGOSUB1920
340 IFC$="4"THENGOSUB2260
350 IFC$="5"THENGOSUB1140
360 IFC$="6"THENGOSUB2600
370 IFC$="7"THENGOSUB1360
380 IFC$="8"THENGOSUB3190
390 GOTO150
400 REM ** ENTER **
410 GRAPHIC0,1
420 IFN>=X-1THEN RETURN
430 CHAR1,2,2,"F1 - CONTINUE"
440 CHAR1,2,4,"F3 - RETURN TO
MENU"
450 GETKEYC$
460 IFC$="1"THEN490
470 IFC$="3"THENGOTO390
480 GOTO450
490 N=N+1
500 SCNCLR
510 CHAR1,10,2,"ENTER NAMES/AD
DRESSES",1
520 PRINT:PRINT"ENTRY #";N;"OF
";X-1
530 INPUT"NAME";N$(N)
540 LETN$(N)=N$(N)+","
550 INPUT"ADDRESS 1";A$(N)
560 LETA$(N)=A$(N)+","
570 INPUT"ADDRESS 2";D$(N)
580 LETD$(N)=D$(N)+","
590 INPUT"ADDRESS 3";V$(N)
600 LETV$(N)=V$(N)+","
610 INPUT"TEL.NO - ";P$(N)
620 LETP$(N)=P$(N)+"."
630 IFD$(N)=""THENLETD$(N)="-"
640 IFV$(N)=""THENLETV$(N)="-"
650 IFP$(N)=""THENLETP$(N)="-"
660 PRINT:PRINTCHR$(18)"ANOTHE
R ENTRY? (Y/N)"
670 GETKEYC$
680 IFN>=X-1 THEN RETURN
690 IFC$="Y"THEN GOTO 490
700 IFC$="N"THEN RETURN
710 GOTO 670
720 REM ** READ **
730 GRAPHIC0,1
740 IFN=0 THEN RETURN
750 CHAR1,10,2,"READ NAMES/ADD
RESSES",1
760 CHAR1,2,4,"F1 - READ ALL"
770 CHAR1,2,6,"F3 - MATCH NAME
"
780 GETKEYC$
790 IFC$="1"THEN820
800 IFC$="3"THEN940
810 GOTO720
820 FORI=1TON
830 SCNCLR
840 PRINT:PRINT" ENTRY NO.";I;
"OF";N
850 PRINT:PRINT" NAME:     ";N
$(I)
860 PRINT" ADDRESS 1:";A$(I)
870 PRINT" ADDRESS 2:";D$(I)
880 PRINT" ADDRESS 3:";V$(I)
890 PRINT" TEL.NO - :";P$(I)
900 PRINT:PRINTCHR$(18)"PRESS
ANY KEY TO CONTINUE"CHR$(146)
910 GETKEYC$
920 NEXTI
930 RETURN
940 SCNCLR
950 PRINT:PRINTTAB(2)"MATCH WH
ICH NAME";:INPUTI$
960 LETI$=I$+","
970 FORI=1TON
980 IFN$(I)=I$THENGOTO1050
990 NEXTI
1000 PRINT:PRINTCHR$(18)"NO SU
CH NAME IN FILE"CHR$(146)
1010 PRINT:PRINT"PRESS ANY KEY
 TO RETURN TO MAIN MENU"
1020 GETC$:IFC$=""THEN1020
1030 RETURN
1040 SCNCLR
1050 PRINT:PRINT" ENTRY NO.";I
;"OF";I
1060 PRINT:PRINT" NAME:     ";N
$(I)
1070 PRINT" ADDRESS 1:";A$(I)
1080 PRINT" ADDRESS 2:";D$(I)
1090 PRINT" ADDRESS 3:";V$(I)
1100 PRINT" TEL.NO - :";P$(I)
1110 PRINT:PRINTCHR$(18)"PRESS
 ANY KEY TO RETURN TO MAIN MEN
U"CHR$(146)
1120 GETC$:IFC$=""THEN1120
1130 RETURN
1140 REM ** PRINT **
1150 GRAPHIC0,1
1160 CHAR1,2,2,"F1 - CONTINUE"
1170 CHAR1,2,4,"F3 - RETURN TO
 MENU"
1180 GETKEYC$
1190 IFC$="1"THEN1220
1200 IFC$="3"THEN RETURN
1210 GOTO 1140
1220 SCNCLR:PRINT:PRINTCHR$(18
)"  PRINTING ADDRESSES  "CHR$(
146)
1230 PRINT:PRINTCHR$(18)"  PLE
ASE WAIT.  "CHR$(146)
1240 OPEN1,4
1250 FORI=1TON
1260 PRINT#4,N$(I)
1270 PRINT#4,A$(I)
1280 PRINT#4,D$(I)
1290 PRINT#4,V$(I)
1300 PRINT#4,P$(I)
1310 PRINT#4
1320 PRINT#4:PRINT#4
1330 NEXTI
1340 CLOSE1
1350 RETURN
1360 REM ** ERASE **
1370 IFN=0 THEN RETURN
1380 GRAPHIC0,1
1390 CHAR1,2,2,"F1 - CONTINUE"
1400 CHAR1,2,4,"F3 - RETURN TO
 MENU"
1410 GETKEYC$
1420 IFC$="1"THEN1450
1430 IFC$="3"THENRETURN
1440 GOTO1360
1450 SCNCLR
1460 CHAR1,10,2,"ERASE NAMES/A
DDRESSES",1
1470 CHAR1,2,4,"F1 - SCAN ALL"
1480 CHAR1,2,6,"F3 - MATCH NAM
E"
1490 GETC$:IFC$=""THEN1490
1500 IFC$="1"THEN1530
1510 IFC$="3"THEN1700
1520 GOTO1360
1530 FORI=1TON
1540 SCNCLR
1550 PRINT"ENTRY #";I;"OF";N
1560 PRINT"NAME:     ";N$(I)
1570 PRINT"ADDRESS 1:";A$(I)
1580 PRINT"ADDRESS 2:";D$(I)
1590 PRINT"ADDRESS 3:";V$(I)
1600 PRINT"TEL.NO - :";P$(I)
1610 PRINT:PRINT"F1 - CONTINUE
"
1620 PRINT:PRINT"F3 - ERASE"
1630 GETKEYC$
1640 IFC$="1"THEN GOTO 1680
1650 IFC$="3"THEN GOSUB 1830:G
OTO1670
1660 GOTO1630
1670 N=N-1:I=I-1
1680 NEXTI
1690 RETURN
1700 SCNCLR
1710 INPUT"MATCH WHICH NAME";I
$
1720 LETI$=I$+","
1730 FORI=1TON
1740 IFN$(I)=I$ THEN GOSUB 183
0
1750 NEXTI
1760 PRINT"NO SUCH NAME IN FIL
E"
1770 PRINT"PRESS ANY KEY TO RE
TURN TO MENU"
1780 GETKEYC$:GOTO1020
1790 RETURN
1800 PRINT:PRINTCHR$(18)"NO SU
CH NAME IN FILE"CHR$(146)
1810 PRINT:PRINT"PRESS ANY KEY
 TO RETURN TO MENU"
1820 GETC$:IFC$=""THEN1020
1830 FORK=ITON
1840 N$(K)=N$(K+1)
1850 A$(K)=A$(K+1)
1860 D$(K)=D$(K+1)
1870 V$(K)=V$(K+1)
1880 P$(K)=P$(K+1)
1890 LETN=N-1:LETI=I-1
1900 NEXTK
1910 GOTO150
1920 REM ** LOAD **
1930 GRAPHIC0,1
1940 CHAR1,2,2,"F1 - CONTINUE"
1950 CHAR1,2,4,"F3 - RETURN TO
 MENU"
1960 GETKEYC$
1970 IFC$="1"THEN2000
1980 IFC$="3"THEN RETURN
1990 GOTO1920
2000 SCNCLR
2010 CHAR1,12,2,"  LOAD FILE
",1
2020 CHAR1,2,4,"F1 LOAD FROM D
ISK"
2030 CHAR1,2,6,"F3 LOAD FROM T
APE"
2040 GETKEYC$
2050 IFC$="1"THEN2080
2060 IFC$="3"THEN2240
2070 GOTO2040
2080 OPEN1,8,3,"0:128 MAIL LIS
T,SEQ"
2090 INPUT#1,N
2100 FORI=1TON
2110 INPUT#1,N$(I)
2120 LETN$(I)=N$(I)+","
2130 INPUT#1,A$(I)
2140 LETA$(I)=A$(I)+","
2150 INPUT#1,D$(I)
```

```
2160 LETD$(I)=D$(I)+","
2170 INPUT#1,V$(I)
2180 LETV$(I)=V$(I)+","
2190 INPUT#1,P$(I)
2200 NEXTI
2210 CLOSE1
2220 IFC$="1"THENPRINT:PRINT:P
RINT"  DISK STATUS - "DS$:SLEE
P3
2230 RETURN
2240 PRINT"";:OPEN1,1,0,"128 M
AIL LIST"
2250 GOTO2090
2260 REM ** SAVE **
2270 IFN=0THEN RETURN
2280 GRAPHIC0,1
2290 CHAR1,2,2,"F1 - CONTINUE"
2300 CHAR1,2,4,"F3 - RETURN TO
 MENU"
2310 GETKEYC$
2320 IFC$="1"THEN2350
2330 IFC$="3"THEN RETURN
2340 GOTO2260
2350 SCNCLR
2360 CHAR1,12,2,"  SAVE FILE
",1
2370 CHAR1,2,4,"F1 - SAVE TO D
ISK"
2380 CHAR1,2,6,"F3 - SAVE TO T
APE"
2390 GETKEYC$
2400 IFC$="1"THEN2430
2410 IFC$="3"THEN2580
2420 GOTO2390
2430 OPEN1,8,15
2440 PRINT#1,"S0:128 MAIL LIST
"
2450 CLOSE 1
2460 OPEN1,8,3,"0:128 MAIL LIS
T,SEQ,WRITE"
2470 PRINT#1,N
2480 FORI=1TON
2490 PRINT#1,N$(I)
2500 PRINT#1,A$(I)
2510 PRINT#1,D$(I)
2520 PRINT#1,V$(I)
2530 PRINT#1,P$(I)
2540 NEXTI
2550 CLOSE1
2560 IFC$="1"THENPRINT:PRINT:P
RINT"  DISK STATUS - "DS$:SLEE
P3
2570 RETURN
2580 PRINT"";:OPEN 1,1,1,"128
MAIL LIST"
2590 GOTO2470
2600 REM ** CHANGE **
2610 IFN=0 THEN RETURN
2620 GRAPHIC0,1
2630 CHAR1,2,2,"F1 - CONTINUE"
2640 CHAR1,2,4,"F3 - RETURN TO
 MENU"
2650 GETKEYC$
2660 IFC$="1"THEN2690
2670 IFC$="3"THENRETURN
2680 GOTO2600
2690 SCNCLR
2700 CHAR1,10,2,"  CHANGE AN E
NTRY  ",1
2710 CHAR1,2,4,"F1 - SCAN & CH
ANGE"
2720 CHAR1,2,6,"F3 - MATCH"
2730 GETKEYC$
2740 IFC$="1"THEN2770
2750 IFC$="3"THEN2910
2760 GOTO2730
2770 FORI=1TON
2780 SCNCLR:PRINTN$(I)
2790 PRINTA$(I)
2800 PRINTD$(I)
2810 PRINTV$(I)
2820 PRINTP$(I)
2830 PRINT:PRINT"F1 - CONTINUE
"
2840 PRINT"F3 - CHANGE"
2850 GETKEYC$
2860 IFC$="1"THEN GOTO 2890
2870 IFC$="3"THEN GOSUB 3060:G
OTO2890
2880 GOTO2850
2890 NEXTI
2900 RETURN
2910 SCNCLR
2920 INPUT"MATCH WITH WHAT NAM
E";M$
2930 LETM$=M$+","
2940 FORI=1TON
2950 IFN$(I)=M$THENGOSUB 3010:
RETURN
2960 NEXTI
2970 PRINTCHR$(18)"NO SUCH NAM
E IN FILE"CHR$(146)
2980 PRINT"PRESS ANY KEY TO RE
TURN TO MAIN MENU"
2990 GETC$:IFC$=""THEN2990
3000 RETURN
3010 SCNCLR:PRINTN$(I)
3020 PRINTA$(I)
3030 PRINTD$(I)
3040 PRINTV$(I)
3050 PRINTP$(I)
3060 PRINT"1 >> NAME"
3070 PRINT"2 >> ADDRESS 1"
3080 PRINT"3 >> ADDRESS 2"
3090 PRINT"4 >> ADDRESS 3"
3100 PRINT"5 >> TEL.NO - "
3110 PRINT:INPUT"CHANGE WHICH"
;C
3120 ONCGOSUB3140,3150,3160,31
70,3180
3130 RETURN
3140 PRINT:INPUT"CHANGE TO WHA
T";Q$:N$(I)=Q$:LETN$(I)=N$(I)+
",":I=I-1:RETURN
3150 PRINT:INPUT"CHANGE TO WHA
T";Q$:A$(I)=Q$:LETA$(I)=A$(I)+
",":I=I-1:RETURN
3160 PRINT:INPUT"CHANGE TO WHA
T";Q$:D$(I)=Q$:LETD$(I)=D$(I)+
",":I=I-1:RETURN
3170 PRINT:INPUT"CHANGE TO WHA
T";Q$:V$(I)=Q$:LETV$(I)=V$(I)+
",":I=I-1:RETURN
3180 PRINT:INPUT"CHANGE TO WHA
T";Q$:P$(I)=Q$:LETP$(I)=P$(I)+
",":I=I-1:RETURN
3190 REM ****** EXIT ******
3200 GRAPHIC1,1
3210 CHAR1,2,5,"HAVE YOU SAVED
 NEW ADDRESSES ?"
3220 GETKEYC$
3230 IFC$="Y"THENGRAPHIC0,1:EN
D
3240 IFC$="N"THENRETURN:ELSE32
00
```



# Binders

## Organise and protect your disk with Commodore Disk User disk binders and data disks.



Why not keep your Commodore Disk User program collection alongside your magazines in a stylish Disk User disk binder? The binder comes complete with 10 disk sleeves to organise and protect your program disks. Why not buy a disk binder to house all of your data disks? We can even supply Commodore Disk User data disks. The Commodore Disk User logo immediately identifies your disks and there's room to title them and document the disks details.
Send for your disks and binders now!

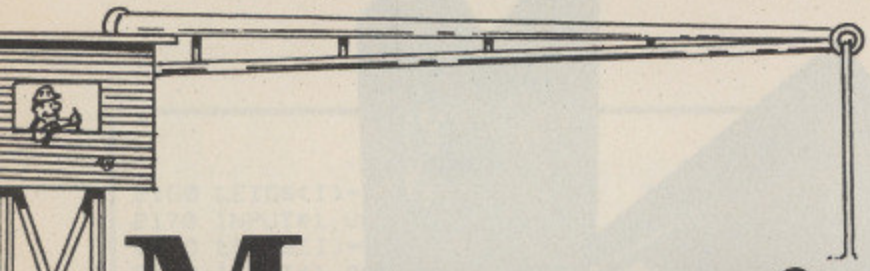
Prices are as follows:
Commodore Disk User Binder £4.95, including 10 sleeves. Order code **BDYU1**
Commodore Disk User Binder with 10 sleeves and 10 disks, £9.95 Order code **BDYU2**
10 sleeves for insertion in binder, £1.50. Order code **BDS10**
20 sleeves for inclusion in binder, £2.75. Order code **BDS20**
10 Commodore Disk User data disks, £5.95. Order code **BDD10**

| PRODUCT NAME | ORDER CODE | QUANTITY | PRICE |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| **Overseas postage add £1.00** | | | |

CHEQUES PAYABLE TO A.S.P LTD

**TOTAL**

**All orders should be sent to: YOUR COMMODORE, READERS SERVICES, ARGUS SPECIALIST PUBLICATIONS, 9 HALL ROAD, HEMEL HEMPSTEAD, HERTS HP2 7BH.**
**Please allow 28 days for delivery.**



# Message Construction Kit

*Produce customised scrolling displays with a suite of editor programs*

by Frank van Tiggelen

The Message Construction Kit is a useful package which can add a pleasing, dual-line banner effect to most programs. It can scroll credits across the screen or add wrap-around instructions. As long as there's room for an interrupt, MCK can help.

Although the messages are referred to as text, they can include user-defined pictures. In fact, the whole text is redefinable to suit your own particular needs.

By raising the start of Basic to $3000 (12288), there is room for a character set from $2000 (8192) upwards and a message can be stored in the space from $0800 to $1FFF (2048–8191) giving room for a string of 6144 characters. The routine which forms the interrupt engine is stored from $C000 to $C166 (49152–49510).

The three MCK editors control text, characters and storage accessed via a master menu which leads to sub-menus for each option.

## Text Editor (T)

There are six options within the text editor.

**E** – selects the 'text edit' facility which allows the scrolling text to be typed in. At first, the program asks for a starting point which should lie somewhere in the text storage range of 2048 to 8191. Entering text is then simply a case of typing in the words, using the DEL key to correct any errors. Pressing RETURN recalls the option menu.

**V** – the View Text option is used to display the text from a given point in memory. While the text is being displayed it can be paused with the F7 key or terminated by pressing the spacebar.

This function also allows you to check the text length which can then be stored by entering the value through the 'M' option from the text menu. This tells the program where the text ends and the wrap-around begins.

**S** – reveals the main scrolling demo where the results of your labours can be viewed. Press the key and the demo is displayed in fully defined characters across the top two screen lines. Pressing the spacebar may cause a panic at first because a warm reset occurs. This is normal and entering RUN will restore the program without losing any character or text information.

**R** – if text has been omitted, it can be inserted using the option. On entry, the program asks for a start and end address of the block of text to be moved, the destination address must then be entered. For the end address you have to know how many characters need to be inserted, this can then be added to the start address to give the end value.

This facility can also be used to repeat blocks of text. The limitation is that only 2000 characters can be moved at a time.

**M** – As mentioned before, this is the 'set message length' option that tells the system where the loop starts and ends. This can be any value from 256 to 6144 in blocks of 256 characters. The number of characters is increased with the '+' key and decreased using '–'.

**X** – Exits from the text menu to the main menu.

## Character Editor (C)

Up to 128 double height characters can be designed and manipulated to create the building blocks for a scrolling message. The screen displays a 16 × 8 grid, a menu of options and a full character set. The screen layout can be seen in Diagram 1.

The functions can be selected by pressing the specially allocated keys displayed alongside the menu but, by pressing the spacebar, the options can be highlighted with a joystick and the option will be executed when the fire button is pressed.

The space character (ASCII 32) is not included in the redefinable set so that the screen does not fill up

with rubbish. This also means that spaces can easily be inserted in the text editor by pressing the spacebar as normal.

When the editing process is complete, the X option will return the program to Basic but the program can be re-run without losing the defined characters from memory.

**SELECT CHARACTER** – This option moves the cursor to the character set at the bottom of the screen. The character to be edited can be selected by moving the cursor onto the relevant character and then pressing fire.

**CLEAR CHARACTER** – When this option is selected, the current character is erased ready for redefining.

**REVERSE CHARACTER** – All of the on pixels are turned off and vice versa when this option is chosen for the character being edited.

**FIRE MODE PLOT** – The fire mode (FM) is set to unit value and a pixel is set when the fire button is pressed.

**FIRE MODE ERASE** – This is like the last option but the FM value is set to zero and the fire button is now used to erase a pixel.

**COPY CHARACTER** – If you want to make a slightly altered version of another character from the set, this option allows you to copy it. First, the program asks for the character to be copied and this is selected on the lower display using the joystick. Next, the character position to which the copy is made is selected in the same way.

**GET CHARACTER** – After selecting a character for display, this function transfers the information to the editing grid.

**PRINT CHARACTER** – A permanent pixel map can be printed out using this facility.

**MIRROR X** – After selection, the character to be mirrored is laterally reversed.

**MIRROR Y** – This is the same as mirroring in the X direction but the character is inverted instead.

**5 CHAR PAINT** – Five consecutive characters can be designed simultaneously on a special 40 × 16 grid. After the first character of the sequence has been selected from the character set display, the screen changes to reveal the new grid. Beneath this display is the fire mode (1 or 0) and the poke values of the characters displayed. The first character must have a screen poke value of less than 123.

The five character frieze can now be created using the '+' and '–' keys to select the fire mode. When finished, the characters are transferred to the computer's memory by pressing the spacebar. F7 displays the characters on the screen as they will appear in the text and, if they look alright, the main editing screen can be re-entered by pressing 'X'.

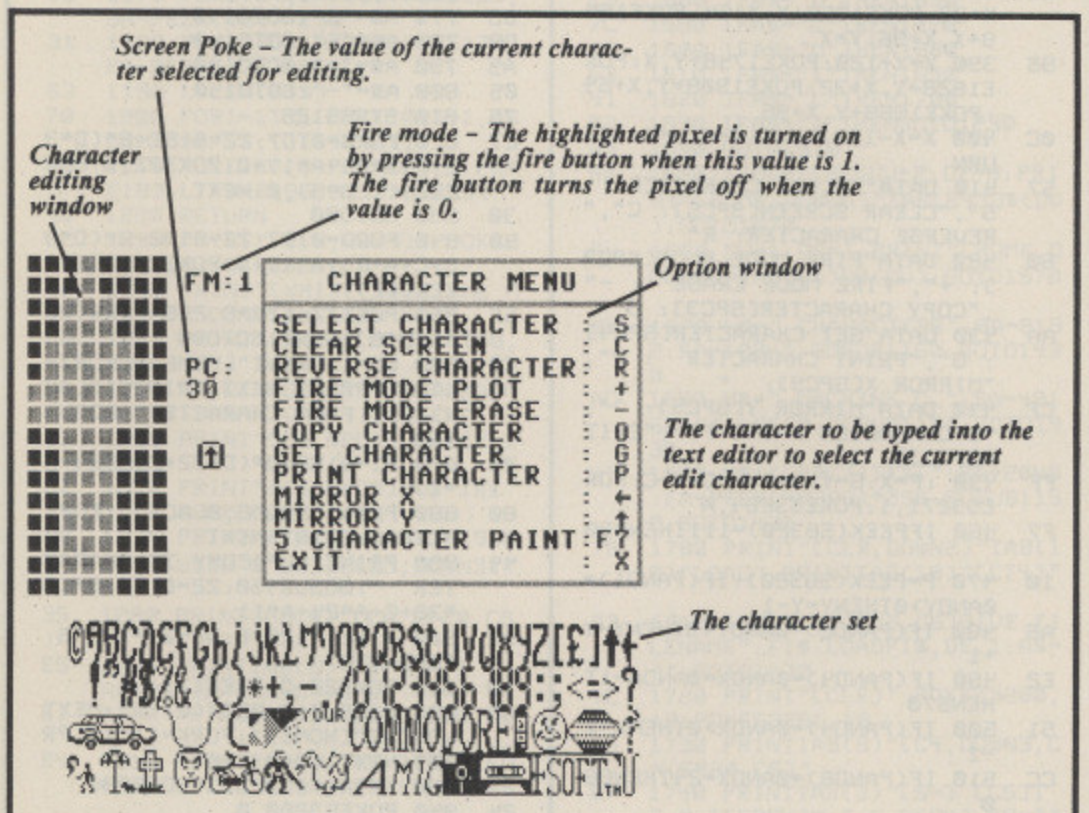



*Diagram 1 – The Character Editor*

## Storage (S)

The options available from the storage menu relate to saving and loading routines.

**CHANGE DEVICE** – The system is initially set for tape operations but this function will toggle from tape to disk.

**SAVE CHARACTERS** – The current character set will be saved under a filename of your own choice.

**SAVE TEXT** – This saves the current text message on to the storage device.

**SAVE MACHINE CODE** – As the scrolling message is modified through the text editor, the machine code is automatically tailored to run the program. This option saves the correct routine to drive the current characters and message.

**LOAD** – Can be used to load any of the three file types saved with the preceding three options.

## Program creation

To use the files with your own programs, first enter the following:

POKE 43,0:POKE 44,48:POKE 12287, 0:NEW

Now the character set, text message and machine code can be loaded in. Before loading your own routine type NEW again.

The following commands will kickstart the scroller:

POKE 899,0:POKE 900,199: POKE 49324,8:SYS 49152

The code routine occupies the locations from 49152 to 49510.

INSTRUCTIONS FOR ENTERING THE PROGRAMS

1) TYPE IN AND SAVE 'MESSAGE CONSTRUCTION KIT' ONTO YOUR MASTER TAPE OR DISK
2) TYPE IN AND SAVE 'M/C MAKER' ONTO A SEPARATE TAPE/DISK
3) RUN 'M/C MAKER' AND ADD IT TO THE MASTER TAPE/DISK AFTER 'MCK'
4) TYPE IN AND SAVE 'CHARACTER MAKER' ONTO A SEPARATE TAPE/DISK
5) RUN 'TEXT MAKER' AND ADD IT TO THE MASTER TAPE/DISK AFTER 'M/C MAKER'
6) TYPE IN AND SAVE 'TEXT MAKER' ONTO A SEPARATE TAPE/DISK
7) RUN 'TEXT MAKER' AND ADD IT TO THE MASTER TAPE/DISK AFTER 'CHARACTER MAKER'
8) LOAD AND RUN 'MCK' AND LOAD THE PROGRAMS IN THIS ORDER:

M/C
CHARS
TEXT

PROGRAM: MESSAGE CONSTRUCTION KIT

```
4A  10 DE=1:DV=1:D$(1)="TAPE":D$
    (2)="DISK":GOTO1230
3A  20 POKE53280,12:POKE53281,15
    :POKE646,0:PRINT"[C4,CLR,BLA
    CK]";:POKE53269,1:POKE53287,
    0
7C  30 FORX=0TO63:POKE832+X,0:NE
    XT:POKE832,255:FORX=2TO6:POK
    E832+X*3,195:NEXT:J=1
6B  40 POKE853,255:POKE856,255:P
    OKE835,255:POKE53248,24:POKE
    53249,50:POKE2040,13
09  50 FORX=0TO15:PRINT"[WHITE,R
    VSON,SO8]":NEXT:PRINT"[HOME,
    DOWN]"TAB(9)"[C4]FM:[WHITE]1
    ":MD=11:GOSUB310
BE  60 A=0:XX=56:SYS49397:FORQ=8
    *32TOQ+7:POKE8192+Q,0:NEXT:P
    RINT"[HOME,DOWN5]"TAB(9)"PC:
    "
A6  70 PRINTTAB(10)"[DOWN2,C5,C@
    ]":PRINTTAB(9)"[CM,C4]@[C5,C
    G]":PRINTTAB(10)"[CT]":GOTO5
    60
D5  80 P=PEEK(56320):IF(PAND1)=0
    ANDY>0THENY=Y-1
82  90 IF(PAND2)=0ANDY<15THENY=Y
    +1
22  100 IF(PAND4)=0ANDX>0THENX=X
    -1
0E  110 IF(PAND8)=0ANDX<7THENX=X
    +1
62  120 IF(PAND16)=0THENPOKE5529
    6+X+Y*40,MD:GOTO280
24  130 POKE53248,24+X*8:POKE532
    49,50+Y*8:GETA$:IFA$=""THENB
    0
60  140 IFA$="+"THENMD=11:PRINT"
    [HOME,DOWN]"TAB(12)"[WHITE]1
    ":GOTO80
48  150 IFA$="-"THENMD=1:PRINT"[
    HOME,DOWN]"TAB(12)"[WHITE]0"
    :GOTO80
89  160 IFA$="S"THEN450
B2  170 IFA$="C"THENPRINT"[HOME]
    ";:FORQ=1TO16:PRINT"[WHITE,R
    VSON,SO8]":NEXT:GOTO820
3E  180 IFA$="R"THENGOTO840
45  190 IFA$=" "THEN680
EA  200 IFA$="X"THEN810
78  210 IFA$="O"THEN860
A5  220 IFA$="G"THEN940
F3  230 IFA$="P"THEN1010
E3  240 IFA$="[F7]"THEN2800
97  250 IFA$="_"THENGOTO2820
FD  260 IFA$="^"THENGOTO2950
B4  270 GOTO680
15  280 ZZ=8192+8*(32*D+C+A*24+A
    *1)+Y:IFY>7THENZZ=ZZ+127*8
30  290 IFMD<>1THENPOKEZZ,PEEK(Z
    Z)OR(2^(7-X)):GOTO130
D8  300 POKEZZ,PEEK(ZZ)AND(255-2
    ^(7-X)):GOTO130
13  310 PRINT"[HOME,C5]"TAB(13)"
    [CA,SC20,CS]"
29  320 PRINTTAB(13)"[SB,SPC3,C4
    ]CHARACTER MENU[C5,SPC3,SB]"

30  330 PRINTTAB(13)"[C5,CQ,SC20
    ,CW]":FORA=1TO12
65  340 PRINTTAB(13)"[SB]"TAB(34
    )"[SB]":NEXT:PRINTTAB(13)"[C
    Z,SC20,CX,C4]"
DD  350 PRINT"[HOME,DOWN2]":POKE
    53264,0:POKE53271,0
87  360 FORX=1TO12:READA$:PRINTT
    AB(14)LEFT$(A$,18)"[WHITE]"R
    IGHT$(A$,2)"[C4]":NEXT:PRINT
    "[HOME]";
8E  370 FORX=0TO31
3B  380 POKE1708+X,X:POKE1788+X,
    X+32:POKE1868+X,X+64:POKE194
    8+X,X+96:Y=X
9B  390 X=X+128:POKE1748+Y,X:POK
    E1828+Y,X+32:POKE1908+Y,X+64
    :POKE1988+Y,X+96
0C  400 X=X-128:NEXT:X=0:Y=0:RET
    URN
57  410 DATA"SELECT CHARACTER :
    S","CLEAR SCREEN[SPC5]: C","
    REVERSE CHARACTER: R"
60  420 DATA"FIRE MODE PLOT[SPC3
    ]: +","FIRE MODE ERASE  : -"
    ,"COPY CHARACTER[SPC3]: O"
AA  430 DATA"GET CHARACTER[SPC4]
    : G","PRINT CHARACTER  : P",
    "MIRROR X[SPC9]: _"
CF  440 DATA"MIRROR Y[SPC9]: ^",
    "5 CHARACTER PAINT:F7","EXIT
    [SPC13]: X"
FF  450 :F=X:G=Y:A=E:Y=D:X=C:POK
    E53271,1:POKE53264,A
F7  460 IFPEEK(56320)=111THEN460

10  470 P=PEEK(56320):IF(PAND1)=
    0ANDY>0THENY=Y-1
A6  480 IF(PAND2)=0ANDY<3THENY=Y
    +1
E2  490 IF(PAND4)=0ANDX=0ANDA=1T
    HEN670
51  500 IF(PAND4)=0ANDX>0THENX=X
    -1
CC  510 IF(PAND8)=0ANDX=24THEN66
    0
0B  520 IF(PAND8)=0ANDX<35THENX=
    X+1
FE  530 IF(PAND16)=0THEN560
2A  540 IFA=1ANDX=7THENX=X-1
73  550 POKE53248,XX+X*8:POKE532
    49,185+16*Y:GOTO470
5E  560 W=E*24+E*1+C+D*80:POKE55
    980+W,11:POKE55980+W+40,11:W
    =A*24+A*1+X+Y*80
E1  570 POKE55980+W,0:POKE55980+
    W+40,0:C=X:D=Y:E=A:X=F:Y=G:P
    OKE53271,0
10  580 QQ=(D*32+C+A*24+A*1):PRI
    NT"[HOME,DOWN6,WHITE]"TAB(8)
    QQ"[LEFT] ";:IFQQ<10THENPRIN
    T" "
CC  590 POKE1024+10+9*40,QQ:PRIN
    T"[HOME]";:FORL=1TO16:PRINT"
    [WHITE,RVSON,SO8]":NEXT
E8  600 POKE53264,0:IFWQ=1THENRE
    TURN
F3  610 P=PEEK(56320):IFP=111THE
    N610
2F  620 GOTO80
91  630 IFDV=1THENDV=2:DN=8:GOTO
    650
B3  640 DV=1:DN=DV
00  650 PRINT"[HOME,WHITE]"TAB(2
    9)D$(DV):GOTO80
D0  660 X=0:XX=0:A=1:POKE53264,A
    :GOTO540
9E  670 X=24:XX=56:A=0:POKE53264
    ,A:GOTO540
20  680 POKE53248,24:POKE53264,1
    :POKE53277,1:I=Y:Y=J
F8  690 P=PEEK(56320):IF(PAND1)=
    0ANDY>1THENY=Y-1
D7  700 IF(PAND2)=0ANDY<12THENY=
    Y+1
D9  710 IF(PAND16)=0THEN730
34  720 POKE53249,65+Y*8:GOTO690

BA  730 POKE53264,0:POKE53277,0
93  740 IFWQ=1THENRETURN
A4  750 P=PEEK(56320):IFP=111THE
    N750
D5  760 J=Y:Y=I:ONJGOTO450,770,7
    80,790,800,860,940,1010,2820
    ,2950,2800,810
B9  770 A$="C":GOTO140
D8  780 A$="F":GOTO140
A5  790 A$="+":GOTO140
05  800 A$="-":GOTO140
75  810 SYS65126
27  820 FORQ=0TO7:ZZ=8192+8*(D*3
    2+C+A*24+A*1)+Q:POKEZZ,0:POK
    EZZ+(128*8),0:NEXT
30  830 GOTO80
40  840 FORQ=0TO7:ZZ=8192+8*(D*3
    2+C+A*24+A*1)+Q:POKEZZ,255-P
    EEK(ZZ)
5F  850 POKEZZ+128*8,255-PEEK(ZZ
    +128*8):NEXT:GOTO80
E8  860 WQ=1:PRINT"[HOME]":FORX=
    1TO15:PRINT:NEXT:PRINT"[BLAC
    K]COPY FROM CHARACTER":GOSUB
    450
20  870 Z1=8192+8*(D*32+C+A*24+A
    *1)
80  880 PRINT"[HOME,BLACK]":FORK
    =1TO16:PRINT:NEXT
44  890 PRINT"[UP]COPY TO CHARAC
    TER  ":GOSUB450:Z2=8192+8*(D
    *32+C+A*24+A*1)
AC  900 W1=Z1+128*8:W2=Z2+128*8:
    FORQ=0TO7
18  910 POKEZ2+Q,PEEK(Z1+Q)
5E  920 POKEW2+Q,PEEK(W1+Q):NEXT
    :PRINT"[HOME]":FORK=1TO15:PR
    INT:NEXT:WQ=0:X=0
8B  930 PRINT"[SPC35]":GOTO80
24  940 POKE53280,0
86  950 ZZ=8192+8*(D*32+C+A*24+A
    *1):WW=ZZ+128*8:FORQ=0TO7:T=
    PEEK(ZZ+Q):S=PEEK(WW+Q)
1B  960 FORR=7TO0STEP-1:IFT-2^R>
    =0THENT=T-2^R:POKE55296+Q*40
    +(7-R),11:GOTO980
91  970 POKE55296+Q*40+7-R,1
20  980 IFS-2^R>=0THENS=S-2^R:PO
    KE55616+Q*40+7-R,11:GOTO1000

E1  990 POKE55616+Q*40+7-R,1
FB  1000 NEXT:NEXT:POKE53280,11:
    GOTO80
```

PROGRAM: MESSAGE CONSTRUCTION KIT

```
A2  1010 PRINT"[HOME]":FORK=1TO1
    5:PRINT:NEXT:PRINT"[BLACK]DO
     YOU OWN A CITIZEN IDP-560?
    (Y/N)"
80  1020 GETA$:IFA$<>"Y"ANDA$<>"
    N"THEN1020
FD  1030 IFA$="Y"THENOPEN6,4,6:P
    RINT#6,CHR$(0):CLOSE6
E0  1040 PRINT"[UP,SPC36]"
0A  1050 OPEN4,4:POKE53265,PEEK(
    53265)AND239
A4  1060 ZZ=8192+8*(D*32+C):FORQ
    =0TO7:T=PEEK(ZZ+Q)
FE  1070 FORR=7TO0STEP-1:IFT-2^R
    >=0THENT=T-2^R:PRINT#4,"[RVS
    ON] [RVSOFF]";:GOTO1090
C0  1080 PRINT#4,"[SO]";
27  1090 NEXT:PRINT#4,"[CG]":NEX
    T
AB  1100 ZZ=8192+8*(D*32+C)+128*
    8:FORQ=0TO7:T=PEEK(ZZ+Q)
9F  1110 FORR=7TO0STEP-1:IFT-2^R
    >=0THENT=T-2^R:PRINT#4,"[RVS
    ON] [RVSOFF]";:GOTO1130
28  1120 PRINT#4,"[SO]";
2A  1130 NEXT:PRINT#4,"[CG]":NEX
    T:PRINT#4,"[CT8]":CLOSE4:POK
    E53265,PEEK(53265)OR16
FD  1140 X=0:GOTO80
1F  1150 PRINT"[CLR,DOWN2,C4]"TA
    B(12)"SAVE "N$:PRINTAB(12)"
    [CT16]"
44  1160 INPUT"[DOWN2,CS]GIVE FI
    LENAME ";FI$:IFFI$=""ANDDE=8
    THENPRINT"[UP3]";:GOTO1160
82  1170 POKE194,SA/256:POKE193,
    SA-PEEK(194)*256
31  1180 POKE175,FA/256:POKE174,
    FA-PEEK(175)*256
53  1190 L=LEN(FI$)
70  1200 FORI=1TOL:POKE1023+I,AS
    C(MID$(FI$,I,1)):NEXT
4D  1210 POKE187,0:POKE188,4:POK
    E183,L:POKE186,DE:SYS62954
46  1220 RETURN
8C  1230 PRINT"[CLR,DOWN]":POKE5
    3280,0:POKE53281,0
DB  1240 PRINT"[WHITE,SPC6]THE M
    ESSAGE CONSTRUCTION KIT"
CA  1245 PRINT"[WHITE,SPC6,CT28,
    DOWN]"
0B  1250 PRINT"[C8,SPC4]PROGRAMM
    ED BY FRANK VAN TIGGELEN"
21  1260 PRINT"[C4,SPC15]MARCH 1
    988"
97  1270 PRINT"[C5,SPC9]FOR THE
    YOUR COMMODORE"
80  1275 PRINT"[YELLOW,SPC10]SER
    IOUS USER'S GUIDE[C5,DOWN3]"

35  1280 PRINTTAB(9)"[CA,S*20,CS
    ]"
E5  1290 PRINTTAB(9)"[S-] M A I
    N[SPC4]M E N U [S-]"
E1  1300 PRINTTAB(9)"[CQ,S*20,CW
    ]"
E4  1310 PRINTTAB(9)"[S-]TEXT ED
    ITOR[SPC6]: [WHITE]T[C5,S-]"

03  1320 PRINTTAB(9)"[S-]CHARACT
    ER EDITOR : [WHITE]C[C5,S-]"

E2  1330 PRINTTAB(9)"[S-]STORAGE
    [SPC10]: [WHITE]S[C5,S-]"
69  1340 PRINTTAB(9)"[CQ,S*20,CW
    ]"
E0  1350 PRINTTAB(9)"[S-]EXIT[SP
    C13]: [WHITE]X[C5,S-]"
D9  1360 PRINTTAB(9)"[CZ,S*20,CX
    ]"
DB  1370 GETA$:IFA$=""THEN1370
7E  1380 IFA$="I"THEN1720
BB  1390 IFA$="C"THEN20
21  1400 IFA$="S"THEN1430
68  1410 IFA$="X"THENSYS65126
81  1420 GOTO1370
1F  1430 PRINT"[C4,CLR]":POKE532
    80,15:POKE53281,15
93  1440 PRINTTAB(9)"[DOWN2,CA,S
    *20,CS]"
C2  1450 PRINTTAB(9)"[S-,SPC3]S
    T O R A G E[SPC4,S-]"
41  1460 PRINTTAB(9)"[CQ,S*20,CW
    ]"
45  1470 PRINTTAB(9)"[S-]  STORA
    GE TO  "D$(DV)"  [SB]"
55  1480 PRINTTAB(9)"[CQ,S*20,CW
    ]"
AA  1490 PRINTTAB(9)"[S-]CHANGE
    DEVICE[SPC4]: [WHITE]D[C4,SB
    ]"
CC  1500 PRINTTAB(9)"[S-]SAVE CH
    ARACTERS  : [WHITE]C[C4,SB]"

C4  1510 PRINTTAB(9)"[S-]SAVE TE
    XT[SPC8]: [WHITE]T[C4,SB]"
40  1520 PRINTTAB(9)"[S-]SAVE MA
    CHINECODE : [WHITE]M[C4,SB]"

64  1530 PRINTTAB(9)"[S-]LOAD[SP
    C13]: [WHITE]L[C4,SB]"
98  1540 PRINTTAB(9)"[CQ,S*20,CW
    ]"
68  1550 PRINTTAB(9)"[S-]EXIT[SP
    C13]: [WHITE]X[C4,S-]"
98  1560 PRINTTAB(9)"[CZ,S*20,CX
    ]"
5A  1570 GETA$:IFA$=""THEN1570
E5  1580 IFA$="D"THEN1650
7C  1590 IFA$="C"THEN1670
9F  1600 IFA$="M"THEN1680
63  1610 IFA$="I"THEN1690
41  1620 IFA$="L"THEN1700
B2  1630 IFA$="X"THENGOTO1230
54  1640 GOTO1570
8B  1650 IFDV=1THENDV=2:DE=8:PRI
    NT"[HOME,DOWN6]"TAB(24)D$(DV
    ):GOTO1570
64  1660 DV=1:DE=1:PRINT"[HOME,D
    OWN6]"TAB(24)D$(DV):GOTO1570

B8  1670 N$=" CHARACTERS":SA=819
    2:FA=10241:GOSUB1150:GOTO143
    0
AC  1680 N$=" MACHINE C.":SA=491
    52:FA=49510:GOSUB1150:GOTO14
    30
41  1690 N$="[SPC7]TEXT":SA=2048
    :FA=PEEK(49386)*256:GOSUB115
    0:GOTO1430
7B  1700 PRINT"[CLR,DOWN2]"TAB(1
    8)"LOAD":PRINTTAB(18)"[CT4]"

33  1710 INPUT"[DOWN2,CS]GIVE FI
    LENAME";FI$:LOADFI$,DE,1:A$=
    "":GOTO1430
4C  1720 PRINT"[CLR]":POKE53280,
    12:POKE53281,15
06  1730 PRINTTAB(9)"[C4,DOWN3,C
    A,S*20,CS]"
D7  1740 PRINTTAB(9)"[S-] [CS]T
    E X T[SPC4]M E N U[C4] [S-]"

52  1750 PRINTTAB(9)"[CQ,S*20,CW
    ]"
0B  1760 PRINTTAB(9)"[S-]EDIT TE
    XT[SPC8]: [WHITE]E[C4,SB]"
A1  1770 PRINTTAB(9)"[S-]VIEW TE
    XT[SPC8]: [WHITE]V[C4,SB]"
82  1780 PRINTTAB(9)"[S-]VIEW SC
    ROLLING[SPC3]: [WHITE]S[C4,S
    B]"
4E  1790 PRINTTAB(9)"[S-]REPLACE
     TEXT[SPC5]: [WHITE]R[C4,SB]
    "
41  1800 PRINTTAB(9)"[S-]SET MES
    S. LENGTH : [WHITE]M[C4,SB]"

E1  1810 PRINTTAB(9)"[CQ,S*20,CW
    ]"
09  1820 PRINTTAB(9)"[S-]EXIT[SP
    C13]: [WHITE]X[C4,S-]"
01  1830 PRINTTAB(9)"[CZ,S*20,CX
    ]"
5B  1840 GETA$:IFA$=""THEN1840
5B  1850 IFA$="E"THEN1920
92  1860 IFA$="V"THEN2110
2E  1870 IFA$="S"THEN2360
55  1880 IFA$="M"THEN2400
76  1890 IFA$="R"THEN3100
72  1900 IFA$="X"THEN1230
61  1910 GOTO1840
07  1920 PRINT"[CLR,C4,DOWN2]ALT
    ER START ADDRESS ? (Y/N) DEF
    .=2048":MS=PEEK(49386)
7F  1930 GETA$:IFA$=""THEN1930
DB  1940 NA=2048:IFA$<>"Y"THEN19
    70
46  1950 INPUT"[CLR,DOWN,C4]GIVE
     NEW START ADDRESS";NA
1B  1960 IFNA<2048ORNA>MS*256THE
    N1950
E8  1970 PRINT"[CLR,DOWN,C4]CHAR
    ACTERS FREE:"MS*256-NA:PRINT
    "[DOWN]EXIT: [WHITE][RETURN]
    [C4]"
BB  1980 Z$="":Z=0:PRINT"[DOWN2,
    CS]INPUT TEXT:[C4] -[LEFT]";

B9  1990 GETA$:IFA$=""THEN1990
D7  2000 IFA$=CHR$(13)THENGOTO17
    20
81  2010 IFA$=CHR$(20)ANDZ>0THEN
    PRINTA$"-[LEFT]";:Z=Z-1:GOTO
    1990
C7  2020 IFA$>CHR$(31)ANDA$<CHR$
    (64)THENVA=ASC(A$):GOTO2070
D4  2030 IFA$>CHR$(63)ANDA$<CHR$
    (96)THENVA=ASC(A$)-64:GOTO20
    70
12  2040 IFA$>CHR$(191)ANDA$<CHR
    $(224)THENVA=ASC(A$)-128:GOT
    O2070
4C  2050 IFA$>CHR$(159)ANDA$<CHR
    $(192)THENVA=ASC(A$)-64:GOTO
    2070
00  2060 GOTO1990
FF  2070 PRINTA$;:POKENA+Z,VA:Z=
    Z+1
D9  2080 IFZ+NA=>MS*256THENPRINT
    :PRINT"[DOWN]MEMORY FULL":GO
    TO2330
75  2090 IFA$=CHR$(34)THENPRINTC
    HR$(20)CHR$(34);
E2  2100 PRINT"-[LEFT]";:GOTO199
    0
A5  2110 INPUT"[CLR,DOWN2,C4]GIV
    E START ADDRESS ? 2048[LEFT6
    ]";TT:MS=PEEK(49386)
73  2120 IFTT<2048ORTT>MS*256THE
    N2110
8F  2130 PRINT"[CLR,DOWN2,C4]PRE
    SS [WHITE][RETURN][C4] TO ST
    ART"
1A  2140 PRINT"PRESS [WHITE][F7]
    [C4] TO PAUSE AND RESTART"
FE  2150 PRINT"PRESS [WHITE][SPA
    CE][C4] TO EXIT":AD=2048
A4  2160 GETA$:IFA$<>CHR$(13)THE
    N2160
2B  2170 PRINT"[DOWN]":FORX=TTO
    MS*256STEP24
7E  2180 X$=STR$(X):X$=RIGHT$(X$
    ,LEN(X$)-1):PRINTTAB(5-LEN(X
    $))"[BLACK]"X$"[SPC5,C4]";
BB  2190 FORY=0TO23
FF  2200 Q=PEEK(X+Y):IFQ>31ANDQ<
    64THEN2250
A6  2210 IFQ=>0ANDQ<32THENQ=Q+64
```

```
    :GOTO2250
07  2220 IFQ=>96ANDQ<128THENQ=Q+
    64:GOTO2250
71  2230 IFQ=>64ANDQ<96THENQ=Q+3
    2:GOTO2250
7A  2240 PRINT"[C5,SV,C4]";:GOTO
    2260
82  2250 PRINTCHR$(Q);:IFQ=34THE
    NPRINTCHR$(20)CHR$(34);
3D  2260 GETA$:IFA$<>""THEN2280
F8  2270 NEXT:PRINT:NEXT:GOTO233
    0
7F  2280 IFA$="[F7]"THEN2310
B3  2290 IFA$=" "THEN2330
F4  2300 GOTO2270
DE  2310 GETA$:IFA$<>"[F7]"THEN2
    310
17  2320 GOTO2270
FF  2330 PRINT:PRINT"[DOWN2,SPC8
    ]PRESS [WHITE][SPACE][C4] T
    O EXIT"
BB  2340 GETA$:IFA$<>" "THEN2340
3A  2350 GOTO1720
3A  2360 PRINT"[C4,CLR]":POKE532
    80,15:POKE53281,15:POKE899,0
    :POKE900,199:POKE901,0
A0  2370 POKE49324,8:SYS49152:PR
    INT"[HOME,DOWN4,C4,SPC9]PRES
    S [WHITE][SPACE][C4] TO EXIT
    "
27  2380 GETA$:IFA$=" "THEN810
43  2390 GOTO2380
F1  2400 PRINT"[CLR]";:MS=PEEK(4
    9386)
35  2410 PRINT"[HOME,DOWN2,C4]ME
    SSAGE LENGTH:[WHITE]"256*(MS
    -8)"[LEFT,C4] CHARACTERS "
68  2420 PRINT"[DOWN2]PRESS [WHI
    TE]+[C4] TO INCREASE AND [WH
    ITE]-[C4] TO DECREASE"
2C  2430 PRINT"NUMBER OF CHARACT
    ERS"
83  2440 PRINT"[DOWN]PRESS [WHIT
    E][SPACE][C4] TO EXIT"
8D  2450 GETA$:IFA$=""THEN2450
D2  2460 IFA$="+"ANDMS<32THENMS=
    MS+1:POKE49386,MS:GOTO2410
55  2470 IFA$="-"ANDMS>9THENMS=M
    S-1:POKE49386,MS:GOTO2410
1F  2480 IFA$=" "THENGOTO1720
4D  2490 GOTO2450
03  2500 PRINT"[C4,CLR]":POKE532
    64,0:POKE53277,0:POKE53271,0
    :R=0:S=0:T=0:TT=24:MM=1
DC  2510 MD=224:FORX=0TO4:POKE18
    40+X,CH+X:POKE1880+X,CH+X+12
    8:NEXT:A1=CH:A2=A1+4
EE  2520 PRINT"[HOME]";:FORX=1TO
    16:PRINT"[C4,SO40]";
69  2530 NEXT
C3  2540 PRINT"[RVSOFF,C4] FM:[W
    HITE]"MM"[C4,SPC3]POKES:[WHI
    TE]"A1"[C4]TO[WHITE]"A2"[C4,
    SPC3]EXIT: [WHITE]X"
AE  2550 P=PEEK(56320):IF(PAND1)
    =0ANDS>0THENS=S-1
8C  2560 IF(PAND2)=0ANDS<15THENS
    =S+1
AD  2570 IF(PAND4)=0ANDT=0ANDR=1
    THENT=28:R=0:TT=24:POKE53264
    ,0:GOTO2620
18  2580 IF(PAND4)=0ANDT>0THENT=
    T-1
07  2590 IF(PAND8)=0ANDT=28THENR
    =1:TT=0:POKE53264,1:T=0:GOTO
    2620
AC  2600 IF(PAND8)=0ANDR=0THENT=
    T+1
0A  2610 IF(PAND8)=0ANDR=1ANDT<1
    0THENT=T+1
7E  2620 IF(PAND16)=0THENPOKE102
    4+S*40+T+28*R+R*1,MD
3E  2630 POKE53248,TT+T*8:POKE53
    249,50+S*8:GETA$:IFA$=""THEN
    2550
5B  2640 IFA$="C"THEN2520
51  2650 IFA$="+"THENMD=224:MM=1
    :PRINT"[UP]";:GOTO2540
7C  2660 IFA$="-"THENMD=79:MM=0:
    PRINT"[UP]";:GOTO2540
70  2670 IFA$="X"THENRESTORE:GOT
    O20
2D  2680 IFA$=" "THEN2710
09  2690 IFA$="[F7]"THENGOTO3020
A9  2700 GOTO2550
25  2710 POKE53280,0
47  2720 FORW=0TO4:FORY=0TO7:FOR
    X=7TO0STEP-1:Z=PEEK(1024+W*8
    +(7-X)+Y*40)
D1  2730 ZZ=8192+CH*8+W*8+Y:IFZ=
    224THENPOKEZZ,PEEK(ZZ)OR(2^X
    ):GOTO2750
39  2740 POKEZZ,PEEK(ZZ)AND(255-
    2^X)
C2  2750 NEXT:NEXT:NEXT
9A  2760 FORW=0TO4:FORY=0TO7:FOR
    X=7TO0STEP-1:Z=PEEK(1344+W*8
    +(7-X)+Y*40)
D4  2770 ZZ=8192+CH*8+W*8+Y+128*
    8:IFZ=224THENPOKEZZ,PEEK(ZZ)
    OR(2^X):GOTO2790
91  2780 POKEZZ,PEEK(ZZ)AND(255-
    2^X)
70  2790 NEXT:NEXT:NEXT:POKE5328
    0,11:GOTO2550
E0  2800 PRINT"[HOME]":FORK=1TO1
    5:PRINT:NEXT:PRINT"[BLACK]5
    CHARS FROM":WQ=1:GOSUB450:WQ
    =0
17  2810 CH=QQ:GOTO2500
02  2820 PRINT"[HOME]":FORK=1TO1
    5:PRINT:NEXT:PRINT"[BLACK]MI
    RROR WHICH CHARACTER":WQ=1
00  2830 GOSUB450:POKE53280,0
D1  2840 Z=QQ*8+8192:WQ=0:POKE53
    264,0:POKE53248,24+X*8:POKE5
    3249,50+Y*8
AC  2850 PRINT"[HOME,BLACK]":FOR
    K=1TO16:PRINT:NEXT:PRINT"[UP
    ,SPC24]"
32  2860 FORQ=0TO7:POKE950+Q,PEE
    K(Z+Q):POKE960+Q,PEEK(Z+128*
    8+Q):NEXT
97  2870 FORK=0TO7:QQ=PEEK(950+K
    ):ZZ=Z+K:FORL=0TO7
29  2880 IFQQ-(2^(7-L))=>0THENPO
    KEZZ,PEEK(ZZ)OR(2^L):QQ=QQ-(
    2^(7-L)):GOTO2900
56  2890 POKEZZ,PEEK(ZZ)AND(255-
    2^L)
95  2900 NEXT:NEXT
89  2910 FORK=0TO7:QQ=PEEK(960+K
    ):ZZ=Z+128*8+K:FORL=0TO7
75  2920 IFQQ-(2^(7-L))=>0THENPO
    KEZZ,PEEK(ZZ)OR(2^L):QQ=QQ-(
    2^(7-L)):GOTO2940
6E  2930 POKEZZ,PEEK(ZZ)AND(255-
    2^L)
67  2940 NEXT:NEXT:POKE53280,11:
    GOTO80
4E  2950 POKE53280,0
1E  2960 PRINT"[HOME]":FORK=1TO1
    5:PRINT:NEXT:PRINT"[BLACK]MI
    RROR WHICH CHARACTER":WQ=1
FF  2970 GOSUB450:Z=QQ*8+8192:WQ
    =0:POKE53264,0:POKE53248,24+
    X*8:POKE53249,50+Y*8
22  2980 PRINT"[HOME,BLACK]":FOR
    K=1TO16:PRINT:NEXT:PRINT"[UP
    ,SPC24]"
14  2990 FORQ=0TO7:POKE950+Q,PEE
    K(Z+Q):POKE960+Q,PEEK(Z+Q+12
    8*8):NEXT
25  3000 FORQ=7TO0STEP-1:POKEZ+Q
    ,PEEK(967-Q):POKEZ+Q+128*8,P
    EEK(957-Q):NEXT
55  3010 POKE53280,11:GOTO80
24  3020 PRINT"[HOME]";:FORX=1TO
    16:PRINT"[C4,SO40]";
63  3030 NEXT
C0  3040 POKE53280,0
0A  3050 FORW=0TO4:FORY=0TO7:QQ=
    8192+CH*8+W*8+Y:Z=PEEK(QQ):Z
    Z=PEEK(QQ+128*8)
67  3060 FORX=0TO7
8C  3070 IFZ-2^(7-X)=>0THENZ=Z-2
    ^(7-X):POKE1024+W*8+Y*40+X,2
    24
28  3080 IFZZ-2^(7-X)=>0THENZZ=Z
    Z-2^(7-X):POKE1344+W*8+Y*40+
    X,224
D3  3090 NEXT:NEXT:NEXT:PRINT:PO
    KE53280,11:GOTO2550
B5  3100 PRINT"[CLR,DOWN2,C4]"TA
    B(14)"REPLACE TEXT":PRINTTAB
    (14)"[CT12]"
E4  3110 INPUT"[DOWN2,C4]GIVE TE
    XT START ADDRESS";TS
77  3120 INPUT"[DOWN]GIVE  TEXT
     END ADDRESS";TE
1B  3130 INPUT"[DOWN]GIVE NEW  S
    TART ADDRESS";NS
C3  3140 IFTE<TSTHENPRINT"[DOWN,
    WHITE]END SHOULD BE HIGHER T
    HAN START":GOTO3240
2A  3150 IFTS<2048ORTS>8191THENG
    OTO3230
9C  3160 IFTE<2048ORTE>8191THENG
    OTO3230
0A  3170 IFNS<2048ORNS>8191THENG
    OTO3230
A6  3180 IFNS+TE-TS>8191THENPRIN
    T"[DOWN,WHITE]NEW START ADDR
    ESS TOO HIGH":GOSUB3240:GOTO
    3100
46  3190 IFTE-TS>2000THENPRINT"[
    DOWN,WHITE]TEXT TOO LONG (LE
    NGTH <= 2000)":GOTO3240
50  3200 PRINT"[DOWN,BLACK]PLEAS
    E WAIT":POKE53280,0
EA  3210 K=TE-TS:FORL=0TOK:POKE1
    0240+L,PEEK(TS+L):NEXT:FORL=
    0TOK
DE  3220 POKENS+L,PEEK(10240+L):
    NEXT:POKE53280,11:GOTO1720
8C  3070 IFZ-2^(7-X)=>0THENZ=Z-2
    ^(7-X):POKE1024+W*8+Y*40+X,2
    24
28  3080 IFZZ-2^(7-X)=>0THENZZ=Z
    Z-2^(7-X):POKE1344+W*8+Y*40+
    X,224
D3  3090 NEXT:NEXT:NEXT:PRINT:PO
    KE53280,11:GOTO2550
B5  3100 PRINT"[CLR,DOWN2,C4]"TA
    B(14)"REPLACE TEXT":PRINTTAB
    (14)"[CT12]"
E4  3110 INPUT"[DOWN2,C4]GIVE TE
    XT START ADDRESS";TS
77  3120 INPUT"[DOWN]GIVE  TEXT
     END ADDRESS";TE
1B  3130 INPUT"[DOWN]GIVE NEW  S
    TART ADDRESS";NS
C3  3140 IFTE<TSTHENPRINT"[DOWN,
    WHITE]END SHOULD BE HIGHER T
    HAN START":GOTO3240
2A  3150 IFTS<2048ORTS>8191THENG
    OTO3230
9C  3160 IFTE<2048ORTE>8191THENG
    OTO3230
0A  3170 IFNS<2048ORNS>8191THENG
    OTO3230
A6  3180 IFNS+TE-TS>8191THENPRIN
    T"[DOWN,WHITE]NEW START ADDR
    ESS TOO HIGH":GOSUB3240:GOTO
    3100
46  3190 IFTE-TS>2000THENPRINT"[
    DOWN,WHITE]TEXT TOO LONG (LE
    NGTH <= 2000)":GOTO3240
```

```
50  3200 PRINT"[DOWN,BLACK]PLEAS
    E WAIT":POKE53280,0
EA  3210 K=TE-TS:FORL=0TOK:POKE1
    0240+L,PEEK(TS+L):NEXT:FORL=
    0TOK
DE  3220 POKENS+L,PEEK(10240+L):
    NEXT:POKE53280,11:GOTO1720
06  3230 PRINT"[WHITE,DOWN]TEXT
    MUST BE BETWEEN 2048 AND 819
    2 !"
B5  3240 FORL=1TO2000:NEXT:GOTO3
    100
```

PROGRAM: M/C MAKER

```
24  10 BL=33   :LN=50   :SA=4915
    2
C0  20 FOR L=0 TO BL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A
B7  30 POKE53280,A:POKE SA+L*16+
    D,A:NEXT D
73  40 READ A:IF A><CX THENPRINT
    "ERROR IN LINE";LN+(L*10):ST
    OP
C4  45 NEXT L:SYS49512
39  50 DATA 120,169,31,141,13,22
    0,141,13,221,173,13,220,173,
    13,221,173,2055
13  60 DATA 17,208,41,127,141,17
    ,208,169,198,141,20,3,169,19
    2,141,21,1813
64  70 DATA 3,169,1,141,26,208,1
    69,66,141,18,208,88,96,169,2
    00,141,1844
02  80 DATA 22,208,169,211,141,2
    0,3,169,192,141,21,3,169,251
    ,141,18,1879
83  90 DATA 208,169,1,141,25,208
    ,76,49,234,169,199,141,22,20
    8,169,198,2217
6E  100 DATA 141,20,3,169,192,14
    1,21,3,169,66,141,18,208,169
    ,1,141,1603
F6  110 DATA 25,208,234,234,234,
    174,131,3,232,224,1,240,6,14
    2,131,3,2222
0E  120 DATA 76,49,234,162,0,142
    ,131,3,174,132,3,202,224,191
    ,208,3,1934
C4  130 DATA 76,140,192,142,22,2
    08,142,132,3,76,49,234,162,1
    99,142,132,2051
B1  140 DATA 3,160,0,185,1,4,153
    ,0,4,185,41,4,153,40,4,200,1
    137
60  150 DATA 192,40,240,3,76,147
    ,192,174,133,3,189,0,8,141,3
    9,4,1581
D8  160 DATA 105,127,141,79,4,23
    4,234,234,234,234,232,142,13
    3,3,169,199,2504
FE  170 DATA 141,74,192,76,219,1
    92,160,2,136,208,253,169,21,
    141,24,208,2216
17  180 DATA 76,45,192,169,25,14
    1,24,208,76,73,192,174,133,3
    ,224,0,1755
B7  190 DATA 240,3,76,49,234,174
    ,172,192,232,224,12,208,2,16
    2,8,142,2130
94  200 DATA 172,192,76,49,234,1
    20,169,31,141,13,220,141,13,
    221,173,13,1978
09  210 DATA 220,173,13,221,173,
    17,208,41,127,141,17,208,169
    ,34,141,20,1923
E0  220 DATA 3,169,193,141,21,3,
    169,1,141,26,208,169,50,141,
    18,208,1661
7B  230 DATA 88,96,169,21,141,24
    ,208,169,62,141,20,3,169,193
    ,141,21,1666
```

```
26  240 DATA 3,169,186,141,18,20
    8,169,1,141,25,208,76,49,234
    ,234,234,2096
B8  250 DATA 234,234,234,234,234
    ,234,234,169,25,141,24,208,2
    38,39,208,169,2859
9C  260 DATA 34,141,20,3,169,193
    ,141,21,3,169,50,141,18,208,
    169,1,1481
52  270 DATA 141,25,208,76,49,23
    4,0,0,169,0,133,250,169,192,
    133,251,2030
8C  280 DATA 169,0,133,174,133,1
    93,169,192,133,175,133,194,1
    69,104,133,252,2456
F5  290 DATA 169,193,133,253,160
    ,0,177,250,145,174,230,250,2
    08,2,230,251,2825
8A  300 DATA 230,174,208,2,230,1
    75,165,250,197,252,208,234,1
    65,251,197,253,3191
49  310 DATA 208,228,169,3,133,1
    87,169,194,133,188,169,3,133
    ,183,169,0,2269
AB  320 DATA 133,185,160,0,185,2
    11,193,240,6,32,210,255,200,
    208,245,32,2495
20  330 DATA 207,255,240,251,201
    ,49,240,4,201,56,48,230,41,1
    5,133,186,2357
A8  340 DATA 76,234,245,147,17,1
    7,73,78,80,85,84,32,68,69,86
    ,73,1464
30  350 DATA 67,69,32,78,85,77,6
    6,69,82,13,17,67,65,83,61,49
    ,980
78  360 DATA 32,47,32,68,73,83,7
    5,61,32,56,32,79,82,32,57,58
    ,899
90  370 DATA 45,32,0,77,47,67,0,
    0,0,0,0,0,0,0,0,0,268
AD  380 DATA 0,0,0,255,255,255,2
    55,0,0,0,0,0,0,0,0,0,1020
```

PROGRAM: CHARACTER MAKER

```
E9  10 BL=139  :LN=50   :SA=3686
    4
C0  20 FOR L=0 TO BL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A
B7  30 POKE53280,A:POKE SA+L*16+
    D,A:NEXT D
73  40 READ A:IF A><CX THENPRINT
    "ERROR IN LINE";LN+(L*10):ST
    OP
C5  45 NEXT L:SYS38920
72  50 DATA 56,108,68,146,186,17
    0,162,162,124,254,198,198,25
    4,254,198,102,2640
58  60 DATA 252,254,198,198,198,
    252,252,198,48,124,110,198,1
    98,192,192,192,3056
BA  70 DATA 248,252,12,54,54,54,
    54,102,60,126,102,192,192,22
    4,112,96,1934
5D  80 DATA 252,254,198,198,192,
    192,96,96,28,62,118,102,96,1
    92,192,192,2460
B5  90 DATA 192,192,192,192,96,9
    6,96,124,24,24,0,24,24,24,24
    ,24,1348
D8  100 DATA 12,12,0,12,12,12,12
    ,6,192,192,192,192,192,204,2
    20,248,1710
61  110 DATA 192,192,96,96,96,48
    ,48,48,130,198,254,254,214,1
    98,198,198,2460
28  120 DATA 220,254,230,198,198
    ,198,102,102,60,126,102,102,
    198,198,198,198,2684
34  130 DATA 240,252,206,198,198
    ,102,110,124,60,126,102,102,
    198,198,198,198,2612
```

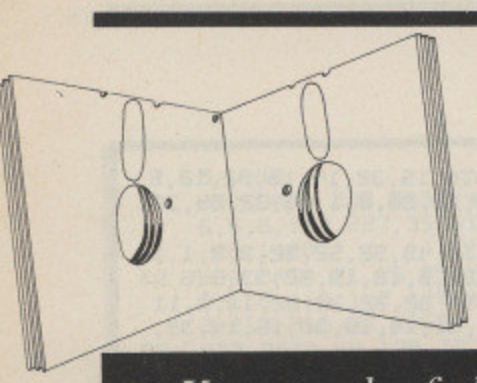
```
D1  140 DATA 248,252,12,198,198,
    198,204,252,56,124,238,198,1
    92,224,120,28,2742
44  150 DATA 96,96,240,120,96,96
    ,96,192,198,198,198,198,198,
    198,198,198,2616
C9  160 DATA 198,198,198,198,198
    ,108,108,108,12,14,6,102,102
    ,198,198,198,2144
55  170 DATA 198,198,198,198,198
    ,108,108,56,198,198,198,198,
    198,102,126,30,2510
16  180 DATA 254,254,6,6,14,12,2
    8,56,60,60,48,48,48,48,48,48
    ,1038
B6  190 DATA 120,252,238,198,192
    ,192,96,104,60,60,12,12,12,1
    2,12,12,1584
83  200 DATA 16,16,56,56,124,124
    ,254,254,16,16,16,48,48,112,
    112,254,1522
3D  210 DATA 0,0,0,0,0,0,0,0,60,
    60,60,60,60,60,60,24,444
CF  220 DATA 238,238,34,102,204,
    0,0,0,54,54,54,54,54,236,252
    ,126,1700
54  230 DATA 60,124,254,222,216,
    248,120,124,98,146,150,102,1
    2,12,24,24,1936
73  240 DATA 48,120,72,216,208,1
    44,240,96,56,56,8,24,48,0,0,
    0,1336
BE  250 DATA 28,56,96,96,192,192
    ,192,192,112,56,12,12,6,6,6,
    6,1260
FB  260 DATA 0,0,0,0,0,16,84,56,
    0,0,0,0,24,24,24,126,354
A6  270 DATA 0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,254,254
F8  280 DATA 0,0,0,0,0,0,0,0,2,2
    ,6,6,12,12,24,24,88
C9  290 DATA 124,186,198,198,198
    ,198,130,0,0,2,6,6,6,6,2,0,1
    260
6E  300 DATA 124,58,6,6,6,6,58,1
    24,124,58,6,6,6,6,58,124,776
B7  310 DATA 0,130,198,198,198,1
    98,186,124,124,184,192,192,1
    92,192,184,124,2616
9D  320 DATA 124,184,192,192,192
    ,192,184,124,124,58,6,6,6,6,
    2,0,1592
0C  330 DATA 124,186,198,198,198
    ,198,186,124,124,186,198,198
    ,198,198,186,124,2824
C0  340 DATA 0,0,0,56,56,56,0,0,
    0,0,0,56,56,56,0,0,336
AC  350 DATA 0,0,0,0,2,12,48,192
    ,0,0,0,0,0,124,124,0,502
5F  360 DATA 0,0,0,0,128,96,24,6
    ,124,254,198,198,102,6,12,28
    ,1176
F0  370 DATA 0,0,0,0,1,2,63,64,0
    ,0,0,255,37,69,250,5,746
C4  380 DATA 0,0,0,255,5,4,2,255
    ,0,0,0,224,24,132,130,125,11
    56
74  390 DATA 0,0,0,0,0,0,0,224,2
    24,24,4,2,2,1,1,1,483
C0  400 DATA 0,0,0,0,0,0,0,0,0,0
    ,0,0,0,1,1,1,3
7B  410 DATA 15,48,64,128,128,0,
    0,0,15,63,112,96,192,192,192
    ,192,1437
0D  420 DATA 128,128,126,124,120
    ,112,96,0,255,127,191,95,175
    ,87,171,85,2020
DF  430 DATA 255,254,255,252,248
    ,225,196,144,137,82,34,34,33
    ,0,0,0,2149
1A  440 DATA 146,82,82,82,142,0,
    0,0,224,144,225,161,145,3,3,
```

```
   3,1442
3A 450 DATA 232,249,145,147,18,
   6,6,6,199,227,35,51,19,27,27
   ,26,1420
E8 460 DATA 31,25,25,153,169,16
   9,169,169,142,140,140,204,21
   2,213,213,85,2259
96 470 DATA 49,120,72,204,132,1
   34,134,134,240,240,144,152,1
   36,141,141,141,2314
80 480 DATA 49,120,72,204,132,1
   34,134,134,241,248,140,140,1
   32,140,140,248,2408
83 490 DATA 252,132,128,128,128
   ,128,136,248,248,252,252,132
   ,164,148,180,132,2788
9A 500 DATA 7,24,32,70,76,140,1
   30,130,224,24,4,50,98,97,1,6
   5,1172
3D 510 DATA 1,3,6,15,21,32,106,
   128,255,255,170,255,85,0,170
   ,0,1502
35 520 DATA 128,192,160,240,88,
   4,170,1,6,3,25,62,88,67,12,1
   2,1258
49 530 DATA 0,0,112,248,136,200
   ,224,96,0,0,0,1,1,193,192,64
   ,1467
02 540 DATA 31,106,177,49,255,3
   6,4,4,0,192,160,144,240,144,
   0,0,1542
DD 550 DATA 31,17,17,245,128,18
   1,128,245,0,0,0,224,32,160,3
   2,224,1664
D4 560 DATA 63,64,88,156,150,14
   4,140,128,248,4,52,114,210,1
   8,98,2,1679
D3 570 DATA 1,1,0,0,0,0,127,62,
   0,192,241,39,96,255,128,65,1
   207
EE 580 DATA 0,0,128,144,240,208
   ,192,124,7,4,9,12,63,127,231
   ,199,1688
66 590 DATA 128,0,64,128,192,22
   5,99,48,30,33,73,112,20,218,
   66,130,1566
19 600 DATA 3,2,4,132,72,49,2,6
   ,0,196,38,69,138,140,76,76,1
   003
C8 610 DATA 0,0,0,0,0,0,0,0,0,0
   ,0,12,28,60,108,204,412
D4 620 DATA 0,0,0,48,24,29,23,1
   8,0,0,0,103,194,194,66,66,76
   5
D5 630 DATA 0,0,0,15,27,33,32,6
   4,255,131,131,131,255,254,25
   3,253,1834
C8 640 DATA 255,255,255,255,255
   ,63,223,223,255,255,192,128,
   128,143,146,146,3177
08 650 DATA 255,255,0,0,0,255,2
   25,225,255,255,3,1,1,241,73,
   73,2117
F0 660 DATA 255,112,32,32,33,33
   ,33,164,255,0,105,155,10,2,2
   ,130,1353
7D 670 DATA 255,0,207,100,36,36
   ,36,36,255,0,223,218,82,2,2,
   2,1490
74 680 DATA 255,7,195,193,65,0,
   0,0,131,68,68,68,68,68,68,68
   ,1322
1D 690 DATA 162,162,170,186,146
   ,68,108,56,102,102,6,6,12,12
   ,12,24,1334
0E 700 DATA 102,102,102,102,108
   ,12,124,120,192,192,198,198,
   198,204,252,120,2326
B6 710 DATA 102,102,198,198,198
   ,198,254,252,192,192,192,192
   ,198,198,254,124,3044
FE 720 DATA 120,248,96,48,48,48
   ,48,48,192,206,222,198,198,2
   06,252,120,2298
53 730 DATA 126,118,102,102,102
   ,102,102,102,24,24,48,48,48,
   48,112,96,1304
E9 740 DATA 6,6,6,198,198,102,1
   26,60,240,248,120,108,108,10
   8,110,102,1846
12 750 DATA 48,48,48,96,96,98,2
   54,252,102,6,6,6,12,12,12,24
   ,1120
BE 760 DATA 102,6,6,6,12,12,28,
   24,198,198,198,198,198,102,1
   24,56,1468
AF 770 DATA 120,96,48,48,48,48,
   48,48,198,198,214,222,222,22
   0,254,114,2146
EC 780 DATA 120,108,108,102,102
   ,198,198,198,12,14,6,6,198,2
   38,124,56,1788
D9 790 DATA 192,192,192,198,198
   ,110,124,56,198,198,198,198,
   198,230,126,62,2670
EA 800 DATA 108,108,56,56,56,56
   ,16,16,198,198,198,198,214,2
   22,246,102,2048
70 810 DATA 108,108,198,198,198
   ,198,198,198,6,6,6,6,12,12,1
   2,24,1488
BF 820 DATA 112,96,224,192,192,
   192,254,254,48,48,48,48,48,4
   8,60,60,1924
C1 830 DATA 248,240,96,96,96,12
   2,254,204,12,12,12,12,12,12,
   60,60,1548
DD 840 DATA 56,56,56,56,56,56,5
   6,56,254,112,112,48,48,48,16
   ,16,1102
AF 850 DATA 0,0,0,0,0,0,0,0,24,
   24,24,24,24,0,24,24,168
16 860 DATA 0,0,0,0,0,0,0,0,238
   ,252,126,206,216,216,216,216
   ,1686
C2 870 DATA 60,62,54,54,246,238
   ,252,248,48,48,96,64,204,210
   ,146,140,2170
2D 880 DATA 224,162,182,148,156
   ,156,222,114,0,0,0,0,0,0,0,0
   ,1364
D9 890 DATA 192,192,192,192,96,
   96,56,28,6,6,6,6,12,12,56,11
   2,1260
FC 900 DATA 254,56,84,16,0,0,0,
   0,126,24,24,24,0,0,0,0,608
4B 910 DATA 0,0,0,56,56,8,24,48
   ,254,0,0,0,0,0,0,0,446
89 920 DATA 0,0,0,0,0,56,56,56,
   48,48,96,96,192,192,128,128,
   1096
E9 930 DATA 130,198,198,198,198
   ,198,186,124,2,6,6,6,6,6,2,0
   ,1464
84 940 DATA 184,192,192,192,192
   ,192,184,124,58,6,6,6,6,6,58
   ,124,1722
1F 950 DATA 58,6,6,6,6,6,2,0,58
   ,6,6,6,6,6,58,124,360
97 960 DATA 186,198,198,198,198
   ,198,186,124,2,6,6,6,6,6,2,0
   ,1520
83 970 DATA 186,198,198,198,198
   ,198,186,124,58,6,6,6,6,6,58
   ,124,1756
83 980 DATA 0,0,56,56,56,0,0,0,
   0,0,56,56,56,8,24,48,416
E8 990 DATA 192,48,12,2,0,0,0,0
   ,0,124,124,0,0,0,0,0,502
0E 1000 DATA 6,24,96,128,0,0,0,
   0,24,24,24,24,24,0,24,24,422

26 1010 DATA 134,135,128,225,25
   5,30,0,0,244,10,249,141,119,
   219,216,112,2217
3E 1020 DATA 2,26,2,2,255,255,0
   ,0,4,196,15,24,247,237,13,7,
   1285
5F 1030 DATA 28,2,129,199,126,1
   88,128,0,1,1,1,0,0,0,0,0,803

AE 1040 DATA 0,0,0,128,128,64,4
   8,15,1,1,1,2,2,4,24,224,642
CF 1050 DATA 0,0,0,0,0,0,0,0,19
   2,192,192,192,96,112,63,15,1
   054
41 1060 DATA 0,32,80,40,84,42,1
   28,128,170,84,168,80,160,64,
   128,0,1388
4B 1070 DATA 65,4,16,1,4,0,1,0,
   0,0,0,0,0,0,0,0,91
DE 1080 DATA 0,0,0,0,0,0,0,0,3,
   3,3,1,1,1,0,0,12
FD 1090 DATA 6,6,6,10,11,145,24
   1,224,26,26,26,18,50,34,226,
   199,1254
8F 1100 DATA 169,169,201,201,20
   1,201,73,95,85,85,101,100,10
   0,100,36,174,2091
47 1110 DATA 134,134,134,132,20
   4,72,120,49,141,141,141,136,
   152,144,240,240,2314
F6 1120 DATA 134,134,134,132,20
   4,72,120,49,176,176,152,152,
   152,152,140,207,2286
26 1130 DATA 248,136,128,128,12
   8,128,132,252,252,132,180,13
   2,252,228,252,248,2956
BF 1140 DATA 153,176,172,67,64,
   32,24,7,145,9,57,194,2,4,24,
   224,1354
92 1150 DATA 128,106,32,21,15,6
   ,3,1,0,170,0,85,255,170,255,
   255,1502
98 1160 DATA 1,170,4,88,240,160
   ,192,128,24,8,16,24,24,0,24,
   24,1127
98 1170 DATA 65,130,12,0,1,2,1,
   3,224,240,208,208,64,32,16,4
   8,1254
BA 1180 DATA 4,4,4,4,4,4,4,31,0
   ,0,0,0,96,252,62,20,489
EC 1190 DATA 17,17,17,17,21,85,
   255,255,0,0,0,0,0,96,192,240
   ,1212
07 1200 DATA 130,129,136,71,68,
   36,16,15,130,2,34,196,68,72,
   16,224,1343
69 1210 DATA 83,140,139,143,139
   ,128,65,62,34,147,253,255,25
   3,129,0,0,1970
16 1220 DATA 226,33,33,49,25,1,
   130,124,232,111,13,29,56,112
   ,97,49,1320
83 1230 DATA 185,156,206,195,19
   3,192,128,224,2,2,2,1,129,19
   2,0,0,1807
DF 1240 DATA 4,1,0,8,145,243,17
   ,62,52,4,132,140,24,240,0,0,
   1072
1A 1250 DATA 1,3,6,12,31,48,96,
   240,140,12,12,12,252,12,12,3
   0,919
BB 1260 DATA 32,32,32,32,65,65,
   65,227,132,132,132,132,8,8,8
   ,156,1258
52 1270 DATA 64,64,128,128,143,
   198,102,60,253,254,255,255,2
   55,255,255,255,2924
E7 1280 DATA 223,55,255,127,127
   ,127,127,255,143,128,255,255
   ,248,240,224,255,3044
FF 1290 DATA 255,0,255,255,0,0,
   0,255,241,1,255,255,31,15,7,
   255,2080
55 1300 DATA 252,164,32,32,32,3
   3,113,249,98,18,10,10,10,10,
   139,113,1315
18 1310 DATA 37,47,37,36,36,36,
```

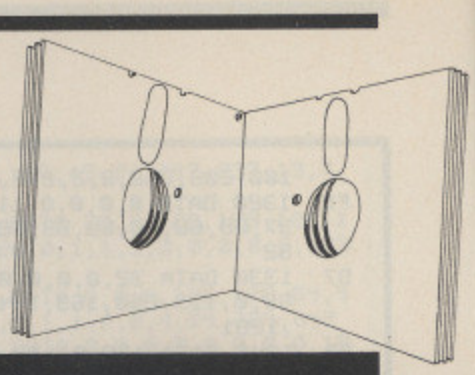
```
    100,206,2,2,2,2,2,2,2,7,556
FD  1320 DATA 0,0,0,0,117,39,37,
    37,68,68,68,68,68,68,68,56,7
    62
D7  1330 DATA 32,0,0,0,0,0,0,0,1
    69,0,133,250,169,144,133,251
    ,1281
7B  1340 DATA 169,0,133,174,133,
    193,169,32,133,175,133,194,1
    69,8,133,252,2200
98  1350 DATA 169,152,133,253,16
    0,0,177,250,145,174,230,250,
    208,2,230,251,2784
A2  1360 DATA 230,174,208,2,230,
    175,165,250,197,252,208,234,
    165,251,197,253,3191
43  1370 DATA 208,228,169,163,13
    3,187,169,152,133,188,169,5,
    133,183,169,0,2389
F5  1380 DATA 133,185,160,0,185,
    115,152,240,6,32,210,255,200
    ,208,245,32,2358
68  1390 DATA 207,255,240,251,20
    1,49,240,4,201,56,48,230,41,
    15,133,186,2357
B0  1400 DATA 76,234,245,147,17,
    17,73,78,80,85,84,32,68,69,8
    6,73,1464
28  1410 DATA 67,69,32,78,85,77,
    66,69,82,13,17,67,65,83,61,4
    9,980
40  1420 DATA 32,47,32,68,73,83,
    75,61,32,56,32,79,82,32,57,5
    8,899
AE  1430 DATA 45,32,0,67,72,65,8
    2,83,0,0,0,0,0,0,0,0,446
95  1440 DATA 0,0,0,255,255,255,
    255,0,0,0,0,0,0,0,0,0,1020
```

PROGRAM: TEXT MAKER

```
7D  10 BL=75   :LN=50   :SA=3814
    4
C0  20 FOR L=0 TO BL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A
B7  30 POKE53280,A:POKE SA+L*16+
    D,A:NEXT D
73  40 READ A:IF A><CX THENPRINT
    "ERROR IN LINE";LN+(L*10):ST
    OP
CD  45 NEXT L:SYS39176
10  50 DATA 20,8,5,32,13,5,19,19
    ,1,7,5,32,3,15,14,19,217
5E  60 DATA 20,18,21,3,20,9,15,1
    4,32,11,9,20,32,23,1,19,267
E1  70 DATA 32,23,18,9,20,20,5,1
    4,32,2,25,32,6,18,1,14,271
E5  80 DATA 11,32,22,1,14,32,20,
    9,7,7,5,12,5,14,32,9,232
D3  90 DATA 14,32,13,1,18,3,8,32
    ,49,57,56,56,32,6,15,18,410
35  100 DATA 32,20,8,5,32,19,5,1
    8,9,15,21,19,32,21,19,5,280
14  110 DATA 18,39,19,32,7,21,9,
    4,5,46,46,46,13,3,11,32,351
D0  120 DATA 1,12,12,15,23,19,32
    ,1,14,25,15,14,5,32,20,15,25
    5
FD  130 DATA 32,3,18,5,1,20,5,32
    ,20,8,5,9,18,32,15,23,246
4C  140 DATA 14,32,19,13,15,15,2
    0,8,45,19,3,18,15,12,12,9,26
    9
88  150 DATA 14,7,44,32,9,14,20,
    5,18,18,21,16,20,32,4,18,292

19  160 DATA 9,22,5,14,32,13,5,1
    9,19,1,7,5,19,32,23,8,233
D3  170 DATA 9,3,8,32,23,9,12,12
    ,32,5,22,5,14,32,18,21,257
2F  180 DATA 14,32,1,12,15,14,7,
    19,9,4,5,32,1,14,15,20,214
67  190 DATA 8,5,18,32,16,18,15,
    7,18,1,13,46,46,46,46,46,381

D9  200 DATA 46,46,46,46,46,46,3
    2,20,8,5,19,5,32,13,5,19,434

20  210 DATA 19,1,7,5,19,32,3,15
    ,14,20,1,9,14,32,21,19,231
0A  220 DATA 5,18,32,4,5,6,9,14,
    5,4,32,7,18,1,16,8,184
58  230 DATA 9,3,19,32,1,14,4,32
    ,3,8,1,18,1,3,20,5,173
57  240 DATA 18,19,44,23,8,9,3,8
    ,32,1,18,5,32,5,1,19,245
9D  250 DATA 9,12,25,32,4,5,19,9
    ,7,14,5,4,32,21,19,9,226
E3  260 DATA 14,7,32,20,8,5,32,3
    ,8,1,18,1,3,20,5,18,195
80  270 DATA 32,5,4,9,20,15,18,3
    2,32,32,32,32,32,32,32,32,39
    1
E5  280 DATA 32,32,32,32,32,20,5
    ,24,20,19,32,3,1,14,32,2,332
6B  290 DATA 5,32,23,18,9,20,20,
    5,14,32,21,19,9,14,7,32,280
60  300 DATA 20,8,5,32,19,9,13,1
    6,12,5,32,20,5,24,20,32,272
59  310 DATA 5,4,9,20,15,18,32,4
    6,46,46,46,46,46,46,46,46,51
    7
59  320 DATA 32,1,32,6,5,23,32,5
    ,24,1,13,16,12,5,19,32,258
B6  330 DATA 15,6,32,23,8,1,20,3
    2,9,19,32,16,15,19,19,9,275
C2  340 DATA 2,12,5,32,23,9,20,8
    ,32,13,3,11,32,58,32,32,324
C9  350 DATA 64,65,66,67,68,32,3
    ,8,9,12,4,9,19,8,32,16,482
D3  360 DATA 9,3,20,21,18,5,19,3
    2,46,46,46,46,46,46,77,78,55
    8
6E  370 DATA 79,80,81,82,83,84,8
    5,86,87,88,32,32,122,123,124
    ,125,1393
FE  380 DATA 126,32,32,112,113,1
    14,115,116,112,113,32,32,73,
    74,32,54,1282
40  390 DATA 52,32,32,32,32,12,1
    5,7,15,19,32,46,46,46,46,46,
    510
1D  400 DATA 46,46,46,46,32,75,7
    6,75,76,75,76,92,93,94,92,93
    ,1133
7C  410 DATA 94,92,93,93,93,94,9
    2,93,94,92,93,94,75,76,75,76
    ,1419
3A  420 DATA 75,76,32,32,2,1,19,
    45,18,5,12,9,5,6,32,7,376
55  430 DATA 18,1,16,8,9,3,19,32
    ,46,46,46,46,46,46,46,32,460

8F  440 DATA 95,96,97,98,99,32,3
    2,32,100,101,32,32,102,103,3
    2,32,1115
C9  450 DATA 32,104,105,106,32,3
    2,32,107,108,109,110,111,32,
    32,32,32,1116
05  460 DATA 32,90,91,32,32,32,3
    2,13,15,18,5,32,3,8,9,12,456

A6  470 DATA 4,9,19,8,32,19,20,2
    1,6,6,32,46,46,46,46,46,406
0C  480 DATA 32,32,32,32,32,20,8
    ,9,19,32,13,5,19,19,1,7,312
77  490 DATA 5,32,3,15,14,20,1,9
    ,14,19,32,49,48,50,52,32,395

53  500 DATA 3,8,1,18,1,3,20,5,1
    8,19,44,2,21,20,32,3,218
73  510 DATA 1,14,32,2,5,32,5,14
    ,12,1,18,7,5,4,32,20,204
96  520 DATA 15,32,14,15,32,12,5
    ,19,19,32,20,8,1,14,32,54,32
    4
76  530 DATA 49,52,52,32,3,8,1,1
    8,1,3,20,5,18,19,32,33,346
F3  540 DATA 32,32,32,32,13,3,11
    ,32,18,21,14,19,32,15,14,32,
    352
E1  550 DATA 73,74,32,54,52,32,1
    ,14,4,32,73,74,32,49,50,56,7
    02
B4  560 DATA 32,40,9,14,32,54,52
    ,32,13,15,4,5,41,32,23,9,407

7C  570 DATA 20,8,32,117,118,32,
    15,18,32,119,120,121,32,46,4
    6,46,922
3A  580 DATA 46,46,46,46,46,32,5
    ,24,16,5,3,20,32,13,15,18,41
    3
5B  590 DATA 5,32,16,18,15,7,18,
    1,13,19,32,21,19,9,14,7,246
DA  600 DATA 32,13,3,11,32,9,14,
    32,20,8,5,32,14,5,1,18,249
33  610 DATA 32,6,21,20,21,18,5,
    32,33,32,32,32,32,32,32,32,4
    12
30  620 DATA 32,20,8,5,32,13,5,1
    9,19,1,7,5,32,3,15,14,230
66  630 DATA 19,20,18,21,3,20,9,
    15,14,32,11,9,20,32,32,0,275

57  640 DATA 32,49,57,56,56,32,6
    ,18,1,14,11,32,22,1,14,32,43
    3
C2  650 DATA 20,9,7,7,5,12,5,14,
    32,38,32,25,15,21,18,32,292
54  660 DATA 3,15,13,13,15,4,15,
    18,5,39,19,32,19,5,18,9,242
A1  670 DATA 15,21,19,32,21,19,5
    ,18,39,19,32,7,21,9,4,5,286
9F  680 DATA 32,32,32,32,32,32,3
    2,32,32,32,32,32,32,32,32,32
    ,512
0C  690 DATA 0,0,0,0,0,0,0,0,169
    ,0,133,250,169,149,133,251,1
    254
11  700 DATA 169,0,133,174,133,1
    93,169,8,133,175,133,194,169
    ,8,133,252,2176
09  710 DATA 169,153,133,253,160
    ,0,177,250,145,174,230,250,2
    08,2,230,251,2785
6F  720 DATA 230,174,208,2,230,1
    75,165,250,197,252,208,234,1
    65,251,197,253,3191
56  730 DATA 208,228,169,163,133
    ,187,169,153,133,188,169,4,1
    33,183,169,0,2389
86  740 DATA 133,185,160,0,185,1
    15,153,240,6,32,210,255,200,
    208,245,32,2359
3F  750 DATA 207,255,240,251,201
    ,49,240,4,201,56,48,230,41,1
    5,133,186,2357
CB  760 DATA 76,234,245,147,17,1
    7,73,78,80,85,84,32,68,69,86
    ,73,1464
06  770 DATA 67,69,32,78,85,77,6
    6,69,82,13,17,67,65,83,61,49
    ,980
EA  780 DATA 32,47,32,68,73,83,7
    5,61,32,56,32,79,82,32,57,58
    ,899
BD  790 DATA 45,32,0,84,69,88,84
    ,0,0,0,0,0,0,0,0,0,402
F3  800 DATA 0,0,0,255,255,255,2
    55,0,0,0,0,0,0,0,0,0,1020
```



# Disk Cat



*Keep track of all your files with this simple but essential cataloguing system*

This double program consists of Disk Cat, which creates a file and File Editor, which allows you to make any changes to it.

With Disk Cat you can create a sequential file on disk containing a program title and a disk number, such as PACMAN 27A. You will be able to print out an alphabetical double-column master list or small notebook size pages to cut out and keep in a ring file.

You will see from your list that PACMAN is on disk number 27, side A. If you do not double your disks then the A or B need not be included in the disk number.

Before using Disk Cat, I use a directory logging program to make a printout of all my program titles and allocate a number to each disk.

When running Disk Cat you will be asked to wait while the sort routine is poked into memory. When prompted, press the space bar and you will be presented with a menu.

If you are creating a new file select option A. The screen will clear and an instruction box will appear at the top of the display. This tells you how to close a file – the only program title you can't use is 'END', on disk number '0'.

Now enter the title (16 characters maximum), press return and enter the disk number (four characters maximum). When you have entered all the titles and disk numbers, simply type END, press the return key, type 0 for the disk number, press return again and the file will be properly closed.

Selecting option B will allow you to extend an existing file. The drive will run for a few seconds as it places the read/write head at the end of the file. Then proceed as in option A.

Option C will read your file into memory and automatically sort it into alphabetical order and then you will be presented with a new menu.

When you choose option D the screen will clear and you'll be asked to enter the date. You should have your printer switched on at this stage. The date should be entered in the form DD/MM/YY. Press return and in a few moments you will have a double column, alphabetical master list of all your program titles and disk numbers. Disk Cat has been tested on the MPS 801,803 and MPS 1000.

Option E will display a submenu, select P for printer or S for screen. You will then be asked to enter a category. If you enter C, every title beginning with C will be listed either to the screen or the printer. If you enter two letters, eg. CL, then only the titles beginning wth CL will be listed.

If you originally selected P the output will be sent to the printer and this will give you a 125×90mm page which can be cut out and put into a small ring file.

If you make a mistake entering a title or disk number don't panic, you can use File Editor to make any corrections, changes or deletions. It will then scratch the old file from disk and replace it with the update. File Editor has on-screen instructions and you should find it simple to follow.

PROGRAM: DISK CAT

```
3C  10 POKE649,0
DF  20 PRINT"[CLR]":POKE53280,15
    :POKE53281,11:PRINTTAB(11)"[
    RVSON,C8,CA,S*17,CS]"
0D  30 PRINTTAB(11)"[RVSON,S-,SP
    C17,S-]"
EC  40 PRINTTAB(11)"[RVSON,S-,SP
    C4]DISK CAT[SPC5,S-]"
A1  50 PRINTTAB(11)"[RVSON,S-,SP
    C17,S-]"
EE  60 PRINTTAB(11)"[RVSON,S-,SP
    C7]BY[SPC8,S-]"
F5  70 PRINTTAB(11)"[RVSON,S-,SP
    C17,S-]"
DA  80 PRINTTAB(11)"[RVSON,S-,SP
    C3]JOHN  HYDE[SPC4,S-]"
49  90 PRINTTAB(11)"[RVSON,S-,SP
    C17,S-]"
EB  100 PRINTTAB(11)"[RVSON,CZ,S
    *17,CX]"
06  110 RR=13:C=6:GOSUB1890:GOSU
    B690
B5  120 DIMX$(1000):L$="[RVSON,C
    A,S*8,CS]":C$="[RVSON,SB]DIS
    K CAT[SB]":B$="[RVSON,CZ,S*8
    ,CX]":CR$=CHR$(13)
97  130 SP$="[SPC39]"
FE  140 NA$="TITLE":DN$="DISK#":
    ML$="MASTER LIST":DC$="DISK
    CAT"
46  150 Z=0:FORJ=49152TO49364:
41  160 READA:POKEJ,A:Z=Z+A:NEXT

B4  170 IFZ<>29842THENPRINT"ERRO
    R IN DATA!":END
9D  180 POKE49248,0:POKE649,10:R
    R=13:C=0:GOSUB1890:NN=7:GOSU
    B1900
39  190 RR=16:C=8:GOSUB1890
5D  200 PRINT"[RVSON,CA,S*21,CS]
    "
31  210 PRINTTAB(8)"[RVSON,S-,SP
    C21,S-]"
90  220 PRINTTAB(8)"[RVSON,S-]HI
    T SPACE TO CONTINUE[S-]"
B5  230 PRINTTAB(8)"[RVSON,S-,SP
    C21,S-]"
63  240 PRINTTAB(8)"[RVSON,CZ,S*
    21,CX]"
F5  250 GETA$:IFA$<>CHR$(32)THEN
    250
21  260 GOTO720
E2  270 OPEN15,8,15:OPEN2,8,2,"0
    :DC TITLE,S,W"
22  280 GOTO310
97  290 OPEN15,8,15:OPEN2,8,2,"0
    :DC TITLE,A"
C4  300 RR=11:GOSUB1890
54  310 GOSUB650:PRINTTAB(13)"[R
    VSON,CA,S*11,CS]"
6A  320 PRINTTAB(13)"[RVSON,S-]E
    NTER TITLE[S-]"
8D  330 PRINTTAB(13)"[RVSON,CZ,S
    *11,CX,RVSOFF]"SPC(58):INPUT
    PN$
```

*by J R Hyde*

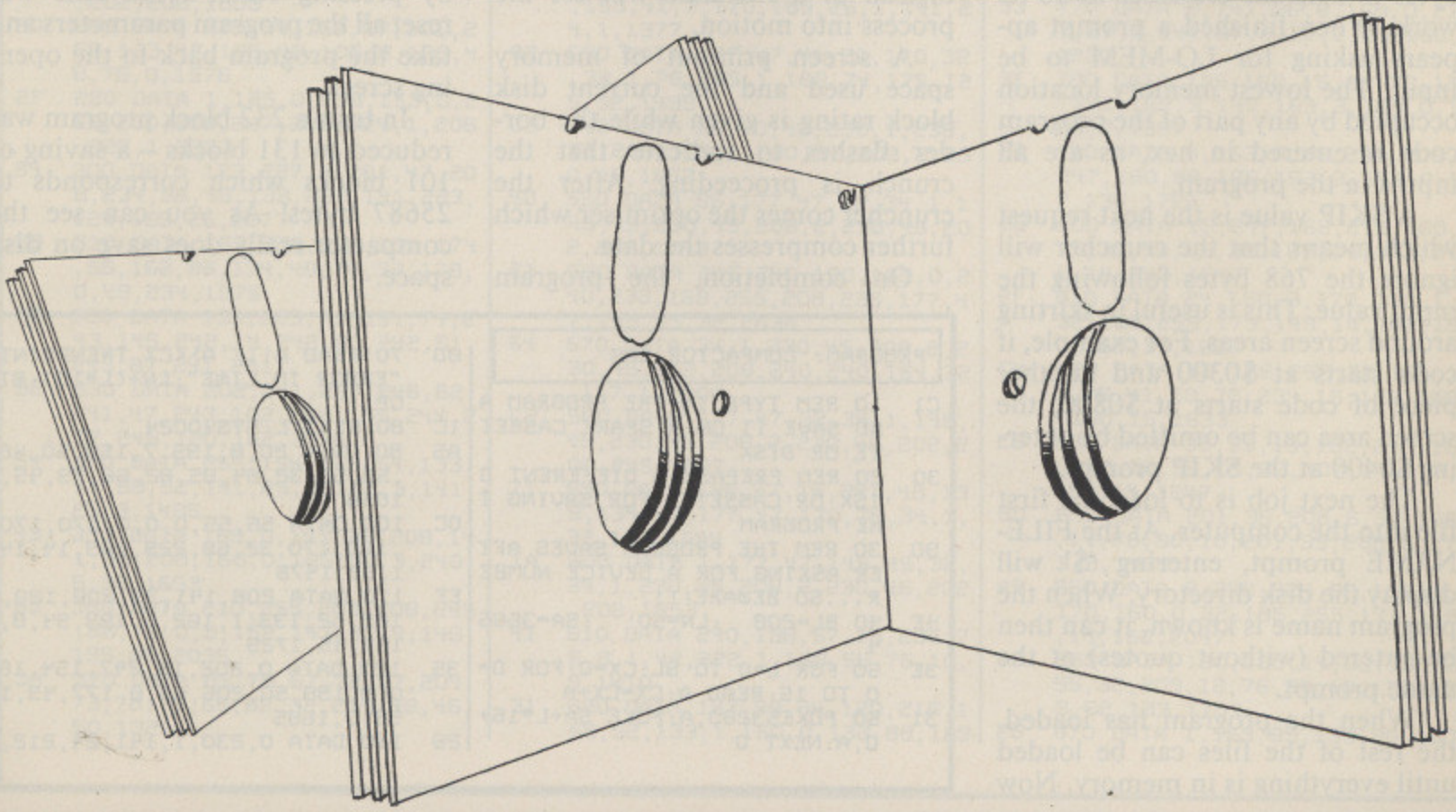
```
91  340 S=LEN(PN$):IFS>16THENRR=
    11:NN=5:GOSUB1900:GOTO1770
DB  350 RR=11:NN=5:GOSUB1900
FF  360 PRINTTAB(9)"[RVSON,CA,S*
    17,CS]"
2D  370 PRINTTAB(9)"[RVSON,S-]EN
    TER DISK NUMBER[S-]"
02  380 PRINTTAB(9)"[RVSON,CZ,S*
    17,CX,RVSOFF]"SPC(56):INPUTN
    $
32  390 SS=LEN(N$):IFSS>4THENRR=
    11:GOSUB1900:GOTO1810
84  400 IFPN$="END"ANDN$="0"THEN
    GOTO440
A8  410 PRINT#2,PN$;CR$;N$
56  420 GOSUB650:NN=13:RR=11:GOS
    UB1900
AC  430 IFPN$<>"END"THEN310
33  440 CLOSE2:CLOSE15:GOTO740
7A  450 RR=10:C=0:GOSUB1890:PRIN
    TTAB(13)"[RVSON,CA,S*12,CS]"
    SPC(26)
A1  460 PRINT"[RVSON,S-]READING
    FILE[S-]"SPC(26)
D4  470 PRINT"[RVSON,CZ,S*12,CX]
    "
DD  480 OPEN15,8,15:OPEN2,8,2,"0
    :DC TITLE,S,R"
2E  490 GOSUB650:I=0:X=0
C3  500 INPUT#2,PN$,N$
A5  510 NU$="0000"+N$:A=LEN(NU$)
    :IFA>=4THENA=4:N$=RIGHT$(NU$
    ,A)
30  520 RS=ST:GOSUB650
DA  530 T$="[SPC20]"
86  540 Q=LEN(PN$):IFQ<=17THENQ=
    21-LEN(PN$)
08  550 T$=LEFT$(T$,Q)
04  560 I=I+1:X=X+1:X$(I)=PN$+T$
    +N$
36  570 IFRS=0THEN500
8F  580 IFRS<>64THENPRINT"STATUS
    =";RS
BE  590 CLOSE2:CLOSE15
AA  600 RR=10:C=0:GOSUB1890:NN=6
    :GOSUB1900
8C  610 PRINTTAB(13)"[RVSON,CA,S
    *12,CS]"SPC(26)
EA  620 PRINT"[RVSON,S-]SORTING
    FILE[S-]"SPC(26)
25  630 PRINT"[RVSON,CZ,S*12,CX]
    "
BA  640 GOSUB1490:GOSUB1900:GOTO
    870
F6  650 INPUT#15,EN,EM$,ET,ES
B1  660 IFEN>0THENPRINTEN,EM$,ET
    ,ES:STOP
2E  670 RETURN
FC  680 PRINTCHR$(147)TAB(15)L$:
    PRINTTAB(15)C$:PRINTTAB(15)B
    $
45  690 PRINT"[RVSON,CA,S*26,CS]
    "
BA  700 PRINTTAB(6)"[RVSON,S-]IN
    ITIALISING...PLEASE WAIT[S-]
    "
45  710 PRINTTAB(6)"[RVSON,CZ,S*
    26,CX]":RETURN
DF  720 PRINTCHR$(147)TAB(55)L$:
    PRINTTAB(15)C$:PRINTTAB(15)B
    $:RR=8:C=9:GOSUB1890
E9  730 GOTO750
44  740 RR=5:C=0:GOSUB1890:NN=18
    :GOSUB1910
CE  750 PRINTTAB(9)"[RVSON,CA,S*
    20,CS]"
05  760 PRINTTAB(9)"[RVSON,S-]A)
     CREATE A NEW FILE[S-]"
BD  770 PRINTTAB(9)"[RVSON,S-,SP
    C20,S-]"
AB  780 PRINTTAB(9)"[RVSON,S-]B)
     ADD TO A FILE[SPC4,S-]"
29  790 PRINTTAB(9)"[RVSON,S-,SP
    C20,S-]"
A2  800 PRINTTAB(9)"[RVSON,S-]C)
     READ A FILE[SPC6,S-]"
05  810 PRINTTAB(9)"[RVSON,CZ,S*
    20,CX]"
54  820 GETA$:IFA$=""THEN820
28  830 IFA$="A"THENGOSUB990:GOT
    O270
6D  840 IFA$="B"THENNN=18:RR=5:G
    OSUB1900:RR=5:GOSUB1890:GOSU
    B1000:GOTO290
C2  850 IFA$="C"THENRR=5:NN=16:G
    OSUB1900:GOTO450
64  860 IFA$<>"A"ORA$<>"B"ORA$<>
    "C"THEN820
E8  870 RR=10:GOSUB1890:PRINTTAB
    (6)"[RVSON,CA,S*27,CS]"
88  880 PRINTTAB(6)"[RVSON,S-]D)
     PRINT MASTER FILE[SPC7,S-]"
4C  890 PRINTTAB(6)"[RVSON,S-,SP
    C27,S-]"
59  900 PRINTTAB(6)"[RVSON,S-]E)
     PRINT ALAPHABETICAL PAGE[S-
    ]"
78  910 PRINTTAB(6)"[RVSON,S-,SP
    C27,S-]"
30  920 PRINTTAB(6)"[RVSON,S-]Z)
     QUIT PROGRAMME[SPC10,S-]"
B4  930 PRINTTAB(6)"[RVSON,CZ,S*
    27,CX]"
1B  940 GETA$:IFA$=""THEN940
51  950 IFA$="D"GOTO1060
E1  960 IFA$="E"GOTO1180
C0  970 IFA$="Z"THENCLR:END
12  980 IFA$<>"D"ORA$<>"E"ORA$<>
    "F"THEN940
3B  990 PRINTCHR$(147)TAB(55)L$:
    PRINTTAB(15)C$:PRINTTAB(15)B
    $
80  1000 PRINTTAB(6)"[RVSON,CA,S
    *27,CS]"
E6  1010 PRINTTAB(6)"[RVSON,S-]T
    O RETURN TO MENU ENTER END[S
    -]"
9E  1020 PRINTTAB(6)"[RVSON,S-,S
    PC27,S-]"
CB  1030 PRINTTAB(6)"[RVSON,S-]
    PRESS RETURN  AND ENTER 0 [S
    -]"
F1  1040 PRINTTAB(6)"[RVSON,CZ,S
    *27,CX]"
90  1050 RETURN
5C  1060 CLOSE2:OPEN2,4:LC=55:NN
    =13:RR=10:GOSUB1900
12  1070 PRINTTAB(14)"[RVSON,CA,
    S*10,CS]"
E4  1080 PRINTTAB(14)"[RVSON,S-]
    ENTER DATE[S-]"
5C  1090 PRINTTAB(14)"[RVSON,CZ,
    S*10,CX]":INPUTDA$
90  1100 GOSUB1900:PRINTTAB(13)"
    [RVSON,CA,S*13,CS]"
94  1110 PRINTTAB(13)"[RVSON,S-]
    PRINTING FILE[S-]"
1B  1120 PRINTTAB(13)"[RVSON,CZ,
    S*13,CX]":GOSUB1440
BF  1130 U=INT((X+1)/2)
30  1140 FORJ=1TOU:X=J:X=U+J
C6  1150 PRINT#2,""TAB(7)X$(J);T
    AB(9)X$(X):LC=LC+1
93  1160 IFLC=55THENFORZ=1TO3:PR
    INT#2,CHR$(10):NEXT:GOSUB144
    0
77  1170 NEXTJ:CLOSE2:GOSUB1900:
    GOTO870
56  1180 NN=8:GOSUB1900:PRINTTAB
    (8)"[RVSON,CA,S*20,CS]"
98  1190 PRINTTAB(8)"[RVSON,S-]P
    ) OUTPUT TO PRINTER[S-]"
57  1200 PRINTTAB(8)"[RVSON,S-,S
    PC20,S-]"
4B  1210 PRINTTAB(8)"[RVSON,S-]S
    ) OUTPUT TO SCREEN [S-]"
B3  1220 PRINTTAB(8)"[RVSON,S-,S
    PC20,S-]"
D4  1230 PRINTTAB(8)"[RVSON,S-]X
    ) RETURN TO MENU[SPC3,S-]"
DF  1240 PRINTTAB(8)"[RVSON,CZ,S
    *20,CX]"
6F  1250 GETA$:IFA$=""THEN1250

CA  1260 IFA$="P"THENGOTO1300
57  1270 IFA$="S"THENGOTO1520
E5  1280 IFA$="X"THENGOTO880
0D  1290 IFA$<>"P"ORA$<>"S"ORA$<
    >"X"THEN1250
CB  1300 RR=10:NN=7:GOSUB1900:PR
    INTTAB(6)"[RVSON,CA,S*28,CS]
    "
78  1310 PRINTTAB(6)"[RVSON,S-]E
    NTER CATAGORY TO PRINT: A-Z[
    S-]"
F6  1320 PRINTTAB(6)"[RVSON,CZ,S
    *28,CX]"
D8  1330 INPUTCA$
26  1340 CLOSE3:OPEN3,4:LC=0:GOS
    UB1470
BC  1350 FORJ=1TOI
F4  1360 IFCA$=LEFT$(X$(J),1)THE
    NPRINT#3,""TAB(3)X$(J);TAB(4
    )CHR$(104):LC=LC+1
0C  1370 IFLC=22THENGOSUB1420
35  1380 NEXTJ
AA  1390 FORZ=LCTO26:PRINT#3,""T
    AB(32)CHR$(104):NEXTZ
CE  1400 FORT=1TO32:PRINT#3,CHR$
    (196);:NEXTT
9C  1410 PRINT#3:CLOSE3:GOSUB190
    0:GOTO870
0E  1420 FORZ=LCTO26:PRINT#3,""T
    AB(32)CHR$(104):NEXT
2D  1430 FORT=1TO32:PRINT#3,CHR$
    (196);:NEXT:PRINT#3,CHR$(10)
    :GOSUB1470:RETURN
FD  1440 PRINT#2:PRINT#2,""TAB(6
    )CHR$(14)DC$;TAB(3)ML$;TAB(3
    )DA$:PRINT#2
61  1450 PRINT#2,""TAB(4)NA$;TAB
    (4)DN$;TAB(3)NA$;TAB(4)DN$;C
    HR$(15):PRINT#2:LC=0
37  1460 RETURN
4C  1470 PRINT#3,CHR$(14)TAB(2)N
    A$;TAB(3)DN$;CHR$(15)TAB(2);
    CHR$(104):PRINT#3
2A  1480 LC=0:RETURN
80  1490 N2=INT(I/256):N1=I-256*
    N2
21  1500 POKE49366,N1:POKE49367,
    N2
AA  1510 SYS49152:RETURN
7E  1520 RR=5:NN=19:GOSUB1900:PR
```

```
    INITTAB(165)"[RVSON,CA,S*28,C
    S]"
43  1530 PRINTTAB(5)"[RVSON,S-]E
    NTER CATAGORY TO PRINT: A-Z[
    S-]"
7D  1540 PRINTTAB(5)"[RVSON,CZ,S
    *28,CX,DOWN]"
B9  1550 INPUTCA$
1A  1560 RR=5:GOSUB1900:LC=0:L=L
    EN(CA$):Z=ASC(LEFT$(CA$,1))
1F  1570 FORJ=1TOI
B7  1580 IFCA$=LEFT$(X$(J),L)THE
    NPRINTTAB(8)X$(J):LC=LC+1
E6  1590 IFASC(LEFT$(X$(J),1))>Z
    THENJ=I
E8  1600 IFLC=14THENGOSUB1710
1C  1610 NEXTJ
50  1620 PRINTTAB(47)"[RVSON,CA,
    S*24,CS]"
75  1630 PRINTTAB(7)"[RVSON,S-]P
    RESS RETURN TO CONTINUE[S-]"
6A  1640 PRINTTAB(7)"[RVSON,S-,S
    PCS]OR X FOR MENU[SPC6,S-]":
    POKE198,0
9B  1650 PRINTTAB(7)"[RVSON,CZ,S
    *24,CX]":POKE198,0
10  1660 GETA$:IFA$=""THEN1660
E5  1670 IFA$="X"THENRR=5:GOSUB1
    900:GOTO870
A5  1680 IFA$=CHR$(13)THEN1520
29  1690 IFA$<>CHR$(13)ORA$<>CHR
    $(88)THEN1660
A3  1700 GOTO1520
7A  1710 PRINTTAB(47)"[RVSON,CA,
    S*24,CS]"
2F  1720 PRINTTAB(7)"[RVSON,S-]P
    RESS RETURN TO CONTINUE[S-]"
28  1730 PRINTTAB(7)"[RVSON,CZ,S
    *24,CX]"
BF  1740 GETA$:IFA$=""THEN1740
99  1750 IFA$<>CHR$(13)THEN1740
FC  1760 GOSUB1900:LC=0:RETURN
F4  1770 PN$="":PRINTTAB(1)"[RVS
    ON,CA,S*35,CS]"
45  1780 PRINTTAB(1)"[RVSON,S-]T
    ITLE MUST BE 16 CHARACTERS O
    R LESS[S-]"
0F  1790 PRINTTAB(1)"[RVSON,CZ,S
    *35,CX]":GOSUB1850:NN=11
E1  1800 GOSUB1900:GOTO310
65  1810 N$="":PRINTTAB(1)"[RVSO
    N,CA,S*36,CS]"
70  1820 PRINTTAB(1)"[RVSON,S-]D
    ISK N0 MUST BE 4 CHARACTERS
    OR LESS[S-]"
C8  1830 PRINTTAB(1)"[RVSON,CZ,S
    *36,CX]":GOSUB1850:NN=11
5E  1840 GOSUB1900:GOTO360
1C  1850 FORA=54272TO54296:POKEA
    ,0:NEXTA
51  1860 POKE54272,4:POKE54273,4
    8:POKE54277,0:POKE54278,249:
    POKE54296,15
F3  1870 FORZ=1TO10:POKE54276,17
    :POKE54276,16:FORV=1TO200:NE
    XTV:NEXTZ
CC  1880 POKE54296,0:RETURN
DA  1890 R=RR:POKE781,R:POKE782,
    C:POKE783,0:SYS65520:RETURN
F3  1900 C=0:GOSUB1890
57  1910 FORP=1TONN:PRINTSP$:NEX
    T
C7  1920 C=0:GOSUB1890:RETURN
F7  1930 DATA173,214,192,208,6,1
    73,215,192,208,1,96,165,47,2
    4,105,10,133,34,165
FB  1940 DATA48,105,0,133,35,76,
    163,192,160,0,177,34,72,177,
    36,145,34,104,145,36
99  1950 DATA200,192,3,208,241,7
    6,116,192,169,0,141,218,192,
    168,177,34,240,59
30  1960 DATA141,213,192,177,36,
    240,219,205,213,192,176,8,14
    1,213,192,169,1,141
3D  1970 DATA218,192,200,177,34,
    133,251,177,36,133,253,200,1
    77,34,133,252,177,36
09  1980 DATA133,254,160,0,177,2
    53,209,251,144,180,208,11,20
    0,206,213,192,208
AD  1990 DATA242,173,218,192,208
    ,167,165,36,24,105,3,133,36,
    165,37,105,0,133,37
21  2000 DATA173,216,192,208,3,2
    06,217,192,206,216,192,173,2
    16,192,208,158,173
2C  2010 DATA217,192,208,153,165
    ,34,24,105,3,133,34,165,35,1
    05,0,133,35,173,214
69  2020 DATA192,208,3,206,215,1
    92,206,214,192,173,214,192,2
    08,6,173,215,192,208
1B  2030 DATA1,96,173,214,192,14
    1,216,192,173,215,192,141,21
    7,192,165,34,24,105
02  2040 DATA3,133,36,165,35,105
    ,0,133,37,76,47,192
```

PROGRAM: DISK CAT EDITOR

```
C9  10 PRINT"[CLR]":POKE53280,15
    :POKE53281,11
3E  15 RR=2:C=13:GOSUB7000:PRINT
    "[C8,RVSON,CA,S*11,CS]"
BC  20 PRINTTAB(13)"[RVSON,S-]FI
    LE EDITOR[S-]"
14  25 PRINTTAB(13)"[RVSON,S-,SP
    C11,S-]"
32  30 PRINTTAB(13)"[RVSON,S-,SP
    C4]BY[SPC5,S-]"
46  35 PRINTTAB(13)"[RVSON,S-,SP
    C11,S-]"
AA  40 PRINTTAB(13)"[RVSON,S-] J
    OHN HYDE [S-]"
48  45 PRINTTAB(13)"[RVSON,CZ,S*
    11,CX]"
BD  50 REM::::::::::::::::::::::
    ::
42  100 CR$=CHR$(13):SP$="[SPC39
    ]"
7D  110 L$="[RVSON,CA,S*11,CS]":
    T$="[RVSON,S-]FILE EDITOR[S-
    ]":B$="[RVSON,CZ,S*11,CX]"
82  120 DIMPR$(1000),DN$(1000)
10  160 GOSUB4000:GOSUB300:RR=10
    :C=13:GOSUB7000:PRINT"[RVSON
    ,CA,S*12,CS]"
9A  161 PRINTTAB(13)"[RVSON,S-]R
    EADING FILE[S-]"
73  162 PRINTTAB(13)"[RVSON,CZ,S
    *12,CX]"
5E  200 CLOSE15:OPEN15,8,15:OPEN
    2,8,2,"0:DC TITLE,S,R"
35  210 GOSUB1000:I=0
1A  220 INPUT#2,PN$,N$
9C  230 RS=ST:GOSUB1000
7F  240 I=I+1:PR$(I)=PN$:DN$(I)=
    N$
6D  250 IFRS=0THEN220
3A  260 IFRS<>64THENPRINT"STATUS
    =";RS
E4  270 CLOSE2:CLOSE15:RR=9:NN=5
    :GOSUB9000:GOTO2000
71  300 PRINTCHR$(147):RR=1:C=13
    :GOSUB7000:PRINTL$
2A  310 PRINTTAB(13)T$
62  320 PRINTTAB(13)B$:RETURN
81  500 RR=8:C=4:GOSUB7000:PRINT
    "[RVSON,CA,S*30,CS]"
8A  501 PRINTTAB(4)"[RVSON,S-]EN
    TER TITLE YOU WISH TO CHANGE
    [S-]"
46  502 PRINTTAB(4)"[RVSON,CZ,S*
    30,CX]":CA$=""
BA  510 INPUT"[DOWN]";CA$
A9  515 IFCA$=""THENRR=7:GOSUB90
    00:GOTO2000
83  530 FORJ=1TOI
C0  540 IFPR$(J)=CA$THENRR=8:NN=
    5:GOSUB9000:RR=5:GOSUB7000:G
    OTO8000
F4  550 NEXTJ
5C  560 IFCA$<>PR$(J)THENRR=12:C
    =11:GOSUB7000:PRINT"[RVSON,C
    A,S*15,CS]"
74  562 PRINTTAB(11)"[RVSON,S-]T
    ITLE NOT FOUND[S-]"
50  565 PRINTTAB(11)"[RVSON,CZ,S
    *15,CX]"
E7  570 FORCO=1TO1200:NEXTCO:RR=
    5:NN=15:GOSUB9000:GOTO2000
80  580 FORA=1TO38:PRINTTAB(1)CH
    R$(192);:NEXTA
62  600 RR=8:C=2:GOSUB7000:PRINT
    "[RVSON,CA,S*35,CS]"
8E  601 PRINTTAB(2)"[RVSON,S-]UP
     ARROW RETAINS TITLE OR DISK
     N0. [S-]"
35  602 PRINTTAB(2)"[RVSON,S-,SP
    C35,S-]"
BE  610 PRINTTAB(2)"[RVSON,S-]BA
    CK ARROW DELETES TITLE & DIS
    K N0.[S-]"
48  611 PRINTTAB(2)"[RVSON,S-,SP
    C35,S-]"
66  612 PRINTTAB(2)"[RVSON,S-,SP
    C10]RETURN TO ABORT[SPC10,S-
    ]"
82  613 PRINTTAB(2)"[RVSON,CZ,S*
    35,CX]"
44  615 REMPN$="":N$=""
6F  620 PN$="":INPUT"[DOWN]";PN$
    :S=LEN(PN$):IFS>16THENGOTO50
    00
FD  625 IFPN$=CHR$(94)THENPN$=PR
    $(J):PR$(J)=PR$(J)
D9  626 IFPN$=""THENPR$(J)=PR$(J
    ):DN$(J)=DN$(J):J=I+1:RR=5:N
    N=16:GOSUB9000:GOTO2000
3F  630 IFPN$="_"THENPR$(J)="":D
    N$(J)="":NN=10:GOSUB9000:GOT
    O2000
E6  635 RR=16:NN=6:GOSUB9000
```

```
B7  640 RR=16:PRINTTAB(9)"[RVSON
    ,CA,S*17,CS]"
3D  641 PRINTTAB(9)"[RVSON,S-]EN
    TER DISK NUMBER[S-]"
E4  642 PRINTTAB(9)"[RVSON,CZ,S*
    17,CX]"
05  650 N$="":INPUTN$:SS=LEN(N$)
    :IFSS>4THENGOTO5010
33  651 IFN$=CHR$(94)THENN$=DN$(
    J):DN$(J)=DN$(J)
53  656 IFN$=""THENPR$(J)=PR$(J)
    :DN$(J)=DN$(J):J=I+1:RR=5:NN
    =16:GOSUB9000:GOTO2000
43  660 PR$(J)=PN$:DN$(J)=N$:J=I
    +1:RR=5:NN=16:GOSUB9000:GOTO
    2000
6F  1000 REM::::::CHECK ERROR CH
    ANNEL::::::
E3  1010 INPUT#15,EN,EM$,ET,ES
E4  1020 IFEN>0THENPRINTEN,EM$,E
    T,ES:STOP
84  1030 RETURN
B9  2000 RR=8:C=6:GOSUB7000:PRIN
    T"[RVSON,CA,S*25,CS]"
84  2010 PRINTTAB(6)"[RVSON,S-]A
    ) CHANGE OR DELETE TITLE[S-]
    "
C7  2011 PRINTTAB(6)"[RVSON,S-,S
    PC25,S-]"
6D  2020 PRINTTAB(6)"[RVSON,S-]B
    ) SAVE CHANGES[SPC10,S-]"
95  2025 PRINTTAB(6)"[RVSON,S-,S
    PC25,S-]"
81  2030 PRINTTAB(6)"[RVSON,S-]C
    ) QUIT PROGRAMME[SPC8,S-]"
73  2031 PRINTTAB(6)"[RVSON,CZ,S
    *25,CX]"
BC  2040 GETA$:IFA$=""THEN2040
7C  2050 IFA$="A"THENRR=7:NN=10:
    GOSUB9000:GOTO500
8F  2060 IFA$="B"THENRR=7:NN=10:
    GOSUB9000:GOTO3000
DF  2070 IFA$="C"THENPRINTCHR$(1
    47):CLR:END
97  2080 IFA$<>"A"ORA$<>"B"ORA$<
    >"C"THEN2040
42  3000 REM::::::SAVE UPDATED F
    ILE::::::
A9  3005 RR=11:C=9:GOSUB7000:PRI
    NT"[RVSON,CA,S*19,CS]"
12  3010 PRINTTAB(9)"[RVSON,S-]S
    AVING UPDATED FILE[S-]"
60  3013 PRINTTAB(9)"[RVSON,CZ,S
    *19,CX]"
77  3015 OPEN15,8,15,"S0:DC TITL
    E"
E7  3020 OPEN2,8,2,"0:DC TITLE,S
    ,W"
91  3030 FORJ=1TOI:PRINT#2,PR$(J
    );CR$;DN$(J):NEXTJ
C2  3040 GOSUB1000:CLOSE2:CLOSE1
    5:GOSUB9000:GOTO2000
41  4000 RR=12:C=7:GOSUB7000:PRI
    NT"[RVSON,CA,S*24,CS]"
C0  4001 PRINTTAB(7)"[RVSON,S-]P
    UT FILE DISK INTO DRIVE[S-]"
54  4005 PRINTTAB(7)"[RVSON,S-,S
    PC24,S-]"
79  4010 PRINTTAB(7)"[RVSON,S-]H
    IT ANY KEY TO CONTINUE [S-]"
86  4011 PRINTTAB(7)"[RVSON,CZ,S
    *24,CX]"
C8  4020 GETA$:IFA$=""THEN4020
43  4030 RETURN
04  5000 RR=16:NN=5:GOSUB9000:PR
    INTTAB(2)"[RVSON,CA,S*35,CS]
    "
48  5001 PRINTTAB(2)"[RVSON,S-]T
    ITLE MUST BE 16 CHARACTERS O
    R LESS[S-]"
FC  5002 PRINTTAB(2)"[RVSON,CZ,S
    *35,CX]":PN$="":GOSUB5020
CC  5008 GOSUB9000:GOTO620
FD  5010 GOSUB9000:PRINTTAB(1)"[
    RVSON,CA,S*36,CS]"
34  5011 PRINTTAB(1)"[RVSON,S-]D
    ISK NO MUST BE 4 CHARACTERS
    OR LESS[S-]"
15  5012 PRINTTAB(1)"[RVSON,CZ,S
    *36,CX]":N$="":GOSUB5020
DD  5015 GOSUB9000:GOTO640
61  5020 FORA=54272TO54296:POKEA
    ,0:NEXT
13  5030 POKE54272,4:POKE54273,4
    8:POKE54277,0:POKE54278,249:
    POKE54296,15
29  5040 FORZ=1TO10:POKE54276,17
    :POKE54276,16:FORV=1TO150:NE
    XTV:NEXTZ
FA  5050 POKE54296,0:RETURN
86  6000 RR=17:C=38:GOSUB7000:FO
    RZ=1TO117:PRINTCHR$(20);:NEX
    T
20  6010 RR=15:C=0:GOSUB7000:RET
    URN
9E  6100 RR=22:C=0:GOSUB7000:FOR
    Z=1TO118:PRINTCHR$(20);:NEXT

D8  6110 RR=19:C=0:GOSUB7000:RET
    URN
64  7000 R=RR:POKE781,R:POKE782,
    C:POKE783,0:SYS65520:RETURN
CE  8000 DEF FNC(X)=20-(LEN(TE$)
    /2)
BC  8010 TE$=PR$(J)+"  "+DN$(J):
    PRINTTAB(FNC(X));TE$:GOTO580

D0  9000 C=0:GOSUB7000
7F  9010 FORP=1TONN:PRINTSP$:NEX
    T
F8  9020 C=0:GOSUB7000:RETURN
```



# Program Compactor

*Reduce long files and link up your programs to save on disk space*

by Richard Bright

After a game has been written there are normally several chunks of code which are spread about in the computer's memory. The Program Compactor will help to tie these routines together and then crunch them down into a neat little file.

The program will not work on Basic programs but it will happily attack anything lying from $0300 to $FFFF. Any number of files can be linked as long as the total area covered does not exceed 234 disk blocks.

When the program runs, it fills up the program memory with zero bytes to help the cruncher to do its work. When finished a prompt appears asking for LO-MEM to be input. The lowest memory location occupied by any part of the program code is entered in hex, as are all inputs in the program.

A SKIP value is the next request which means that the cruncher will ignore the 768 bytes following the input value. This is useful in skirting around screen areas. For example, if code starts at $0300 and another piece of code starts at $0800, the screen area can be omitted by entering $0400 at the SKIP prompt.

The next job is to load the first file into the computer. At the FILENAME prompt, entering '$' will display the disk directory. When the program name is known, it can then be entered (without quotes) at the name prompt.

When the program has loaded, the rest of the files can be loaded until everything is in memory. Now

it's crunch time and entering '!' instead of a filename will set the process into motion.

A screen printout of memory space used and the current disk block rating is given while the border flashes to indicate that the crunch is proceeding. After the cruncher comes the optimiser which further compresses the data.

On completion, the program prints the new memory usage and block count details before asking for the hex boot location for the coded program. Next a filename for the saved program is needed and this is the last point at which disks can be changed. Enter the new filename, press return and the job's complete.

Another copy can now be saved with a new filename, if necessary, or pressing the return key without entering anything will reset the program ready for a new compacting session.

If anything goes wrong at any point the program can be restarted by pressing RESTORE. This will reset all the program parameters and take the program back to the opening screen.

In tests a 232 block program was reduced to 131 blocks – a saving of 101 blocks which corresponds to 25687 bytes! As you can see the compactor really does save on disk space.

PROGRAM: COMPACTOR.BAS

```
C1  10 REM TYPE IN THE PROGRAM A
    ND SAVE IT ON A SPARE CASSET
    TE OR DISK
30  20 REM PREPARE A DIFFERENT D
    ISK OR CASSETTE FOR SAVING T
    HE PROGRAM
9D  30 REM THE PROGRAM SAVES AFT
    ER ASKING FOR A DEVICE NUMBE
    R...SO BEWARE!!!
4E  40 BL=208  :LN=50   :SA=3686
    4
3E  50 FOR L=0 TO BL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A
31  60 POKE53280,A:POKE SA+L*16+
    D,A:NEXT D
8D  70 READ A:IF A><CX THENPRINT
    "ERROR IN LINE";LN+(L*10):ST
    OP
1C  80 NEXT L:SYS40024
A5  90 DATA 20,8,195,7,158,50,48
    ,55,52,32,84,85,82,66,79,45,
    1066
0C  100 DATA 56,55,0,0,0,170,170
    ,170,170,32,68,229,169,14,14
    1,32,1476
EF  110 DATA 208,141,33,208,120,
    169,52,133,1,162,5,189,94,8,
    157,45,1725
35  120 DATA 0,202,16,247,154,16
    0,0,198,50,206,65,8,177,49,1
    53,0,1685
26  130 DATA 0,230,1,141,24,212,
```

```
      198,1,200,208,241,165,50,201
      ,8,208,2088
AD    140 DATA 230,185,100,8,153,0
      ,1,200,208,247,76,0,1,0,3,5,
      1417
8E    150 DATA 245,84,20,177,47,42
      ,42,42,42,41,7,170,189,26,1,
      141,1316
98    160 DATA 24,1,177,47,41,31,1
      70,32,34,1,76,255,1,168,74,1
      75,1307
A1    170 DATA 125,92,66,70,48,230
      ,1,238,32,208,198,1,230,47,2
      08,2,1796
54    180 DATA 230,48,96,177,47,32
      ,34,1,145,45,230,45,208,2,23
      0,46,1616
5A    190 DATA 202,208,245,240,190
      ,169,0,240,239,169,255,208,2
      35,177,47,145,2969
B8    200 DATA 45,32,34,1,230,45,2
      08,2,230,46,202,208,240,240,
      164,32,1959
F6    210 DATA 117,1,177,47,32,34,
      1,145,45,230,45,208,2,230,46
      ,202,1562
77    220 DATA 208,245,198,57,16,2
      41,48,139,134,57,177,47,170,
      76,34,1,1848
3E    230 DATA 32,117,1,177,47,145
      ,45,32,34,1,230,45,208,2,230
      ,46,1392
5B    240 DATA 202,208,240,198,57,
      16,236,76,0,1,32,222,1,169,5
      5,133,1846
4E    250 DATA 1,88,234,234,234,32
      ,89,166,76,48,3,224,0,240,23
      5,169,2073
FC    260 DATA 3,44,169,8,133,255,
      177,47,145,45,200,196,255,20
      8,247,24,2156
EB    270 DATA 165,45,101,255,133,
      45,165,46,105,0,133,46,160,0
      ,202,208,1809
8D    280 DATA 229,24,165,47,101,2
      55,133,47,165,48,105,0,133,4
      8,76,0,1576
2F    290 DATA 1,185,0,239,153,0,2
      55,200,208,247,206,224,1,206
      ,227,1,2353
94    300 DATA 173,227,1,201,47,20
      8,234,96,45,139,227,141,223,
      124,165,26,2277
9B    310 DATA 167,228,167,134,174
      ,55,162,96,134,49,76,72,178,
      0,49,234,1975
7B    320 DATA 159,223,71,254,74,2
      43,145,242,14,242,80,242,51,
      243,87,241,2611
92    330 DATA 202,241,237,246,62,
      241,47,243,102,254,165,244,2
      37,245,120,162,3048
54    340 DATA 255,154,169,54,133,
      1,169,52,141,24,3,169,3,141,
      25,3,1496
43    350 DATA 169,0,141,32,208,14
      1,33,208,160,0,185,95,3,240,
      6,32,1653
6B    360 DATA 210,255,200,208,245
      ,88,76,0,9,152,147,8,14,146,
      195,82,2035
3A    370 DATA 85,78,67,72,47,204,
      73,78,75,69,82,32,214,49,46,
      50,1321
3D    380 DATA 13,13,87,82,73,84,8
      4,69,78,32,66,89,32,212,85,8
      2,1181
37    390 DATA 66,79,32,195,82,65,
      67,75,69,82,45,56,55,13,13,1
      63,1157
EA    400 DATA 35,133,1,88,227,234
      ,96,83,32,89,166,76,174,167,
      224,0,1825
83    410 DATA 240,235,169,3,44,16
      9,8,133,255,177,47,145,45,20
      0,196,255,2321
7A    420 DATA 208,247,24,165,45,1
      01,255,133,45,165,46,105,0,1
      33,46,160,1878
14    430 DATA 0,202,208,229,24,16
      5,47,101,255,133,47,165,48,1
      05,0,133,1862
DC    440 DATA 48,76,0,1,185,0,239
      ,153,0,255,200,208,247,206,2
      24,1,2043
E3    450 DATA 206,227,1,173,227,1
      ,201,223,208,234,96,172,35,7
      7,79,78,2238
EB    460 DATA 132,1,0,50,20,8,195
      ,7,158,50,48,55,52,32,84,85,
      977
F6    470 DATA 82,66,79,45,56,55,1
      63,228,170,96,38,32,68,229,1
      69,14,1590
2A    480 DATA 141,32,208,141,33,2
      08,120,169,52,133,1,162,5,18
      9,94,8,1696
CE    490 DATA 157,45,0,202,16,247
      ,154,160,0,198,50,206,65,8,1
      77,49,1734
72    500 DATA 153,162,60,230,1,14
      1,24,212,198,1,200,208,241,1
      65,50,201,2247
1B    510 DATA 8,208,230,185,100,8
      ,153,0,1,200,208,247,76,0,1,
      198,1823
9C    520 DATA 34,177,47,228,42,96
      ,153,41,7,170,189,26,1,141,2
      4,1,1377
B2    530 DATA 177,47,41,31,170,32
      ,34,1,76,255,1,168,74,175,12
      5,92,1499
C3    540 DATA 66,70,48,230,1,238,
      32,208,198,1,230,47,208,2,23
      0,48,1857
4D    550 DATA 96,177,47,32,34,1,1
      45,45,230,45,208,2,230,46,20
      2,208,1748
E1    560 DATA 245,240,190,169,0,2
      40,239,169,255,208,235,177,4
      7,145,45,32,2636
64    570 DATA 34,1,230,45,208,2,2
      30,46,202,208,240,240,164,32
      ,117,1,2000
C2    580 DATA 177,47,32,34,1,145,
      45,230,45,208,2,230,46,202,2
      08,245,1897
71    590 DATA 198,57,16,241,48,13
      9,134,57,177,47,170,76,34,1,
      32,117,1544
1A    600 DATA 1,177,47,145,45,32,
      34,1,230,45,208,2,230,46,202
      ,208,1653
41    610 DATA 240,198,57,16,236,7
      6,0,1,44,222,1,169,55,76,10,
      9,1410
31    620 DATA 167,99,84,120,216,1
      69,52,133,1,160,0,132,88,169
      ,16,133,1739
25    630 DATA 89,152,141,5,9,141,
      6,9,141,9,9,145,88,200,208,2
      51,1603
33    640 DATA 230,89,208,247,169,
      55,133,1,88,162,152,160,15,3
      2,72,15,1828
BD    650 DATA 176,247,142,3,9,140
      ,4,9,142,94,8,140,95,8,162,2
      27,1606
4A    660 DATA 160,15,32,72,15,176
      ,19,140,8,9,169,255,141,7,9,
      152,1379
02    670 DATA 24,105,15,141,237,3
      ,169,32,208,7,169,0,141,7,9,
      169,1436
1B    680 DATA 44,141,251,8,32,83,
      10,32,2,14,162,203,160,15,32
      ,55,1244
4A    690 DATA 15,169,212,32,210,2
      55,120,169,52,133,1,173,5,9,
      133,90,1778
9B    700 DATA 173,6,9,133,91,169,
      255,133,88,133,89,160,0,177,
      90,145,1851
4C    710 DATA 88,165,88,208,2,198
      ,89,198,88,165,90,208,2,198,
      91,198,2076
C4    720 DATA 90,165,91,201,15,20
      8,230,230,88,208,2,230,89,16
      9,55,133,2204
78    730 DATA 1,88,169,195,32,210
      ,255,32,147,10,169,13,32,210
      ,255,165,1983
67    740 DATA 90,141,5,9,141,96,8
      ,56,233,255,141,98,8,165,91,
      141,1678
98    750 DATA 6,9,141,97,8,233,7,
      141,99,8,56,173,96,8,233,88,
      1403
C3    760 DATA 141,96,8,173,97,8,2
      33,17,141,97,8,56,169,0,237,
      96,1577
33    770 DATA 8,141,96,8,169,0,23
      7,97,8,141,97,8,32,2,14,162,
      1220
9F    780 DATA 135,160,15,32,72,15
      ,142,162,3,140,163,3,160,0,1
      85,0,1387
BE    790 DATA 8,153,0,16,200,208,
      247,160,88,185,152,3,153,0,1
      7,136,1726
E8    800 DATA 16,247,162,216,160,
      15,32,55,15,32,107,14,173,14
      9,14,240,1647
2F    810 DATA 24,169,8,170,160,1,
      32,186,255,173,149,14,162,15
      0,160,14,1827
C4    820 DATA 32,189,255,32,154,1
      3,76,41,10,76,234,15,162,169
      ,160,15,1633
C9    830 DATA 32,55,15,162,216,16
      0,15,32,55,15,32,107,14,173,
      150,14,1247
BD    840 DATA 201,36,208,6,32,166
      ,14,76,90,10,201,33,208,6,17
      3,9,1469
A5    850 DATA 9,240,224,96,169,8,
      170,160,0,32,186,255,173,149
      ,14,162,2047
A9    860 DATA 150,160,14,32,189,2
      55,32,209,12,76,90,10,120,16
      9,52,133,1703
26    870 DATA 1,169,89,133,90,160
```

```
     ,0,169,17,133,91,132,112,230
     ,1,238,1765
26   880 DATA 32,208,198,1,165,11
     2,240,3,76,108,12,177,88,200
     ,209,88,1917
7F   890 DATA 240,94,162,1,165,88
     ,72,165,89,72,160,0,132,113,
     32,161,1746
13   900 DATA 11,240,31,177,88,20
     0,209,88,208,13,201,0,240,20
     ,201,255,2182
9B   910 DATA 240,16,200,209,88,2
     40,11,32,40,12,232,224,31,20
     8,223,76,2082
4F   920 DATA 170,11,104,133,89,1
     04,133,88,160,0,224,1,208,10
     ,177,88,1700
41   930 DATA 201,0,240,28,201,25
     5,240,24,138,9,32,32,13,12,1
     77,88,1690
C6   940 DATA 32,13,12,32,161,11,
     202,208,245,165,89,208,144,7
     6,24,12,1634
5A   950 DATA 201,0,240,9,201,255
     ,240,5,200,209,88,208,149,16
     2,1,32,2200
9A   960 DATA 161,11,240,11,209,8
     8,208,7,232,224,31,208,242,2
     40,44,134,2290
E8   970 DATA 92,201,0,208,6,169,
     160,5,92,208,19,201,255,208,
     6,169,1999
FE   980 DATA 192,5,92,208,9,72,1
     69,224,5,92,32,13,12,104,32,
     13,1274
24   990 DATA 12,165,89,240,3,76,
     164,10,76,24,12,134,92,162,0
     ,134,1393
B0   1000 DATA 93,72,32,161,11,24
     0,28,104,209,88,72,208,22,16
     5,92,208,1805
7A   1010 DATA 2,230,93,230,92,16
     5,93,201,31,208,231,165,92,2
     01,255,208,2497
47   1020 DATA 225,198,92,165,93,
     9,128,32,13,12,165,92,32,13,
     12,104,1385
AF   1030 DATA 32,13,12,165,89,24
     0,122,76,164,10,160,0,230,88
     ,208,2,1611
4F   1040 DATA 230,89,96,134,92,1
     34,113,162,0,134,93,32,161,1
     1,240,45,1766
72   1050 DATA 177,88,200,209,88,
     208,13,201,0,240,34,201,255,
     240,30,200,2384
D2   1060 DATA 209,88,240,25,32,4
     0,12,165,92,208,2,230,93,230
     ,92,165,1923
0D   1070 DATA 93,201,31,208,214,
     165,92,201,255,208,208,198,9
     2,165,93,9,2433
1C   1080 DATA 96,32,13,12,165,92
     ,32,13,12,104,133,89,104,133
     ,88,177,1295
18   1090 DATA 88,32,13,12,32,161
     ,11,198,92,208,244,198,93,16
     ,240,165,1803
4A   1100 DATA 89,240,14,76,164,1
     0,160,0,145,90,230,90,208,2,
     230,91,1839
FC   1110 DATA 96,169,0,32,13,12,
     169,55,133,1,88,169,11,141,3
     2,208,1329
D2   1120 DATA 96,165,89,201,255,
     208,1,96,169,8,32,58,12,169,
     3,32,1594
6E   1130 DATA 58,12,96,133,116,1
     33,115,10,133,117,160,0,132,
     114,164,114,1607
8F   1140 DATA 177,88,200,132,114
     ,164,115,209,88,240,1,96,200
     ,132,115,196,2267
BA   1150 DATA 117,208,235,169,25
     5,133,112,228,104,96,94,165,
     113,240,3,76,2348
D8   1160 DATA 228,11,76,233,10,1
     62,1,160,0,132,114,132,112,1
     64,116,132,1783
2F   1170 DATA 115,165,116,133,11
     7,164,114,177,88,200,132,114
     ,164,115,209,88,2211
FD   1180 DATA 208,12,200,132,115
     ,198,117,208,236,232,224,31,
     208,227,138,164,2650
34   1190 DATA 116,192,8,208,2,9,
     64,32,13,12,177,88,132,115,3
     2,13,1213
1E   1200 DATA 12,164,115,200,196
     ,116,208,242,160,0,138,166,1
     16,224,8,240,2305
F8   1210 DATA 8,133,255,10,24,10
     1,255,208,3,227,10,96,210,24
     ,101,88,1753
8F   1220 DATA 133,88,165,89,105,
     0,133,89,76,164,10,162,142,1
     60,13,32,1561
70   1230 DATA 55,15,32,144,255,1
     69,8,32,192,255,162,8,32,198
     ,255,32,1844
31   1240 DATA 90,13,133,88,32,90
     ,13,133,89,173,7,9,240,11,16
     5,89,1375
46   1250 DATA 205,8,9,144,4,233,
     16,133,89,24,165,89,105,16,1
     33,89,1462
BE   1260 DATA 56,165,88,237,3,9,
     133,88,165,89,237,4,9,133,89
     ,32,1537
B6   1270 DATA 90,13,120,160,52,1
     32,1,160,0,145,88,160,55,132
     ,1,230,1539
4A   1280 DATA 88,208,2,230,89,36
     ,144,80,230,32,115,13,165,88
     ,208,2,1730
F8   1290 DATA 198,89,198,88,173,
     6,9,197,89,176,19,208,7,173,
     5,9,1644
6A   1300 DATA 197,88,176,10,165,
     88,141,5,9,165,89,141,6,9,16
     9,255,1713
A2   1310 DATA 141,9,9,96,32,19,2
     38,170,165,144,74,74,176,2,1
     38,96,1583
94   1320 DATA 104,104,88,32,115,
     13,162,123,160,13,76,55,15,1
     69,8,32,1269
01   1330 DATA 195,255,76,231,255
     ,35,35,32,196,73,83,75,32,19
     7,82,82,1934
9A   1340 DATA 79,82,83,32,35,35,
     13,0,210,69,65,68,73,78,71,2
     27,1220
9C   1350 DATA 46,97,66,13,0,162,
     247,160,13,32,55,15,32,143,2
     46,120,1447
0B   1360 DATA 169,97,133,185,32,
     213,243,165,186,32,12,237,16
     5,185,32,185,2271
F4   1370 DATA 237,169,1,133,88,1
     69,16,133,89,169,1,32,221,23
     7,169,8,1872
94   1380 DATA 32,221,237,120,160
     ,52,132,1,160,0,177,88,160,5
     5,132,1,1728
B4   1390 DATA 32,221,237,165,88,
     205,5,9,208,7,165,89,205,6,9
     ,240,1891
CE   1400 DATA 8,230,88,208,222,2
     30,89,208,218,88,32,63,246,1
     69,13,76,2188
5F   1410 DATA 210,255,215,82,73,
     84,73,78,71,46,46,13,0,162,7
     0,160,1638
D2   1420 DATA 14,32,55,15,173,5,
     9,133,136,56,173,6,9,233,16,
     133,1198
44   1430 DATA 137,166,136,32,205
     ,189,162,88,160,14,32,55,15,
     160,1,165,1717
05   1440 DATA 137,240,16,200,24,
     165,136,233,253,133,136,165,
     137,233,0,133,2341
F2   1450 DATA 137,208,236,152,17
     0,169,0,32,205,189,162,96,16
     0,14,76,55,2061
F5   1460 DATA 15,13,208,82,79,71
     ,82,65,77,32,76,69,78,71,84,
     72,1174
5D   1470 DATA 58,32,0,32,194,89,
     84,69,83,61,0,32,194,76,79,6
     7,1150
92   1480 DATA 75,83,46,13,13,0,1
     69,0,133,198,32,96,165,169,0
     ,141,1333
7E   1490 DATA 149,14,160,0,132,1
     24,185,0,2,240,17,200,174,14
     9,14,224,1784
BC   1500 DATA 16,240,9,232,142,1
     49,14,157,149,14,208,234,173
     ,149,14,96,1996
00   1510 DATA 48,49,50,51,52,53,
     54,55,56,57,48,49,50,51,52,5
     3,828
51   1520 DATA 54,32,68,229,169,8
     ,170,160,15,32,186,255,169,2
     ,162,53,1764
70   1530 DATA 160,15,32,189,255,
     32,192,255,32,231,255,169,8,
     160,0,170,2155
5D   1540 DATA 32,186,255,173,149
     ,14,162,150,160,14,32,189,25
     5,32,192,255,2250
2F   1550 DATA 162,8,32,198,255,5
     ,32,14,15,96,127,133,88,32,1
     4,15,1226
10   1560 DATA 168,166,88,152,32,
     205,189,169,32,32,210,255,32
     ,14,15,240,1999
BE   1570 DATA 5,32,210,255,208,2
     46,169,13,32,210,255,208,213
     ,76,24,15,2171
A0   1580 DATA 32,207,255,170,165
     ,144,208,2,138,96,32,204,255
     ,169,8,170,2255
8B   1590 DATA 32,195,255,32,231,
     255,104,104,169,45,160,27,32
     ,210,255,136,2242
75   1600 DATA 208,250,169,13,76,
     210,255,73,48,134,88,132,89,
     160,0,177,2082
37   1610 DATA 88,240,6,32,210,25
```

```
    5,200,208,246,96,32,55,15,32
    ,107,14,1836
41  1620 DATA 173,149,14,240,46,
    201,4,208,244,162,1,32,101,1
    5,72,32,1694
8E  1630 DATA 101,15,170,104,168
    ,24,96,32,116,15,228,10,96,1
    29,133,92,1529
0C  1640 DATA 32,116,15,5,92,96,
    189,149,14,232,201,58,144,2,
    233,7,1585
26  1650 DATA 41,15,96,162,174,1
    60,167,56,96,197,78,84,69,82
    ,32,210,1719
C7  1660 DATA 213,206,45,193,196
    ,210,58,32,36,0,197,78,84,69
    ,82,32,1731
E6  1670 DATA 204,207,215,45,205
    ,197,205,58,32,36,0,13,204,7
    3,78,75,1847
83  1680 DATA 69,82,58,13,39,36,
    39,45,32,196,73,82,44,32,39,
    33,912
0F  1690 DATA 39,45,195,82,85,78
    ,67,72,32,46,13,13,0,195,82,
    85,1129
F2  1700 DATA 78,67,72,73,78,71,
    32,58,32,0,198,73,76,69,78,6
    5,1120
F8  1710 DATA 77,69,58,32,0,211,
    75,73,80,32,36,0,76,52,3,163
    ,1037
43  1720 DATA 98,29,162,0,189,0,
    16,157,52,3,232,224,192,208,
    245,76,1883
7D  1730 DATA 52,3,208,218,88,32
    ,63,246,169,13,76,210,255,21
    1,65,86,1995
3D  1740 DATA 73,78,71,227,46,96
    ,218,13,0,162,70,160,14,32,5
    5,15,1330
47  1750 DATA 173,5,9,133,136,56
    ,173,6,9,233,16,133,137,166,
    136,32,1553
2F  1760 DATA 205,189,162,88,160
    ,14,32,55,15,160,1,165,137,2
    40,16,200,1839
F3  1770 DATA 24,165,136,233,253
    ,133,136,165,137,233,0,133,1
    37,208,236,152,2481
F3  1780 DATA 170,169,0,32,205,1
    89,162,96,160,14,76,55,15,13
    ,208,82,1646
86  1790 DATA 79,71,82,65,77,32,
    76,69,78,71,84,72,58,32,0,32
    ,978
22  1800 DATA 194,89,84,69,83,61
    ,0,32,194,76,79,67,75,83,46,
    13,1245
CC  1810 DATA 13,0,169,0,133,198
    ,32,96,165,169,0,141,149,14,
    160,0,1439
74  1820 DATA 132,124,185,0,2,24
    0,17,200,174,149,14,224,16,2
    40,9,232,1958
C1  1830 DATA 142,149,14,157,149
    ,14,208,234,173,149,14,96,48
    ,49,50,51,1697
7A  1840 DATA 52,53,54,55,56,57,
    48,49,50,51,52,53,54,32,68,2
    29,1013
06  1850 DATA 169,8,170,160,15,3
    2,186,255,169,2,162,53,160,1
```

```
    5,32,189,1777
0D  1860 DATA 255,32,192,255,32,
    231,255,169,8,160,0,170,32,1
    86,255,173,2405
0E  1870 DATA 149,14,162,150,160
    ,14,32,189,255,32,192,255,16
    2,8,32,198,2004
B4  1880 DATA 255,5,32,14,15,96,
    127,133,88,32,14,15,168,166,
    88,152,1400
71  1890 DATA 32,205,189,169,32,
    32,210,255,32,14,15,240,5,32
    ,210,255,1927
46  1900 DATA 208,246,169,13,32,
    210,255,208,213,76,24,15,32,
    207,255,170,2333
42  1910 DATA 165,144,208,2,138,
    96,32,204,255,169,8,170,32,1
    95,255,32,2105
D3  1920 DATA 231,255,104,104,16
    9,45,160,27,32,210,255,136,2
    08,250,169,13,2368
1D  1930 DATA 76,210,255,73,48,1
    34,88,132,89,160,0,177,88,24
    0,6,32,1808
09  1940 DATA 210,255,200,208,24
    6,96,32,55,15,32,107,14,173,
    149,14,240,2046
47  1950 DATA 46,201,4,208,244,1
    62,1,32,101,15,72,32,101,15,
    170,104,1508
02  1960 DATA 168,24,96,32,116,1
    5,228,10,96,129,133,92,32,11
    6,15,5,1307
23  1970 DATA 92,96,189,149,14,2
    32,201,58,144,2,233,7,41,15,
    96,162,1731
78  1980 DATA 174,160,167,56,96,
    197,78,84,69,82,32,210,213,2
    06,45,193,2062
93  1990 DATA 196,210,58,32,36,0
    ,197,78,84,69,82,32,204,207,
    215,45,1745
60  2000 DATA 205,197,205,58,32,
    36,0,13,204,73,78,75,69,82,5
    8,13,1398
BD  2010 DATA 39,36,39,45,32,196
    ,73,82,44,32,39,33,39,45,195
    ,82,1051
BA  2020 DATA 85,78,67,72,32,46,
    13,13,0,195,82,85,78,67,72,7
    3,1058
FE  2030 DATA 78,71,32,58,32,0,1
    98,73,76,69,78,65,77,69,58,3
    2,1066
57  2040 DATA 0,211,75,73,80,32,
    36,0,76,52,3,163,48,162,0,18
    9,1200
AD  2050 DATA 0,16,157,52,3,232,
    224,192,208,245,76,52,3,0,9,
    243,1712
C4  2060 DATA 234,0,201,0,0,0,0,
    0,169,0,133,250,169,144,133,
    251,1684
22  2070 DATA 169,1,133,174,133,
    193,169,8,133,175,133,194,16
    9,88,133,252,2257
29  2080 DATA 169,156,133,253,16
    0,0,177,250,145,174,230,250,
    208,2,230,251,2788
0F  2090 DATA 230,174,208,2,230,
    175,165,250,197,252,208,234,
    165,251,197,253,3191
1D  2100 DATA 208,228,169,243,13
    3,187,169,156,133,188,169,9,
    133,183,169,0,2477
34  2110 DATA 133,185,160,0,185,
    195,156,240,6,32,210,255,200
    ,208,245,32,2442
DF  2120 DATA 207,255,240,251,20
    1,49,240,4,201,56,48,230,41,
    15,133,186,2357
2B  2130 DATA 76,234,245,147,17,
    17,73,78,80,85,84,32,68,69,8
    6,73,1464
33  2140 DATA 67,69,32,78,85,77,
    66,69,82,13,17,67,65,83,61,4
    9,980
0F  2150 DATA 32,47,32,68,73,83,
    75,61,32,56,32,79,82,32,57,5
    8,899
39  2160 DATA 45,32,0,67,79,77,8
    0,65,67,84,79,82,0,0,0,0,757

16  2170 DATA 0,0,0,255,255,255,
    255,255,0,0,0,0,0,0,0,0,1275
```

# Super Index

*Dig out those old magazines and get them organised with this super database*

Have you ever spent fruitless hours sifting through a mountain of back-dated magazines, searching for a particular article, sub-routine or program? Now with Super Index and a little typing skill on your part, all this thumbing through pages can be a thing of the past.

This database will enable you to select a subject from a cursor-driven menu and then tell you immediately the titles and page numbers where the subject is broached in your pile of magazines. Press the 'R' key, and you are returned instantly to the subject list, ready to make another enquiry.

This high speed shuttling between subjects and magazine titles is possible because the database is committed to memory only once before you actually use the program. The system is designed this way so that the READ statement is not required to re-search the database or arrays. The resultant speed means that information comes to you as fast as Basic can print it on the screen.

The program begins by defining two user-defined variables. The print limit (PL) is concerned with the size of the selection list displayed on the screen. With PL set to 20 you get the maximum display of 20 items. If you wish, this can be reduced.

The next variable involves the alphabetical sort (AS), and is simply the memory location for the machine-code sort. Though you may relocate the sort, you certainly can't omit it.

The main task of determining the size and shape of the database is solved by reading all the data statements and concluding with values for DM (DiMension) and CL (CoLumn).

What may have caught your eye in the listing, is the isolated data statements that lie between the program and data information. Lines 1760 to 1790 serve to format your screen displays:

```
TL$(0)–1760 DATA C/MAGS 20MAY88      Title of the data base
TL$(1)–1770 DATA "MAG:"          }   Headings for
TL$(2)–1780 DATA "PAGE:"         }   the information lists
TL$(3)–1790 DATA *                   An end marker
```

What is so important about these particular data statements? Well, the entire usable database is structured on two parallel arrays. The first is isolated and contains the various subjects on file and is known as the A$() array.

The second, which is two dimensional, contains the information you're looking for and is known as the B$(–,–) array. Its structure relies on the format defined by the above data statements (lines 1760 to 1790). The aim of this particular database is to locate the exact edition of the magazine plus the page number against any subject you have chosen from the subject list. For example, an article entitled MODEMS can be found in the March 1988 edition of Your Commodore page 96. That precise information is written down as:

```
1860 DATA MODEMS,YOUR COMMODORE MAR 88,96
          ↑                    ↑       ↑
         A$()                 B$(–,–)
```

The beauty of a system using a two dimensional array is its sheer flexibility, consider adding the following line:

```
1765 DATA "VOLUME:"
```

You could now go on to create an extra element in your database, and line 1860 could now look like this:

```
1860 DATA MODEMS,4,YOUR
     COMMODORE MAR 88,96
```

So if you store your magazines in folders or volumes, you could easily grab Volume 4, in this case, whip out the mag and flick through to the correct page.

If you really wish to impress your friends, you can use the program for other purposes. Car performance figures could be yours at the touch of a button. Try these lines:

```
1760 DATA CAR PERFORMANCE          The title of your new database
1765 DATA "0–60:"              }
1770 DATA "ECONOMY:"           }   Six statements to
1775 DATA "POWER:"             }   control six elements
1780 DATA "SERVICE:"           }   of the database
1785 DATA "MAX SPEED:"         }
1790 DATA *                        DON'T FORGET THE END
                                   MARKER
```

*by N Hart*

And a typical example for your database would be:

```
1860 DATA FORD FIESTA,
     16.1 SECS,41 MPG,
     40 BHP,6000 MILES,80 MPH
```

The only thing to remember is that your formatting statements (lines 1760 to 1790) must end with an asterisk (*), and that your database statements must contain the appropriate number of elements – a total of three in the case of Super Index, and six in the Car Performance example.

With the database sized up and about to be committed to memory, it would seem appropriate to prepare a list directly for the screen, ie print A$(). But first think about the disadvantages this may have. Such a simple list would be chronological, in other words, in the order that it's read, not alphabetical. Perhaps you would consider an alphabetical-sort to be a bit of a luxury, and I would agree if the database was small but, with anything larger than 80 to 100 items, luxury takes on a new meaning.

The second problem would be the lack of a condensed list. It would be far better if the list routine examined itself for multiple entries and formed single entries in their place. For example, modems are a very popular subject and occur three times in the database I've provided. However, if 'Modems' is to be listed on the screen it need only appear once, any more would be unnecessary.

The third problem is making a single item list seem intelligent. Each subject in the list needs to keep track of where it came from in the array and then it can instantly reveal the corresponding information. Select MODEMS and you will be told that articles on this subject can be found in three separate magazines.

How is all this achieved? The first trick is to reconfigure the A$() array so that each element includes its own subscript. For example, the flavour of the month, MODEMS, occurs at positions 1, 20 and 30 in the array, and what you have at this point is:

```
A$(1) = "MODEMS"
A$(20) = "MODEMS"
A$(30) = "MODEMS"
```

What you need is:

```
A$(1) = "MODEMS1"
A$(20) = "MODEMS20"
A$(30) = "MODEMS30"
```

The method I've chosen involves printing to and reading off the screen. Notice the OPEN 1,3 in line 330. To see what's involved, examine lines 540 to 640 and you will notice that I've formatted the printing on screen so that the variable (I) always begins four spaces from the end of each printed string, and that a colon(:) appears at the end of each string to minimise the string length to be read. In this example I've used a "–" to illustrate spacings.

```
                MODEMS
Changes to      MODEMS–20–:
Which becomes   MODEMS–20–
```

All this can be observed as it actually happens by changing the value of POKE 53281 in line 330 to POKE 53281,11 (ie change the colour of the screen).

Now that the A$() array has been rewritten, a quick excursion through the land of machine code (SYS AS,A in line 680) is all that you need to return the array nicely sorted and ready for final processing before appearing on the screen. If you follow the simplified block diagram (Fig. 1) and read what is to follow, this last process will become clear.

The overall task of the routine (lines 700 to 830) is to extract the stored information held by the A$() array, and totally reconfigure and condense that information and present it as the L$() array (the list array). This takes place while the A$() array is progressively nullified in order to save memory. The resulting L$() array eventually exists as an alternating series of words and numbers, hence the double dimension requirement expressed in line 470.

The relationship between the L$() lines is very important. Consider the following:

```
L$(29) = "MODEMS"
L$(30) = "1––20–30–"
```

Take the 1, 20 and 30 from L$(30) and apply them to the B$(–,–) array:

As you can see, L$(30) holds the key to the whereabouts of the corresponding information in the B$(–,–) array for L$(29). Each of the numbers held by L$ is designated three spaces in the string, so the carrying capacity is the maximum string length divided by three (approximately 85 locations). In our modems example you would need more than 85 articles entitled MODEMS before the program crashes with a 'string too long' error.

If you've checked the program listing (lines 700 to 830) against the block diagram (Fig. 1), you will see that I've had to tell a few white lies in order to save space in the diagram. Remember that the diagram serves only to show the essence of what is occurring when the program is running; the real details are in the program itself. You'll also notice that my first step (at line 700) is to nullify A$(0). In fact, when the A$() array is sorted, A$(0) is kept out of the alphabetical order because it is normally regarded by the sort-routine as a reserved string (usually a title string identifying the array). As you can see, I've reserved my titles elsewhere in a separate array, TL$(), choosing to bypass A$(0) as part of my database. The overall result does not suffer in any way and the procedures are easier to follow with the database titles separated off.

At last you're on to the screen. Lines 880 to 1020 deal with the subject list and, if you've paid attention, you'll realise that only the odd numbered subscripts are printed to the screen – L$(1), L$(3), L$(5) and so on. The variable 'I' is the fundamental counter at this point and the mathematics involved is pretty straightforward; check lines 890 to 910. All the user has to do is to select the required page (SPACE or SHIFT/SPACE), press RETURN, move the arrow cursor alongside the appropriate subject, press RETURN again, and the information (the magazine titles and page numbers) are printed on the screen immediately.

Why is this process so speedy? Well, if you've selected MODEMS, it is displayed as number 15 in the subject list, and if you take 15 and

```
B$(0,1)  = "YOUR COMMODORE MAR 88" :B$(1,1) = "96"
B$(0,20) = "YOUR COMMODORE APR 88" :B$(1,20) = "39"
B$(0,30) = "YOUR COMMODORE DEC 87" :B$(1,30) = "56"
```

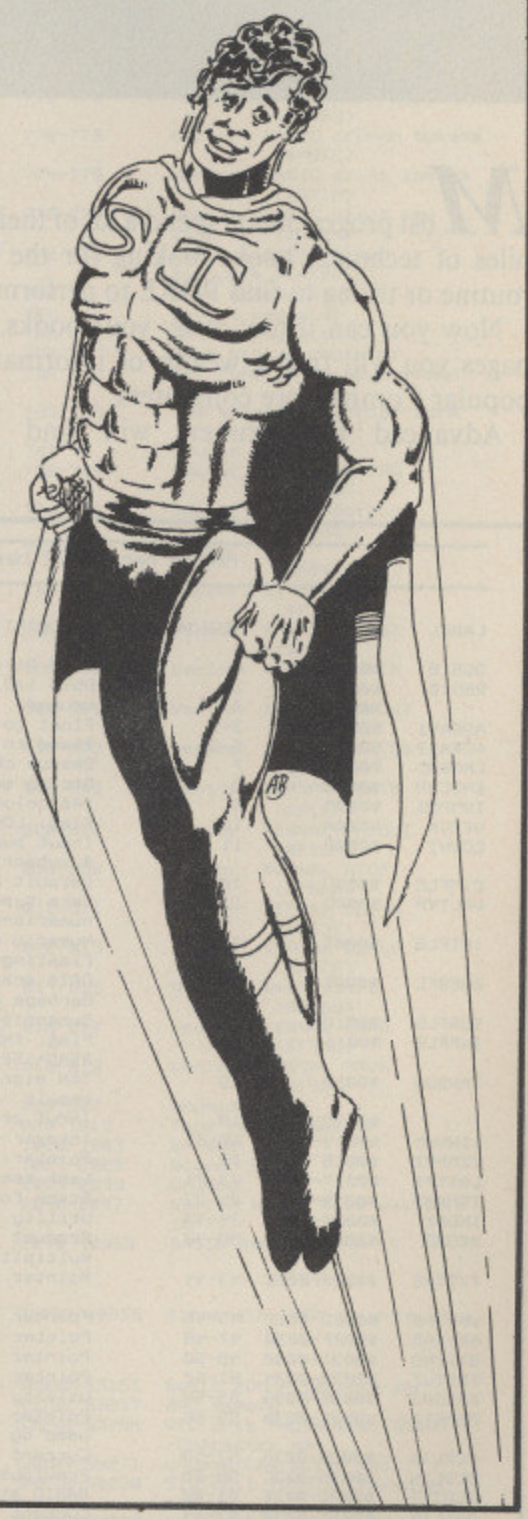
multiply it by two you get 30. If you then take L$(30), you should get the string "1--20-30-". Take the three values (1,20, and 30 – the variable 'Z' in line 1090) and apply them to the B$(-,-) array and the process is complete. The value of 'Z' is obtained by taking a look at the numbers in the appropriate L$() string in jumps of three (P=P+3 in line 1070) and, in this way, three digit values can be catered for. Press the 'R' key and you are returned, without delay, to the subject list ready for another go. The business of darting from one screen to another uses no memory, so you can play around without fear of crashing the program.

Before you start key bashing, a few bits of information and advice. The program lines containing the colons only are purely decorative and are there to separate the various routines throughout the program. By contrast, the full stops in the DATA statements act as shorthand repeater statements, study the DATA statements carefully to understand what is going on. As to the capacity, I've tested the existing database to approximately 640 lines, without any problems – and that's more than ten items per monthly magazine for five years.

**PROGRAM: SUPER INDEX**

```
F3  10 REM ******************
F4  20 REM *** SUPER INDEX ***
E5  30 REM ***             ***
AD  40 REM *** WRITTEN  BY ***
01  50 REM ***             ***
26  60 REM ***   N. HART.  ***
67  70 REM ******************
DF  80 REM
FF  90 REM  *****************
1A  100 REM *  20 MAY 1988  *
8B  110 REM *****************
F7  120 REM
0D  130 REM
41  140 PL=20:   REM PRINT LIMIT

AC  150 :
5B  160 AS=50000:REM ALPHABETICA
    L M/C SORT
90  170 :
8E  180 :
31  190 PRINTCHR$(8):POKE53281,1
    5:POKE53280,11:OPEN1,3
B2  200 PL=PL-2:IFPL>20THENPL=20

7E  210 PRINT"[BLACK,CLR,DOWN9]"
    SPC(3)"PLEASE WAIT:"
ED  220 PRINTTAB(3)"[DOWN]THE IN
    DEX IS BEING SORTED":L=2:PRI
    NT"[HOME,DOWN6]":GOSUB1140
DC  230 :
CA  240 :
7D  250 POKE53280,0:POKE53280,2:
    READX:IFX<>96THEN250
17  260 READTL$:PRINT"[RIGHT,UP6
    ,RVSON]"TL$:CL=CL+1
25  270 READTL$:IFTL$<>"*"THENCL
    =CL+1:GOTO270
99  280 DM=-1:CL=CL-2
64  290 POKE53280,6:READA$:DM=DM
    +1:POKE53280,0:IFA$<>"[END O
    F FILE]"THEN290
C9  300 DM=DM/(CL+2)+1
0D  310 :
7B  320 :
0D  330 DIMA$(DM),B$(CL,DM),L$(2
    *DM),TL$(CL+2):RESTORE
6F  340 :
9D  350 J=-1
D5  360 POKE53280,0:J=J+1:POKE53
    280,5:READX:POKEAS+J,X:IFX<>
    96THEN360
B0  370 READTL$(I):IFI<(CL+2)THE
    NI=I+1:GOTO370
C8  380 TL$=TL$(0)
50  390 I=1:E=0
55  400 POKE53280,7
11  410 READA$(I)
ED  420 PRINT"[HOME,DOWN,SPC39,U
    P]"
30  430 PRINT"[C8]"A$(I)"[SPC4]:
    [LEFT5]"I"[LEFT,UP]"
6A  440 IFLEN(A$(I))>31THENPRINT
    "[BLACK,CLR,DOWN,RIGHT]STRIN
    G IN DATA TOO LONG ERROR[DOW
    N]":PRINTA$(I):STOP
1B  450 READB$(E,I)
D2  460 IFB$(E,I)="."THENB$(E,I)
    =B$(E,I-1)
8E  470 E=E+1:IFE>CLTHENE=0:GOTO
    490
D9  480 GOTO450
81  490 INPUT#1,A$(I)
34  500 POKE53280,0:IFLEFT$(A$(I
    ),13)<>"[END OF FILE]"THENI=
    I+1:GOTO400
C5  510 :
30  520 :
94  530 POKE53280,8:PRINT"[CLR,B
    LACK,DOWN7,RIGHT2,RVSON]ALPH
    ABETICAL SORT"
17  540 SYSAS,A:POKE53280,3:I=1:
    X=0:PRINT"[CLR,C8,DOWN]"
12  550 :
D9  560 POKE53280,0:A$(X)=""
06  570 POKE53280,13:X=X+1
A7  580 IFLEFT$(A$(X),13)="[END
    OF FILE]"THEN710
16  590 IFLEFT$(A$(X),LEN(A$(X))
    -3)=LEFT$(A$(X+1),LEN(A$(X+1
    ))-3)THEN660
DB  600 POKE53280,4
94  610 PRINTRIGHT$(A$(X),3)":[L
    EFT]";
17  620 PRINT"[HOME,DOWN]":L$(I)
    =LEFT$(A$(X),LEN(A$(X))-4)
18  630 IFPEEK(1144)=32THENINPUT
    #1,L$(I+1):GOTO690
B1  640 INPUT#1,Y$(Y):IFY$(Y)=""
    THEN680
B4  650 Y=Y+1:GOTO640
E1  660 PRINTRIGHT$(A$(X),3)":[L
    EFT]";:P=P+1:IFP=12THENP=0:P
    RINT
98  670 GOTO560
34  680 FORZ=0TOY-1:L$(I+1)=L$(I
    +1)+Y$(Z):Y$(Z)="":NEXT
9A  690 I=I+2:Y=0:P=0:PRINT"[CLR
    ,DOWN]":GOTO560
84  700 :
36  710 I=1:POKE53280,14:PRINT"[
    CLR,BLACK,DOWN7,RIGHT2,RVSON
    ]CLEARING GARBAGE[C8]"
2F  720 PRINTFRE(8):POKE53280,15
E6  730 :
D9  740 PRINT"[CLR,BLACK,DOWN]":
    L=0
72  750 PRINTTAB(2)I/2+.5:PRINTT
    AB(6)"[UP]:";:IFI/2+.5=NTHEN
    PRINT"[RVSON]";
AC  760 PRINTL$(I):IFPEEK(1144+(
    PL*40)+6)<>32THEN800
7F  770 IFL$(I+2)<>""THENI=I+2:L
    =L+1:GOTO750
B2  780 L=L+1:PRINTTAB(10)"*END
    OF SUBJECT LIST*"
CD  790 PRINT"[HOME]":GOSUB1130:
    L=L-1:GOTO810
D9  800 PRINT"[HOME]":GOSUB1130
8E  810 PRINT"[RIGHT,RVSON](SHIF
    T)[SPACE]-PAGE[RVSOFF,SPC3,R
    VSON][RETURN]-ENQUIRY"
0D  820 :
81  830 POKE198,0:WAIT198,1:C=PE
    EK(631)
99  840 IFC=13THENLL=L:GOSUB1380
    :PRINT:GOTO880
28  850 IFC=32ANDL$(I+2)<>""THEN
    I=I+2:GOTO740
6F  860 IFC=160ANDI<>2*L+1THENI=
    I-(2*(L+PL+2)):GOTO740
5B  870 GOTO830
91  880 P=-2:IFN=0THENI=(I)-(LL*
    2):GOTO740
47  890 :
BD  900 :
AB  910 :
AC  920 PRINT"[CLR,DOWN2,RIGHT2,
    RVSON]"L$(N*2-1)"[DOWN]":L=1

92  930 P=P+3
A8  940 IFP>LEN(L$(N*2))THENPRIN
    TTAB(16)"[UP]..END..":P=1:GO
    TO1040
DC  950 Z=VAL(MID$(L$(N*2),P,3))

1A  960 PRINT"[UP]";:FORTS=0TOCL
    :PRINTTAB(5)CHR$(160)
FA  970 IFPEEK(1869)<>32THENTS=C
    L:NEXT:PRINT"[UP]";:L=L-1:GO
    TO1040
55  980 NEXT
B5  990 FORTS=1TOCL:PRINT"[UP]";
    :NEXT:PRINT"[RIGHT4]:[LEFT4]
    "(P+2)/3"[UP]"
20  1000 L=L+1:PRINTTAB(5)TL$(D+
    1);:PRINTB$(D,Z):IFD=CLTHEND
    =0:GOTO1020
```

```
14  1010 D=D+1:GOTO1000
4D  1020 PRINT:L=L+1
BF  1030 GOTO930
1E  1040 GOSUB1130
BE  1050 PRINT"[BLACK,RIGHT,RVSO
    N][SPACE]-PAGE[RVSOFF,SPC15,
    RVSON][R]-RESTART"
12  1060 :
CE  1070 POKE198,0:WAIT198,1:C=P
    EEK(631)
E8  1080 IFC=32THENPRINT"[CLR,DO
    WN2]":P=P-3:GOTO920
46  1090 IFC=82THENI=(I)-(LL*2):
    GOTO740
40  1100 GOTO1060
60  1110 :
5E  1120 :
17  1130 PRINT"[HOME]";
24  1140 PRINTTAB(28)"[RVSON]SUP
    ER INDEX"
F9  1150 PRINT"[RIGHT,RVSON,UP]"
    TL$
5B  1160 PRINT"[RIGHT,SO,CY36,SP
    ]"
43  1170 FORX=0TOL:PRINT"[RIGHT,
    CH]"SPC(36)"[CN]":NEXT
EA  1180 PRINT"[RIGHT,SL,CP36,S@
    ]"
24  1190 RETURN
8E  1200 :
84  1210 :
B8  1220 DATA32,115,0,133,97,169
    ,128,133,98,32,115,0,240,7,9
    ,128,133,98,32,115
B1  1230 DATA0,165,47,133,99,165
    ,48,133,100,160,0,165,97,209
    ,99,208,7,200,165,98
9E  1240 DATA209,99,240,20,24,16
    0,2,177,99,101,99,72,200,177
    ,99,101,100,133
B7  1250 DATA100,104,133,99,144,
    221,160,5,177,99,133,102,200
    ,177,99,133,101,208
E7  1260 DATA2,198,102,198,101,2
    4,165,99,105,7,133,99,165,10
    0,105,0,133,100,165,101
3C  1270 DATA208,2,198,102,198,1
    01,208,4,165,102,240,18,133,
    105,162,0,134,103,134
30  1280 DATA104,165,99,133,106,
    165,100,133,107,240,224,240,
    114,24,165,106,105
B4  1290 DATA3,133,106,165,107,1
    05,0,133,107,230,103,208,2,2
    30,104,160,2,177,106
D9  1300 DATA153,109,0,136,16,24
    8,160,5,177,106,153,109,0,13
    6,192,2,208,246,170
BD  1310 DATA56,229,109,144,2,16
    6,109,160,255,232,200,202,20
    8,8,165,112,197,109
9D  1320 DATA144,10,176,34,177,1
    13,209,110,240,238,16,26,160
    ,2,185,112,0,145
FF  1330 DATA106,136,16,248,160,
    5,185,106,0,145,106,136,192,
    2,208,246,169,0,133
9A  1340 DATA105,165,101,197,103
    ,208,152,165,102,197,104,208
    ,146,165,105,240,138,96
71  1350 :
6F  1360 :
65  1370 :
F9  1380 PRINT"[RVSON,UP,RIGHT]M
    OVE CRSR[RVSOFF] [RVSON][R]-
    RESTART"
16  1390 PRINT"[HOME]"SPC(28)"[S
    PC3,RVSON][S]-SAVE[DOWN]"
3D  1400 N=0:A=0:PRINT"[RIGHT2]>
    "
D3  1410 POKE198,0:WAIT198,1:C=P
    EEK(631)
32  1420 IFC=82THENRETURN
50  1430 IFC=83THEN1550
8D  1440 IFA=LLANDC=17THENA=A-1:
    PRINT"[UP2]"
62  1450 IFA=0ANDC=145THENA=A+1:
    PRINT
DD  1460 IFC=17THENA=A+1:PRINT"[
    RIGHT2,UP] [LEFT,DOWN]>"
98  1470 IFC=145THENA=A-1:PRINT"
    [RIGHT2,UP] [LEFT,UP]>"
35  1480 IFC=13THENPRINT"[RIGHT3
    ,UP]";:GOTO1500
DC  1490 GOTO1410
08  1500 INPUT#1,N
65  1510 RETURN
CF  1520 :
C5  1530 :
30  1540 :
07  1550 PRINT"[CLR,DOWN3]SAVE"C
    HR$(34)TL$CHR$(34)"[UP3]"
EB  1560 POKE631,13:POKE198,1:EN
    D
12  1570 :
08  1580 :
06  1590 :
7C  1600 :
6A  1610 :
5C  1620 DATAC/MAGS 20MAY88
FA  1630 DATA"MAG :"
38  1640 DATA"PAGE:"
25  1650 DATA*
B8  1660 :
B6  1670 :
AC  1680 :
9A  1690 :
90  1700 :
8E  1710 :
1F  1720 DATAMODEMS,YOUR COMMODO
    RE MAR 88,96
51  1730 DATASUPER MOUSE,.,99
C8  1740 DATABASIC COMPILER,COM
    DISK USER MAR/APR 1988,16
11  1750 DATAPROFESSIONAL PROGRA
    MMING,.,40
45  1760 DATAMULTI COLOUR,.,14
B6  1770 DATABASIC+,.,20
C8  1780 DATATAPE ARCHIVE,.,21
F8  1790 DATALINK AND CRUNCH M/C
    ODE,.,22
FC  1800 DATAMONITOR,.,24
09  1810 DATADISK LIBRARIAN II,.
    ,26
AC  1820 DATAPROGRAM COMPRESSION
    ,.,31
70  1830 DATAMONITOR,.,33
6F  1840 DATABYTING INTO THE 651
    0,YOUR COMMODORE JAN 88,13
FE  1850 DATALIST PAUSE,YOUR COM
    MODORE APR 88,14
B7  1860 DATACARTRIDGE-FINAL CAR
    TRIDGE III,.,18
BD  1870 DATACARTRIDGE-ACTION RE
    PLAY,.,.
81  1880 DATAAUTO START,.,22
88  1890 DATACLEARING TEXT SCREE
    N,.,24
1D  1900 DATATABULATION ON SCREE
    N,.,30
AD  1910 DATAMODEMS,.,39
BE  1920 DATAMUSIC,.,45
AB  1930 DATAMAKING GEOS BRITISH
    ,.,47
13  1940 DATAARTIFICIAL INTELLIG
    ENCE,.,59
E2  1950 DATAMUSIC,.,65
1F  1960 DATAEXTENDED BACKGROUND
    S,.,69
FC  1970 DATATAPE TURBO,.,.
6E  1980 DATACP/M ASSEMBLY LANGU
    AGE,.,70
E5  1990 DATAMUSIC,.,75
DD  2000 DATAOLD ROUTINE,.,79
37  2010 DATAMODEMS,YOUR COMMODO
    RE DEC 87,56
62  2020 DATA"[END OF FILE]",,,,
    ,,,,,,,,,,,,,,,
```



# Technical Information

*All you ever wanted to know about your Commodore but were afraid to ask.*

Most programmers spend a lot of their time sifting through piles of technical books looking for the address of a certain routine or trying to find POKE to perform a certain function.

Now you can throw away your books, as on the following pages you will find a wealth of information about all of the popular Commodore computers.

Advanced programmers will find the memory maps invaluable while both beginners and old hands alike will find the Hex converter, the hints and tips and much more, to their liking.

Most of the information provided here is useful by itself. Some information, such as the addresses of routines within the ROMs, will be of more use when used together with a ROM disassembly.

MEMORY MAP OF THE C64

| LABEL | HEX | DECIMAL | DESCRIPTION |
|---|---|---|---|
| D6510 | $0000 | 0 | 6510 Direction register |
| R6510 | $0001 | 1 | 6510 I/O, memory and tape |
| | $0002 | 2 | Unused |
| ADRAY1 | $0003-0004 | 3-4 | Float to fixed vector |
| ADRAY2 | $0005-0006 | 5-6 | Fixed to float vector |
| CHARAC | $0007 | 7 | Search character |
| ENDCHR | $0008 | 8 | String scan-quotes flag |
| TRMPOS | $0009 | 9 | TAB column |
| VERCK | $000A | 10 | Flag: LOAD=0, VERIFY=1 |
| COUNT | $000B | 11 | Input buffer pointer/ # subscripts |
| DIMFLG | $000C | 12 | Default DIM flag: default=0 |
| VALTYP | $000D | 13 | Data type: string=255, numeric=0 |
| INTFLG | $000E | 14 | Numeric data type: floating=0, integer=128 |
| GARBFL | $000F | 15 | DATA scan/LIST quote/ Garbage collect flag |
| SUBFLG | $0010 | 16 | Subscript/FN flag |
| INPFLG | $0011 | 17 | Flag: INPUT=0, GET=64, READ=152 |
| TANSGN | $0012 | 18 | TAN sign/comparison result |
| | $0013 | 19 | INPUT prompt flag |
| LINNUM | $0014-0015 | 20-21 | Integer value |
| TEMPPT | $0016 | 22 | Pointer: temp string stack |
| LASTPT | $0017-0018 | 23-24 | Last temp string address |
| TEMPST | $0019-0021 | 25-33 | Stack for temp strings |
| INDEX | $0022-002S | 34-37 | Utility pointer area |
| RESHO | $0026-002A | 38-42 | Product area for multiplication |
| TXTTAB | $002B-002C | 43-44 | Pointer start of BASIC (S0801) |
| VARTAB | $002D-002E | 45-46 | Pointer start of variables |
| ARYTAB | $002F-0030 | 47-48 | Pointer start of arrays |
| STREND | $0031-0032 | 49-50 | Pointer end of arrays +1 |
| FRETOP | $0033-0034 | 51-52 | Pointer bottom of strings |
| FRESPC | $0035-0036 | 53-54 | Utility string pointer |
| MEMSIZ | $0037-003B | 55-56 | Pointer highest address used by BASIC |
| CURLIN | $0039-003A | 57-58 | Current BASIC line number |
| OLDLIN | $003B-003C | 59-60 | Previous BASIC line number |
| OLDTXT | $003D-003E | 61-62 | BASIC statement for CONT |
| DATLIN | $003F-0040 | 63-64 | Current DATA line |
| DATPTR | $0041-0042 | 65-66 | Current DATA address |
| INPPTR | $0043-0044 | 67-68 | INPUT routine vector |
| VARNAM | $0045-0046 | 69-70 | Pointer: current variable name |
| VARPNT | $0047-0048 | 71-72 | Pointer: current variable data |
| FORPNT | $0049-004A | 73-74 | Pointer: variable for FOR/NEXT |
| | $004B-004C | 75-76 | Y-save/op-save/BASIC pointer save |
| | $004D | 77 | Comparison symbol accumulator |
| | $004E-0053 | 78-83 | Misc work area |
| | $0054-0056 | 84-86 | Jump vector for functions |
| | $0057-0060 | 87-96 | Misc numeric work area |
| FACEXP | $0061 | 97 | FAC#1 - exponent |
| FACHO | $0062-0065 | 98-101 | FAC#1 - mantissa |
| FACSGN | $0066 | 102 | FAC#1 - sign |
| SGNFLG | $0067 | 103 | Pointer: series evaluation constant |
| BITS | $0068 | 104 | FAC#1 - overflow digit |
| ARGEXP | $0069 | 105 | FAC#2 - exponent |
| ARGHO | $006A-006D | 106-109 | FAC#2 - mantissa |
| ARGSGN | $006E | 110 | FAC#2 - sign |
| ARISGN | $006F | 111 | FAC#1/#2 sign comparison result |
| FACOV | $0070 | 112 | FAC#1 - low order rounding |
| FBUFPT | $0071-0072 | 113-114 | Pointer: cassette buffer |
| CHRGET | $0073-008A | 115-138 | Subroutine: get next BASIC byte |
| CHRGOT | $0079 | 121 | Entry point to get same byte |
| TXTPTR | $007A-007B | 122-123 | Pointer current byte of BASIC |
| RNDX | $008B-008F | 139-143 | RND seed value |
| STATUS | $0090 | 144 | Kernal I/O status (ST) |
| STKEY | $0091 | 145 | STOP key/RVS key switch |
| SVXT | $0092 | 146 | Timing constant for tape |
| VERCK | $0093 | 147 | Flag: LOAD=0, VERIFY=1 |
| C3PO | $0094 | 148 | Serial bus: buffered char flag |
| BSOUR | $0095 | 149 | Serial bus: buffered output character |
| SYNO | $0096 | 150 | EOT tape signal received |
| | $0097 | 151 | Register save |
| LDTND | $0098 | 152 | Number of files open/File table index |
| DFLTN | $0099 | 153 | Input device (default=0) |
| DFLTO | $009A | 154 | Output device (default=3) |
| PRTY | $009B | 155 | Tape char parity |
| DPSW | $009C | 156 | Flag: tape byte received |
| MSGFLG | $009D | 157 | BASIC mode: Program=0, Direct=128 |
| PTR1 | $009E | 158 | Tape pass 1 error log |
| PTR2 | $009F | 159 | pass 2 error log |
| TIME | $00A0-00A2 | 160-162 | Real-time jiffy clock |
| | $00A3 | 163 | Serial bit count/EOI flag |
| | $00A4 | 164 | Cycle count |

| | | | |
|---|---|---|---|
| CNTDN | $00A5 | 165 | Tape sync countdown/bit count |
| BUFPNT | $00A6 | 166 | Pointer: tape I/O buffer |
| INBIT | $00A7 | 167 | RS232 input bits/tape(write ldr/read count) |
| BITCI | $00AB | 168 | RS232 input bit count tape write ldr/read count |
| RINONE | $00A9 | 169 | Flag: RS232 start bit |
| RIDATA | $00AA | 170 | RS232 input byte buffer/ tape (scan/counter/ldr) |
| RIPRTY | $00AB | 171 | RS232 input parity/ tape (write ldr length/read checksum) |
| SAL | $00AC-00AD | 172-173 | Pointer: tape buffer/screen scrolling |
| EAL | $00AE-00AF | 174-175 | Tape program end address |
| CMPO | $00B0-00B1 | 176-177 | Tape timing constants |
| TAPE1 | $00B2-00B3 | 178-179 | Pointer: start of tape buffer |
| BITTS | $00B4 | 180 | RS232 out bit count/tape timer enabled=1 |
| NXTBIT | $00B5 | 181 | RS232 next bit to send/tape EOT |
| RODATA | $00B6 | 182 | RS232 out byte buffer/read character error |
| FNLEN | $00B7 | 183 | Current filename length |
| LA | $00B8 | 184 | Current logical file number |
| SA | $00B9 | 185 | Current secondary address |
| FA | $00BA | 186 | Current device number |
| FNADR | $00BB-00BC | 187-188 | Pointer: filename address |
| ROPRTY | $00BD | 189 | RS232 out parity/tape read input char |
| FSBLK | $00BE | 190 | Blocks left for tape read/write |
| MYCH | $00BF | 191 | Serial word buffer |
| CAS1 | $00C0 | 192 | Tape motor sensor |
| STAL | $00C1-00C2 | 193-194 | I/O start address |
| MEMUSS | $00C3-00C4 | 195-196 | Kernal setup pointer/tape temp address |
| LSTX | $00C5 | 197 | Last key pressed |
| NDX | $00C6 | 198 | Keyboard queue length |
| RVS | $00C7 | 199 | Flag: reverse chars: on=1, off=0 |
| INDX | $00C8 | 200 | Pointer: end of line for |
| LXSP | $00C9-00CA | 201-202 | Cursor row, column at start of INPUT |
| SFDX | $00CB | 203 | Current key pressed: no key=64 |
| BLNSW | $00CC | 204 | Cursor blink phase: on=1, off=0 |
| BLNCT | $00CD | 205 | Cursor countdown timer |
| GDBLN | $00CE | 206 | Character at cursor position |
| BLNON | $00CF | 207 | Cursor blink phase on/off |
| CRSW | $00D0 | 208 | Flag: INPUT from screen/GET from keyboard |
| PNT | $00D1-00D2 | 209-210 | Pointer: current screen line address |
| PNTR | $00D3 | 211 | Cursor column on current lines |
| QTSW | $00D4 | 212 | Flag: quote mode status: no quotes=0, in quotes >0 |
| LNMX | $00D5 | 213 | Physical screen line length |
| TBLX | $00D6 | 214 | Current row location of cursor |
| | $00D7 | 215 | Last inkey/checksum/buffer temp data |
| INSRT | $00D8 | 216 | Number of inserts outstanding |
| LDTB1 | $00D9-00F2 | 217-242 | Screen line link table |
| USER | $00F3-00F4 | 243-244 | Pointer: current cursor colour RAM location |
| KEYTAB | $00F5-00F6 | 245-246 | Keyboard decode table address |
| RIBUF | $00F7-00F8 | 247-248 | Pointer: RS232 input buffer |
| ROBUF | $00F9-00FA | 249-250 | Pointer: RS232 output buffer |
| FREKZP | $00FB-00FE | 251-254 | Free zero page area |
| BASZPT | $00FF | 255 | BASIC temp data area |
| | $0100-01FF | 256-511 | Processor stack |
| | $0100-010A | 256-266 | Float to ASCII work area |
| BAD | $0100-013E | 256-318 | Tape error log |
| BUF | $0200-025B | 512-600 | System input buffer |
| LAT | $0259-0262 | 601-610 | Logical file table |
| FAT | $0263-026C | 611-620 | File device number table |
| SAT | $026D-0276 | 621-630 | Secondary address table |
| KEYD | $0277-0280 | 631-640 | Keyboard buffer |
| MEMSTR | $0281-0282 | 641-642 | Start of BASIC memory |
| MEMSIZ | $0283-0284 | 643-644 | Top of BASIC memory |
| TIMOUT | $0285 | 645 | Serial bus time out flag |
| COLOR | $0286 | 646 | Current character colour |
| GDCOL | $0287 | 647 | Background colour under cursor |
| HIBASE | $0288 | 648 | Screen location page number |
| XMAX | $0289 | 649 | Size of keyboard buffer |
| RPTFLG | $028A | 650 | Repeat key flag: default=0, repeat all=128, no repeats=64 |
| KOUNT | $028B | 651 | Repeat speed counter |
| DELAY | $028C | 652 | Repeat delay counter |
| SHFLAG | $028D | 653 | Flag: SHIFT=1, CBM=2, CTRL=4 |
| LSTSF | $028E | 654 | Last shift pattern flag |
| KEYLOG | $028F-0290 | 655-656 | Keyboard setup table pointer |
| MODE | $0291 | 657 | Flag: 0=disable shift keys 128=enable shifts |
| AUTODN | $0292 | 658 | Scroll: enable=0 |
| M51CTR | $0293 | 659 | RS232 control register image |
| M51CDR | $0294 | 660 | RS232 command register image |
| M51AJB | $0295-0296 | 661-662 | RS232 non-standard baud rate |
| RSSTAT | $0297 | 663 | RS232 status register image |
| BITNUM | $0298 | 664 | RS232 bits left to send |
| BAUDOF | $0299-029A | 665-666 | RS232 baud rate |
| RIDBE | $029B | 667 | RS232 index to end of input buffer |
| RIDBS | $029C | 668 | RS232 page number of start of input buffer |
| RODBS | $029D | 669 | RS232 page number of start of output buffer |
| RODBE | $029E | 670 | RS232 index to end of output buffer |
| IRQTMP | $029F-02A0 | 671-672 | IRQ vector during tape save |
| ENABL | $02A1 | 673 | RS232 enable/CIA2 (NMI) interrupt control |
| | $02A2 | 674 | CIA 1 timer A control log during tape I/O |
| | $02A3 | 675 | CIA 1 interrupt log tape read |
| | $02A4 | 676 | CIA 1 timer A enable log tape read |
| | $02A5 | 677 | Screen line marker |
| | $02A6 | 678 | PAL/NTSC flag: 0=NTSC, 1=PAL |
| | $02A7-02FF | 679-767 | Unused |
| IERROR | $0300-0301 | 768-769 | Vector: BASIC error messages ($E38B) |
| IMAIN | $0302-0303 | 770-771 | Vector: BASIC warm start ($A483) |
| ICRNCH | $0304-0305 | 772-773 | Vector: BASIC crunch tokens ($A57C) |
| IQPLOP | $0306-0307 | 774-775 | Vector: BASIC print tokens ($A71A) |
| IGONE | $0308-0309 | 776-777 | Vector: BASIC start new line ($A7E4) |
| IEVAL | $030A-030B | 778-779 | Vector: BASIC token evaluate ($AE86) |
| SAREG | $030C | 780 | Save accumulator |
| SXREG | $030D | 781 | Save X register |
| SYREG | $030E | 782 | Save Y register |
| SPREG | $030F | 783 | Save status register |
| USRPOK | $0310 | 784 | USR function jump command ($4C) |
| USRADD | $0311-0312 | 785-786 | USR address low/high form ($B248) |
| | $0313 | 787 | Unused |
| CINV | $0314-0315 | 788-789 | Vector: Hardware IRQ ($EA31) |
| CBINV | $0316-0317 | 790-791 | Vector: BRK interrupt ($FE66) |
| NMINV | $0318-0319 | 792-793 | Vector: NMI ($FE47) |
| IOPEN | $031A-031B | 794-795 | Vector: KERNAL OPEN ($F34A) |
| ICLOSE | $031C-031D | 796-797 | Vector: KERNAL CLOSE ($F291) |
| ICHKIN | $031E-031F | 798-799 | Vector: KERNAL CHKIN ($F20E) |
| ICKOUT | $0320-0321 | 800-801 | Vector: KERNAL CHKOUT ($F250) |
| ICLRCH | $0322-0323 | 802-803 | Vector: KERNAL CLRCHN ($F333) |
| IBASIN | $0324-0325 | 804-805 | Vector: KERNAL CHRIN ($F157) |
| IBSOUT | $0326-0327 | 806-807 | Vector: KERNAL CHROUT ($F1CA) |
| ISTOP | $0328-0329 | 808-809 | Vector: KERNAL STOP ($F6ED) |
| IGETIN | $032A-032B | 810-811 | Vector: KERNAL GETIN ($F13E) |
| ICLALL | $032C-032D | 812-813 | Vector: KERNAL CLALL ($F32F) |
| USRCMD | $032E-032F | 814-815 | Vector: WARM start ($FE66) |
| ILOAD | $0330-0331 | 816-817 | Vector: KERNAL LOAD ($F4A5) |
| ISAVE | $0332-0333 | 818-819 | Vector: KERNAL SAVE ($F5ED) |
| | $0334-033B | 820-827 | Unused |
| TBUFFER | $033C-03FB | 828-1019 | Tape header buffer |
| | $03FC-03FF | 1020-1023 | Unused |
| VICSCN | $0400-07E7 | 1024-2023 | Screen RAM |
| | $07E8-07F7 | 2024-2039 | Unused |
| | $07F8-07FF | 2040-2047 | Sprite block data pointers (0-7) |
| | $0800-9FFF | 2048-40959 | BASIC RAM (TXTTAB-1) |
| | $8000-9FFF | 32768-40959 | Alternate: ROM plug-in area |
| | $A000-BFFF | 40960-49151 | Basic ROM/Alternate RAM |
| | $C000-CFFF | 49152-53247 | RAM memory |
| | $D000-D02E | 53248-53294 | VIC chip registers (6566) /Character set |
| | $D02F-D3FF | 53295-54271 | Unused/Character set |
| | $D400-D41C | 54272-54230 | SID chip (6581)/Character set |
| | $D41D-D4FF | 54231-54527 | Unused/Character set |
| | $D500-D7FF | 54528-55295 | SID images/Character set |
| | $D800-DBFF | 55296-56319 | Colour nybble memory /Character set |
| | $DC00-DC0F | 56320-56335 | CIA 1 Interface IRQ (6526) /Character set |
| | $DC10-DCFF | 56336-56575 | Unused/Character set |

| | | |
|---|---|---|
| $DD00-DD0F | 56576-56591 | CIA 2 Interface NMI (6526) /Character set |
| $DD10-DDFF | 56592-57342 | Unused/Character set |
| $E000-FF80 | 57344-65408 | KERNAL ROM/RAM memory |
| $FF81-FFF5 | 65409-65525 | KERNAL jump table/RAM memory |

MEMORY MAP OF THE COMMODORE 128

| LABEL | HEX | DECIMAL | DESCRIPTION |
|---|---|---|---|
| D6510 | $0000 | 0 | 6510 Direction register |
| R6510 | $0001 | 1 | 6510 I/O, memory and tape |
| BANK | $0002-0004 | 2-4 | Jump call for SYS |
| PREG | $0005 | 5 | .P register for SYS |
| AREG | $0006 | 6 | .A register for SYS |
| XREG | $0007 | 7 | .X register for SYS |
| YREG | $0008 | 8 | .Y register for SYS |
| STKPTR | $0009 | 9 | Search character |
| ENDCHR | $000A | 10 | String scan-quotes flag |
| TRMPOS | $000B | 11 | TAB column |
| VERCK | $000C | 12 | Flag: LOAD=0, VERIFY=1 |
| COUNT | $000D | 13 | Input buffer pointer/ # subscripts |
| DIMFLG | $000E | 14 | Default DIM flag: default=0 |
| VALTYP | $000F | 15 | Data type: string=255, numeric=0 |
| INTFLG | $0010 | 16 | Numeric data type: floating=0, integer=128 |
| GARBFL | $0011 | 17 | DATA scan/LIST quote/ Garbage collect flag |
| SUBFLG | $0012 | 18 | Subscript/FN flag |
| INPFLG | $0013 | 19 | Flag: INPUT=0, GET=64, READ=152 |
| TANSGN | $0014 | 20 | TAN sign/comparison result |
| CHANNL | $0015 | 21 | Current I/O channel |
| LINNUM | $0016-0017 | 22-23 | Integer value |
| TEMPPT | $0018 | 24 | Pointer: temp string stack |
| LASTPT | $0019-001A | 25-26 | Last temp string address |
| TEMPST | $001B-0023 | 27-35 | Stack for temp strings |
| INDEX | $0024-0027 | 36-39 | Utility pointer area |
| RESHO | $0028-002C | 40-44 | Product area for multiplication |
| TXTTAB | $002D-002E | 45-46 | Pointer start of BASIC |
| VARTAB | $002F-0030 | 47-48 | Pointer start of variables |
| ARYTAB | $0031-0032 | 49-50 | Pointer start of arrays |
| STREND | $0033-0034 | 51-52 | Pointer end of arrays +1 |
| FRETOP | $0035-0036 | 53-54 | Pointer bottom of strings |
| FRESPC | $0037-0038 | 55-56 | Utility string pointer |
| MXMEM1 | $0039-003A | 57-58 | Pointer: top of Bank 1 storage |
| CURLIN | $003B-003C | 59-60 | Current BASIC line number |
| TXTPTR | $003D-003E | 61-62 | Current byte of BASIC text |
| FNDPNT | $003F-0040 | 63-64 | Pointer: item found by search |
| DATLIN | $0041-0042 | 65-66 | Current DATA line |
| DATPTR | $0043-0044 | 67-68 | Current DATA address |
| INPPTR | $0045-0046 | 69-70 | INPUT routine vector |
| VARNAM | $0047-0048 | 71-72 | Pointer: current variable name |
| VARPNT | $0049-004A | 73-74 | Pointer: current variable data |
| FORPNT | $004B-004C | 75-76 | Pointer: variable for FOR/NEXT |
| OPPTR | $004D-004E | 77-78 | Operator table displacement |
| OPMASK | $004F | 79 | Comparison symbol accumulator |
| DEFPNT | $0050-0051 | 80-81 | Pointer: current FN descriptor |
| DSCPNT | $0052-0054 | 82-84 | Pointer: current string descriptor |
| HELPER | $0055 | 85 | Flag: HELP/LIST |
| JMPER | $0056-0057 | 86-87 | 6502 JMP to function |
| | $0058-0062 | 88-98 | Temp data area for strings |
| FACEXP | $0063 | 99 | FAC#1 - exponent |
| FACHO | $0064-0067 | 100-103 | FAC#1 - mantissa |
| FACSGN | $0068 | 104 | FAC#1 - sign |
| SGNFLG | $0069 | 105 | Pointer: series evaluation constant |
| ARGEXP | $006A | 106 | FAC#2 - exponent |
| ARGHO | $006B-006E | 107-110 | FAC#2 - mantissa |
| ARGSGN | $006F | 111 | FAC#2 - sign |
| ARISGN | $0070 | 112 | FAC#1/#2 sign comparison result |
| FACOV | $0071 | 113 | FAC#1 - low order rounding |
| FBUFPT | $0072-0073 | 114-115 | Pointer: cassette buffer |
| AUTINC | $0074-0075 | 116-117 | Increment value for AUTO |
| MOFLAG | $0076 | 118 | Graphics area set flag (0=no) |
| NOZE | $0077 | 119 | Sprite temp/zero counter for USING |
| MULP | $0078 | 120 | Counter |
| SYNTMP | $0079 | 121 | Temp for indirect loads |
| DSDESC | $007A-007C | 122-124 | Pointer: DS$ descriptor |
| TOS | $007D-007E | 125-126 | Pointer: top of run-time stack |
| RUNMOD | $007F | 127 | Flag: program/direct modes |
| PARSTS | $0080 | 128 | Disk command syntax check |
| PARSTX | $0081 | 129 | Disk command syntax check |
| OLDSTK | $0082 | 130 | |
| COLSEL | $0083 | 131 | Current colour |
| MULTI1 | $0084 | 132 | Multicolour 1 |
| MULTI2 | $0085 | 133 | Multicolour 2 |
| FORGND | $0086 | 134 | Bit map foreground colour |
| SCALEX | $0087-0088 | 135-136 | Scale factor X |
| SCALEY | $0089-008A | 137-138 | Scale factor Y |
| STOPNB | $008B | | Flag: Stop paint |
| VTEMP | $008C-008F | 139-143 | Temp data area |
| STATUS | $0090 | 144 | Kernal I/O status (ST) |
| STKEY | $0091 | 145 | STOP key/RVS key switch |
| SVXT | $0092 | 146 | Timing constant for tape |
| VERCK | $0093 | 147 | Flag: LOAD=0, VERIFY=1 |
| C3P0 | $0094 | 148 | Flag: Serial bus data flag |
| BSOUR | $0095 | 149 | Serial bus: character for output |
| SYNO | $0096 | 150 | EOT tape signal received |
| | $0097 | 151 | Register save |
| LDTND | $0098 | 152 | Number of files open/File table index |
| DFLTN | $0099 | 153 | Input device (default=0) |
| DFLTO | $009A | 154 | Output device (default=3) |
| PRTY | $009B | 155 | Tape char parity |
| DPSW | $009C | 156 | Flag: tape byte received |
| MSGFLG | $009D | 157 | BASIC mode: Program=0, Direct=128 |
| PTR1 | $009E | 158 | Tape pass 1 error log |
| PTR2 | $009F | 159 | pass 2 error log |
| TIME | $00A0-00A2 | 160-162 | Real-time jiffy clock |
| R2D2 | $00A3 | 163 | Serial bit count/EOI flag |
| BSOUR1 | $00A4 | 164 | Cycle count |
| CNTDN | $00A5 | 165 | Tape sync countdown/bit count |
| BUFPNT | $00A6 | 166 | Pointer: tape I/O buffer |
| INBIT | $00A7 | 167 | RS232 input bits/tape(write ldr/read count) |
| BITCI | $00A8 | 168 | RS232 input bit count tape write ldr/read count |
| RINONE | $00A9 | 169 | Flag: RS232 start bit |
| RIDATA | $00AA | 170 | RS232 input byte buffer/ tape (scan/counter/ldr) |
| RIPRTY | $00AB | 171 | RS232 input parity/ tape (write ldr length/read checksum) |
| SALK | $00AC-00AD | 172-173 | Pointer: tape buffer/screen scrolling |
| EALK | $00AE-00AF | 174-175 | Tape program end address |
| CMP0 | $00B0-00B1 | 176-177 | Tape timing constants |
| TAPE1 | $00B2-00B3 | 178-179 | Pointer: start of tape buffer |
| BITTS | $00B4 | 180 | RS232 out bit count/tape timer enabled=1 |
| NXTBIT | $00B5 | 181 | RS232 next bit to send/tape EOT |
| RODATA | $00B6 | 182 | RS232 out byte buffer/read character error |
| FNLEN | $00B7 | 183 | Current filename length |
| LA | $00B8 | 184 | Current logical file number |
| SA | $00B9 | 185 | Current secondary address |
| FA | $00BA | 186 | Current device number |
| FNADR | $00BB-00BC | 187-188 | Pointer: filename address |
| ROPRTY | $00BD | 189 | RS232 out parity/tape read input char |
| FSBLK | $00BE | 190 | Blocks left for tape read/write |
| MYCH | $00BF | 191 | Serial word buffer |
| CAS1 | $00C0 | 192 | Tape motor sensor |
| STALK | $00C1-00C2 | 193-194 | I/O start address |
| MEMUSS | $00C3-00C4 | 195-196 | Kernal setup pointer/tape temp address |
| DATA | $00C5 | 197 | Tape read/write data |
| BA | $00C6 | 198 | Bank for LOAD/SAVE/VERIFY |
| FNBANK | $00C7 | 199 | Bank holding filename (FNADR) |
| RIBUF | $00C8-00C9 | 200-201 | Pointer: RS232 input buffer |
| ROBUF | $00CA-00CB | 201-202 | Pointer: RS232 output buffer |
| KEYTAB | $00CC-$00CD | 204-205 | Pointer: keyboard table |
| IMPARM | $00CE-$00CF | 206-207 | Pointer: String for Kernal PRIMM |
| NDX | $00D0 | 208 | Index to keyboard buffer |
| KYNDX | $00D1 | 209 | Flag: Function keypress |
| KEYIDX | $00D2 | 210 | Index to function key string |
| SHFLG | $00D3 | 211 | Flag: SHIFT=1, CBM=2, CTRL=4 |
| SFDX | $00D4 | 212 | Current key pressed 64=no key |
| LSTX | $00D5 | 213 | Last key pressed |
| CRSW | $00D6 | 214 | Flag: INPUT from screen or GET from keyboard |
| MODE | $00D7 | 215 | Flag: 40/80 columns 0=40 columns |
| GRAPHM | $00DB | 216 | Current GRAPHIC mode |
| CHAREN | $00D9 | 217 | Flag: VIC fetch from ROM/RAM |
| | $00DA-00DF | 218-223 | Programmable key variables |
| PNT | $00E0-00E1 | 224-225 | Pointer: screen line |
| USER | $00E2-00E3 | 226-227 | Pointer: current cursor colour RAM location |
| SCTOP | $00E4 | 228 | Top line of window |
| SCBOT | $00E5 | 229 | Bottom line of window |
| SCLF | $00E6 | 230 | Left row of window |
| SCRT | $00E7 | 231 | Right row of window |
| LSXP | $00E8 | 232 | INPUT start column |
| LSTP | $00E9 | 233 | INPUT start line |
| INDX | $00EA | 234 | INPUT end line |
| TBLX | $00EB | 235 | Cursor row |
| PNTR | $00EC | 236 | Cursor column |
| LINES | $00ED | 237 | Maximum number of screen lines |
| COLUMS | $00EE | 238 | Maximum number of screen columns |
| DATAX | $00EF | 239 | Current character for printing |
| LSTCHR | $00F0 | 240 | Previous character printed (ESC test) |
| COLOR | $00F1 | 241 | Current character colour |

| | | | |
|---|---|---|---|
| TCOLOR | $00F2 | 242 | Saved character colour for INST/DEL |
| RVS | $00F3 | 243 | Flag: RVS characters 0=off 1=on |
| QTSW | $00F4 | 244 | Flag: 1=quotes mode on 0=edit mode |
| INSRT | $00F5 | 245 | Number of inserts outstanding |
| INSFLG | $00F6 | 246 | Flag: Auto-insert mode |
| LOCKS | $00F7 | 247 | FLag: SHIFT or CBM pressed |
| SCROLL | $00F8 | 248 | Screen scroll disable 0=enabled |
| BEEPER | $00F9 | 249 | CTRL-G disable |
| FREKZP | $00FA-00FF | 250-255 | Free zero page area |
| FBUFFER | $0100-010F | 256-271 | Filename construction area |
| XCNT | $0110 | 272 | DOS loop counter |
| DOSF1L | $0111 | 273 | Length of DOS Filename 1 |
| DOSDS1 | $0112 | 274 | First drive number |
| DOSF1A | $0113-0114 | 275-276 | Address of DOS filename 1 |
| DOSF2L | $0115 | 277 | Length of DOS filename 2 |
| DOSDS2 | $0116 | 278 | Second drive number |
| DOSF2A | $0117-0118 | 279-280 | Address of DOS filename 2 |
| DOS0FL | $0119-011A | 281-282 | Start address for BLOAD/BSAVE |
| DOS0FH | $011B-011C | 283-284 | End address for BSAVE |
| DOSLA | $011D | 285 | DOS logical file number |
| DOSFA | $011E | 286 | DOS device number |
| DOSSA | $011F | 287 | DOS secondary address |
| DOSRCL | $0120 | 288 | DOS record length |
| DOSBNK | $0121 | 289 | DOS bank number |
| DOSDID | $0122-0123 | 290-291 | DOS identifier |
| DIDCHK | $0124 | 292 | DOS did flag |
| BNR | $0125 | 293 | Pointer. USING begin number |
| ENR | $0126 | 294 | Pointer: USING end number |
| DOLR | $0127 | 295 | Flag: USING dollar |
| FLAG | $0128 | 296 | Flag: USING comma |
| SWO | $0129 | 297 | USING counter |
| USGN | $012A | 298 | Sign exponent |
| UEXP | $012B | 299 | Pointer: exponent |
| UN | $012C | 300 | Number of digits before decimal point |
| CHSN | $012D | 301 | Using justify flag |
| VF | $012E | 302 | Number of field characters before decimal point |
| NF | $012F | 303 | Number of field decimal places |
| POSP | $0130 | 304 | Flag: +/- in USING field |
| FESP | $0131 | 305 | Flag: USING exponent |
| ETOF | $0132 | 306 | Switch |
| CFORM | $0133 | 307 | Field character counter |
| SND | $0134 | 308 | Sign number |
| BLFD | $0135 | 309 | Flag: blank or asterisk |
| BEGFD | $0136 | 310 | Pointer: beginning of field |
| LFOR | $0137 | 311 | Length of format |
| ENDFD | $0138 | 312 | Pointer: end of field |
| STACK | $0139-01FF | 313-511 | System stack |
| BUF | $0200-0258 | 512-600 | System input buffer for BASIC and MONITOR |
| FETCH | $02A2 | 674 | Subroutine: LDA( ),Y from any bank |
| STASH | $02AF | 687 | Subroutine: STA( ),Y to any bank |
| CMPARE | $02BE | 702 | Subroutine: CMP( ),Y in any bank |
| JSRFAR | $02CD | 717 | JSR to any bank |
| JMPFAR | $02E3 | 739 | JMP to any bank |
| ICRNCH | $030C-030D | 780-781 | Vector: BASIC crunch tokens |
| IQPLOP | $030E-030F | 782-783 | Vector: LIST |
| IEVAL | $0310-0311 | 784-785 | Vector: execute hook |
| IGONE | $0312-0313 | 786-787 | Vector: BASIC character despatch |
| IIRQ | $0314-0315 | 788-789 | Vector: Hardware IRQ |
| IBRK | $0316-0317 | 790-791 | Vector: BRK interrupt |
| INMI | $0318-0319 | 792-793 | Vector: NMI |
| IOPEN | $031A-031B | 794-795 | Vector: KERNAL OPEN |
| ICLOSE | $031C-031D | 796-797 | Vector: KERNAL CLOSE |
| ICHKIN | $031E-031F | 798-799 | Vector: KERNAL CHKIN |
| ICKOUT | $0320-0321 | 800-801 | Vector: KERNAL CHKOUT |
| ICLRCH | $0322-0323 | 802-803 | Vector: KERNAL CLRCHN |
| IBASIN | $0324-0325 | 804-805 | Vector: KERNAL CHRIN |
| IBSOUT | $0326-0327 | 806-807 | Vector: KERNAL CHROUT |
| ISTOP | $0328-0329 | 808-809 | Vector: KERNAL STOP |
| IGETIN | $032A-032B | 810-811 | Vector: KERNAL GETIN |
| ICLALL | $032C-032D | 812-813 | Vector: KERNAL CLALL |
| EXMON | $032E-032F | 814-815 | Vector: indirect monitor commands |
| ILOAD | $0330-0331 | 816-817 | Vector: KERNAL LOAD |
| ISAVE | $0332-0333 | 818-819 | Vector: KERNAL SAVE |
| CTLVEC | $0334-0335 | 820-821 | Vector: CTRL code link |
| SHFVEC | $0336-0337 | 822-823 | Vector: SHIFT code link |
| ESCVEC | $0338-0339 | 824-825 | Vector: ESC sequence link |
| KEYVEC | $033A-033B | 826-827 | Vector: keyscan (indirect) |
| KEYCHK | $033C-033D | 828-829 | Vector: store keypress |
| DECODE | $033E-033F | 830-831 | Vector: keyboard decode tables |
| | $0340-0349 | 832-841 | |
| KEYD | $034A-0353 | 842-851 | Keyboard buffer |
| TABMAP | $0354-035D | 852-861 | Bit map TAB stops |
| BITABL | $035E-0361 | 862-865 | Screen line link table |
| LAT | $0362-036B | 866-875 | Logical file table |
| FAT | $036C-036D | 876-877 | Device number table |
| SAT | $036E-037F | 878-895 | Secondary address table |
| CHRGET | $0380-039E | 896-926 | Subroutine: get next BASIC byte |
| CHRGOT | $0386-039E | 902-926 | Subroutine: get current BASIC byte |
| INDSB1 | $039F-03AA | 927-938 | Subroutine: fetch into RAM 0 |
| INDSB2 | $03AB-03B6 | 939-950 | Subroutine: fetch into RAM 1 |
| INDIN1 | $03B7-03BF | 951-959 | Subroutine: fetch INDEX1 indirect |
| INDIN2 | $03C0-03C8 | 960-968 | Subroutine: fetch INDEX2 indirect |
| INDIXT | $03C9-03D1 | 969-977 | Subroutine: fetch IXIPTR indirect |
| ZERO | $03D2-03D4 | 978-980 | Floating point constant from ROM |
| CURBA | $03D5 | 981 | Bank for PEEK/POKE/SYS |
| TMPDES | $03D6 | 982 | Temp area for INSTR |
| FINBNK | $03DA | 986 | Bank for string-number conversion |
| SAVSIZ | $03DB-03DE | 987-990 | Temp area for SSHAPE |
| BITS | $03DF | 991 | FAC#1 overflow digit |
| SPRTMP | $03E0-03FF | 992-1023 | Temp area for SPRSAV |
| VICSCN | $0400-07E7 | 1024-2023 | 40 column screen memory |
| SPRPTR | $07E8-07FF | 2024-2047 | Sprite pointers |
| RUNSTK | $0800-09FF | 2048-2559 | BASIC pseudo stack |
| SVECT | $0A00-0A01 | 2560-2561 | Vector: restart system |
| DEJAVU | $0A02 | 2562 | Warm/cold start status |
| PALNTS | $0A03 | 2563 | Flag: PAL/NTSC |
| INITST | $0A04 | 2564 | Flag: Reset vs NMI for initialisation |
| MEMSTR | $0A05-0A06 | 2565-2566 | Bottom of system bank memory |
| MEMSIZ | $0A07-0A08 | 2567-2568 | Top of system bank memory |
| IRQTEMP | $0A09-0A0A | 2569-2570 | Temp store for IRQ vector during tape I/O |
| CASTON | $0A0B | 2571 | TOD sence during tape ops |
| STUPID | $0A0C-0A0D | 2572-2573 | Tape read temps |
| TIMOUT | $0A0E | 2574 | Serial bus time out flag |
| ENABL | $0A0F | 2575 | RS232 enable (NMI) interrupt control |
| MS1CTR | $0A10 | 2576 | RS232 control register image |
| MS1CDR | $0A11 | 2577 | RS232 command register image |
| MS1AJB | $0A12-0A13 | 2578-2579 | RS232 non-standard baud rate |
| RSSTAT | $0A14 | 2580 | RS232 status register image |
| BITNUM | $0A15 | 2581 | RS232 bits left to send |
| BAUDOF | $0A16-0A17 | 2582-2583 | RS232 baud rate |
| RIDBE | $0A18 | 2584 | RS232 index to end of input buffer |
| RIDBS | $0A19 | 2585 | RS232 page number of start of input buffer |
| RODBS | $0A1A | 2586 | RS232 page number of start of output buffer |
| RODBE | $0A1B | 2587 | RS232 index to end of output buffer |
| SERIAL | $0A1C | 2588 | Flag: fast serial internal/external op |
| TIMER | $0A1D-0A1F | 2589-2591 | Decrementing jiffy register |
| XMAX | $0A20 | 2592 | Size of keyboard buffer |
| PAUSE | $0A21 | 2593 | Flag: CTRL-S |
| RPTFLG | $0A22 | 2594 | Repeat key flag: default=0, repeat all=128, no repeats=64 |
| KOUNT | $0A23 | 2595 | Repeat speed counter |
| DELAY | $0A24 | 2596 | Repeat delay counter |
| LSTSHF | $0A25 | 2597 | Last shift pattern flag |
| BLNON | $0A26 | 2598 | Flag: VIC cursor blink |
| BLNSW | $0A27 | 2599 | VIC cursor blink enable |
| BLNCT | $0A28 | 2600 | VIC cursor blink timer |
| GDBLN | $0A29 | 2601 | VIC character under cursor |
| GDCOL | $0A2A | 2602 | VIC background colour under cursor |
| CURMOD | $0A2B | 2603 | VDC active cursor mode |
| VM1 | $0A2C | 2604 | VIC text screen start page |
| VM2 | $0A2D | 2605 | VIC bit map start page |
| VM3 | $0A2E | 2606 | VDC text screen base |
| VM4 | $0A2F | 2607 | VDC colour map |
| LINTMP | $0A30 | 2608 | Temp pointer for LOOP |
| SAVB0 | $0A31-0A34 | 2609-2612 | Temp data for VDC screen handling |
| CURCOL | $0A35 | 2613 | VDC colour under cursor |
| SPLIT | $0A36 | 2614 | VIC split screen raster value |
| FNADRX | $0A37 | 2615 | X register save for bank ops |
| PALCNT | $0A38 | 2616 | Jiffy adjustment for PAL system |
| XCNT | $0A80-0A9F | 2688-2719 | MLM compare buffer |
| | $0AA0-0AAA | 2720-2730 | MLM temp data |
| LENGTH | $0AAB | 2731 | Flag: Assemble/disassemble |
| | $0AAC-0AB1 | 2732-2737 | Temp MLM values |
| XSAV | $0AB2 | 2738 | X save during indirect subroutine calls |
| DIRCTN | $0AB3 | 2739 | Direction indicator for transfer |
| TEMPS | $0AB4-0ABF | 2740-2751 | MLM temps |
| CURBANK | $0AC0 | 2752 | Function key ROM bank being polled |
| PAT | $0AC1-0AFF | 2753-2815 | Table of logged ROM cards |
| TBUFFR | $0B00-0BBF | 2816-3007 | Tape buffer |
| | $0BC0-0BFF | 3008-3071 | Disk boot page |
| RS232I | $0C00-0CFF | 3072-3327 | RS232 input buffer |
| RS232O | $0D00-0DFF | 3328-3583 | RS232 output buffer |
| | $0E00-0FFF | 3584-4095 | Free space |
| PKTBUF | $1000-1009 | 4096-4105 | Function key string lengths table |
| PKYDEF | $100A-10FF | 4106-4351 | Function key definition table |
| | $1100-110B | 4352-4360 | CP/M reset subroutine |
| XPOS | $1131-1132 | 4401-4402 | Current pixel X position |
| YPOS | $1133-1134 | 4403-4404 | Current pixel Y position |
| XDEST | $1135-1136 | 4405-4406 | X co-ordinate destination |
| YDEST | $1137-1138 | 4407-4408 | Y co-ordinate destination |
| XABS | $1139-113A | 4409-4410 | X position for DRAW |
| YABS | $113B-113C | 4411-4412 | Y position for DRAW |
| XSGN | $113D-113E | 4413-4414 | X parameter sign |
| YSGN | $113F-1140 | 4415-4416 | Y parameter sign |

| | | | |
|---|---|---|---|
| | $1141-1144 | 4417-4420 | Line drawing temps |
| ERRVAL | $1145-1146 | 4421-4422 | Graphics error value |
| LESSER | $1147 | 4423 | Graphics lesser marker |
| GREATR | $1148 | 4424 | Graphics greater marker |
| ANGSGN | $1149 | 4425 | Sign of angle |
| SINVAL | $114A-114B | 4426-4427 | Sin value of angle |
| COSVAL | $114C-114D | 4428-4429 | Cosine value of angle |
| ANGCNT | $114E-114F | 4430-4431 | Temps for angle-distance routines |
| XCIRCL | $1150-1151 | 4432-4433 | CIRCLE centre X pos/BOX point 1 X |
| YCIRCL | $1152-1153 | 4434-4435 | CIRCLE centre Y pos/BOX point 1 Y |
| STRSZ | $1153 | 4435 | Shape string length |
| XRADUS | $1154-1155 | 4436-4437 | CIRCLE X radius/BOX rotation angle |
| GETTYP | $1154 | 4436 | Replace shape mode |
| STRPTR | $1155 | 4437 | String position counter |
| YRADUS | $1156-1157 | 4438-4439 | CIRCLE Y radius |
| OLDBYT | $1156 | 4438 | Old bit map byte |
| NEWBYT | $1157-1158 | 4439-4440 | New string or bit map byte |
| ROTANG | $1158-1159 | 4440-4441 | Circle rotation angle |
| XSIZE | $1159-115A | 4441-4442 | Shape - column length |
| BOXLEN | $115A-115B | 4442-4443 | BOX legth of a side |
| YSIZE | $115B-115C | 4443-4444 | Shape - row length |
| ANGBEG | $115C-115D | 4444-4445 | Arc angle start |
| ANGEND | $115E-115F | 4446-4447 | Arc angle end |
| STRADR | $115F-1160 | 4447-4448 | Save shape string descriptor |
| XRCOS | $1160-1161 | 4448-4449 | X radius * COS(angle) |
| BITIDX | $1161 | 4449 | Bit index into byte |
| YRSIN | $1162-1163 | 4450-4451 | Y radius * SIN(angle) |
| XRSIN | $1164-1165 | 4452-4453 | X radius * SIN(angle) |
| YRCOS | $1166-1167 | 4454-4455 | Y radius * COS(angle) |
| CHRPAG | $1168 | 4456 | High byte of character ROM address |
| BITCNT | $1169 | 4457 | Temp for GSHAPE |
| SCALEM | $116A | 4458 | Flag: scale mode |
| WIDTH | $116B | 4459 | Flag: double width |
| FILFLG | $116C | 4460 | Flag: fill box |
| BITMSK | $116D-116E | 4461-4462 | Temp for bitmask |
| TRCFLG | $116F | 4463 | 0=trace off, 255=trace on |
| RENUM | $1170-1173 | 4464-4467 | Temps for RENUMBER |
| GTEMP | $1174-1179 | 4468-4473 | Graphics temp storage |
| ADRAY1 | $117A-117B | 4474-4475 | Flag: convert floating point to integer |
| ADRAY2 | $117C-117D | 4476-4477 | Flag: convert integer to floating point |
| SDATA | $117E-11D5 | 4478-4565 | Sprite speed and direction table |
| VICSAV | $11D6-11FF | 4566-4607 | Copy of VIC registers |
| OLDLIN | $1200-1201 | 4608-4609 | Previous BASIC line |
| OLDTXT | $1202-1203 | 4610-4611 | BASIC statement for CONT |
| FUFILL | $1204 | 4612 | Fill symbol for USING |
| FUCOMA | $1205 | 4613 | Comma symbol for USING |
| FUDOT | $1206 | 4614 | Decimal point symbol for USING |
| FUMONY | $1207 | 4615 | Dollar/pound symbol for USING |
| ERRNUM | $1208 | 4616 | Last error number |
| ERRLIN | $1209-120A | 4617-4618 | Last error line number (65535=none) |
| TRAPNO | $120B-120C | 4619-4620 | Line number for TRAP (255=off) |
| TMPTRP | $120D-120F | 4621-4623 | Temp for TRAP number |
| TXTTOP | $1210-1211 | 4624-4625 | Pointer: top of BASIC text |
| MXMEM0 | $1212-1213 | 4626-4627 | Pointer: top of bank 0 storage |
| TMPTXT | $1214-1217 | 4628-4631 | DO/LOOP temp |
| USRPOK | $1218-121A | 4632-4634 | USR vector code |
| RNDX | $121B-121F | 4635-4639 | RND seed value |
| CIRCLE | $1220 | 4640 | Degrees per circle segment |
| DEJAVU | $1221 | 4641 | Cold/warm reset status |
| TEMPO | $1222 | 4642 | Tempo rate |
| VOICES | $1223-1228 | 4643-4648 | |
| NTIME | $1229-122A | 4649-4650 | |
| OCTAVE | $122C | 4652 | |
| PITCH | $122D-122E | 4653-4654 | |
| VOICE | $122F | 4655 | |
| WAVE0 | $1230-1232 | 4656-4658 | |
| DNOTE | $1233 | 4659 | |
| FLTSAV | $1234-1237 | 4660-4663 | |
| FLTFLG | $1238 | 4664 | |
| NIBBLE | $1239 | 4665 | Temp stor for ENVELOPE parameters |
| TONNUM | $123A | 4666 | Current ENVELOPE number |
| TONVAL | $123B-123D | 4667-4669 | Current ADSR and waveform |
| PARCNT | $123E | 4670 | Counter for envelope parameters |
| ATKTAB | $123F-1248 | 4671-4680 | ENVELOPE attack/decay table |
| SUSTAB | $1249-1252 | 4681-4690 | ENVELOPE sustain/release table |
| WAVTAB | $1253-125C | 4691-4700 | ENVELOPE waveform table |
| PULSLO | $125D-1266 | 4701-4710 | Pulse width low byte table |
| PULSHI | $1267-1270 | 4711-4720 | Pulse width high byte table |
| FILTRS | $1271-1275 | 4721-4725 | Filter values table |
| IRPFLG | $1276-1278 | 4726-4728 | Flags: interrupt handling tripped |
| SSINTL | $1279 | 4729 | Line for sprite-sprite collision IRQ handling (low) |
| SDINTL | $127A | 4730 | Line for sprite-data collision IRQ handling (low) |
| SPINTL | $127B | 4731 | Line for lightpen IRQ handling (low) |
| SSINTH | $127C | 4732 | Line for sprite-sprite IRQ (hi) |
| SDINTH | $127D | 4733 | Line for sprite-data IRQ (hi) |
| SPINTH | $127E | 4734 | Line for lightpen IRQ (hi) |
| INTVAL | $127F | 4735 | Flag: collision enabled |
| COLTYP | $1280 | 4736 | Collision interrupt type |
| VOICE | $1281 | 4737. | Voice number for SOUND |
| TIMELO | $1282-1284 | 4738-4740 | SOUND time low bytes |
| TIMEHI | $1285-1287 | 4741-4743 | SOUND time hi bytes |
| MAXLO | $1288-128A | 4744-4746 | SOUND |
| MAXHI | $128B-128D | 4747-4749 | SOUND |
| MINLO | $128E-1290 | 4750-4752 | SOUND |
| MINHI | $1291-1293 | 4753-4755 | SOUND |
| DIRCTN | $1294-1296 | 4756-4758 | SOUND direction table |
| STEPLO | $1297-1299 | 4759-4761 | SOUND step values low byte table |
| STEPHI | $129A-129C | 4762-4764 | SOUND step values hi byte table |
| FREQLO | $129D-129F | 4765-4767 | SOUND frequency values lo-byte table |
| FREQHI | $12A0-12A2 | 4768-4770 | SOUND Frequency values hi-byte table |
| TIME | $12A3-12A4 | 4771-4772 | Duration for SOUND |
| | $12A5-12B0 | 4773-4784 | Temps for SOUND |
| POTMP | $12B1-12B2 | 4785-4786 | Temp store for lightpen co-ordinates |
| | $12B7-12FF | 4791-4863 | SPRSAV/SPRDEF storage |

## COMMODORE 128 MEMORY OVERVIEW

| HEX | DECIMAL | DESCRIPTION |
|---|---|---|
| $0000-12FF | 0- 4863 | BASIC workspace |
| $4000-AA6D | 16384-43629 | BASIC ROM |
| $AA6E-AEFF | 43630-44799 | Empty ROM space |
| $AF00-AFA7 | 44800-44967 | BASIC jump table |
| $AFA8-AFFF | 44968-45055 | Empty ROM space |
| $B000-BFFF | 45056-49151 | MONITOR |
| $C000-CFFF | 49152-53247 | Screen/keyboard routines |
| $D000-D02E | 53248-53294 | VIC chip (as C64) |
| $D02F | 53295 | 128 mode extra keyboard lines (KEYLIN) |
| $D030 | 53296 | 128 mode system clock speed register |
| $D400-D41C | 54272-54300 | SID chip (as C64) |
| $D500 | 54528 | MMU primary configuration register |
| $D501 | 54529 | MMU Preconfiguration register A |
| $D502 | 54530 | MMU Preconfiguration register B |
| $D503 | 54531 | MMU Preconfiguration register C |
| $D504 | 54532 | MMU Preconfiguration register D |
| $D505 | 54533 | MMU mode configuration register |
| $D506 | 54534 | MMU RAM configuration register |
| $D507 | 54535 | Page 0 pointer lo |
| $D508 | 54536 | Page 0 pointer hi |
| $D509 | 54537 | Page 1 pointer lo |
| $D50A | 54538 | Page 1 pointer hi |
| $D50B | 54539 | MMU version/reset register |
| $D600 | 54784 | VDC address register |
| $D700 | 55040 | VDC data register |
| $E000-FC3D | 57344-64573 | Kernal ROM |
| $FC3E-FEFF | 64574-65279 | Unused ROM |
| $FF00-FF46 | 65280-65350 | MMU registers |
| $FF47-FFF3 | 65351-65523 | Kernal jump table |
| $FFF4-FFFF | 65524-65535 | Hardware vectors |

## USEFUL BASIC INTERPRETER ADDRESSES

| C64 HEX | C128 HEX | DESCRIPTION OF ROUTINE |
|---|---|---|
| $A000 | $4000 | Start vector |
| $A002 | | NMI vector |
| $A004 | | 'CBMBASIC' |
| $A00C | $46FC | Addresses of the BASIC commands minus 1 |
| $A052 | $47D8 | Addresses of the BASIC functions |
| $A080 | $4828 | Hierarchy-codes and addresses of the BASIC operators |
| $A09E | $4417 | List of BASIC command words |
| $A19E | $4848 | BASIC error messages |
| $A364 | | Messages of the BASIC interpreter |
| $A38A | $4FAA | Stack search-routine for FOR-NEXT and GOSUB |
| $A3B8 | $7C66 | Block-shifting routine |
| $A3FB | $4FFE | Checks on space in stack |
| $A408 | $5017 | Makes space in memory |
| $A435 | | Output of 'Out of memory' |
| $A437 | $4D3C | Output of error messages |
| $A469 | $4DA5 | Break vector |
| $A474 | $4D37 | Ready vector |
| $A480 | $4DC3 | Input waiting-loop |
| $A49C | $4DE2 | Clear and inserting program lines |
| $A533 | $4F4F | Tie BASIC program lines anew |
| $A560 | $4F93 | Gets a line into input buffer |
| $A571 | | Output of 'String too long' |
| $A579 | $430A | Change of a line into interpreter-code |
| $A613 | $5064 | Look for start address of a BASIC line |
| $A642 | $51D6 | BASIC-command NEW |
| $A65E | $51F8 | BASIC-command CLR |
| $A68E | $5254 | Set program pointer to BASIC start |
| $A69C | $50E2 | BASIC-command LIST |
| $A717 | $514E | Change interpreter code to command word |
| $A742 | $5DF9 | BASIC-command FOR |
| $A7AE | $4AF6 | Interpreter loop, carries out BASIC commands |
| $A7E4 | $4A9F | Execute next BASIC statement |
| $A7ED | $4B74 | Carries out BASIC command |
| $A81D | $5ACA | BASIC-command RESTORE |
| $A82C | $4BC1 | Interrupts program at pressed stop-key |
| $A82F | $4BCB | BASIC-command STOP |
| $A831 | $4BCD | BASIC-command END |
| $A857 | $5A60 | BASIC-command CONT |
| $A871 | $5A9B | BASIC-command RUN |

```
$A883 $59CF BASIC-command GOSUB
$A8A0 $59DB BASIC-command GOTO
$A8D2 $5262 BASIC-command RETURN
$A8EB $528F BASIC-command DATA
$A906 $52A2 Looks for next statement
$A909 $52A5 Looks for next line
$A928 $52C5 BASIC-command IF
$A93B $529D BASIC-command REM
$A948 $53A3 BASIC-command ON
$A96B $50A0 Looks for address of a BASIC line
$A9A5 $53C6 BASIC-command LET
$AA80 $553A BASIC-command PRINT#
$AA86 $5540 BASIC-command CMD
$AAA0 $555A BASIC-command PRINT
$AB1E $55E2 Output string
$AB3E       Output empty character (Or cursor right)
$AB4D $574D Error handling for INPUT
$AB7B $5612 BASIC-command GET
$ABA5 $5648 BASIC-command INPUT#
$ABBF $5662 BASIC-command INPUT
$ABF9 $569C Print INPUT prompt and handle input
$AC06 $56A9 BASIC-command READ
$ACFC       Output '?extra ignored' and '?redo from start'
$AD1E $57F4 BASIC-command NEXT
$ADBA $77D7 Evaluate numeric expression
$ADBD $77DA Checks on numeric
$ADBF $77DD Checks on string
$AD99       Output of 'Type mismatch'
$AD9E $77EF Evaluate expression
$AE83 $78D7 Get arithmetic term
$AEA8 $78FE Floating point constant for PI
$AED4 $7930 BASIC-command NOT
$AEF1 $7950 Gets term in parenthesis
$AEF7 $7956 Checks on parenthesis closed
$AEFA $7959 Checks on parenthesis open
$AEFD $795C Checks on comma
$AEFF $795E Checks on characters in accumulator
$AF08 $796C Output of 'Syntax error'
$AF28 $797B Gets variable
$AFA7 $4BF7 Set up references
$AFE6 $4CB6 BASIC-command OR
$AFE9 $4C89 BASIC-command AND
$B016 $4CB6 Comparison operations
$B081 $587B BASIC-command DIM
$B08B $7AAF Search for or create variable descriptor
$B113       Checks for letter
$B11D $7B46 Create new 7 byte descriptor
$B185 $7B4A Return address of variable
$B194       Calculates pointer to first array-element
$B1A5 $849A Floating point constant -32768
$B1AA $849F Change FAC to INTEGER
$B1B2 $84A7 Input and convert floating to integer
$B1BF $84B4 FAC integer
$B1D1 $7CAB Search for or create array
$B245 $7D25 Output of 'Bad subscript'
$B248 $7D28 Output of 'Illegal-quantity'
$B34C       Calculates array size
$B37D $8000 BASIC-function FRE
$B391 $8C70 Integer to FAC
$B39E $84D0 BASIC-function POS
$B3A6 $84D9 Checks on direct-mode
$B3AB       Output of 'Illegal direct'
$B3AE       Output of 'UndeF'd function'
$B3B3 $847A BASIC-command DEF
$B3E1 $8528 Checks on FN syntax
$B3F4 $853B BASIC-function FN
$B465 $85AE BASIC-function STR$
$B475       String administration, calculate pointer on
            string
$B487 $869A Establish string
$B4F4 $9299 Allocate string memory space
$B526 $92EA Garbage collection, remove unwanted strings
$B5BD $93B3 Is current string highest in memory?
$B63D $870D String concatenate '+'
$B67A $874E Transfer string to memory
$B6A3 $877E String administration FRESTR
$B6DB $87E0 Delete entry from temp string stack
$B6EC $85BF BASIC-function CHR$
$B700 $85D6 BASIC-function LEFT$
$B72C $860A BASIC-function RIGHT$
$B737 $861C BASIC-function MID$
$B761 $864D Pull string parameters off stack
$B77C $8668 BASIC-function LEN
$B782 $866E Get string parameter
$B78B $8677 BASIC-function ASC
$B79B $87F1 Gets byte term (0-255)
$B7AD $804A BASIC-function VAL
$B7EB $8803 Gets address (0-65535) and byte value (0-255)
$B7F7       Change FAC to address-format (Range 0-65535)
$B80D $80C5 BASIC-function PEEK
$B824 $80ED BASIC-command POKE
$B82D $6C2D BASIC-command WAIT
$B849 $8A0E FAC = FAC + 0.5
$B850 $8B2E Minus FAC = constant (A/Y) - FAC
$B853 $8B31 Minus FAC = ARG - FAC
$B867 $8A45 Plus FAC = constant (A/Y) - FAC
$B86A $8B48 Plus FAC = ARG + FAC
$B947 $8926 Complement FAC
$B97E $895D Output of 'Overflow'
$B983 $8962 Single byte multiply
$B98C $899C Floating point constant for LOG
$B9EA $89CA BASIC-function LOG
$BA28 $8A0B Multiplication FAC = constant (A/Y) * FAC
$BA2B $8A0B Multiplication FAC = ARG * FAC
$BA8C $8AB9 ARG = constant (A/Y)
$BAB7 $8AEC Add exponent FAC to Exponent of ARG
$BAE2 $8B17 FAC = FAC * 10
$BAF9 $8B2E Floating point constant 10
$BAFE $8B38 FAC = FAC/10
$BB07 $8B3F Divide ARG by memory
$BB0F $8B49 FAC = constant (A/Y) / FAC
$BB12 $8B4C FAC = ARG/FAC
$BB8A $8B33 Output of 'Division by zero'
$BBA2 $8BD4 FAC = constant (A/Y)
$BBC7 $8BF9 Accu#4 = FAC
$BBCA $8BFC Accu#3 = FAC
$BBD0 $8C00 Variable = FAC
$BBFC $8C2B FAC = ARG
$BC0C $8C3B ARG = FAC
$BC0F $8C3B Move FAC to ARG
$BC1B $8C47 Round FAC
$BC2B $8C57 Get signs of FAC
$BC39 $8C65 BASIC-function SGN
$BC58 $8C84 BASIC-function ABS
$BC5B $8C58 Compare constant (A/Y) with FAC
$BC9B $8CC7 Change from FAC to integer
$BCCC $8CFB BASIC-function INT
$BCF3 $8D22 Change ASCII to floating point
$BD7E $8DB4 Get new ASCII digit
$BDB3 $8E17 Floating point constants for floating point to
            ASCII
$BDC2 $8E26 Output of line number at error message
$BDCD $8E32 Output of positive integer number (0-65535)
$BDDD $8E42 Change FAC to ASCII format
$BF11 $8F76 Floating point constant 0.5
$BF16 $8F81 Binary numbers for change of FAC to ASCII
$BF71 $8FB7 BASIC-function SQR
$BF78       FAC = constant (A/Y) to the power of FAC
$BF7B $8FC1 FAC = ARG to the power of FAC
$BFB4 $8FFA BASIC negation function
$BF8F $9005 Floating point constant for EXP
$BFED $9033 BASIC-function EXP
```

VIC CHIP ADDRESSES: $D000-$D02E (53248-53294)

| ADDRESS HEX | DECIMAL | BIT | DESCRIPTION |
|---|---|---|---|
| $D000 | 53248 | | Sprite 0 - X position (bits 0-8) |
| $D001 | 53249 | | Sprite 0 - Y position (bits 0-8) |
| $D002 | 53250 | | Sprite 1 - X position |
| $D003 | 53251 | | Sprite 1 - Y position |
| $D004 | 53252 | | Sprite 2 - X position |
| $D005 | 53253 | | Sprite 2 - Y position |
| $D006 | 53254 | | Sprite 3 - X position |
| $D007 | 53255 | | Sprite 3 - Y position |
| $D008 | 53256 | | Sprite 4 - X position |
| $D009 | 53257 | | Sprite 4 - Y position |
| $D00A | 53258 | | Sprite 5 - X position |
| $D00B | 53259 | | Sprite 5 - Y position |
| $D00C | 53260 | | Sprite 6 - X position |
| $D00D | 53261 | | Sprite 6 - Y position |
| $D00E | 53262 | | Sprite 7 - X position |
| $D00F | 53263 | | Sprite 7 - Y position |
| $D010 | 53264 | | 9th bit of sprite X co-ordinate |
| | | 0 | Sprite 0 |
| | | 1 | Sprite 1 |
| | | 2 | Sprite 2 etc through sprite 7 |
| $D011 | 53265 | | VIC Control Register |
| | | 7 | Raster compare register. Bit 9 |
| | | 6 | 1=Enable extended colour text mode |
| | | 5 | 1=Enable bit map mode |
| | | 4 | 1=Blank screen to border |
| | | 3 | 1=25 row text display. 0=24 row text display |
| | | 2-0 | Smooth scroll to Y dot position |
| $D012 | 53266 | | Raster compare register. Position of raster on screen |
| $D013 | 53267 | | Light pen X position |
| $D014 | 53268 | | Light pen Y position |
| $D015 | 53269 | | Enable or disable sprite |
| | | 0 | 1=Enable sprite 0 |
| | | 1 | 1=Enable sprite 1 |
| | | 2 | 1=Enable sprite 2 etc through sprite 7 |
| $D016 | 53270 | | VIC Control Register |
| | | 4 | 1=Multicolour mode on |
| | | 3 | 1=40 Column text:0=38 coloumn text |
| | | 2-0 | Smooth scroll to X position |
| $D017 | 53271 | | Sprite Vertical Expansion |
| | | 0 | Expand sprite 0 Vertically |
| | | 1 | Expand sprite 1 Vertically |
| | | 2 | Expand sptite 2 Vertically etc through to sprite 7 |
| $D018 | 53272 | | VIC Memory Control |
| | | 7-4 | Video matrix base address |
| | | 3-0 | Character set base address |
| $D019 | 53273 | | VIC Interrupt Flags |
| | | 7 | Set to any VIC IRQ condition |
| | | 3 | Light pen triggered (bit 7) |
| | | 2 | Sprite vs sprite triggered (bit 7) |
| | | 1 | Sprite vs background triggered (bit 7) |
| | | 0 | Raster compare triggered (bit 7) |
| $D01A | 53274 | | VIC Interrupt Switches |
| | | 3 | 1=Enable light pen interrupt |
| | | 2 | 1=Sprite vs sprite enabled |
| | | 1 | 1=Sprite vs background enabled |
| | | 0 | 1=Raster compare enabled |
| $D01B | 53275 | | Sprite Priority Registers |
| | | 0-7 | Each bit relates to corresponding sprite, 1=Sprite/background priority |
| $D01C | 53276 | | Sprite multi-colour select |
| | | 0-7 | Each bit sets corresponding sprite to multicolour |
| $D01D | 53277 | 0-7 | Sprite Horizontal Expansion |
| $D01E | 53278 | | Sprite vs sprite collision detection. If any sprite is touching another sprite, the bits corresponding to both sprites are turned on. |
| $D01F | 53279 | | Sprite/background collision detection. If sprite has hit text or background character, the relevant bit is set. |

| | | |
|---|---|---|
| $D020 | 53280 | Border colour |
| $D021 | 53281 | Background colour |
| $D022 | 53282 | Multi-colour 1 |
| $D023 | 53283 | Multi-colour 2 |
| $D024 | 53284 | Multi-colour 3 |
| $D025 | 53285 | Sprite multi-colour |
| $D026 | 53286 | Sprite multi-colour |
| $D027 | 53287 | Sprite 0 colour |
| $D028 | 53288 | Sprite 1 colour |
| $D029 | 53289 | Sprite 2 colour |
| $D02A | 53290 | Sprite 3 colour |
| $D02B | 53291 | Sprite 4 colour |
| $D02C | 53292 | Sprite 5 colour |
| $D02D | 53293 | Sprite 6 colour |
| $D02E | 53294 | Sprite 7 colour |

## USEFUL SPRITE DATA STORAGE LOCATIONS

| | | |
|---|---|---|
| $02C0-02FE | 704- 766 | Sprite block 11 |
| $0340-037E | 832- 894 | Sprite block 13 |
| $0380-03BE | 896- 958 | Sprite block 14 |
| $03C0-03FE | 960-1022 | Sprite block 15 |

## SID CHIP ADDRESSES: $D400-$D41C (54272-54300)

| ADDRESS HEX | ADDRESS DECIMAL | BIT | DESCRIPTION |
|---|---|---|---|
| $D400 | 54272 | | Voice 1: low byte of frequency |
| $D401 | 54273 | | Voice 1: High byte of frequency |
| $D402 | 54274 | | Voice 1: Low byte of pulse width |
| $D403 | 54275 | 3-0 | Voice 1: High byte of pulse width |
| $D404 | 54276 | | Voice 1 Control Register |
| | | 7 | 1=Random noise |
| | | 6 | 1=Pulse waveform on |
| | | 5 | 1=Sawtooth waveform on |
| | | 4 | 1=Triangle waveform on |
| | | 3 | 1=Disable voice 1 |
| | | 2 | 1=Ring modulate voice 1 with voice 3 |
| | | 1 | 1=Synchronize voice 1 with freq of voice 3 |
| | | 0 | 1=Start attack,decay,sustain<br>0=Start release |
| $D405 | 54277 | | Voice 1 Attack/decay |
| | | 7-4 | Attack cycle duration |
| | | 3-0 | Decay cycle duration |
| $D406 | 54278 | | Voice 1 Sustain/release |
| | | 7-4 | Sustain cycle duration |
| | | 3-0 | Release cycle duration |
| $D407 | 54279 | | Voice 2: low byte of frequency |
| $D408 | 54280 | | Voice 2: high byte of frequency |
| $D409 | 54281 | | Voice 2: low byte of pulse width |
| $D40A | 54282 | 3-0 | Voice 2: high byte of pulse width |
| $D40B | 54283 | | Voice 2 Control Register |
| | | 7 | 1=Random noise on |
| | | 6 | 1=Pulse waveform on |
| | | 5 | 1=Sawtooth waveform on |
| | | 4 | 1=Triangle waveform on |
| | | 3 | 1=Disable oscillator 1 |
| | | 2 | 1=Ring modulate oscillator 2 with oscillator 1 |
| | | 1 | 1=Synchronize oscillator 2 with oscillator 1 frequency |
| | | 0 | 1=Start attack,decay,sustain<br>0=Start release |
| $D40C | 54284 | | Voice 2 Attack/decay |
| | | 7-4 | Attack cycle duration |
| | | 3-0 | Decay cycle duration |
| $D40D | 54285 | | Voice 2 Sustain/release |
| | | 7-4 | Sustain cycle duration |
| | | 3-0 | Release cycle duration |
| $D40E | 54286 | | Voice 3: low byte of frequency |
| $D40F | 54287 | | Voice 3: high byte of frequency |
| $D410 | 54288 | | Voice 3: low byte of pulse width |
| $D411 | 54289 | 3-0 | Voice 3: high byte of pulse width |
| $D412 | 54290 | | Voice 3 Control Register |
| | | 7 | 1=Random noise on |
| | | 6 | 1=Pulse waveform on |
| | | 5 | 1=Sawtooth waveform on |
| | | 4 | 1=Triangle waveform on |
| | | 3 | 1=Disable voice |
| | | 2 | 1=Ring modulate oscillator 3 with oscillator 2 output |
| | | 1 | 1=Synchronize oscillator 3 with freq of oscillator 2 |
| | | 0 | 1=Start attack,decay,sustain<br>0=Start release |
| $D413 | 54291 | | Voice 3 Attack/decay |
| | | 7-4 | Attack cycle duration |
| | | 3-0 | Decay cycle duration |
| $D414 | 54292 | | Voice 3 Sustain/release |
| | | 7-4 | Sustain cycle duration |
| | | 3-0 | Release cycle duration |
| $D415 | 54293 | 2-0 | Filter cut-off low nybble |
| $D416 | 54294 | | Filter cut-off high byte |
| $D417 | 54295 | | Filter Control |
| | | 7-4 | Filter resonance |
| | | 3 | 1=External input to filter |
| | | 2 | 1=Voice 3 to filter |
| | | 1 | 1=Voice 2 to filter |
| | | 0 | 1=Voice 1 to filter |
| $D418 | 54296 | | Filter Volume And Mode |
| | | 7 | 1=Turn off voice 3 output |
| | | 6 | 1=High pass filter on |
| | | 5 | 1=Band pass filter on |
| | | 4 | 1=Low pass filter on |
| | | 3-0 | Output volume |
| $D419 | 54297 | | A/D convertor for paddle 1 |
| $D41A | 54298 | | A/D convertor for paddle 2 |
| $D41B | 54299 | | Produces random number when voice 3 set to noise |
| $D41C | 54300 | | Output of voice 3 envelope generator |

## KERNAL ROM ROUTINES

| C64 HEX | C128 HEX | Description of Routine |
|---|---|---|
| $E043 | $9086 | Series 1 polynomial calculation |
| $E059 | $909C | Series 2 polynomial calculation |
| $E08D | $8490 | Floating point constants for RND |
| $E097 | $8434 | BASIC-function RND |
| $E107 | | Output of 'Break' |
| $E10C | $90DF | BSOUT output of character |
| $E112 | $90E5 | BASIN receive a character |
| $E118 | $90EB | CKOUT establish output-device |
| $E11E | $90FD | CHKIN establish input-device |
| $E124 | $9109 | GETIN get a character |
| $E12A | $5885 | BASIC-command SYS |
| $E156 | $9112 | BASIC-command SAVE |
| $E165 | $9129 | BASIC-command VERIFY |
| $E168 | $912C | BASIC-command LOAD |
| $E1BE | $918D | BASIC-command OPEN |
| $E1C7 | $919A | BASIC-command CLOSE |
| $E1D4 | $91AE | Get parameters for LOAD/VERIFY/SAVE |
| $E200 | $91DD | Get integer in X |
| $E206 | $91E3 | Get current char and check for line end |
| $E20E | $91EB | Check character follows comma |
| $E219 | $91F6 | Get parameter for OPEN/CLOSE |
| $E264 | $9409 | BASIC-function COS |
| $E26B | $9410 | BASIC-function SIN |
| $E2B4 | $9459 | BASIC-function TAN |
| $E2E0 | $9485 | Floating point constants for COS/SIN/TAN |
| $E2E5 | $948A | 2*PI in floating point |
| $E2EA | $948F | 1/4 in floating point |
| $E2EF | $9494 | More constants for COS/SIN/TAN |
| $E30E | $94B3 | BASIC-function ATN |
| $E33E | $94E3 | Floating point constants for ATN |
| $E37B | $4009 | BASIC NMI jump-in |
| $E38B | $4D3F | Error message handler |
| $E394 | $4D23 | BASIC cold start |
| $E3A2 | $4279 | Copy of the CHRGET routine |
| $E3BA | | Start value for the RND function |
| $E3BF | $4045 | Initialize RAM for BASIC |
| $E447 | $4267 | Table of BASIC vectors |
| $E453 | $4251 | Load BASIC vectors |
| $E45F | $41BB | Messages of the operating system |
| $E4E0 | $C3F4 | Waits for Commodore key |
| $E4EC | | Constants for RS"#" timing |
| $E500 | | Gets BASIC-address of CIA or VIA |
| $E505 | | Gets screen format line/column |
| $E50A | $CC6A | Set cursor or get cursor position |
| $E518 | $C07B | Screen reset |
| $E544 | $C142 | Clear screen |
| $E566 | $C150 | Cursor home |
| $E5A0 | | Initialize video controller |
| $E5B4 | $C6AD | Get character from keyboard buffer |
| $E5CA | | Waiting loop for keyboard input |
| $E632 | $C29B | Get a character from the screen |
| $E684 | $C2FF | Checks for quote |
| $E6B6 | | Calculate MSB for line starts |
| $E8DA | | Table of colour codes |
| $E8EA | $C3A6 | Scroll screen |
| $E9C8 | $C40D | Shift line up |
| $E9FF | $C4A5 | Clear screen line |
| $EA1C | $C7E5 | Set character and colour on screen |
| $EA24 | | Calculate pointer on colour RAM |
| $EA31 | $FA65 | Interrupt routine |
| $EAB7 | | Keyboard prompt |
| $EB48 | | Checks on SHIFT,CTRL and CBM keys |
| $EB79 | $C06F | Pointer on keyboard decoding tables |
| $EB81 | $FA80 | Decoding tables |
| $EC44 | | Checks on control character |
| $EC78 | | Decoding tables |
| $ECB9 | $E2C7 | Constants for video controller |
| $ECE7 | | 'Load (cr) Run (cr)' |
| $ECF0 | $CE74 | LSB tables of screen starts |
| $ED09 | $E3BB | Send TALK |
| $ED0C | $E343 | Send LISTEN |
| $ED40 | $E3E2 | Output of byte on IEC-bus |
| $ED89 | | Send secondary address for LISTEN |
| $EDC7 | | Send secondary address for TALK |
| $EDEF | $E515 | Send UNTALK |
| $EDFE | $E526 | Send UNLISTEN |
| $EE13 | $EF5C | Get a byte from the IEC-bus |
| $EE83 | | One milliesecond delay |
| $EE8B | $E5FF | Output RS232 |
| $EF4A | $E68E | Calculate number of RS232 data-bits |
| $F014 | $E75C | Output in RS232 buffer |
| $F086 | $E7CE | GET of RS232 |
| $F0A4 | | Set timer for IEC time-out |
| $F0BD | | Error messages of the operating system |
| $F128 | $F71E | Put out messages |
| $F157 | $EF06 | BASIN get a character |
| $F1CA | $EF79 | BSOUT output a character |
| $F20E | $F106 | CHKIN fixing of the input-device |
| $F250 | $F14C | CHKOUT fixing of the output-device |
| $F291 | $F18B | CLOSE |
| $F30F | | Look for logical file number |

```
$F31F        Set file parameter
$F32F $F222 CLALL closes all I/O channels
$F34A $EFBD OPEN
$F49E $F265 LOAD
$F5AF        Output 'Searching for file name'
$F5D2        Output 'Loading/verifying'
$F5DD $F53E SAVE
$F68F        Output 'Saving filename'
$F69B $F5F8 UDTIM increase running time
$F6DD $F6SE Get time
$F6E4 $F665 Set time
$F6ED $EA8F Test stop-key
$F6FB        Put out error messages of the operating system
$F72C $EBD0 Read program header of tape
$F76A $E919 Write header on tape
$F7D0        Get start address of tape buffer
$F7D7        Set start and end address of the tape buffer
$F7EA $E99A Look for name on tape-header
$F80D $E9BE Increase tape buffer pointer
$F817 $E9CA Waits for tape key for reading
$F82E        Asks for tape key
$F838 $E9E9 Waits for tape key for writing
$F841 $E9F2 Read block of tape
$F84A $E9FB Load program oFF tape
$F864 $EA15 Write tape buffer to tape
$F868 $E919 Write block or program on tape
$F8BE $EA7D Wait for I/O end
$F8E1        Checks on stop key
$F92C $EAEB Interrupt routine for tape read
$F897 $ED5A Set bit counter for serial output
$F8A6 $ED69 Write one bit to tape
$F8CD $ED90 Interrupt routine for tape write
$FC88 $C207 Set IRQ vector
$FCCA $EEB0 Switch off tape drive
$FCD1        Checks on reaching of end address
$FCDB        Increase address pointer
$FCE2 $FF3D RESET
$FD02        Checks on ROM in $8000 or $A000
$FD10        ROM module identification
$FD15        Set or get hardware and I/O vectors
$FD30        Table of hardware and I/O vectors
$FD50        Initialize work memory
$FD9B $EEA8 Table of IRQ vectors
$FDF9        Set parameter for file names
$FE00        Set parameter for active file
$FE07        Get status
$FE18        Set flag for messages of the operating system
$FE1C        Set status
$FE21        Set timeout flag for IEC-bus
$FE25        Set or get RAM-upper limit
$FE34        Set or get RAM-lower limit
$FE43 $FA40 NMI routine
$FEC2 $E850 Constants for RS232 baud rate
$FF48        Interrupt handler
```

C64 KERNAL JUMP TABLE

| ADDRESS | CONTENTS | PURPOSE |
|---|---|---|
| $FF84 | JMP $FDA3 | Initialize CIA's |
| $FF87 | JMP $FD50 | Clear or check RAM |
| $FF8A | JMP $FD15 | Initialize I/O |
| $FF8D | JMP $FD1A | Initialize I/O vectors |
| $FF90 | JMP $FE18 | Set status |
| $FF93 | JMP $EDB9 | Send LISTEN se condary address |
| $FF96 | JMP $EDC7 | Send TALK Secondary address |
| $FF99 | JMP $FE25 | Set/get RAM end |
| $FF9C | JMP $FE34 | Set/get RAM start |
| $FF9F | JMP $EA87 | Scan keyboard |
| $FFA2 | JMP $FE21 | Set IEC-bus time out flag |
| $FFA5 | JMP $EE13 | Input for IEC-bus |
| $FFA8 | JMP $EDDD | Output to IEC-bus |
| $FFAB | JMP $EDEF | Send UNTALK |
| $FFAE | JMP $EDFE | Send UNLISTEN |
| $FFB1 | JMP $ED0C | Send LISTEN |
| $FFB4 | JMP $ED09 | Send TALK |
| $FFB7 | JMP $FE07 | Get status |
| $FFBA | JMP $FE00 | Set file parameter |
| $FFBD | JMP $FDF9 | Set filename parameter |
| $FFC0 | JMP ($031A) | $F34A OPEN |
| $FFC3 | JMP ($031C) | $F291 CLOSE |
| $FFC6 | JMP ($031E) | $F20E CHKIN set input device |
| $FFC9 | JMP ($0320) | $F250 CKOUT set output device |
| $FFCC | JMP ($0322) | $F333 CLRCH |
| $FFCF | JMP ($0324) | $F157 BASIN input character |
| $FFD2 | JMP ($0326) | $F1CA BSOUT output character |
| $FFD5 | JMP $F49E | LOAD |
| $FFD8 | JMP $F5DD | SAVE |
| $FFDB | JMP $F6E4 | Set time |
| $FFDE | JMP $F6DD | Get time |
| $FFE1 | JMP ($0328) | $F6ED Scan stop-key |
| $FFE4 | JMP ($032A) | $F13E GET |
| $FFE7 | JMP ($032C) | $F32F CLALL |
| $FFEA | JMP $F69B | Increase time |
| $FFED | JMP $E505 | SCREEN get number lines and columns |
| $FFF0 | JMP $E50A | Set/get cursor position |
| $FFF3 | JMP $E500 | Get start of I/O element |
| $FFFA | JMP $FE43 | NMI vector |
| $FFFC | JMP $FCE2 | RESET vector |

SCREEN COLOUR CODES AND MODES

Value to POKE for each colour:

| COLOUR | LOW NYBBLE VALUE | HIGH NYBBLE VALUE | MULTI-COLOUR |
|---|---|---|---|
| Black | 0 | 0 | 8 |
| White | 1 | 16 | 9 |
| Red | 2 | 32 | 10 |
| Cyan | 3 | 48 | 11 |
| Purple | 4 | 64 | 12 |
| Green | 5 | 80 | 13 |
| Blue | 6 | 96 | 14 |
| Yellow | 7 | 112 | 15 |
| Orange | 8 | 128 | -- |
| Brown | 9 | 144 | -- |
| Light red | 10 | 160 | -- |
| Dark grey | 11 | 176 | -- |
| Mid grey | 12 | 192 | -- |
| Light green | 13 | 208 | -- |
| Light blue | 14 | 224 | -- |
| Light grey | 15 | 240 | -- |

Where to POKE colour values for each mode:

| MODE (i) | BIT OR BIT-PAIR | LOCATION | COLOUR VALUE |
|---|---|---|---|
| Regular text | 0 | 53281 | Low nybble |
| | 1 | Colour memory | Low nybble |
| Multicolour text | 00 | 53281 | Low nybble |
| | 01 | 53282 | Low nybble |
| | 10 | 53283 | Low nybble |
| | 11 | Colour memory | Multicolour |
| Extended colour text (ii) | 00 | 53281 | Low nybble |
| | 01 | 53282 | Low nybble |
| | 10 | 53283 | Low nybble |
| | 11 | 53284 | Low nybble |
| Bitmapped | 0 | Screen memory | Low nybble (iii) |
| | 1 | Screen memory | High nybble (iii) |
| Multicolour bitmapped | 00 | 53281 | Low nybble (iii) |
| | 01 | Screen memory | High nybble (iii) |
| | 10 | Screen memory | Low nybble (iii) |
| | 11 | Colour memory | Low nybble |

(i) For all modes, the screen border colour is controlled by POKEing 53280 with the low nybble colour value.

(ii) In extended colour mode, bits 6 & 7 of each byte of screen memory serve as the bit-pair controlling background colour. Because only bits 0-5 are available for character selection, only characters with screen codes 0-63 can be used in this mode.

(iii) In the bitmapped modes, the high and low nybble colour values are ORed together and POKed into the SAME LOCATION in screen memory to control the colours of the corresponding CELL in the bitmap. For example: to control the colours of cell 0 of the bitmap, OR the high and low nybble values and POKE the result into location 0 of screen memory.

C128 COLOUR CODES

Command: COLOR source, colour

| SOURCE NUMBER | SOURCE |
|---|---|
| 0 | 40-column background colour |
| 1 | Foreground for graphics screen |
| 2 | Foreground for multicolour 1 |
| 3 | Foreground for multicolour 2 |
| 4 | 40-column border (text and graphics) |
| 5 | Text colour for 40- or 80-column screen |
| 6 | 80-column background colour |

| 40-COLUMN MODE | | 80 COLUMN MODE | |
|---|---|---|---|
| COLOUR VALUE | COLOUR | COLOUR VALUE | COLOUR |
| 1 | Black | 1 | Black |
| 2 | White | 2 | White |
| 3 | Red | 3 | Red |
| 4 | Cyan | 4 | Light cyan |
| 5 | Purple | 5 | Light purple |
| 6 | Green | 6 | Light green |
| 7 | Blue | 7 | Dark blue |
| 8 | Yellow | 8 | Light yellow |
| 9 | Orange | 9 | Dark purple |
| 10 | Brown | 10 | Brown |
| 11 | Light red | 11 | Light red |
| 12 | Dark grey | 12 | Dark cyan |
| 13 | Medium grey | 13 | Medium grey |
| 14 | Light green | 14 | Light green |
| 15 | Light blue | 15 | Light blue |
| 16 | Light grey | 16 | Light grey |

### STANDARD CBM TOKENS

| HEX | DEC | TOKEN | HEX | DEC | TOKEN | HEX | DEC | TOKEN |
|---|---|---|---|---|---|---|---|---|
| $20 | 32 | SPACE | $4F | 79 | O | $9E | 158 | SYS |
| $21 | 33 | ! | $50 | 80 | P | $9F | 159 | OPEN |
| $22 | 34 | " | $51 | 81 | Q | $A0 | 160 | CLOSE |
| $23 | 35 | # | $52 | 82 | R | $A1 | 161 | GET |
| $24 | 36 | $ | $53 | 83 | S | $A2 | 162 | NEW |
| $25 | 37 | % | $54 | 84 | T | $A3 | 163 | TAB( |
| $26 | 38 | & | $55 | 85 | U | $A4 | 164 | TO |
| $27 | 39 | ' | $56 | 86 | V | $A5 | 165 | FN |
| $28 | 40 | ( | $57 | 87 | W | $A6 | 166 | SPC( |
| $29 | 41 | ) | $58 | 88 | X | $A7 | 167 | THEN |
| $2A | 42 | * | $59 | 89 | Y | $A8 | 168 | NOT |
| $2B | 43 | + | $5A | 90 | Z | $A9 | 169 | STEP |
| $2C | 44 | , | $5B | 91 | [ | $AA | 170 | + ADD |
| $2D | 45 | - | $5C | 92 | £ | $AB | 171 | - MINUS |
| $2E | 46 | . | $5D | 93 | ] | $AC | 172 | * MULTIPLY |
| $2F | 47 | / | $5E | 94 | ↑ | $AD | 173 | / DIVIDE |
| $30 | 48 | 0 | $5F | 95 | L.ARROW | $AE | 174 | ^ POWER |
| $31 | 49 | 1 | $80 | 128 | END | $AF | 175 | AND |
| $32 | 50 | 2 | $81 | 129 | FOR | $B0 | 176 | OR |
| $33 | 51 | 3 | $82 | 130 | NEXT | $B1 | 177 | > GREATER |
| $34 | 52 | 4 | $83 | 131 | DATA | $B2 | 178 | = EQUAL |
| $35 | 53 | 5 | $84 | 132 | INPUT# | $B3 | 179 | < LESS |
| $36 | 54 | 6 | $85 | 133 | INPUT | $B4 | 180 | SGN |
| $37 | 55 | 7 | $86 | 134 | DIM | $B5 | 181 | INT |
| $38 | 56 | 8 | $87 | 135 | READ | $B6 | 182 | ABS |
| $39 | 57 | 9 | $88 | 136 | LET | $B7 | 183 | USR |
| $3A | 58 | : | $89 | 137 | GOTO | $B8 | 184 | FRE |
| $3B | 59 | ; | $8A | 138 | RUN | $B9 | 185 | POS |
| $3C | 60 | < | $8B | 139 | IF | $BA | 186 | SQR |
| $3D | 61 | = | $8C | 140 | RESTORE | $BB | 187 | RND |
| $3E | 62 | > | $8D | 141 | GOSUB | $BC | 188 | LOG |
| $3F | 63 | ? | $8E | 142 | RETURN | $BD | 189 | EXP |
| $40 | 64 | @ | $8F | 143 | REM | $BE | 190 | COS |
| $41 | 65 | A | $90 | 144 | STOP | $BF | 191 | SIN |
| $42 | 66 | B | $91 | 145 | ON | $C0 | 192 | TAN |
| $43 | 67 | C | $92 | 146 | WAIT | $C1 | 193 | ATN |
| $44 | 68 | D | $93 | 147 | LOAD | $C2 | 194 | PEEK |
| $45 | 69 | E | $94 | 148 | SAVE | $C3 | 195 | LEN |
| $46 | 70 | F | $95 | 149 | VERIFY | $C4 | 196 | STR$ |
| $47 | 71 | G | $96 | 150 | DEF | $C5 | 197 | VAL |
| $48 | 72 | H | $97 | 151 | POKE | $C6 | 198 | ASC |
| $49 | 73 | I | $98 | 152 | PRINT# | $C7 | 199 | CHR$ |
| $4A | 74 | J | $99 | 153 | PRINT | $C8 | 200 | LEFT$ |
| $4B | 75 | K | $9A | 154 | CONT | $C9 | 201 | RIGHT$ |
| $4C | 76 | L | $9B | 155 | LIST | $CA | 202 | MID$ |
| $4D | 77 | M | $9C | 156 | CLR | $CB | 203 | GO |
| $4E | 78 | N | $9D | 157 | CMD | | | |

### C128 EXTENDED TOKENS

| HEX | DEC | TOKEN | HEX | DEC | TOKEN | HEX | DEC | TOKEN |
|---|---|---|---|---|---|---|---|---|
| $CC | 204 | RGR | $DD | 221 | PUDEF | $EE | 238 | DIRECTORY |
| $CD | 205 | RCLR | $DE | 222 | GRAPHIC | $EF | 239 | DSAVE |
| $CE | 206 | reserved | $DF | 223 | PAINT | $F0 | 240 | DLOAD |
| $CF | 207 | JOY | $E0 | 224 | CHAR | $F1 | 241 | HEADER |
| $D0 | 208 | RDOT | $E1 | 225 | BOX | $F2 | 242 | SCRATCH |
| $D1 | 209 | DEC | $E2 | 226 | CIRCLE | $F3 | 243 | COLLECT |
| $D2 | 210 | HEX$ | $E3 | 227 | GSHAPE | $F4 | 244 | COPY |
| $D3 | 211 | ERR$ | $E4 | 228 | SSHAPE | $F5 | 245 | RENAME |
| $D4 | 212 | INSTR | $E5 | 229 | DRAW | $F6 | 246 | BACKUP |
| $D5 | 213 | ELSE | $E6 | 230 | LOCATE | $F7 | 247 | DELETE |
| $D6 | 214 | RESUME | $E7 | 231 | COLOR | $F8 | 248 | RENUMBER |
| $D7 | 215 | TRAP | $E8 | 232 | SCNCLR | $F9 | 249 | KEY |
| $D8 | 216 | TRON | $E9 | 233 | SCALE | $FA | 250 | MONITOR |
| $D9 | 217 | TROFF | $EA | 234 | HELP | $FB | 251 | USING |
| $DA | 218 | SOUND | $EB | 235 | DO | $FC | 252 | UNTIL |
| $DB | 219 | VOL | $EC | 236 | LOOP | $FD | 253 | WHILE |
| $DC | 220 | AUTO | $ED | 237 | EXIT | $FE | 254 | reserved |

### CBM128 DOUBLE BYTE TOKENS

$CE followed by:

| HEX | DEC | TOKEN | HEX | DEC | TOKEN | HEX | DEC | TOKEN |
|---|---|---|---|---|---|---|---|---|
| $02 | 2 | POT | $05 | 5 | RSPPOS | $08 | 8 | XOR |
| $03 | 3 | BUMP | $06 | 6 | RSPRITE | $09 | 9 | RWINDOW |
| $04 | 4 | PEN | $07 | 7 | RSPCOLOR | $0A | 10 | POINTER |

$FE followed by:

| HEX | DEC | TOKEN | HEX | DEC | TOKEN | HEX | DEC | TOKEN |
|---|---|---|---|---|---|---|---|---|
| $02 | 2 | BANK | $0E | 14 | APPEND | $1B | 27 | BOOT |
| $03 | 3 | FILTER | $0F | 15 | DCLOSE | $1C | 28 | WIDTH |
| $04 | 4 | PLAY | $10 | 16 | BSAVE | $1D | 29 | SPRDEF |
| $05 | 5 | TEMPO | $11 | 17 | BLOAD | $1E | 30 | QUIT |
| $06 | 6 | MOVSPR | $12 | 18 | RECORD | $1F | 31 | SPRDEF |
| $07 | 7 | SPRITE | $13 | 19 | CONCAT | $20 | 32 | QUIT |
| $08 | 8 | SPRCOLOR | $14 | 20 | DVERIFY | $21 | 33 | STASH |
| $09 | 9 | RREG | $15 | 21 | DCLEAR | $22 | 34 | FETCH |
| $0A | 10 | ENVELOPE | $16 | 22 | SPRSAV | $23 | 35 | SWAP |
| $0B | 11 | SLEEP | $17 | 23 | COLLISION | $24 | 36 | OFF |
| $0C | 12 | CATALOG | $18 | 24 | BEGIN | $25 | 37 | FAST |
| $0D | 13 | DOPEN | $19 | 25 | BEND | $26 | 38 | SLOW |
| | | | $1A | 26 | WINDOW | | | |

### 1541 DISK DRIVE - USEFUL MEMORY LOCATIONS

DOS ADDRESS

| HEX | DECIMAL | DESCRIPTION |
|---|---|---|
| $0000-$07FF | 0-2047 | DOS RAM CHIP |
| $0000 | 0 | Command code for buffer 0 |
| $0001 | 1 | Command code for buffer 1 |
| $0002 | 2 | Command code for buffer 2 |
| $0003 | 3 | Command code for buffer 3 |
| $0004 | 4 | Command code for buffer 4 |
| $0006-0007 | 6-7 | Track and sector for buffer 0 |
| $0008-0009 | 8-9 | Track and sector for buffer 1 |
| $000A-000B | 10-11 | Track and sector for buffer 2 |
| $000C-000D | 12-13 | Track and sector for buffer 3 |
| $000E-000F | 14-15 | Track and sector for buffer 4 |
| $0012-0013 | 18-19 | ID for drive 0 |
| $0014-0015 | 20-21 | ID for drive 1 |
| $0016-0017 | 22-23 | Current ID |
| $0020-0021 | 32-33 | Flag for head transport |
| $0030-0031 | 48-49 | Buffer pointer for disk controller |
| $0039 | 57 | Constant 8 - mark for beginning of data block header |
| $003A | 58 | Parity for data buffer |
| $003D | 61 | Drive number for disk controller |
| $003F | 63 | Buffer number for disk controller |
| $0043 | 67 | Number of sectors per track for formatting |
| $0047 | 71 | Constant 7 - mark for begining of data block header |
| $0049 | 73 | Stack pointer |
| $004A | 74 | Step counter for head transport |
| $0051 | 81 | Actual track number for formatting |
| $0069 | 105 | Step size for sector division (=10) |
| $006A | 106 | Number of read attempts (5) |
| $006F-0070 | 111-112 | Pointer to address for M and B commands |
| $0077 | 119 | Device number+ $20(32) for LISTEN |
| $0078 | 120 | Device number+ $40(64) for TALK |
| $0079 | 121 | Flag for LISTEN (I/O) |
| $007A | 122 | Flag for TALK (I/O) |
| $007C | 124 | Flag for ATN from serial bus receiving |
| $007D | 125 | Flag for EOI from serial bus |
| $007F | 127 | Drive number (0) |
| $0080 | 128 | Current track number |
| $0081 | 129 | Current sector number |
| $0082 | 130 | Current channel number |
| $0083 | 131 | Current file number |
| $0084 | 132 | Current secondary address |
| $0085 | 133 | Current data byte |
| $008B-008D | 139-141 | Work storage for division |
| $0094-0095 | 148-149 | Actual buffer pointer |
| $0099-009A | 153-154 | Address of buffer 0 ($0300) |
| $009B-009C | 155-156 | Address of buffer 1 ($0400) |
| $009D-009E | 157-158 | Address of buffer 2 ($0500) |
| $009F-00A0 | 159-160 | Address of buffer 3 ($0600) |
| $00A1-00A2 | 161-162 | Address of buffer 4 ($0700) |
| $00A3-00A4 | 163-164 | Pointer to input buffer $0200 |
| $00A5-00A6 | 165-166 | Pointer to buffer error message ($02DS) |
| $00B5-00BA | 181-186 | Record number LO, block number LO |
| $00BB-00C0 | 187-192 | Record number HI, block number HI |
| $00C1-00C6 | 193-198 | Write pointer for REL file |
| $00C7-00CC | 199-204 | Record length for REL file |
| $00D4 | 212 | Pointer in record for REL file |
| $00D5 | 213 | Side sector number |
| $00D6 | 214 | Pointer to data block in side sector |
| $00D7 | 215 | Pointer to record in REL file |
| $00E7 | 231 | File type |
| $00F9 | 249 | Buffer number |
| $0100-0145 | 256-325 | Stack |
| $0200-0228 | 512-552 | Buffer for command string |
| $024A | 586 | File type |
| $0258 | 600 | Record length |
| $0259 | 601 | Track side-sector |
| $025A | 602 | Sector side-sector |
| $0274 | 628 | Length of input line |
| $0278 | 632 | Number of file names |
| $0297 | 663 | File control method |
| $0280-0284 | 640-644 | Track of a file |
| $0285-0289 | 645-649 | Sector of a file |
| $02D5-02F9 | 725-761 | Buffer for error messages |
| $02FA-02FC | 762-764 | Number of BLOCKS FREE |
| $0300-03FF | 768-1023 | Buffer 0 - main work buffer |
| $0400-04FF | 1024-1279 | Buffer 1 - disk directory |
| $0500-05FF | 1280-1535 | Buffer 2 - user buffer |
| $0600-06FF | 1536-1791 | Buffer 3 - disk directory |
| $0700-07FF | 1792-2047 | Buffer 4 - BAM map |
| $0800-FFFF | 2048-65535 | DOS ROM CHIP |
| $0800-17FF | 2048-6143 | Unused |
| $1800-180F | 6144-6159 | IEEE bus controller 6522 |
| $1810-1BFF | 6160-7167 | Unused |
| $1C00-1C0F | 7168-7183 | Drive controller 6522 |
| $1C10-C0FF | 7184-49407 | Unused |
| $C100-FFFF | 49408-65535 | Disk operating system routines |

## 1541 DISK ERROR MESSAGES AND THEIR CAUSES

The following list contains the error messages recognised by the 1541 DOS.
Note that TT and SS denote Track and Sector respectively.

| ERROR NUMBER | DESCRIPTION |
|---|---|
| 00,OK,00,00 | The last disk operation was error free or no disk access has been made since the last error message was read. |
| 20,READ ERROR,TT,SS | The 'header' of a block was not found. It is usually the result of a defective disk. TT and SS denote the track and sector in which the error occurred. Remedy: change the disk. |
| 21,READ ERROR,TT,SS | The SYNC marker of a block was not found. The cause may be an unformatted disk, or no disk in the drive. This error can also be caused by a misaligned read/write head. Remedy: Either insert a disk and format it if necessary, or have the head re-aligned. |
| 22,READ ERROR,TT,SS | A checksum error has occurred in the header of a data block, which may have been caused by the incorrect writing of a block or rough handling of the disk. |
| 23,READ ERROR,TT,SS | A data block was read into the DOS buffer but a checksum error has occurred. One or more data bytes are incorrect. Remedy: Save as many files as possible onto another disk. |
| 24,READ ERROR,TT,SS | This error also results from a checksum error in the data block or in the preceding data header. Incorrect bytes have been read. Remedy: Same as for error 23. |
| 25,WRITE ERROR,TT,SS | This is actually a VERIFY error. After writing every block the data is read again, checked against the data in the buffer. This error is produced if the data are not identical. Remedy: Repeat the command that caused the error. If this does not work, the block-allocate command must be used to lock out the offending block from future use. |
| 26,WRITE PROTECT ON,TT,SS | An attempt was made to write to a disk with a write protect tab on. Remedy: Remove the tab. |
| 27,READ ERROR,TT,SS | A checksum error has occurred in the header of a data block. Remedy: Repeat command or rescue block. |
| 28,WRITE ERROR,TT,SS | After writing a data block, the SYNC characters of the next data block were not found. Remedy: Format the disk again, or exchange it. |
| 29,DISK ID MISMATCH,TT,SS | The ID in the DOS memory does not agree with the ID on the disk. The disk either was not initialised or has an error in the header of a data block. Remedy: initialise the disk. |
| 30,SYNTAX ERROR,00,00 | The DOS cannot understand the command that it is receiving. Remedy: Correct the command. |
| 31,SYNTAX ERROR,00,00 | A command was not recognized by the DOS. Remedy: Do not use the command. |
| 32,SYNTAX ERROR,00,00 | The command sent was over 40 characters long. Remedy: Shorten the command. |
| 33,SYNTAX ERROR,00,00 | A wildcard, ("*" or "?") was used in an OPEN or SAVE command. Remedy: Remove wildcard. |
| 34,SYNTAX ERROR,00,00 | The DOS cannot find the filename in a command. The cause may be a forgotten colon after the command word. Remedy: Check the command. |
| 39,FILE NOT FOUND,00,00 | User program (USR) was not found for automatic execution. Remedy: Check filename. |
| 50,RECORD NOT PRESENT,00,00 | A non-existent record was addressed in a relative data file. When writing a record this is not really an error. Remedy: You can avoid this message if you write CHR$(255) with the highest record number when initialising the file. |
| 51,OVERFLOW IN RECORD,00,00 | The number of characters sent when writing a record in a relative file was greater than the record length. The excess characters are ignored. |
| 52,FILE TOO LARGE,00,00 | The record number within a relative file is too big; the disk does not have enough capacity. Remedy: Use another disk or reduce the number of records. |
| 60,WRITE FILE OPEN,00,00 | An attempt was made to OPEN a file that had not previously been CLOSEd after writing. Remedy: Use mode 'M' in the OPEN command to read the file. |
| 61,FILE NOT OPEN,00,00 | Access was attempted to a file that has not been OPENed. Remedy: OPEN the file or check the filename. |
| 62,FILE NOT FOUND,00,00 | An attempt was made to load a program or open a file that does not exist on the disk. Remedy: Check the filename. |
| 63,FILE EXISTS,00,00 | An attempt was made to establish a new file with the same name as one already on the disk. Remedy: Use a different name or use @0: |
| 64,FILE TYPE MISMATCH,00,00 | The file type used in the OPEN command does not agree with the file type in the directory. Remedy: Correct the filetype. |
| 65,NO BLOCK,TT,SS | This message is given in association with the block-allocate command when the specified block is no longer free. In this case,the DOS automatically searches for a free block with a higher sector and/or track number and gives these values as the track and sector number in the error message. If no block with a greater number is free, two zeros will be given. |
| 66,ILLEGAL TT or SS,TT,SS | An attempt has been made to access a non-existent block using the block commands. |
| 67,ILLEGAL TT or SS,TT,SS | The track/sector combination of a file contains values for a non-existent track or sector. |
| 70,NO CHANNEL,00,00 | An attempt has been made to open more file channels than are available or a direct access channel is already reserved. Remedy: Always close a channel after it has been accessed. |
| 71,DIR ERROR,TT,SS | The number of free blocks in the DOS storage does not agree with the BAM. Often this means the disk has notbeen initialised. Remedy: If the disk has been initialised, validate it. |
| 72,DISK FULL,00,00 | Fewer than three blocks are free on the disk or the maximum number of directory entries have been used (144 on the 1541). Remedy: Use a different disk or try validating to free any blocks that may be available. |
| 73,CBM DOS v.26 1541,00,00 | The message is the power-up message of the 1541. It appears as an error message when an attempt is made to write to a disk that was not formatted with the same DOS version. |
| 74,DRIVE NOT READY,00,00 | The drive does not have a disk inserted. |
| 75,FORMAT SPEED ERROR,00,00 | This error only occurs on the CBM 8250. |

## LOCATION 197 C64 KEYCODE VALUES

| KEY | KEYCODE | KEY | KEYCODE |
|---|---|---|---|
| A | 10 | 5 | 16 |
| B | 28 | 6 | 19 |
| C | 20 | 7 | 24 |
| D | 18 | 8 | 27 |
| E | 14 | 9 | 32 |
| F | 21 | 0 | 35 |
| G | 26 | + | 40 |
| H | 29 | - | 43 |
| I | 33 | £ | 48 |
| J | 34 | CLR/HOME | 51 |
| K | 37 | INST/DEL | 0 |
| L | 42 | LEFT ARROW | 57 |
| M | 36 | @ | 46 |
| N | 39 | * | 49 |
| O | 38 | ^ | 54 |
| P | 41 | : | 45 |
| Q | 62 | ; | 50 |
| R | 17 | = | 53 |
| S | 13 | RET | 1 |
| T | 22 | , | 47 |
| U | 30 | . | 44 |
| V | 31 | / | 55 |
| W | 9 | CSR UP/DOWN | 7 |
| X | 23 | CSR LT/RT | 2 |
| Y | 25 | F1 | 4 |
| Z | 12 | F3 | 5 |
| 1 | 56 | F5 | 6 |
| 2 | 59 | F7 | 3 |
| 3 | 8 | SPACE | 60 |
| 4 | 11 | RUN/STOP | 63 |

NO KEY PRESSED = 64

## C64 VALUES FOUND AT LOCATION 653

| CODE | DESCRIPTION |
|---|---|
| 0 | No key pressed |
| 1 | SHIFT |
| 2 | CBM |
| 3 | SHIFT and CBM |
| 4 | CTRL |
| 5 | SHIFT and CTRL |
| 6 | CBM and CTRL |
| 7 | SHIFT, CTRL and CBM |

## 6510 ADDRESSING MODES AND OPERATION CODES

The following table gives the hex values for the various opcodes in their individual addressing modes. The following key to be used for the Address Mode:

A = Accumulator
# = Immediate
ZP = Zero page
AB = Absolute
ABX = Absolute X
ABY = Absolute Y
ZPX = Zero page X
ZPY = Zero page Y
,X) = Indexed X
),Y = Indexed Y

| MNEMONIC | ADDRESSING MODE | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | A | # | ZP | AB | ABX | ABY | ZPX | ZPY | ,X) | ),Y |
| ADC | -- | 69 | 65 | 6D | 7D | 79 | 75 | -- | 61 | 71 |
| AND | -- | 29 | 25 | 2D | 3D | 39 | 35 | -- | 21 | 31 |
| ASL | 0A | -- | 06 | 0E | 1E | -- | 16 | -- | -- | -- |
| BIT | -- | -- | 24 | 2C | -- | -- | -- | -- | -- | -- |
| CMP | -- | C9 | C5 | CD | DD | D9 | D5 | -- | C1 | D1 |
| CPX | -- | E0 | E4 | EC | -- | -- | -- | -- | -- | -- |
| CPY | -- | C0 | C4 | CC | -- | -- | -- | -- | -- | -- |
| DEC | -- | -- | C6 | CE | DD | -- | D6 | -- | -- | -- |
| EOR | -- | 49 | 45 | 4D | 5D | 59 | 55 | -- | 41 | 51 |
| INC | -- | -- | E6 | EE | FD | -- | F6 | -- | -- | -- |
| LDA | -- | A9 | A5 | AD | BD | B9 | B5 | -- | A1 | B1 |
| LDX | -- | A2 | A6 | AE | -- | BE | -- | B6 | -- | -- |
| LDY | -- | A0 | A4 | AC | BC | -- | B4 | -- | -- | -- |
| LSR | 4A | -- | 46 | 4E | 5E | -- | 56 | -- | -- | -- |
| ORA | -- | 09 | 05 | 0D | 1D | 19 | 15 | -- | 01 | 11 |
| ROL | 2A | -- | 26 | 2E | 3E | -- | 36 | -- | -- | -- |
| ROR | 6A | -- | 66 | 6E | 7E | -- | 76 | -- | -- | -- |
| SBC | -- | E9 | E5 | ED | FD | F9 | F5 | -- | E1 | F1 |
| STA | -- | -- | 85 | 8D | 9D | 99 | 95 | -- | 81 | 91 |
| STX | -- | -- | 86 | 8E | -- | -- | -- | 96 | -- | -- |
| STY | -- | -- | 84 | 8C | -- | -- | 94 | -- | -- | -- |

## GROUPED INSTRUCTIONS

Branch Instructions

| BPL | BMI | BVC | BVS | BCC | BCS | BNE | BEQ |
|---|---|---|---|---|---|---|---|
| 10 | 30 | 50 | 70 | 90 | B0 | D0 | F0 |

Transfer Instructions

| TXA | TAX | TYA | TAY | TSX | TXS |
|---|---|---|---|---|---|
| 8A | AA | 98 | A8 | BA | 9A |

Stack Instructions

| PHP | PLP | PHA | PLA |
|---|---|---|---|
| 08 | 28 | 48 | 68 |

Jump Instructions

| BRK | JSR | RTI | RTS | JMP | JMP | NOP |
|---|---|---|---|---|---|---|
| 00 | 20 | 40 | 60 | 4C | 6C | EA |

Flag Instructions

| CLC | SEC | CLI | SEI | CLV | CLD | SED |
|---|---|---|---|---|---|---|
| 18 | 38 | 58 | 78 | B8 | D8 | F8 |

INC/DEC Instructions

| DEY | INY | DEX | INX |
|---|---|---|---|
| 88 | C8 | CA | E8 |

## HEX TO DECIMAL CONVERTER

| HEX | DECIMAL LOW | DECIMAL HIGH | HEX | DECIMAL LOW | DECIMAL HIGH | HEX | DECIMAL LOW | DECIMAL HIGH |
|---|---|---|---|---|---|---|---|---|
| $00 | 0 | 0 | $56 | 86 | 22016 | $AC | 172 | 44032 |
| $01 | 1 | 256 | $57 | 87 | 22272 | $AD | 173 | 44288 |
| $02 | 2 | 512 | $58 | 88 | 22528 | $AE | 174 | 44544 |
| $03 | 3 | 768 | $59 | 89 | 22784 | $AF | 175 | 44800 |
| $04 | 4 | 1024 | $5A | 90 | 23040 | $B0 | 176 | 45056 |
| $05 | 5 | 1280 | $5B | 91 | 23296 | $B1 | 177 | 45312 |
| $06 | 6 | 1536 | $5C | 92 | 23552 | $B2 | 178 | 45568 |
| $07 | 7 | 1792 | $5D | 93 | 23808 | $B3 | 179 | 45824 |
| $08 | 8 | 2048 | $5E | 94 | 24064 | $B4 | 180 | 46080 |
| $09 | 9 | 2304 | $5F | 95 | 24320 | $B5 | 181 | 46336 |
| $0A | 10 | 2560 | $60 | 96 | 24576 | $B6 | 182 | 46592 |
| $0B | 11 | 2816 | $61 | 97 | 24832 | $B7 | 183 | 46848 |
| $0C | 12 | 3072 | $62 | 98 | 25088 | $B8 | 184 | 47104 |
| $0D | 13 | 3328 | $63 | 99 | 25344 | $B9 | 185 | 47360 |
| $0E | 14 | 3584 | $64 | 100 | 25600 | $BA | 186 | 47616 |
| $0F | 15 | 3840 | $65 | 101 | 25856 | $BB | 187 | 47872 |
| $10 | 16 | 4096 | $66 | 102 | 26112 | $BC | 188 | 48128 |
| $11 | 17 | 4352 | $67 | 103 | 26368 | $BD | 189 | 48384 |
| $12 | 18 | 4608 | $68 | 104 | 26624 | $BE | 190 | 48640 |
| $13 | 19 | 4864 | $69 | 105 | 26880 | $BF | 191 | 48896 |
| $14 | 20 | 5120 | $6A | 106 | 27136 | $C0 | 192 | 49152 |
| $15 | 21 | 5376 | $6B | 107 | 27392 | $C1 | 193 | 49408 |
| $16 | 22 | 5632 | $6C | 108 | 27648 | $C2 | 194 | 49664 |
| $17 | 23 | 5888 | $6D | 109 | 27904 | $C3 | 195 | 49920 |
| $18 | 24 | 6144 | $6E | 110 | 28160 | $C4 | 196 | 50176 |
| $19 | 25 | 6400 | $6F | 111 | 28416 | $C5 | 197 | 50432 |
| $1A | 26 | 6656 | $70 | 112 | 28672 | $C6 | 198 | 50688 |
| $1B | 27 | 6912 | $71 | 113 | 28928 | $C7 | 199 | 50944 |
| $1C | 28 | 7168 | $72 | 114 | 29184 | $C8 | 200 | 51200 |
| $1D | 29 | 7424 | $73 | 115 | 29440 | $C9 | 201 | 51456 |
| $1E | 30 | 7680 | $74 | 116 | 29696 | $CA | 202 | 51712 |
| $1F | 31 | 7935 | $75 | 117 | 29952 | $CB | 203 | 51968 |
| $20 | 32 | 8192 | $76 | 118 | 30208 | $CC | 204 | 52224 |
| $21 | 33 | 8448 | $77 | 119 | 30464 | $CD | 205 | 52480 |
| $22 | 34 | 8704 | $78 | 120 | 30720 | $CE | 206 | 52736 |
| $23 | 35 | 8960 | $79 | 121 | 30976 | $CF | 207 | 52992 |
| $24 | 36 | 9216 | $7A | 122 | 31232 | $D0 | 208 | 53248 |
| $25 | 37 | 9472 | $7B | 123 | 31488 | $D1 | 209 | 53504 |
| $26 | 38 | 9728 | $7C | 124 | 31744 | $D2 | 210 | 53760 |
| $27 | 39 | 9984 | $7D | 125 | 32000 | $D3 | 211 | 54016 |
| $28 | 40 | 10240 | $7E | 126 | 32256 | $D4 | 212 | 54272 |
| $29 | 41 | 10496 | $7F | 127 | 32512 | $D5 | 213 | 54528 |
| $2A | 42 | 10752 | $80 | 128 | 32768 | $D6 | 214 | 54784 |
| $2B | 43 | 11008 | $81 | 129 | 33024 | $D7 | 215 | 55040 |
| $2C | 44 | 11264 | $82 | 130 | 33280 | $D8 | 216 | 55296 |
| $2D | 45 | 11520 | $83 | 131 | 33536 | $D9 | 217 | 55552 |
| $2E | 46 | 11776 | $84 | 132 | 33792 | $DA | 218 | 55808 |
| $2F | 47 | 12032 | $85 | 133 | 34048 | $DB | 219 | 56064 |
| $30 | 48 | 12288 | $86 | 134 | 34304 | $DC | 220 | 56320 |
| $31 | 49 | 12544 | $87 | 135 | 34560 | $DD | 221 | 56576 |
| $32 | 50 | 12800 | $88 | 136 | 34816 | $DE | 222 | 56832 |
| $33 | 51 | 13056 | $89 | 137 | 35072 | $DF | 223 | 57088 |
| $34 | 52 | 13312 | $8A | 138 | 35328 | $E0 | 224 | 57344 |
| $35 | 53 | 13568 | $8B | 139 | 35584 | $E1 | 225 | 57600 |
| $36 | 54 | 13824 | $8C | 140 | 35840 | $E2 | 226 | 57856 |
| $37 | 55 | 14080 | $8D | 141 | 36096 | $E3 | 227 | 58112 |
| $38 | 56 | 14336 | $8E | 142 | 36352 | $E4 | 228 | 58368 |
| $39 | 57 | 14592 | $8F | 143 | 36608 | $E5 | 229 | 58624 |
| $3A | 58 | 14848 | $90 | 144 | 36864 | $E6 | 230 | 58880 |
| $3B | 59 | 15104 | $91 | 145 | 37120 | $E7 | 231 | 59136 |
| $3C | 60 | 15360 | $92 | 146 | 37376 | $E8 | 232 | 59392 |
| $3D | 61 | 15616 | $93 | 147 | 37632 | $E9 | 233 | 59648 |
| $3E | 62 | 15872 | $94 | 148 | 37888 | $EA | 234 | 59904 |
| $3F | 63 | 16128 | $95 | 149 | 38144 | $EB | 235 | 60160 |
| $40 | 64 | 16384 | $96 | 150 | 38400 | $EC | 236 | 60416 |
| $41 | 65 | 16640 | $97 | 151 | 38656 | $ED | 237 | 60672 |
| $42 | 66 | 16896 | $98 | 152 | 38912 | $EE | 238 | 60928 |
| $43 | 67 | 17152 | $99 | 153 | 39168 | $EF | 239 | 61184 |
| $44 | 68 | 17408 | $9A | 154 | 39424 | $F0 | 240 | 61440 |
| $45 | 69 | 17664 | $9B | 155 | 39680 | $F1 | 241 | 61696 |
| $46 | 70 | 17920 | $9C | 156 | 39936 | $F2 | 242 | 61952 |
| $47 | 71 | 18176 | $9D | 157 | 40192 | $F3 | 243 | 62208 |
| $48 | 72 | 18432 | $9E | 158 | 40448 | $F4 | 244 | 62464 |
| $49 | 73 | 18688 | $9F | 159 | 40704 | $F5 | 245 | 62720 |
| $4A | 74 | 18944 | $A0 | 160 | 40960 | $F6 | 246 | 62976 |
| $4B | 75 | 19200 | $A1 | 161 | 41216 | $F7 | 247 | 63232 |
| $4C | 76 | 19456 | $A2 | 162 | 41472 | $F8 | 248 | 63488 |
| $4D | 77 | 19712 | $A3 | 163 | 41728 | $F9 | 249 | 63744 |
| $4E | 78 | 19968 | $A4 | 164 | 41984 | $FA | 250 | 64000 |
| $4F | 79 | 20224 | $A5 | 165 | 42240 | $FB | 251 | 64256 |
| $50 | 80 | 20480 | $A6 | 166 | 42496 | $FC | 252 | 64512 |
| $51 | 81 | 20736 | $A7 | 167 | 42752 | $FD | 253 | 64768 |
| $52 | 82 | 20992 | $A8 | 168 | 43008 | $FE | 254 | 65024 |
| $53 | 83 | 21248 | $A9 | 169 | 43264 | $FF | 255 | 65280 |
| $54 | 84 | 21504 | $AA | 170 | 43520 | | | |
| $55 | 85 | 21760 | $AB | 171 | 43776 | | | |

# Want to use the programs in this guide?

# Can't afford the time to type them in?

# Why not buy them all on disk or cassette?

Typing in long programs can be a pretty daunting task. Once you've entered the program there will probably be typing errors that need to be corrected. Why not save yourself time and trouble by buying a disk or cassette containing all of the programs from this magazine at a bargain price of £6.00 (disk) or £4.00 (cassette).

The disk and cassette are only available by mail order from the address on the order form. A cheque payable to ASP Ltd for the correct amount should be included with the order. Overseas customers should add £1.00 for postage.

ORDER FORM — PLEASE COMPLETE IN BLOCK CAPITALS

| NAME | QUANTITY | PRICE | ORDER CODE | TOTAL |
|---|---|---|---|---|
| SERIOUS USER DISK | | £6.00 | YSUGD2 | |
| SERIOUS USER TAPE | | £4.00 | YSUGC2 | |
| OVERSEAS POSTAGE | | £1.00 | | |
| | | | TOTAL | |

NAME.....

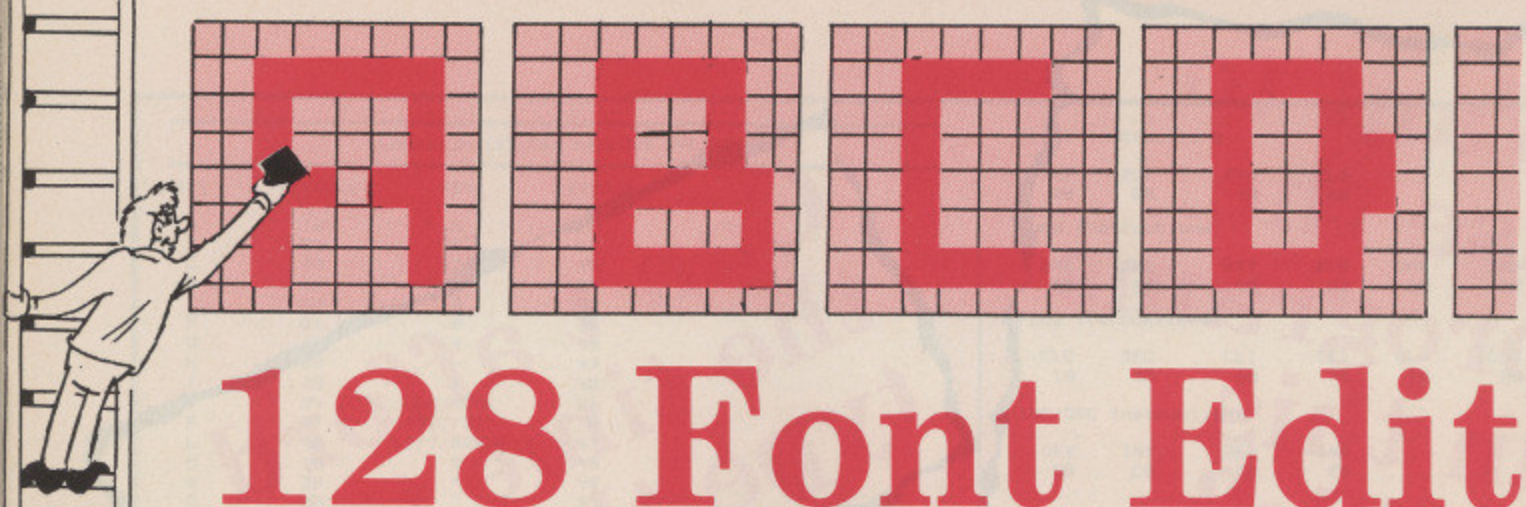
ADDRESS.....

.....POSTCODE.....

I enclose a cheque/postal order for £..... made payable to ASP LTD. for the Your Commodore Serious User Guide Disk/Tape.

All orders should be sent to: Your Commodore, Readers Services, Argus Specialist Publications, 9 Hall Road, Hemel Hempstead, Herts HP2 7BH.

Please allow 28 days for delivery.




# 128 Font Editor

*A character designer to turn your C128 into the definitive machine*

by Julian Mackintosh

User-defined characters can turn a so-so screen into a work of art but their creation involves the programmer in hours of preparatory work. Font Editor makes the C128's brain take the strain by providing an environment which is easy to master and can be expanded as the user's needs dictate.

Font Editor is a machine code program located between $3000 and $4000 (12288–16384), with two characters sets stored at $2000 (8184) and $2800 (10240). These locations are normally reserved for the screen in graphic modes 1, 2, 3 and 4, which means that the Font Editor can't be used alongside these four modes.

Before the editor is used, it is important to understand how the C128 character sets are normally used. The two sets are categorised as upper case with graphics, and upper case with lower case. Only one set can be used at a time and they can be swapped by holding down the SHIFT key and pressing the CBM key.

The character shapes are copied from a table in ROM but it is possible, by pointing the operating system towards an area of RAM, to load in two user-defined character sets. The 128 Font Editor has been written to help with the design of these alternative sets.

## Controlling the editor

The Font Editor screen is divided into several windows, or areas, which control all of the functions necessary for redefining the characters. By using a joystick in Port 2, the screen pointer can be moved from area to area and, when the desired mode is highlighted, the fire button is pressed to select the new mode.

The function of each window is shown in Diagram 1.

## The options

The 20 option menu allows complex operations, like mirroring and rotation of characters, to be performed at the touch of a button. Tables 1 and 2 show the variety of the functions available through these menus. The load and save operations only operate on the current character set, so the set has to be selected from the menu before these options are executed.

These tables only show the standard features but complex manipulations can be achieved by using two or more options in sequence. Any character can be flipped in the diagonal plane by first rotating the character through 90 or 270 degrees and then by flipping the character in the horizontal plane.

The best way to execute these special operations is through user-defined routines written in Basic or code. To execute such a function, select the End Editor option and the computer will return to the mode it was in when the editor was entered. For example, if the editor was called from within a program by a line such as:

10 BOOT"FONT EDITOR"

it will return to program control at the next program line. If it was entered from direct mode then the READY prompt will appear.

When the defined routine has been executed, the Font Editor can be re-entered with SYS DEC ("382F") as long as the option screen has not been corrupted by the new routine.

By using calls to the editor's code, it is possible to add extremely complex functions to the editor whenever your needs dictate.

## Defining new options

There are several things to bear in mind when defining an option. The editor code and character definition areas must not be corrupted by the new routines, but the input window (Area 4) can be used. To facilitate extended options, the function keys can be used when the program is in input mode but Area 4 should always be cleared before re-entering the editor.

The RUN/STOP and RESTORE function is not disabled by the program but, if the program is interrupted by using these keys, it can be restarted by SYS 12288. If, however, the screen has been corrupted, the editor will have to be rebooted before this system call.

Always remember that the cur-

rent character set remains operative when the End Editor option is used. This also applies to whether the set is taken from ROM or RAM.

The following breakdown of the Font Editor memory map will help when defining your own routines.

*$0A2C The lower nybble indicates the current source of the character set.*
*%xxxx0100 ($4) for ROM set 1*
*%xxxx0110 ($6) for ROM set 2*
*%xxxx1000 ($8) for RAM set 1*
*%xxxx1010 ($A) for RAM set 2*

*$0C00 Start of the relocated interrupt routine for moving the cursor around the screen. It appears here to avoid problems if the graphics screen is cleared.*

*$0D63 The X position of the pointer when the fire button was last pressed.*

*$0D64 The Y position corresponding to $0D63.*

*$0D65 Register for the currently selected menu option in Area 3. No option is selected if the value is $FF.*

*$0D66 Flag to indicate that the fire button has been pressed. This is reset from 1 to 0 when the signal is acknowledged.*

*$3000 SYS called for a cold start only after RUN/STOP and RESTORE have been pressed at the same time.*

*$30AE This routine clears the input window, Area 4.*

*$30D6 Forces Area 1 to display the current character held in the accumulator. It also updates Areas 2, 5, 6 and 7 accordingly. From Basic this is called by*
*SYS DEC("30D6"),sp*
*where sp is the screen poke value for the character to be displayed.*

*$3260 This inverts the pixel at the location specified by the accumulator (column) and the X register (row). From Basic:*
*SYS DEC("$3260"),c,r*
*where both variables are within the range zero to seven.*

*$3662 Over-writes the RAM characters with the ROM set.*

*$382F WARM START. This is the call for restarting the editor when re-entering from a user-defined option. Before calling this address remember to reset any windows that may have been used, especially Area 4.*

The following commands only affect the character indicated by Area 5. Any other characters which need to be altered must first be called to the screen by the routine at $30D6.

*$32D5 Clears the character in Area 1.*

*$32EC Inverts the current character.*

*$3306 Mirrors the bottom half of the current character into the top area.*

*$332D Mirrors the top of the current character into the bottom half.*

*$3354 Flip the character in the horizontal plane.*

*$3384 Rotates the character by 90 degrees.*

*$33D7 Mirrors the right half of the character onto the left side.*

*$340E Mirrors the left side of the character onto the right half.*

*$3445 Flips the character in the vertical plane.*

*$36A0 Copies one character from ROM into the current RAM slot.*

When the End Editor option is used, the pointer remains operative and can be used in the user-defined function.

Redefining characters can be enjoyable and satisfying, the 128 Font Editor can increase the enjoyment if it is used wisely.

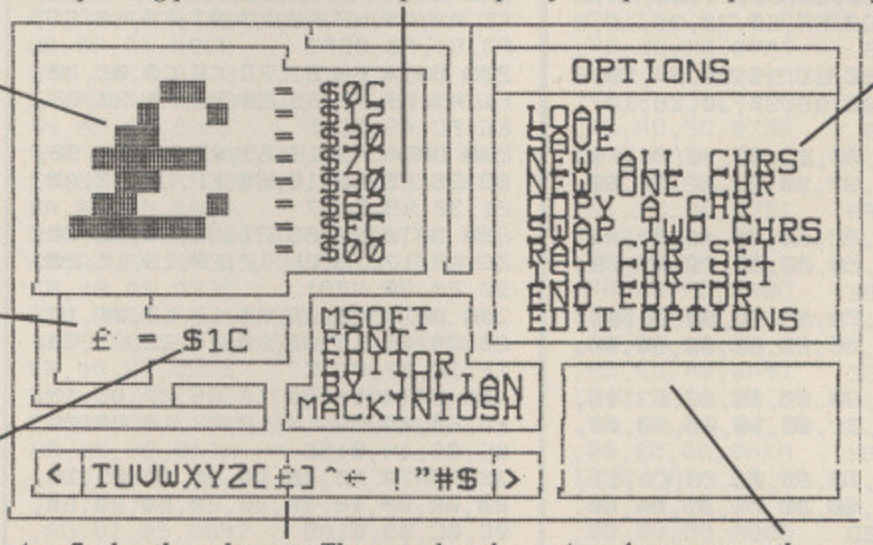
*Table 1: The Function Menu*

| *COMMAND* | *FUNCTION* |
|---|---|
| *LOAD* | *Loads a font from disk after allowing its name to be input via Area 4. The font is placed into the RAM locations for the currently selected character set.* |
| *SAVE* | *The current character set is saved to disk after its filename is input through Area 4.* |
| *NEW ALL* | *This replaces all of the current RAM characters with a copy of the corresponding ROM set* |

*Area 1 – displays an enlarged pixel map of the current character. By moving the pointer onto one of the pixels and pressing the fire button, the pixel is inverted. In other words, a pixel will be turned on if it is off and vice versa.*

*Area 2 – contains the current hexadecimal values of all the rows of the character in Area 1. This facility allows the user to enter a hexadecimal digit which is automatically poked into the corresponding character row. If, for example, the 'F' key is pressed when in this mode, the selected nybble of the current character row would be fully set (%1111). Such an alteration would be reflected by the turning on of all the corresponding pixels on the Area 1 pixel map.*

*Area 3 – main menu window which is subdivided into twenty options in two banks of ten. The pointer can be moved up and down the options until the desired action is highlighted. When the fire button is pressed, the option is executed. The first menu also shows which of the two character sets is currently on display and the final option swaps one menu for the other.*




*Area 5 – a similar colour function to Area 4 but this time the foreground is affected. It is also this area which displays the current character at its normal size.*

*Area 6 – displays the hexadecimal value of the screen poke for the character displayed in Areas 1 and 5. The value can be changed in the same way as the hexadecimal value in Area 2 and the corresponding character will appear in Areas 1 and 5 ready for editing.*

*Area 7 – has three elements. The central section shows a portion of the current character set which surround the character shown in Area 1. Two mid-line markers indicate which of these characters is currently displayed. By clicking the fire button on the arrow indicators at each side of the window the row can be scrolled backwards and forwards to select a new character for editing.*

*Area 4 – serves as the command line but can also be used to change the background colour. When an input is required, a prompt appears in this zone and the response is displayed as the characters are typed in.*
*To change the colour, the pointer is moved to this zone and, each time the fire button is pressed, the background is cycled on to the next of the sixteen colours in the range.*

*Diagram 1 – The Screen Layout*

| | |
|---|---|
| *NEW ONE* | *The character corresponding to the current RAM character is transferred from ROM.* |
| *COPY CHR* | *Replaces the current character with the one whose screen poke hexadecimal value is specified in the input window.* |
| *SWAP CHR* | *Behaves in a similar way to COPY CHAR but the two character shapes are swapped.* |
| *ROM/RAM CHR SET* | *The character set used to form the editor screen display is toggled between the ROM and the RAM sets.* |
| *1st/2nd CHR SET* | *Toggles between the first and second character sets.* |
| *END EDITOR* | *Returns control to the Basic mode from which the editor was entered.* |
| *OTHER OPTIONS* | *Selects the second set of menu options.* |

*Table 2: The Manipulation Menu*

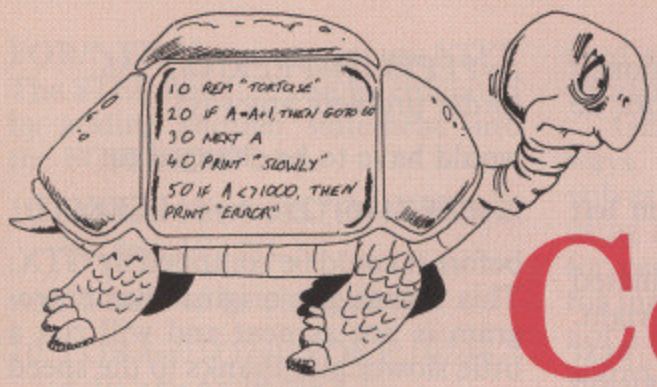
| *COMMAND* | *FUNCTION* |
|---|---|
| *MIRROR UP* | *Mirrors the bottom four lines of the character in the top four lines.* |
| *MIRROR DOWN* | *The reverse procedure of the MIRROR UP function.* |
| *MIRROR RIGHT* | *Takes a mirror image of the four right-hand columns and copies them into the left four columns.* |
| *MIRROR LEFT* | *Has the reverse effect of the MIRROR RIGHT function.* |
| *FLIP VERT* | *The current character is laterally reversed.* |
| *FLIP HORZ* | *Inverts the current character.* |
| *ROTATE* | *Rotates the current character through 90 degrees. Repeatedly select this option to rotate through 180 and 270 degrees.* |
| *INVERT* | *Each pixel is reversed out so that any on pixels are turned off and vice versa.* |
| *CLEAR* | *The current character is erased.* |
| *MAIN MENU* | *Selects the function menu.* |

PROGRAM: 128 FONT EDITOR

```
5 GRAPHIC1,1:REM MOVE BASIC AB
OVE M.CODE
6 GRAPHIC0,1
10 DATA 3000
20 DATA 4C,04,30,20,A9,28,8D,0
3,30,20,62,36,A9,20,8D,03,30,2
0;62,36,052A
30 DATA AD,2C,0A,29,F0,09,08,8
D,2C,0A,A2,00,BD,00,3C,9D,00,0
4,BD,00,05C9
40 DATA 3D,9D,00,05,BD,00,3E,9
D,00,06,BD,00,3F,9D,00,07,BD,0
0,3A,9D,05B1
50 DATA 00,0C,BD,00,3B,9D,00,0
D,E8,D0,D9,A9,0B,20,D2,FF,20,0
0,0C,4C,075C
60 DATA 2F,38,AE,96,30,E8,E0,1
0,D0,02,A2,00,EC,97,30,F0,F4,8
E,96,30,0A12
70 DATA 8A,48,A2,D8,86,FC,A2,0
0,86,FB,A0,00,91,FB,C8,D0,FB,E
6,FC,A6,0D38
80 DATA FC,E0,DC,D0,F1,68,AA,B
D,86,30,20,D2,FF,60,90,05,1C,9
F,9C,1E,0B59
90 DATA 1F,9E,81,95,96,97,98,9
9,9A,9B,01,02,AE,97,30,E8,E0,1
0,D0,02,0988
100 DATA A2,00,EC,96,30,F0,F4,
8E,21,D0,8E,97,30,60,A2,C2,86,
FB,A2,06,0AF9
110 DATA 86,FC,A2,00,A0,00,A9,
20,91,FB,C8,C0,0B,D0,F9,A5,FB,
18,69,28,0ABE
120 DATA 85,FB,A5,FC,69,00,85,
FC,E8,E0,05,D0,E3,60,8D,5D,06,
48,20,FB,0B3E
130 DATA 30,8D,62,06,8E,63,06,
68,48,20,10,31,68,18,69,F8,A0,
00,99,4D,0694
140 DATA 07,18,69,01,C8,C0,11,
D0,F5,60,18,48,20,05,31,AA,68,
4A,4A,4A,06ED
150 DATA 4A,29,0F,C9,0A,90,02,
E9,39,69,30,60,20,16,31,4C,31,
31,AE,03,05C8
160 DATA 30,86,FC,A2,00,86,FD,
A2,03,18,0A,26,FD,CA,D0,F9,85,
FB,A5,FD,0B76
170 DATA 65,FC,85,FC,60,A2,04,
8E,4F,31,A2,A4,8E,4E,31,A0,00,
A2,00,B1,093C
180 DATA FB,3D,88,31,F0,05,A9,
A0,4C,4D,31,A9,20,9D,E4,05,E8,
E0,08,D0,09E8
190 DATA EA,AD,4F,31,8D,6E,31,
AD,4E,31,8D,6D,31,B1,FB,20,FB,
30,48,8A,0963
200 DATA A2,0D,48,68,9D,BC,05,
CA,E0,0B,D0,F7,A9,28,18,6D,4E,
31,8D,4E,08E9
210 DATA 31,90,03,EE,4F,31,C8,
C0,08,D0,B6,60,80,40,20,10,08,
04,02,01,06A7
220 DATA 48,8A,48,AD,5D,06,20,
16,31,68,18,65,FB,85,FB,90,02,
E6,FC,68,08CD
230 DATA AA,BD,88,31,A2,00,41,
FB,81,FB,AD,5D,06,4C,10,31,C9,
0A,F0,20,08FA
240 DATA 18,0A,A8,88,88,B9,C7,
31,48,B9,C6,31,48,60,F7,34,57,
35,61,36,0879
250 DATA 9F,36,05,37,52,37,A5,
37,D8,37,2B,38,A2,22,86,FD,A2,
32,86,FE,0887
260 DATA A2,DF,86,FB,A2,04,86,
FC,A9,0A,48,A0,00,B1,FB,48,B1,
FD,91,FB,0BF3
270 DATA 68,91,FD,C8,C0,0E,D0,
F1,A5,FB,18,69,28,85,FB,90,02,
E6,FC,A5,0C2F
280 DATA FD,18,69,0E,85,FD,90,
02,E6,FE,68,18,69,FF,D0,D2,AD,
21,32,49,0A57
290 DATA 01,8D,21,32,60,00,20,
0D,09,12,12,0F,12,20,15,10,20,
20,20,20,0281
300 DATA 20,0D,09,12,12,0F,12,
20,04,0F,17,0E,20,20,20,0D,09,
12,12,0F,017C
310 DATA 12,20,12,09,07,08,14,
20,20,0D,09,12,12,0F,12,20,0C,
05,06,14,0156
320 DATA 20,20,20,06,0C,09,10,
20,08,0F,12,1A,20,20,20,20,20,
06,0C,09,01A9
330 DATA 10,20,16,05,12,14,20,
20,20,20,20,12,0F,14,01,14,05,
20,20,20,01C0
340 DATA 20,20,20,20,20,03,0C,
05,01,12,20,20,20,20,20,20,20,
20,20,09,01F0
350 DATA 0E,16,05,12,14,20,20,
20,20,20,20,20,20,A0,8D,81,89,8E,
A0,8F,90,05B3
360 DATA 94,89,8F,8E,93,A0,C9,
0A,D0,03,4C,D8,31,18,0A,AB,88,
88,B9,C4,09BF
370 DATA 32,48,B9,C3,32,48,60,
05,33,2C,33,0D,34,D6,33,53,33,
44,34,83,0632
380 DATA 33,D4,32,EB,32,AD,5D,
06,20,16,31,A0,00,98,91,FB,C8,
C0,08,D0,08F1
390 DATA F9,AD,5D,06,20,10,31,
60,AD,5D,06,20,16,31,A0,00,B1,
FB,49,FF,07D5
400 DATA 91,FB,C8,C0,08,D0,F5,
AD,5D,06,20,10,31,60,AD,5D,06,
20,16,31,0829
410 DATA A0,04,A2,03,86,FD,B1,
FB,84,FE,A4,FD,91,FB,E6,FE,88,
84,FD,A4,0DB8
420 DATA FE,98,C9,08,D0,EC,AD,
5D,06,20,10,31,60,AD,5D,06,20,
16,31,A2,080D
430 DATA 04,A0,03,86,FD,B1,FB,
84,FE,A4,FD,91,FB,C6,FE,C8,84,
FD,A4,FE,0E34
440 DATA 98,C9,FF,D0,EC,AD,5D,
06,20,10,31,60,AD,5D,06,20,16,
31,A0,00,0804
450 DATA A2,07,86,FD,B1,FB,48,
84,FE,A4,FD,B1,FB,85,FD,68,91,
FB,A5,FD,0E07
460 DATA 84,FD,A4,FE,91,FB,C6,
FD,C8,C0,04,D0,E3,AD,5D,06,20,
10,31,60,0882
470 DATA AD,5D,06,20,16,31,A2,
00,8A,9D,CF,33,E8,E0,08,D0,F8,
A2,00,A0,091C
480 DATA 00,B1,FB,3D,88,31,F0,
0A,B9,C7,33,18,7D,CF,33,9D,CF,
33,C8,C0,0A0D
490 DATA 08,D0,EA,E8,E0,08,D0,
E3,A0,00,B9,CF,33,91,FB,C8,C0,
08,D0,F6,0C82
500 DATA AD,5D,06,20,10,31,60,
01,02,04,08,10,20,40,80,00,10,
00,E3,0F,03D2
510 DATA 11,31,C7,AD,5D,06,20,
16,31,A0,00,B1,FB,29,F0,91,FB,
C8,C0,08,0901
520 DATA D0,F5,A0,00,A2,00,B1,
FB,3D,88,31,F0,08,BD,C7,33,18,
71,FB,91,0A6D
530 DATA FB,E8,E0,04,D0,EC,C8,
C0,08,D0,E5,AD,5D,06,20,10,31,
```

```
60,AD,5D,0AA3
540 DATA 06,20,16,31,A0,00,B1,
FB,29,0F,91,FB,C8,C0,08,D0,F5,
A0,00,A2,0914
550 DATA 04,B1,FB,3D,88,31,F0,
08,BD,C7,33,18,71,FB,91,FB,E8,
E0,08,D0,0B05
560 DATA EC,C8,C0,08,D0,E5,AD,
5D,06,20,10,31,60,A2,00,8A,9D,
CF,33,E8,09B5
570 DATA E0,08,D0,F8,AD,5D,06,
20,16,31,A0,00,A2,00,B1,FB,3D,
88,31,F0,08FB
580 DATA 0A,B9,CF,33,18,7D,C7,
33,99,CF,33,E8,E0,08,D0,EA,C8,
C0,08,D0,0AD9
590 DATA E3,A0,00,B9,CF,33,91,
FB,C8,C0,08,D0,F6,AD,5D,06,20,
10,31,60,09F1
600 DATA 18,A2,11,A0,1A,20,F0,
FF,60,A2,00,A9,20,9D,EC,34,E8,
E0,0B,D0,09BF
610 DATA F8,A2,00,86,FB,20,E4,
FF,F0,FB,48,A9,07,20,D2,FF,68,
C9,0D,F0,0C20
620 DATA 1A,C9,14,F0,17,C9,23,
90,E8,C9,7B,B0,E4,A4,FB,E6,FB,
99,EC,34,0C73
630 DATA 20,D2,FF,C0,0A,90,D6,
60,A4,FB,F0,D1,C6,FB,A9,9D,20,
D2,FF,A9,0D82
640 DATA 20,20,D2,FF,A9,9D,20,
D2,FF,A9,20,88,99,EC,34,4C,A1,
34,46,2E,09E7
650 DATA 20,20,20,20,20,20,20,
20,20,20,20,20,20,88,34,A2,00,
BD,AE,35,049E
660 DATA 20,D2,FF,E8,E0,17,D0,
F5,18,A2,13,A0,1A,20,F0,FF,20,
91,34,A9,0AB9
670 DATA 00,20,90,FF,A9,0D,A2,
EA,A0,34,20,BD,FF,A9,00,A2,08,
A0,00,20,08B4
680 DATA BA,FF,A9,00,A2,00,AC,
03,30,20,D5,FF,B0,07,AD,5D,06,
20,10,31,07FF
690 DATA 60,18,A2,15,A0,1D,20,
F0,FF,A2,00,BD,53,35,20,D2,FF,
E8,E0,05,09A0
700 DATA D0,F5,60,45,52,52,4F,
52,20,88,34,A2,00,BD,C6,35,20,
D2,FF,E8,09BE
710 DATA E0,17,D0,F5,A2,13,A0,
1A,18,20,F0,FF,20,91,34,A9,00,
20,90,FF,098F
720 DATA A9,0D,A2,EA,A0,34,20,
BD,FF,A9,00,A2,08,A0,01,20,BA,
FF,AE,03,0970
730 DATA 30,86,FC,A2,00,86,FB,
A9,FB,A2,00,AC,03,30,C8,C8,C8,
C8,C8,C8,0BAA
740 DATA C8,C8,20,D8,FF,B0,96,
AD,5D,06,20,10,31,60,20,20,20,
4C,4F,41,07DA
750 DATA 44,11,9D,9D,9D,9D,9D,
9D,46,4F,4E,54,20,4E,41,4D,45,
20,20,20,06DB
760 DATA 20,53,41,56,45,11,9D,
9D,9D,9D,9D,9D,46,4F,4E,54,20,
4E,41,4D,0741
770 DATA 45,20,20,20,20,43,4F,
50,59,11,9D,9D,9D,9D,9D,9D,43,
48,41,52,06DD
780 DATA 41,43,54,45,52,20,11,
9D,9D,9D,9D,9D,9D,9D,9D,9D,9D,
4E,55,4D,08B2
790 DATA 42,45,52,20,24,20,20,
20,4D,4F,56,45,11,9D,9D,9D,9D,
9D,9D,43,06B6
800 DATA 48,41,52,41,43,54,45,
52,20,11,9D,9D,9D,9D,9D,9D,9D,
9D,9D,9D,089D
810 DATA 4E,55,4D,42,45,52,20,
24,11,20,20,45,4E,54,45,52,20,
41,9D,9D,0577
820 DATA 9D,9D,9D,9D,9D,9D,9D,
11,48,45,58,49,44,45,43,49,4D,
41,4C,9D,0816
830 DATA 9D,9D,9D,9D,9D,9D,9D,
11,4E,55,4D,42,45,52,AD,00,FF,
48,A9,01,08C3
840 DATA 8D,00,FF,A9,00,85,FB,
AD,03,30,18,69,B0,85,FC,A9,00,
85,FD,AD,0A1F
850 DATA 03,30,85,FE,18,69,08,
AA,A0,00,B1,FB,91,FD,C8,D0,F9,
E6,FC,E6,0C1C
860 DATA FE,E4,FE,D0,EF,68,8D,
00,FF,AD,5D,06,20,10,31,60,AD,
5D,06,48,09BC
870 DATA 20,16,31,A5,FC,69,B0,
85,FE,A5,FB,85,FD,AD,00,FF,48,
A9,01,8D,0AF1
880 DATA 00,FF,A0,00,B1,FD,91,
FB,C8,C0,08,D0,F7,68,8D,00,FF,
68,20,10,0ABC
890 DATA 31,60,20,AE,30,20,88,
34,A0,00,B9,34,36,20,D2,FF,C8,
C0,2E,D0,08A5
900 DATA F5,20,E4,FF,F0,FB,A2,
00,DD,F6,36,F0,08,E8,E0,10,D0,
F6,4C,E1,0D51
910 DATA 36,60,30,31,32,33,34,
35,36,37,38,39,41,42,43,44,45,
46,20,88,04E0
920 DATA 34,A2,00,BD,DE,35,20,
D2,FF,E8,E0,2B,D0,F5,20,E1,36,
20,D2,FF,0B77
930 DATA 8A,0A,0A,0A,0A,85,FD,
20,E1,36,20,D2,FF,8A,05,FD,20,
16,31,A5,07F4
940 DATA FB,48,A5,FC,48,AD,5D,
06,20,16,31,68,85,FE,68,85,FD,
A0,00,B1,09C9
950 DATA FD,91,FB,C8,C0,08,D0,
F7,AD,5D,06,20,10,31,60,20,88,
34,A2,00,092F
960 DATA BD,09,36,20,D2,FF,E8,
E0,2B,D0,F5,20,E1,36,20,D2,FF,
8A,0A,0A,0A6B
970 DATA 0A,0A,85,FD,20,E1,36,
20,D2,FF,8A,05,FD,20,16,31,A5,
FB,48,A5,093E
980 DATA FC,48,AD,5D,06,20,16,
31,68,85,FE,68,85,FD,A0,00,B1,
FB,48,B1,09D5
990 DATA FD,91,FB,68,91,FD,C8,
C0,08,D0,F1,AD,5D,06,20,10,31,
60,AD,2C,0A7A
1000 DATA 0A,29,0C,C9,04,D0,15
,AD,2C,0A,29,F3,09,08,8D,2C,0A
,AD,D1,05,0647
1010 DATA 29,80,09,01,8D,D1,05
,60,AD,2C,0A,29,F3,09,04,8D,2C
,0A,AD,D1,06C3
1020 DATA 05,29,80,09,0F,8D,D1
,05,60,AD,2C,0A,29,02,F0,23,A9
,8E,20,D2,06D3
1030 DATA FF,A2,00,BD,F8,05,29
,80,1D,26,38,9D,F8,05,E8,E0,03
,D0,F0,A9,0A4D
1040 DATA 20,8D,03,30,AD,5D,06
,20,10,31,60,A9,0E,20,D2,FF,A2
,00,BD,F8,07B0
1050 DATA 05,29,80,1D,29,38,9D
,F8,05,E8,E0,03,D0,F0,A9,28,8D
,03,30,AD,088F
1060 DATA 5D,06,20,10,31,60,31
,13,14,32,0E,04,68,68,60,AE,66
,0D,F0,FB,05FC
1070 DATA AD,65,0D,C9,FF,F0,1B
,48,A0,00,8C,66,0D,20,AE,30,68
,AE,21,32,0840
1080 DATA D0,06,20,B4,31,4C,2F
,38,20,AE,32,4C,2F,38,AD,63,0D
,AC,64,0D,067B
1090 DATA C9,04,90,22,C9,0C,B0
,1E,C0,04,90,1A,C0,0C,B0,16,88
,88,88,88,0842
1100 DATA 84,FB,A6,FB,18,69,FC
,20,90,31,A0,00,8C,66,0D,4C,2F
,38,C9,04,089D
1110 DATA 90,17,C9,07,B0,13,C0
,0E,90,0F,C0,11,B0,0B,20,52,30
,A0,00,8C,0701
1120 DATA 66,0D,4C,2F,38,C9,1A
,90,17,C9,25,B0,13,C0,11,90,0F
,C0,16,B0,0757
1130 DATA 0B,20,98,30,A0,00,8C
,66,0D,4C,2F,38,C9,02,90,1D,C9
,05,B0,19,0654
1140 DATA C0,14,90,15,C0,17,B0
,11,EE,5D,06,AD,5D,06,20,D6,30
,A0,00,8C,07C4
1150 DATA 66,0D,4C,2F,38,C9,16
,90,1D,C9,19,B0,19,C0,14,90,15
,C0,17,B0,075D
1160 DATA 11,CE,5D,06,AD,5D,06
,20,D6,30,A0,00,8C,66,0D,4C,2F
,38,C9,10,06A3
1170 DATA 90,23,C9,12,B0,1F,C0
,04,90,1B,C0,0C,B0,17,18,69,F0
,88,88,88,0868
1180 DATA 88,85,FB,98,A6,FB,20
,42,39,A9,00,8D,66,0D,4C,2F,38
,C9,0A,90,089B
1190 DATA 1A,C9,0C,B0,16,C0,0F
,90,12,C0,10,B0,0E,18,69,F6,20
,7D,39,A0,07A1
1200 DATA 00,8D,66,0D,4C,2F,38
,4C,2F,38,48,8A,48,20,CE,36,86
,FE,AD,5D,0732
1210 DATA 06,20,16,31,68,D0,19
,68,A8,A5,FE,0A,0A,0A,0A,85,FE
,B1,FB,29,07F1
1220 DATA 0F,05,FE,91,FB,AD,5D
,06,20,D6,30,60,68,A8,B1,FB,29
,F0,05,FE,0A0C
1230 DATA 91,FB,AD,5D,06,20,D6
,30,60,48,20,CE,36,86,FE,A9,5D
,85,FB,A9,0A41
1240 DATA 06,85,FC,68,AA,A9,00
,48,E0,00,D0,D8,4C,53,39,00,00
,00,00,00,06EA
1250 DATA 00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00
,00,00,00,0000
1260 DATA 00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00
,00,00,00,0000
1270 DATA 00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00
,00,00,00,0000
1280 DATA 00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00
```

```
,00,00,00,0000
1290 DATA 00,00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00,00
,00,00,00,0000
1300 DATA A2,40,CA,BD,67,0D,9D
,00,0E,E0,00,D0,F5,AD,04,0A,25
,FE,8D,04,089C
1310 DATA 0A,AD,15,D0,29,FE,09
,01,8D,15,D0,A9,5E,8D,00,D0,8D
,01,D0,AD,08AE
1320 DATA 10,D0,29,FE,8D,10,D0
,A9,38,8D,F8,07,A9,F1,8D,27,D0
,AD,14,03,09C3
1330 DATA 8D,5F,0D,AD,15,03,8D
,60,0D,78,A9,52,8D,14,03,A9,0C
,8D,15,03,0629
1340 DATA 58,60,AD,65,0D,C9,FF
,F0,03,20,38,0D,AD,00,DC,AA,29
,01,D0,09,082D
1350 DATA A9,FC,18,6D,01,D0,8D
,01,D0,8A,29,02,D0,09,A9,04,18
,6D,01,D0,07EA
1360 DATA 8D,01,D0,8A,29,04,D0
,13,A9,FC,18,6D,00,D0,8D,00,D0
,B0,08,AD,08B4
1370 DATA 10,D0,49,01,8D,10,D0
,8A,29,08,D0,13,A9,04,18,6D,00
,D0,8D,00,06C4
1380 DATA D0,90,08,AD,10,D0,49
,01,8D,10,D0,8A,48,AD,00,D0,4A
,AA,AD,10,08AC
1390 DATA D0,29,01,F0,05,8A,18
,69,80,AA,8A,4A,4A,18,69,FD,8D
,61,0D,AD,0868
1400 DATA 01,D0,4A,4A,18,69,01
,4A,18,69,F9,8D,62,0D,68,29,10
,D0,4E,AE,0714
1410 DATA 66,0D,D0,10,AD,61,0D
,CD,63,0D,D0,08,AD,62,0D,CD,64
,0D,F0,11,07DE
1420 DATA AD,61,0D,8D,63,0D,AD
,62,0D,8D,64,0D,A9,01,8D,66,0D
,AD,61,0D,06F7
1430 DATA C9,25,B0,19,C9,17,90
,15,AD,62,0D,18,69,FB,C9,0A,B0
,0B,69,01,07CC
1440 DATA 8D,65,0D,20,38,0D,4C
,5C,0D,A9,FF,8D,65,0D,4C,5C,0D
,A2,00,8E,06A5
1450 DATA 63,0D,A0,00,8C,64,0D
,4C,01,0D,EA,EA,A0,B6,84,B2,A0
,04,84,B3,08A2
1460 DATA AA,A9,28,18,65,B2,90
,02,E6,B3,85,B2,CA,D0,F2,A0,0E
,A9,80,18,0A87
1470 DATA 71,B2,91,B2,88,D0,F6
,60,6C,5F,0D,00,00,00,00,00,00
,FF,00,C0,07AB
1480 DATA 00,00,E0,00,00,E0,00
,00,70,18,00,70,FE,00,7B,EF,00
,3F,77,00,05D6
1490 DATA 3B,B1,80,3D,8D,C0,1D
,BD,C0,1C,7B,E0,1D,F7,E0,0B,EF
,F0,0F,DF,0AD3
1500 DATA F0,07,1F,F8,07,CF,F8
,03,EF,F0,01,F7,E0,00,FF,C0,00
,7F,00,00,09D4
1510 DATA 1C,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00
,00,00,00,001C
1520 DATA 00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00
,00,00,00,0000
1530 DATA 00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00
,00,00,00,0000
1540 DATA 00,00,00,00,00,00,00
,00,00,00,00,00,00,00,00,00,00
,00,00,00,0000
1550 DATA 00,00,00,00,00,00,00
,00,00,00,00,00,20,20,20,20,20
,20,20,20,0100
1560 DATA 20,20,20,20,20,20,20
,20,20,20,20,20,20,20,20,20,20
,20,20,20,0280
1570 DATA 20,20,20,20,20,20,20
,20,20,20,20,20,20,70,40,40,40
,40,40,40,0390
1580 DATA 40,40,40,40,40,40,72
,40,40,40,40,40,40,72,40,40,40
,40,40,40,0564
1590 DATA 40,40,40,40,40,40,40
,40,40,40,6E,20,20,5D,70,40,40
,40,40,40,053B
1600 DATA 40,40,40,40,40,6E,6D
,40,40,40,40,40,6E,5D,70,40,40
,40,40,40,05D6
1610 DATA 40,40,40,40,40,40,40
,40,40,6E,5D,20,20,5D,5D,20,20
,20,20,20,04A5
1620 DATA 20,20,20,20,20,6D,40
,40,40,40,40,6E,5D,5D,5D,20,20
,20,0F,10,0451
1630 DATA 14,09,0F,0E,13,20,20
,20,20,5D,5D,20,20,5D,5D,20,20
,20,20,20,0321
1640 DATA 20,20,20,20,20,3D,20
,24,30,30,20,5D,5D,5D,6B,40,40
,40,40,40,0463
1650 DATA 40,40,40,40,40,40,40
,40,40,73,5D,20,20,5D,5D,20,20
,20,20,20,04AA
1660 DATA 20,20,20,20,20,3D,20
,24,30,30,20,5D,5D,5D,5D,20,0C
,0F,01,04,0355
1670 DATA 20,20,20,20,20,20,20
,20,20,5D,5D,20,20,5D,5D,20,20
,20,20,20,0374
1680 DATA 20,20,20,20,20,3D,20
,24,30,30,20,5D,5D,5D,5D,20,13
,01,16,05,0364
1690 DATA 20,20,20,20,20,20,20
,20,20,5D,5D,20,20,5D,5D,20,20
,20,20,20,0374
1700 DATA 20,20,20,20,20,3D,20
,24,30,30,20,5D,5D,5D,5D,20,0E
,05,17,20,037F
1710 DATA 01,0C,0C,20,03,08,12
,13,20,5D,5D,20,20,5D,5D,20,20
,20,20,20,02DD
1720 DATA 20,20,20,20,20,3D,20
,24,30,30,20,5D,5D,5D,5D,20,0E
,05,17,20,037F
1730 DATA 0F,0E,05,20,03,08,12
,20,20,5D,5D,20,20,5D,5D,20,20
,20,20,20,02F3
1740 DATA 20,20,20,20,20,3D,20
,24,30,30,20,5D,5D,5D,5D,20,03
,0F,10,19,0370
1750 DATA 20,01,20,03,08,12,20
,20,20,5D,5D,20,20,5D,5D,20,20
,20,20,20,0312
1760 DATA 20,20,20,20,20,3D,20
,24,30,30,20,5D,5D,5D,5D,20,13
,17,01,10,0370
1770 DATA 20,14,17,0F,20,03,08
,12,13,5D,5D,20,20,5D,5D,20,20
,20,20,20,02FE
1780 DATA 20,20,20,20,20,3D,20
,24,30,30,20,5D,5D,5D,5D,20,12
,01,0D,20,0375
1790 DATA 03,08,12,20,13,05,14
,20,20,5D,5D,20,20,5D,5D,20,20
,20,20,20,02FD
1800 DATA 20,20,20,20,20,70,40
,40,40,40,40,7D,6D,7D,5D,20,31
,13,14,20,04AC
1810 DATA 03,08,12,20,13,05,14
,20,20,5D,5D,20,20,5D,6D,72,40
,40,40,72,0411
1820 DATA 40,40,40,40,40,7D,70
,40,40,40,40,40,40,6E,5D,20,05
,0E,04,20,04CF
1830 DATA 05,04,09,14,0F,12,20
,20,20,5D,5D,20,20,6B,6E,5D,20
,20,20,6D,03A4
1840 DATA 40,40,40,40,40,6E,5D
,0D,13,0F,06,14,20,5D,5D,20,05
,04,09,14,0374
1850 DATA 20,0F,10,14,09,0F,0E
,13,20,5D,5D,20,20,5D,5D,5D,20
,20,20,3D,035A
1860 DATA 20,24,32,30,20,5D,5D
,06,0F,0E,14,20,20,5D,6D,40,40
,40,40,40,0401
1870 DATA 40,40,40,40,40,40,40
,40,40,7D,5D,20,20,5D,5D,5D,20
,20,20,70,0541
1880 DATA 40,40,40,40,40,7D,5D
,05,04,09,14,0F,12,6D,40,40,6E
,70,40,40,04AC
1890 DATA 40,40,40,40,40,40,40
,40,40,6E,5D,20,20,5D,5D,6D,40
,40,40,7D,05AF
1900 DATA 70,40,40,40,6E,70,7D
,02,19,20,0A,15,0C,09,01,0E,5D
,5D,20,20,0403
1910 DATA 20,20,20,20,20,20,20
,20,20,5D,5D,20,20,5D,6D,40,40
,40,40,40,0424
1920 DATA 71,40,40,40,73,5D,0D
,01,03,0B,09,0E,14,0F,13,08,5D
,5D,20,03,034F
1930 DATA 31,32,38,20,04,09,13
,03,20,5D,5D,20,20,6B,40,40,40
,40,40,40,03E3
1940 DATA 40,40,40,40,7D,6D,40
,40,40,40,40,40,40,40,40,40,7D
,5D,20,20,0584
1950 DATA 16,05,12,13,09,0F,0E
,20,20,5D,5D,20,20,5D,70,40,72
,40,40,40,03DF
1960 DATA 40,40,40,40,40,72,40
,40,40,40,40,40,40,72,40,6E
,5D,20,28,0577
1970 DATA 03,29,20,20,31,39,38
,37,20,5D,5D,20,20,5D,5D,3C,5D
,18,19,1A,03FD
1980 DATA 1B,1C,1D,1E,1F,20,21
,22,23,24,25,26,27,28,5D,3E,5D
,5D,20,20,036A
1990 DATA 20,20,20,20,20,20,20
,20,20,5D,5D,20,20,5D,6D,40,71
,40,40,40,0455
2000 DATA 40,40,40,40,40,71,40
,40,40,40,40,40,40,40,71,40,7D
,6D,40,40,05CC
2010 DATA 40,40,40,40,40,40,40
,40,40,7D,5D,20,20,6D,40,40,40
,40,40,40,0547
2020 DATA 40,40,40,40,40,40,40
,40,40,40,40,40,40,40,40,40,40
,40,40,40,0500
2030 DATA 40,40,40,40,40,40,40
,40,40,40,7D,20,20,20,20,20,20
,20,20,20,041D
2040 DATA 20,20,20,20,20,20,20
,20,20,20,20,20,20,20,20,20,20
,20,20,20,0280
2050 DATA 20,20,20,20,20,20,20
,20,20,20,20,20,00,00,00,00,00
,00,00,00,0180
2060 DATA END
63995 PRINT"DATA ERROR IN LINE
"PEEK(65)+256*PEEK(66):END
63999 LOOP:INPUT"FILENAME FOR
SAVE:";N$:BSAVE(N$),B0,P(S)TOP
(E):END
```




# GTX Compiler

*Supercharge your Basic programs by converting them to pure machine code*

The GTX Compiler is designed to convert a Commodore 64 program written in Basic into high speed, machine code. This code makes calls to standard functions stored in memory, called the run-time library which means that the compiled program is shorter because frequently-used routines such as add or subtract can be stored as subroutines.

Before using GTX, the program to be compiled must already be saved on tape or disk. When the compiler runs, the title page will appear and three prompts for inputs given in sequence.

The SOURCE NAME is the name of the Basic program to be compiled. If using tape, press RETURN to load the program.

The OBJECT NAME is the name that the compiled program will be saved under. If using tape, RETURN will cause the program to be saved without a name.

SIGN SPACE refers to the printing of numbers. In Basic a number is preceded by a sign space and ended with a space. Pressing RETURN will print numbers in this way. Altering the Y to N will print numbers with neither of the two spaces.

The program specified by the source name will then be loaded and compiling will begin. There are two parts to this process:

**PASS 1**
Each command is deciphered and the corresponding machine code is produced. The current Basic program line number is printed along with the present length of the compiled program but, if any errors occur, the compiler stops and the error message is printed. The Basic program must then be re-loaded and modified if it is to be compiled successfully.

Apart from errors, any number of warnings may be given. These do not stop the compiler since the program can still be compiled but the results obtained with the original, Basic version will differ from those in the compiled version. Again, the original program must be modified if the results are to tally.

**PASS 2**
All jumps such as GOTO and GOSUB have the correct addresses inserted and any DATA is transferred within the program.

If compiling succeeds, then the Basic program length and the compiled program length are printed along with the following options:

**R** Run the compiled program
**S** Save the compiled program
**O** Output the run-time library
**C** Compile another program
**E** exit

It is advised that the program is saved before running (option S) in case there is a problem. If using tape, a copy of the run-time library must be saved directly after the compiled program (option O). If using disk, a single copy of the run-time library will serve all the programs on the disk. Finally, the compiled program can be run using option R.

## Acceptable Basic

The compiler does not accept the full set of commands available in Basic. Those that can be compiled are:

PRINT (including , and ;)
POKE
PEEK
IF/THEN
READ/DATA/RESTORE
FOR/NEXT
GOSUB/RETURN
GOTO
CLR
SYS
REM
AND
OR
END
STOP

There are also restrictions placed on these commands but, although all this seems rather daunting, most programs require very little attention before they can be successfully compiled.

**All numbers are 16 bit un-signed integers**
This obviously means that GTX can not be used for complex programs

*by R D Bell*

that require floating point numbers but the main concept for GTX is to transform relatively simple Basic programs, such as games, to fast machine code programs. This type of program usually only requires integer numbers so this limitation is not a major one. The lack of negative numbers can cause problems in converting a program that was not specifically written to be compiled. However, this limitation can normally be solved by changing the style of programming to incorporate numbers between 0 and 65535.

For example, part of a normal Basic program consists of the following:

```
10 PRINT"[CLS]"
20 P=10*40+1024+20:D=1
30 A=P:P=P+D
40 IFP>=10*40+1024+39
   THEND=-D
50 IFP<=10*40+1024
   THEND=-D
60 FORB=1TO500:NEXT
70 POKEP,88:POKEA,32:
   GOTO30
```

As it stands this would not compile because of the negation of D in lines 40 and 50. The possible values for D are 1 and −1, the latter being out of range on GTX. Therefore, before the program could be compiled, a few simple modifications would need to be made.

Lines 30 to 50 must be changed to the following:

```
30 A=P:P=P-1:IFD=1
   THENP=P+2
40 IFP>=10*40+1024+39
   THEND=1-D
50 IFP<=10*40+1024
   THEND=1-D
```

Now the program could be compiled. In fact, when it was compiled it ran 36 times faster than the original.

**Variables can only be single letters**
This can cause problems if the original has a large number of variables. To assist with this problem there is a single array available, Z, which is pre-dimensioned to 64 elements, ie. Z(0) to Z(63). The DIM in the original program is not required by GTX but if included it will be skipped. Any further dimensioned arrays will cause an error to be flagged.

**Expressions are evaluated from left to right**
In Basic expressions are calculated in a set order, not from left to right. Brackets take top priority, next come indices, followed by division and multiplication, then addition and subtraction, with Boolean expressions (AND, OR, NOT) taking lowest priority. For example,

```
A=3+6*2
```

In Basic this expression would evaluate A as being equal to 15: 6*2=12 and 12+3=15. If compiled A would be calculated as being equal to 18: 3+6=9 and 9*2+18. To ensure the compiled program and the Basic program obtain the same value the expression would need to be changed to:

```
A=6*2+3
```

This can cause problems when a program was written without compiling in mind. By checking the order of all expressions and changing this where necessary, the compiled program should obtain the same results as the Basic version. During Pass 1 of compiling the order of expressions are checked and, if discrepancies are likely to occur, a warning is given. The indicated expressions should be re-ordered to ensure the program will operate correctly.

**Brackets are not valid expressions**
Brackets are only allowed with a PEEK command. Any other brackets must be removed before the program can be compiled. Usually a number of dummy variables are required to achieve this. An expression such as

```
A=5+((3*X)-8)/7-T/(4*Y-2)
```

would have to be changed to:

```
A=3*X-8/7+5:B=4*Y-2:B=T/B:
A=A-B
```

**Only one PEEK per expression**
An expression such as this:

```
A=PEEK(45)+PEEK(46)*256
```

(order problem anyway)

would have to be changed to:

```
A=PEEK(46)*256:A=A+PEEK(45)
```

before it could be compiled by GTX. This means the original Basic program is not as neat and will run a little slower but, thanks to the speed increase given by GTX, this is insignificant when compiled.

**No strings**
Strings are not incorporated at all. The use of strings in the type of programs intended to be compiled is usually very infrequent. Their only use is usually connected with a GET. There are a number of methods of emulating GET with the commands available, this is discussed later.

**AND and OR can not be used in IF/THEN comparisons**
A program line which takes the form

```
10 IFA=3ORA=6THEN 1000
```

would have to be split into two separate lines before it could be compiled:

```
10 IFA=3THEN1000
11 IFA=6THEN1000
```

The AND statement must also be changed when used in the context:

```
10 IFA=4ANDA=7THEN1000
```

This would become:

```
10 IFA=4THENIFA=7THEN1000
```

AND and OR are included in the list of available commands but are only legal when acting as arithmetic operators. Take the situation where it is necessary to strip off the five most significant bits from the contents of location 197.

```
B=PEEK(197)AND7
```

In this case, the AND is not operating as a logic comparator so the expression can be compiled. If AND or OR are used in an IF/THEN statement their operation would still be treated as arithmetical and the Basic program would not operate in the same way as the compiled version.

**FOR/NEXT loops can not use STEP**
The STEP function can be emulated by adding another statement into the FOR/NEXT loop.

```
10 FORA=0TO940STEP40:POKE
   1024+A,102:NEXT
```

would have to be changed to:

```
10 FORA=0TO940:POKE:1024+
   A,102:A=A+39:NEXT
```

before it could be compiled. Note that A is only increased by 39 because the NEXT statement automatically increments it by 1, resulting in a step of 40 for each loop.

If the STEP is negative, it creates more of a problem.

```
10 FORA=20TO0STEP-2:
   PRINTA:NEXT
```

would have to be changed to:

```
10 FORB=0TO20:A=20-B:
---PRINTA:B=B+1:NEXT
```

Either of the above cases could be solved by a conditional GOTO loop. Taking another example:

```
10 FOREA=920TO40STEP-40:
   POKE1024+A,160:NEXT
```

could be changed to:

```
 9 A=920
10 POKE1024+A,160:A=A-40:
   IFA>=40THEN10
```

**READ/DATA**
This is almost the same as Basic except that the numbers stored in the DATA statements can only be eight bits (0 to 255 decimal). This is not a serious limitation in games because most DATA statements are used for storing user-defined characters or sprite data, where the values are only eight bits anyway.

RESTORE is exactly the same as in Basic.

The Z( ) array is restricted
This array must always be used through another variable.

```
10 POKE Z(A),44
```

must be changed to:

```
10 B=Z(A):POKEB,44
```

## Missing commands

The missing commands are rarely used and can often be forgotten or emulated using the legal commands.

GET can be emulated in two ways. The first, and simplest, involves reading the keyboard directly using a PEEK. The values returned for each key are non-standard but for inputting only a few keys this is not a problem. The second method allows the keys to be read in ASCII and actually uses the same Kernal ROM routine as GET.

```
10 GETA$:IFA$="A"THEN100
```

can be changed to either:

```
10 IFPEEK(197)=10THEN100
```

or

```
10 SYS65508:IFPEEK(780)=
   65THEN100
```

In the first case, a value is returned for as long as the key is held down, unlike the second case where the value is returned only once for each key press.

RND can be emulated by reading certain memory locations that change frequently. Two of the best locations to use are the raster register (53266) and the CIA timer A (56324).

```
10 A=INT(RND(1)*10)
```

would have to be changed to:

```
10 A=PEEK(56324)AND15:
   IFA>9THEN10
```

A different type of random statement would be:

```
10 IFRND(1)<.1THENPRINT
   "HELLO"
```

Such a statement has a 10% probability of printing 'HELLO'. The best way to achieve this with the compiler is to change it to:

```
10 IFPEEK(56324)<25THEN
   PRINT"HELLO"
```

This also gives a 10% probability because there are 256 possible values for location 56324. The statement only reacts to 25 of these values, so the probability calculation becomes (25/256)*100, which equals 10.

CHR$ statements can usually be changed to a direct PRINT. If it is not possible to do this in your program then this alternative may be used:

```
10 PRINT CHR$(A*4-9)
```

can be emulated by:

```
10 POKE780,A*4-9:SYS65490:
   PRINT
```

The extra PRINT is required because the character is printed without a carriage return.

## Typing it in

If you are using a disk then the programs can be entered and saved in any order. With a tape they must be saved in the correct order.

First, enter Program 1, the loader that changes the address at which the compiler is loaded. GTX must always be loaded at this new location or it will be over-written by the program to be compiled. Save this loader at the start of the tape.

Next, enter Program 2, the actual compiler. Do not run it yet because it must be loaded at the new address. It can be typed in at its run-location by entering the following direct command before starting.

```
POKE44,100:POKE43,1:POKE
25600,0:NEW
```

When the program has been entered, save it using the filename "GTX COMPILER", following on directly from Program 1.

Ensure that the computer has been reset before typing in Program 3, the run-time library. Before running, save this program on a spare tape or disk in case of fatal errors. Run the program and insert the original tape or disk, taking care not to erase the other two programs. The run-time library will be saved with the filename "RTL". The run-time library must always be saved with this name even when working with a cassette tape.

You should now have a complete Basic compiler.

## Testing the compiler

The compiler is now ready to be tested. Enter Program 4, the test program, and save it. Load and run GTX and compile the program. If

any errors are detected, correct your test program and try again. Run the compiled program using option 'R' and you should see GTX COMPILER filling the screen.

If you run the program in Basic first, you will notice the speed increase, typically 45 times. Pressing 'R' will re-start and 'E' will exit back to Basic.

## Memory requirements

The compiler uses the following locations which should not be altered by the compiled program. If they are, crashes will inevitably occur.

**$C000–$C2FF (49152–49919)** These locations contain the compilers run-time library. This must always be present whenever the compiler itself or a compiled program is used. In both cases, it will be loaded automatically by the program. If it can't be found, the program will crash.

**$CF00–$CFFF (52992–53247)** The compiled program's variables are stored here. This location can be changed, if required, by altering the value of VA in line 11 of the compiler.

A compiled program also uses a few page zero locations which, again, must not be altered by the compiled program:

$02 (2)
$14–$15 (20–21)
$39–$3C (57–60)
$3F–$42 (63–66)
$FB–$FE (251–254)

## Program Description

**Line numbers**

| | |
|---|---|
| 30–890 | Prints current line number and length. Deciphers next command and jumps to corresponding part of the program. |
| 1000–2510 | Evaluates expression. |
| 3000–3070 | GOTO and GOSUB |
| 3200–3930 | IF/THEN |
| 4000–4600 | PRINT |
| 5000–5030 | POKE |
| 5100–5160 | FOR |
| 5200 5260 | NEXT |
| 5300–5310 | REM |
| 5400–5420 | DIM |
| 5500–5570 | SYS |
| 5600–5670 | DATA |
| 5700–5770 | READ |
| 8000–8030 | Prints 'insert tape', or disk, depending on current device and then waits for a key to be pressed. |
| 9000–9070 | Opening title page. Inputs source name, object name and sign space. |
| 9100–9120 | Loads program specified by the source name from the current device to $0801 (2049 decimal). |
| 9500–9505 | Relocates program to run at $0820 (2080 decimal). |
| 9510–9570 | Prints original length and compiled length along with the total number of warnings. The options are also printed. |
| 9580–9650 | Inputs selected option. |
| 9700–9720 | Checks if run-time library is present, if not it is loaded off the current device to $C000 (49152 decimal). |
| 9800–9830 | Saves the compiled program to current device. |
| 10000–10190 | Inserts the correct addresses for all GOTOs and GOSUBs. |
| 10200–10220 | Copies all DATA from its temporary store, starting at $C700 (50944 decimal), to its correct place in the compiled program. |
| 20000–20550 | Copies the machine code out of the run-time library to its correct place in the compiled program. |
| 25000–26050 | Simplifies the machine code, if possible. |
| 30000–32000 | Error messages. |

**Variables**

| | |
|---|---|
| PR | Pointer to the next byte of the Basic program. |
| L%( ) | Line numbers of the Basic program. |
| A%( ) | Start address of the machine code for each Basic line. |
| J%( ) | Address of each jump, such as GOTO and GOSUB. |
| T%( ) | Address of each THEN. |
| N% | Number of Basic lines encountered so far. |
| CM% | Current Basic token being processed. |
| VA | Start address of compiled program's variables. |
| JP | Number of GOTOs and GOSUBs so far. |
| TH | Number of THENs encountered so far. |
| DD | Start of temporary store for DATA ($C700, 50944 decimal). |
| EN | End address of program being compiled. |
| DA | Start address of machine code. |
| AD | Address to store next byte of machine code. |
| AJ | Adjust value to relocate program to $0820 (2080 decimal). |
| DV | Current I/O device (1=tape and 8=disk). |

PROGRAM: GTX COMPILER - LOADER

```
5E  10 PRINT"[CLR,DOWN3]POKE44,1
    00:POKE43,1:POKE25600,0:NEW"

37  20 PRINT"[DOWN2]LOAD"CHR$(34
    )"GTX COMPILER"CHR$(34)","PE
    EK(186)
60  30 PRINT"[DOWN9]RUN[HOME]"
CF  40 FORA=0TO9:POKE631+A,13:NE
    XT:POKE198,10
B2  50 END
```

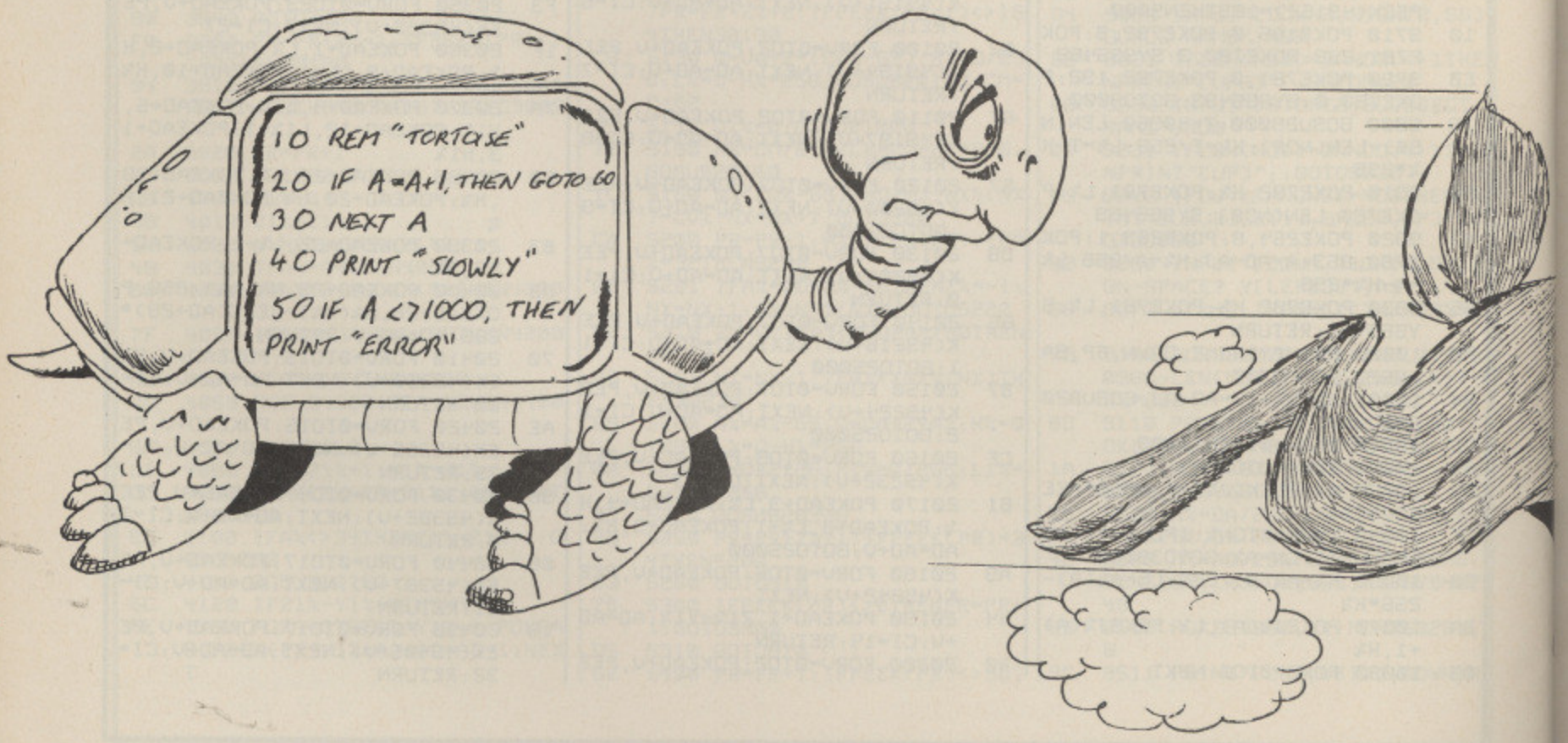
PROGRAM: GTX COMPILER - MAIN

```
16  1 GOTO9700
A3  10 PR=2049:DIML%(300),A%(300
    ),J%(300),I%(300),N%(25)
1F  11 N%=-1:VA=52992:JP=0:TH=0:
    DD=50944:WA$="[HOME,DOWN2]"
F1  15 PRINT"[CLR,SP,SA,SS2] 1"
F0  20 EN=PEEK(173)*256+PEEK(172
    ):AD=EN+4:IFAD<2110THENAD=21
    10
F5  25 DA=AD:AJ=AD-2107:GOSUB205
    50:GOSUB20500
BA  30 LN=PEEK(PR+3)*256+PEEK(PR
    +2):PR=PR+4
CA  31 IFPR>=ENTHEN10000
49  33 IFAD>25500THEN30210
58  35 PRINT"[HOME,DOWN]"LN;AD-D
    A
65  40 N%=N%+1:L%(N%)=LN:A%(N%)=
    AD-AJ
10  50 CM%=PEEK(PR)
FE  60 IFCM%=153THEN4000
A3  70 IFCM%>64ANDCM%<91THEN2000

26  80 IFCM%=137ORCM%=141THEN300
    0
D5  90 IFCM%=128ORCM%=142ORCM%=1
    44THENGOSUB20270:Z%=0:GOTO90
    0
0F  100 IFCM%=139THEN3200
AF  110 IFCM%=151THEN5000
93  120 IFCM%=129THEN5100
B4  130 IFCM%=130THEN5200
D4  140 IFCM%=156THENGOSUB20500:
    Z%=0:GOTO900
07  150 IFCM%=143THEN5300
74  160 IFCM%=134THEN5400
79  170 IFCM%=158THEN5500
51  180 IFCM%=131THEN5600
F3  190 IFCM%=140THENGOSUB20540:
    Z%=0:GOTO900
50  200 IFCM%=135THEN5700
4E  210 IFCM%=136THENPR=PR+1:GOT
    O50
A9  890 GOTO30200
25  900 PR=PR+Z%+1
B4  910 IFPEEK(PR)=58THENPR=PR+1
    :GOTO50
67  920 IFPEEK(PR)=0THENPR=PR+1:
    GOTO30
C5  930 GOTO30150
14  1000 Z%=0:WA%=9
2C  1010 Z9%=PEEK(PR+Z%):IFZ9%=0
    ORZ9%=58ORZ9%=44ORZ9%=164THE
    N1020
44  1011 IFZ9%=59ORZ9%=34ORZ9%=1
    78ORZ9%=179ORZ9%=177ORZ9%=16
    7THEN1020
D8  1013 Z%=Z%+1:GOTO1010
4B  1020 Z%=Z%-1:A%=0:B%=0:C%=0:
    FORA=0TO2%
35  1030 IFPEEK(PR+A)=194ANDA%=1
    THEN30000
4C  1040 IFPEEK(PR+A)=194ANDPEEK
    (PR+A+1)<>40THEN30010
99  1050 IFPEEK(PR+A)=194THENA%=
    1:C%=1:GOTO1100
3C  1060 IFPEEK(PR+A)=40ANDB%=1T
    HEN30020
41  1070 IFPEEK(PR+A)=40THENB%=1
    :GOTO1100
8B  1080 IFPEEK(PR+A)=41ANDB%=0T
    HEN30030
70  1090 IFPEEK(PR+A)=41THENB%=0
    :A%=0
D2  1100 NEXT
7C  1110 IFC%=1THEN1150
FE  1120 X%=0:Y%=Z%:O4%=0:GOSUB2
    100
94  1130 GOSUB20120
F6  1140 RETURN
DE  1150 O6%=0:FORG=0TO2%:IFPEEK
    (PR+G)=194THENO6%=O6%+1
18  1160 NEXT:X%=2:Y%=X%
73  1170 IFPEEK(PR+Y%)<>41THENY%
    =Y%+1:GOTO1170
B8  1180 Y%=Y%-1:O1%=0:GOSUB2100
    :GOSUB20130
64  1190 GOSUB20150:X%=Y%+2:IFO6
    %=1THEN1260
B4  1200 GOSUB20120:O4%=1:A%=PEE
    K(PR+X%):GOSUB2230:O2%=O1%
16  1210 IFPEEK(PR+X%+1)<>194THE
    N30000
D2  1220 X%=X%+3:Y%=X%
C6  1230 IFPEEK(PR+Y%)<>41THENY%
    =Y%+1:GOTO1230
EC  1240 Y%=Y%-1:O1%=0:GOSUB2100
    :GOSUB20130
22  1250 GOSUB20140:O1%=O2%:GOSU
    B2500:X%=Y%+2
47  1260 O4%=2:O1%=0:Y%=Z%:IFX%>
    =Y%THEN1280
0D  1270 GOSUB2100:O1%=0
11  1280 GOSUB20120:RETURN
A2  2000 PR=PR+1
C7  2001 IFPEEK(PR)=40ANDCM%<>90
    THEN30150
91  2002 IFPEEK(PR)=40THENOO=PR+
    1:GOTO2050
A6  2009 OO=0:IFPEEK(PR)<>178THE
    N30040
B7  2010 PR=PR+1:IFPEEK(PR+1)=40
    ANDPEEK(PR)<>90THEN2015
62  2011 IFPEEK(PR+1)=40THENGOSU
    B2090:OO=1:GOTO2019
B7  2015 O1%=0:GOSUB1000
E5  2019 Q=VA+2*(CM%-65):H%=Q/25
    6:L%=Q-H%*256
1F  2020 GOSUB20160
31  2030 IFOO=1THEN2056
CF  2040 GOTO900
CE  2050 IFPEEK(PR)<>178THENPR=P
    R+1:GOTO2050
D6  2051 PR=PR+1
12  2052 IFPEEK(PR+1)=40ANDPEEK(
    PR)=90THEN30150
00  2053 O1%=0:GOSUB1000
B9  2054 GOSUB20310
24  2055 PR=OO-2:GOSUB2090
81  2056 IFPEEK(PR)=0ORPEEK(PR)=
    58THEN910
C0  2057 PR=PR+1:GOTO2056
A5  2080 X%=0:Y%=0
36  2081 IFPEEK(PR+Y%)<>41THENY%
    =Y%+1:GOTO2081
14  2082 Y%=Y%-1:RETURN
E8  2090 PR=PR+2:O1%=0:GOSUB2080
    :GOSUB2100
4F  2091 IFOO=0THENGOSUB20120:GO
    SUB20310:GOSUB20480:RETURN
EA  2092 GOSUB20490:RETURN
5D  2100 A$="":O%=0:FORA=X%TOY%+
    1:A%=PEEK(PR+A)
29  2101 IFO4%=2THENO4%=0:GOTO22
    30
CF  2110 IFA%>47ANDA%<58ANDO%=1T
    HEN30050
04  2111 IFA%>47ANDA%<58THENA$=A
    $+CHR$(A%):GOTO2300
B4  2120 IFA$=""THEN2160
8C  2130 Q=VAL(A$):H%=Q/256:L%=Q
    -H%*256:A$=""
CF  2131 IFO1%<>0THEN2151
9B  2140 GOSUB20000
2F  2150 O%=1:GOTO2230
90  2151 GOSUB20040
87  2152 O%=1:GOSUB2500:GOTO2230

AF  2160 IFA%>64ANDA%<91ANDO%=1T
    HEN30050
D1  2170 IFA%>64ANDA%<91THEN2190

07  2180 GOTO2220
4A  2190 Q=VA+2*(A%-65):H%=Q/256
    :L%=Q-H%*256
FA  2191 IFO1%<>0THEN2211
DD  2200 GOSUB20020
C1  2210 O%=1:GOTO2300
DE  2211 GOSUB20060
09  2212 O%=1:GOSUB2500:GOTO2300

AC  2220 IFO%=0THEN30050
12  2230 IFA%=170THENO1%=3:O%=0:
    GOTO2300
B4  2240 IFA%=172THENO1%=5:O%=0:
    GOTO2300
F0  2250 IFA%=171THENO1%=4:O%=0:
    GOTO2300
67  2251 IFA%=173THENO1%=6:O%=0:
    GOTO2300
69  2252 IFA%=175THENO1%=1:O%=0:
    GOTO2300
89  2260 IFA%=176THENO1%=2:O%=0:
    GOTO2300
CB  2261 IFA%=0ORA%=58ORA%=44ORA
    %=41ORA%=59ORA%=34THEN2300
AC  2262 IFA%=178ORA%=179ORA%=17
    7ORA%=167ORA%=164THEN2300
24  2270 GOTO30060
37  2300 IFO4%=1THENO4%=0:RETURN
49  2310 NEXT:RETURN
C3  2500 ONO1%GOSUB20460,20470,2
    0080,20100,20090,20110
AC  2510 IFWA%<INT((O1%+1)/2)THE
    NWN=WN+1:WA$=WA$+"[DOWN]":PR
    INTWA$"[SL]INE"LN"WARNING"
E4  2520 WA%=INT((O1%+1)/2):RETU
    RN
BB  3000 PR=PR+1:A$="":FORZ=0TO6
    :A%=PEEK(PR+Z)
5F  3010 IFA%=58ORA%=0THEN3040
2C  3020 IFA%>47ANDA%<58THENA$=A
```

```
    $+CHR$(A%):NEXT
1C  3030 GOTO30070
EB  3040 Q=VAL(A$):H%=Q/256:L%=Q
    -H%*256:Z%=Z-1
00  3050 IFCM%=141THEN3070
F1  3060 GOSUB20230:GOTO900
69  3070 GOSUB20250:GOTO900
10  3200 PR=PR+1:IFPEEK(PR)=167T
    HEN30090
B8  3201 O1%=0
C2  3210 GOSUB1000:PR=PR+Z%+1:A%
    =PEEK(PR):A1%=PEEK(PR+1)
F2  3220 CP%=0:IFA%=167THEN3700
90  3230 PR=PR+2:IF(A%=179ANDA1%
    =177)OR(A%=177ANDA1%=179)THE
    NCP%=1:GOTO3300
6D  3240 IF(A%=177ANDA1%=178)OR(
    A%=178ANDA1%=177)THENCP%=2:G
    OTO3300
4A  3250 IF(A%=179ANDA1%=178)OR(
    A%=178ANDA1%=179)THENCP%=3:G
    OTO3300
AF  3255 IF(A1%<64ORA1%>90)AND(A
    1%<47ORA1%>58)ANDA1%<>194THE
    N30120
F4  3260 PR=PR-1:IFA%=177THENCP%
    =4:GOTO3300
28  3270 IFA%=178THENCP%=5:GOTO3
    300
F6  3280 IFA%=179THENCP%=6:GOTO3
    300
22  3290 GOTO30110
13  3300 GOSUB20310
BC  3310 O1%=0:GOSUB1000:IFPEEK(
    PR+Z%+1)<>167THEN30120
FB  3320 ONCP%GOSUB20330,20410,2
    0450,20440,20320,20420
81  3330 GOTO3800
C4  3700 GOSUB20300
F1  3710 Z%=-1:GOTO3800
CF  3800 T%(TH)=AD+1:TH=TH+1
19  3810 GOSUB20280
80  3840 GOTO3900
F0  3900 PR=PR+Z%+1:IFPEEK(PR)<>
    167THEN30120
94  3910 IFPEEK(PR+1)>47ANDPEEK(
    PR+1)<58THENCM%=137:GOTO3000

2A  3920 PR=PR+1
25  3930 GOTO50
85  4000 Z1%=0
48  4010 Z1%=Z1%+1
14  4020 A%=PEEK(PR+Z1%)
48  4030 IFA%=34THEN4080
E8  4050 IFA%>64ANDA%<91THEN4200

7F  4051 IFA%>47ANDA%<58THEN4200

A9  4052 IFA%=194THEN4200
AD  4060 GOSUB4500:IFA=1THEN4400

EA  4061 GOTO4010
D1  4080 Z1%=Z1%+1:Y1%=Z1%
CC  4090 A%=PEEK(PR+Z1%):IFA%=0T
    HEN4110
9E  4100 IFA%<>34THENZ1%=Z1%+1:G
    OTO4090
4E  4110 GOSUB20180
6C  4120 IFZ1%-Y1%=0THEN4161
6E  4160 FORA=0TOZ1%-Y1%-1:POKEA
    D,PEEK(PR+Y1%+A):AD=AD+1:NEX
    T
B4  4161 IFA%=0THENGOSUB4500:GOT
    O4400
E0  4170 Z1%=Z1%+1:A%=PEEK(PR+Z1
    %):GOSUB4500:IFA=1THEN4400
72  4180 GOTO4010
E9  4200 R1=PR:PR=PR+Z1%
F4  4209 O4%=0:O1%=0:O2%=0:O%=0:
    GOSUB1000
C2  4210 GOSUB20210
E7  4220 PR=R1:Z1%=Z1%+Z%:A%=PEE
    K(PR+Z1%)
6C  4230 GOSUB4500:IFA=1THEN4400

A4  4231 IFA%=34THENZ1%=Z1%-1
AE  4240 GOTO4010
D0  4400 Z%=Z1%-1:GOTO900
25  4500 A=0:IFA%=59ORA%=34THENR
    ETURN
E9  4510 IFA%=44THEN4550
3C  4520 IFA%>64ANDA%<91THENRETU
    RN
CD  4530 IFA%=194THENRETURN
59  4531 IFA%=0ORA%=58THEN4570
43  4540 RETURN
0E  4550 GOSUB20200
6F  4560 RETURN
23  4570 A=1:IFPEEK(PR+Z1%-1)=59
    ORPEEK(PR+Z1%-1)=44THENRETUR
    N
2E  4580 GOSUB20220
87  4600 RETURN
3E  5000 PR=PR+1:O1%=0:GOSUB1000
    :GOSUB20310
3F  5010 PR=PR+Z%+2:O1%=0:GOSUB1
    000
D3  5020 GOSUB20340
89  5030 GOTO900
74  5100 PR=PR+1:CV%=PEEK(PR):IF
    CV%<65ORCV%>90THEN30130
E2  5110 PR=PR+1:IFPEEK(PR)<>178
    THEN30130
54  5120 PR=PR+1:O1%=0:GOSUB1000
    :PR=PR+Z%+2:IFPEEK(PR-1)<>16
    4THEN30130
C8  5130 Q=VA+2*(CV%-65):H%=Q/25
    6:L%=Q-H%*256:GOSUB20160:CM=
    Q+52
95  5140 O1%=0:GOSUB1000
7B  5150 H%=CM/256:L%=CM-H%*256:
    GOSUB20160
1C  5160 N%(CV%-65)=AD-AJ:NX%(NX
    )=CV%:NX=NX+1:GOTO900
CA  5200 PR=PR+1:A%=PEEK(PR):IFA
    %=32THEN5200
6B  5201 IFA%=0ORA%=58THENZ%=-1:
    NX=NX-1:A%=NX%(NX):GOTO5220
EF  5210 Z%=0:IFA%<65ORA%>90THEN
    30140
B3  5211 NX=NX-1:IFA%<>NX%(NX)TH
    EN30230
99  5220 A%=A%-65:Q=VA+2*A%:H%=Q
    /256:L%=Q-H%*256
0A  5230 Q=Q+2*26:H1%=Q/256:L1%=
    Q-H1%*256
33  5240 GOSUB20350
45  5250 PR=PR+Z%+1:IFPEEK(PR)=3
    4THEN5200
83  5260 GOTO910
75  5300 IFPEEK(PR)<>0THENPR=PR+
    1:GOTO5300
D5  5310 GOTO910
6E  5400 PR=PR+1:IFPEEK(PR)<>90T
    HEN30160
3F  5410 IFPEEK(PR)<>0ANDPEEK(PR
    )<>58THENPR=PR+1:GOTO5410
60  5420 GOTO910
27  5500 PR=PR+1:O1%=0:GOSUB1000

76  5560 GOSUB20510
D7  5570 GOTO900
35  5600 A$="":PR=PR+1
8E  5610 A%=PEEK(PR):IFA%<48ORA%
    >57THEN5630
71  5620 A$=A$+CHR$(A%):PR=PR+1:
    GOTO5610
F4  5630 Q=VAL(A$):IFQ>255THEN30
    170
EF  5640 POKEDD,Q:DD=DD+1
67  5650 IFA%=44THEN5600
89  5660 IFA%=58ORA%=0THEN910
61  5670 GOTO30180
DC  5700 PR=PR+1:A%=PEEK(PR)
19  5710 IFA%<65ORA%>90THEN30190
62  5720 Q=VA+2*(A%-65):H%=Q/256
    :L%=Q-H%*256
AE  5730 GOSUB20520
D4  5740 PR=PR+1:A%=PEEK(PR)
7A  5750 IFA%=44THEN5700
FF  5760 IFA%=0ORA%=58THEN910
D4  5770 GOTO30190
FD  8000 PRINT"[DOWN,SPC5]";:IFD
    V=1THENPRINT"POSITION TAPE";
    :GOTO8020
8F  8010 PRINT" INSERT DISK";
38  8020 PRINT" AND PRESS A KEY[
    DOWN]":POKE198,0:WAIT198,1:P
    OKE198,0:RETURN
46  9000 POKE53280,14:POKE53281,
    6:POKE198,0
7C  9005 PRINTCHR$(14)"[CLR,WHIT
    E,DOWN]  [SG,ST,SX] [SC,SO,S
    M,SP,SI,SL,SE,SR,SPC10][SC]
    ) [SR] [SD] [SB]ELL"
1C  9010 PRINT"  [CY12]"
D4  9020 INPUT"[DOWN2,RIGHT2,SS]
    OURCE NAME";NS$
D6  9021 IFLEN(NS$)=0ANDDV<>1THE
    NPRINT"[UP3]";:GOTO9020
68  9030 INPUT"[RIGHT2,SO]BJECT
    NAME";NO$
0F  9031 IFLEN(NO$)=0ANDDV<>1THE
    NPRINT"[UP]";:GOTO9030
05  9040 IFDV<>1ANDNS$=NO$THENPR
    INT"[UP,RIGHT14,SPC20,UP]":G
    OTO9030
32  9050 INPUT"[DOWN,RIGHT2,SS]I
    GN-SPACE? Y[LEFT3]";A$
6E  9070 SP%=0:IFA$="Y"THENSP%=1

7B  9100 GOSUB8000:POKE185,0:T=4
    0960-LEN(NS$):H%=T/256:L%=T-
    H%*256
8D  9110 POKE782,H%:POKE781,L%:P
    OKE780,LEN(NS$):SYS65469
1A  9120 POKE781,1:POKE782,8:POK
    E780,0:SYS65493:GOTO10
E9  9500 H%=DA/256:L%=DA-H%*256:
    POKE251,L%:POKE252,H%
42  9505 H%=AD/256:L%=AD-H%*256:
    POKE253,L%:POKE254,H%:SYS497
    46
4B  9510 POKE53280,14:POKE53281,
    6
9C  9511 PRINT"[CLR,WHITE,DOWN,R
```

```
    IGHT3,C@16]":PRINT"[RIGHT3,R
    VSON,SP,SR,SO,SG,SR,SA,SM,SS
    PC,SC,SO,SM,SP,SI,SL,SE,SD,R
    VSOFF]"
D1  9512 PRINT"[DOWN,RIGHT3,SB,S
    A,SS,SI,SC] LENGTH:"EN-2049;
    TAB(26)"[2049";-EN"[LEFT]]"
9C  9513 PRINT"[RIGHT3,SC]OMPILE
    D LENGTH:"AD-DA+58;TAB(26)"[
    2049";-(2107+AD-DA)"[LEFT]]"
2A  9514 PRINT"[DOWN,RIGHT3,SW]A
    RNINGS:"WN
EB  9520 PRINT"[DOWN2,RIGHT3,SS]
    ELECT ONE OF THE FOLLOWING:-
    "
48  9530 PRINT"[DOWN,RIGHT6,RVSO
    N,SS,RVSOFF]  SAVE COMPILED
    PROGRAM"
30  9540 PRINT"[DOWN,RIGHT6,RVSO
    N,SR,RVSOFF]  RUN COMPILED P
    ROGRAM AND EXIT"
05  9550 PRINT"[DOWN,RIGHT6,RVSO
    N,SO,RVSOFF]  OUTPUT RUN-TIM
    E LIBRARY"
64  9560 PRINT"[DOWN,RIGHT6,RVSO
    N,SC,RVSOFF]  COMPILE ANOTHE
    R PROGRAM"
29  9570 PRINT"[DOWN,RIGHT6,RVSO
    N,SE,RVSOFF]  RETURN TO BASI
    C"
9A  9580 POKE198,0
46  9600 GETA$:IFA$="R"THENSYS20
    80:END
57  9610 IFA$="C"THENRUN
5D  9620 IFA$="S"THENGOSUB9800:G
    OTO9510
CE  9630 IFA$="O"THENGOSUB8000:S
    YS49854:GOTO9510
C5  9640 IFA$="E"THENEND
E1  9650 GOTO9600
6E  9700 POKE1020,82:POKE1021,84
    :POKE1022,76:DV=PEEK(186):IF
    PEEK(49152)=169THEN9000
10  9710 POKE185,0:POKE782,3:POK
    E781,252:POKE780,3:SYS65469
EB  9720 POKE781,0:POKE782,192:P
    OKE780,0:SYS65493:GOTO9000
19  9800 GOSUB8000:T=40960-LEN(N
    S$)-LEN(NO$):H%=T/256:L%=T-H
    %*256
70  9810 POKE782,H%:POKE781,L%:P
    OKE780,LEN(NO$):SYS65469
43  9820 POKE254,8:POKE253,1:POK
    E780,253:A=AD-AJ:H%=A/256:L%
    =A-H%*256
55  9830 POKE782,H%:POKE781,L%:S
    YS65496:RETURN
FD  10000 PRINT"[HOME,DOWN,SP,SA
    ,SS2] 2[SPC7]"
34  10001 A%(N%+1)=AD-AJ:GOSUB20
    270
DD  10010 IFJP=0THEN10100
43  10020 FORA=0TOJP-1
69  10030 A%=PEEK(J%(A))+256*PEE
    K(J%(A)+1)
DF  10040 FORB=0TON%:IFL%(B)<>A%
    THENNEXT:LN=A%:GOTO30080
D0  10050 H%=A%(B)/256:L%=A%(B)-
    256*H%
BA  10070 POKEJ%(A),L%:POKEJ%(A)
    +1,H%
06  10080 FORB=0TO0:NEXT
13  10090 NEXT
26  10100 IFTH=0THEN10200
51  10120 FORA=0TOTH-1
59  10130 A%=PEEK(T%(A))+256*PEE
    K(T%(A)+1)
BB  10140 FORB=0TON%:IFL%(B)<>A%
    THENNEXT:LN=A%:GOTO30080
55  10150 B=B+1:H%=A%(B)/256:L%=
    A%(B)-256*H%
FE  10170 POKET%(A),L%:POKET%(A)
    +1,H%
A2  10180 FORB=0TO0:NEXT
77  10190 NEXT
B5  10200 IFDD=50944THEN9500
B3  10210 FORA=0TODD-50945:POKEA
    D+A,PEEK(50944+A):NEXT:B=AD-
    AJ
84  10220 H%=B/256:L%=B-H%*256:P
    OKEDA+3,H%:POKEDA+1,L%:AD=AD
    +A:GOTO9500
B8  20000 FORV=0TO7:POKEAD+V,PEE
    K(49152+V):NEXT
53  20010 POKEAD+1,L%:POKEAD+5,H
    %:AD=AD+V:CI=1:GOTO25000
7F  20020 FORV=0TO9:POKEAD+V,PEE
    K(49160+V):NEXT
97  20030 POKEAD+1,L%:POKEAD+2,H
    %:POKEAD+6,L%+1:POKEAD+7,H%:
    AD=AD+V:CI=2:GOTO25000
2C  20040 FORV=0TO7:POKEAD+V,PEE
    K(49170+V):NEXT
25  20050 POKEAD+1,L%:POKEAD+5,H
    %:AD=AD+V:CI=3:GOTO25000
7E  20060 FORV=0TO9:POKEAD+V,PEE
    K(49178+V):NEXT
19  20070 POKEAD+1,L%:POKEAD+2,H
    %:POKEAD+6,L%+1:POKEAD+7,H%:
    AD=AD+V:CI=4:GOTO25000
C3  20080 FORV=0TO2:POKEAD+V,PEE
    K(49188+V):NEXT:AD=AD+V:CI=5
    :RETURN
52  20090 FORV=0TO2:POKEAD+V,PEE
    K(49191+V):NEXT:AD=AD+V:CI=6
    :RETURN
62  20100 FORV=0TO2:POKEAD+V,PEE
    K(49194+V):NEXT:AD=AD+V:CI=7
    :RETURN
6E  20110 FORV=0TO2:POKEAD+V,PEE
    K(49197+V):NEXT:AD=AD+V:CI=8
    :RETURN
67  20120 FORV=0TO7:POKEAD+V,PEE
    K(49200+V):NEXT:AD=AD+V:CI=9
    :GOTO25000
DB  20130 FORV=0TO7:POKEAD+V,PEE
    K(49208+V):NEXT:AD=AD+V:CI=1
    0:RETURN
85  20140 FORV=0TO7:POKEAD+V,PEE
    K(49216+V):NEXT:AD=AD+V:CI=1
    1:GOTO25000
B7  20150 FORV=0TO7:POKEAD+V,PEE
    K(49224+V):NEXT:AD=AD+V:CI=1
    2:GOTO25000
CF  20160 FORV=0TO9:POKEAD+V,PEE
    K(49232+V):NEXT:CI=13
61  20170 POKEAD+3,L%:POKEAD+4,H
    %:POKEAD+8,L%+1:POKEAD+9,H%:
    AD=AD+V:GOTO25000
A9  20180 FORV=0TO6:POKEAD+V,PEE
    K(49242+V):NEXT
44  20190 POKEAD+1,Z1%-Y1%:AD=AD
    +V:CI=14:RETURN
B0  20200 FORV=0TO2:POKEAD+V,PEE
    K(49249+V):NEXT:AD=AD+V:CI=1
    5:RETURN
57  20210 IFSP%=1THENGOSUB20430
8A  20211 FORV=0TO6:POKEAD+V,PEE
    K(49252+V):NEXT:AD=AD+V:IFSP
    %=1THENGOSUB20430
D7  20212 CI=16:RETURN
DD  20220 FORV=0TO4:POKEAD+V,PEE
    K(49259+V):NEXT:AD=AD+V:CI=1
    7:RETURN
D4  20230 FORV=0TO2:POKEAD+V,PEE
    K(49264+V):NEXT:J%(JP)=AD+1
A0  20240 POKEAD+1,L%:POKEAD+2,H
    %:AD=AD+V:JP=JP+1:CI=18:RETU
    RN
4F  20250 FORV=0TO2:POKEAD+V,PEE
    K(49267+V):NEXT:J%(JP)=AD+1
C5  20260 POKEAD+1,L%:POKEAD+2,H
    %:AD=AD+V:JP=JP+1:CI=19:RETU
    RN
9D  20270 POKEAD,PEEK(49270):AD=
    AD+1:CI=20:RETURN
32  20280 FORV=0TO2:POKEAD+V,PEE
    K(49271+V):NEXT
E3  20290 POKEAD+2,LN/256:POKEAD
    +1,LN-PEEK(AD+2)*256:AD=AD+V
    :CI=21:RETURN
26  20300 FORV=0TO7:POKEAD+V,PEE
    K(49274+V):NEXT:AD=AD+V:CI=2
    2:RETURN
A0  20310 FORV=0TO7:POKEAD+V,PEE
    K(49282+V):NEXT:AD=AD+V:CI=2
    3:GOTO25000
1B  20320 FORV=0TO11:POKEAD+V,PE
    EK(49290+V):NEXT:AD=AD+V:CI=
    24:RETURN
16  20330 FORV=0TO11:POKEAD+V,PE
    EK(49302+V):NEXT:AD=AD+V:CI=
    25:RETURN
7B  20340 FORV=0TO5:POKEAD+V,PEE
    K(49314+V):NEXT:AD=AD+V:CI=2
    6:RETURN
F3  20350 FORV=0TO29:POKEAD+V,PE
    EK(49320+V):NEXT:CI=27
1F  20360 POKEAD+1,L%:POKEAD+2,H
    %:POKEAD+9,L%+1:POKEAD+10,H%
7A  20370 POKEAD+4,L1%:POKEAD+5,
    H1%:POKEAD+12,L1%+1:POKEAD+1
    3,H1%
BD  20380 POKEAD+17,L%:POKEAD+18
    ,H%:POKEAD+20,L%:POKEAD+21,H
    %
B3  20390 POKEAD+25,L%+1:POKEAD+
    26,H%
DB  20400 POKEAD+29,N%(A%)/256:P
    OKEAD+28,N%(A%)-PEEK(AD+29)*
    256:AD=AD+V:RETURN
70  20410 FORV=0TO15:POKEAD+V,PE
    EK(49350+V):NEXT:AD=AD+V:CI=
    28:RETURN
AE  20420 FORV=0TO15:POKEAD+V,PE
    EK(49366+V):NEXT:AD=AD+V:CI=
    29:RETURN
96  20430 FORV=0TO4:POKEAD+V,PEE
    K(49382+V):NEXT:AD=AD+V:CI=3
    0:RETURN
08  20440 FORV=0TO17:POKEAD+V,PE
    EK(49387+V):NEXT:AD=AD+V:CI=
    31:RETURN
18  20450 FORV=0TO17:POKEAD+V,PE
    EK(49405+V):NEXT:AD=AD+V:CI=
    32:RETURN
```

```
71  20460 FORV=0TO2:POKEAD+V,PEE
    K(49423+V):NEXT:AD=AD+V:CI=3
    3:RETURN
45  20470 FORV=0TO2:POKEAD+V,PEE
    K(49426+V):NEXT:AD=AD+V:CI=3
    4:RETURN
F4  20480 FORV=0TO2:POKEAD+V,PEE
    K(49429+V):NEXT:AD=AD+V:CI=3
    5:RETURN
EF  20490 FORV=0TO2:POKEAD+V,PEE
    K(49432+V):NEXT:AD=AD+V:CI=3
    6:RETURN
13  20500 FORV=0TO2:POKEAD+V,PEE
    K(49435+V):NEXT:AD=AD+V:CI=3
    7:RETURN
99  20510 FORV=0TO11:POKEAD+V,PE
    EK(49438+V):NEXT:AD=AD+V:CI=
    38:RETURN
A2  20520 FORV=0TO6:POKEAD+V,PEE
    K(49450+V):NEXT
AE  20530 POKEAD+1,L%:POKEAD+3,H
    %:AD=AD+V:CI=39:RETURN
19  20540 FORV=0TO2:POKEAD+V,PEE
    K(49457+V):NEXT:AD=AD+V:CI=4
    0:RETURN
32  20550 FORV=0TO11:POKEAD+V,PE
    EK(49460+V):NEXT:AD=AD+V:CI=
    41:RETURN
2E  25000 IFLI=1ANDCI=9THENAD=AD
    -8:POKEAD-5,20:POKEAD-1,21:L
    I=42:RETURN
79  25010 IFLI=2ANDCI=9THENAD=AD
    -8:POKEAD-6,20:POKEAD-1,21:L
    I=43:RETURN
AA  25020 IFLI=9ANDCI=13THEN2600
    0
41  25030 IFLI=9ANDCI=23THENAD=A
    D-8:POKEAD-5,63:POKEAD-1,64:
    LI=45:RETURN
83  25040 IFLI=11ANDCI=44THEN260
    20
50  25050 IFLI=11ANDCI=45THENAD=
    AD-8:POKEAD-5,63:POKEAD-1,64
    :LI=47:RETURN
00  25060 IFLI=12ANDCI=44THEN260
    40
D9  25070 IFLI=12ANDCI=45THENAD=
    AD-8:POKEAD-5,63:POKEAD-1,64
    :LI=49:RETURN
56  25900 LI=CI:RETURN
5F  26000 AD=AD-18:FORV=0TO9:POK
    EAD+V,PEEK(AD+8+V):NEXT
DD  26010 POKEAD+1,251:POKEAD+6,
    252:AD=AD+10:LI=44:RETURN
7B  26020 AD=AD-18:FORV=0TO9:POK
    EAD+V,PEEK(AD+8+V):NEXT
63  26030 POKEAD+1,20:POKEAD+6,2
    1:AD=AD+10:LI=46:RETURN
B7  26040 AD=AD-18:FORV=0TO9:POK
    EAD+V,PEEK(AD+8+V):NEXT
41  26050 POKEAD+1,253:POKEAD+6,
    254:AD=AD+10:LI=48:RETURN
26  30000 PRINT"[ST]OO MANY [SP,
    SE2,SK]S";:GOTO32000
27  30010 PRINT"[SP,SE2,SK] WITH
    OUT BRACKETS";:GOTO32000
58  30020 PRINT"[SB]RACKETS TOO
    COMPLEX";:GOTO32000
D9  30030 PRINT"[SB]RACKET NESTI
    NG ERROR";:GOTO32000
4F  30040 PRINT"[SL,SE,ST] WITH
    NO EQUAL";:GOTO32000
74  30050 PRINT"[SE]XPECTING OPE
    RATOR";:GOTO32000
96  30060 PRINT"[SI]LLEGAL OPERA
    ND";:GOTO32000
2B  30070 PRINT"[SG,SO,ST,SO]/[S
    G,SO,SS,SU,SB] WITH NO NUMBE
    R";:GOTO32000
E1  30080 PRINT"[SL]INE NUMBER D
    OESN'T EXIST";:GOTO32000
2D  30090 PRINT"[SI,SF]-[ST,SH,S
    E,SN] WITH NO EXPRESSION";:G
    OTO32000
48  30100 PRINT"[SI,SF] WITH NO
    [ST,SH,SE,SN]";:GOTO32000
6A  30110 PRINT"[SC]OMPARISON ER
    ROR";:GOTO32000
ED  30120 PRINT"[SI,SF]-[ST,SH,S
    E,SN] GARBAGE";:GOTO32000
79  30130 PRINT"[SF,SO,SR]-[SN,S
    E,SX,ST] GARBAGE";:GOTO32000
7B  30140 PRINT"[SN,SE,SX,ST] WI
    THOUT [SF,SO,SR]";:GOTO32000
85  30150 PRINT"[SA]RRAY GARBAGE
    ";:GOTO32000
3A  30160 PRINT"[SD,SI,SM] GARBA
    GE";:GOTO32000
89  30170 PRINT"[SN]UMBER TOO LA
    RGE";:GOTO32000
4C  30180 PRINT"[SD,SA,ST,SA] GA
    RBAGE";:GOTO32000
EA  30190 PRINT"[SR,SE,SA,SD] GA
    RBAGE";:GOTO32000
7A  30200 PRINT"[SS]YNTAX ERROR"
    ;:GOTO32000
C4  30210 PRINT"[SM]EMORY FULL -
     PROGRAM TOO LONG":END
33  30230 PRINT"[SF,SO,SR]-[SN,S
    E,SX,ST] NESTING ERROR";:GOT
    O32000
E7  32000 PRINT" IN"LN:END
```



# Bus Route 64

*Connect two Commodore 64 computers through their serial ports or link in to the C16 and Plus 4*

by Simon Clarke

There are many ways of connecting one Commodore computer to another but most methods require custom-made cables and complex software. This routine simply uses the serial bus which already has a cheap and easily available connector in the form of the disk drive/printer cable.

The serial bus has two lines which are capable of input and output on both computers: the CLocK and DATa lines. There is also the ATN (attention) line that is used to interrupt external devices, but unfortunately it cannot accept a data signal. There is a line designated SRQ IN (serial request in) on the C64 but this has been deleted on the C16 and Plus/4 microcomputers.

The SRQ IN line allows external devices to interrupt the C64 and can only be used as an input. The CLK and DAT lines are so called because of their use in the Commodore Kernal ROM during communication with serial bus devices. It is the CLK line that governs which bits are valid on the DAT line and in a sense it 'clocks' the bits going out.

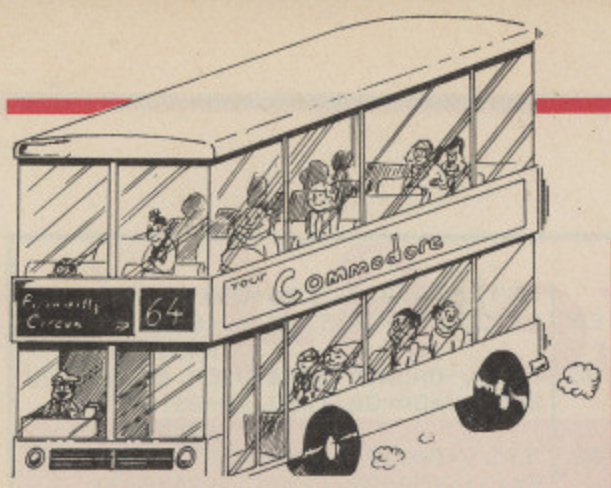
Recently a few dual-player games have appeared on the market where two Commodore computers are connected together and a game is loaded into both. This routine has possibilities in such environments, requiring very little modification to the actual code.

Two copies must be made to get the routine up and running, one per computer, with a slight modification at the beginning of the routine 'INTVAR' in each version. Where it has LDA #XX, the XX must be 00 in one and 01 in the other. This is in order to condition TLKFLG (TaLK FLaG) so that one computer has the initial talk priority. This is toggled after every press of the return key.

When converting the routine to work with other Commodore computers which have a serial bus, the following table of register/bit numbers may be useful

```
10                ;SERIAL BUS COMMUNICATIONS
20                .ORG    $0801
30 COLOUR        =$0286
40                ;VIDEO INTERFACE CHIP II REGISTERS
50 VIC           =$D000
60 EXTCOL        =VIC+$20
70 BGCOL0        =VIC+$21
80                ;COMPLEX INTERFACE ADAPTOR #2 REGISTERS
90 CI2           =$DD00
100 CI2PRA       =CI2+$00
110 C2DDRA       =CI2+$02
120               ;KERNAL ROM ROUTINES
130 CHROUT       =$FFD2
140 GETIN        =$FFE4
150               ;BASIC HEADER
160               .WORD   EOB
170               .WORD   1987        ;LINE NUMBER
180               .BYTE   $9E         ;SYS TOKEN
190               .BYTE   '2061'
200               .BYTE   $00         ;SIGNALS END OF BASIC LINE
210 EOB           .BYTE   $00,$00     ;SIGNALS END OF BASIC TEXT
220 SERCOM        JSR INTVID          ;INITIALISE VIDEO
230               JSR LINREL          ;RELEASE SERIAL BUS LINES
240               JSR INTVAR          ;INITIALISE VARIABLES
250 CHKFLG        LDA TLKFLG
260               BNE TALK            ;BRANCH TO 'TALK OR 'LISTEN
270               JMP LISTEN
280 TALK          JSR GETIN
290               CMP #$00
300               BEQ TALK            ;WAIT FOR CHAR FROM KEYBOARD
310               STA BYTE
320               PHA                 ;SAVE CHAR ON STACK
330               LDX #$0D            ;LIGHT GREEN TEXT
340               STX COLOUR
350               JSR CHROUT          ;PRINT CHARACTER TO SCREEN
360               SEI                 ;THEN SEND ON SERIAL BUS
370               JSR CLKLO           ;SEND ATTENTION
380 WDATLO        JSR GETBIT
390               BCS WDATLO          ;WAIT FOR ACKNOWLEDGE
400               LDX #$08
410 NXTBIT        JSR CLKLO           ;MAKE DATA NOT VALID
420               ASL BYTE            ;BIT TO SEND IN .C
430               BCS HIDAT
440               JSR DATLO           ;SEND ZERO BIT
450               JMP DATVAL
460 HIDAT         JSR DATHI           ;SEND ONE BIT
470 DATVAL        JSR CLKHI           ;MAKE DATA VALID
480 WCLKLO        JSR GETBIT
490               BMI WCLKLO
500 WCLKHI        JSR GETBIT
510               BPL WCLKHI          ;WAIT FOR BIT ACKNOWLEDGE
520               DEX
530               BNE NXTBIT
540               JSR DATHI           ;RELEASE DATA LINE
550               CLI                 ;ALLOW MASKABLE INTERRUPTS
560               PLA
570               CMP #$0D            ;IF 'RETURN' KEY THEN LISTEN
580               BNE CHKFLG
590               DEC TLKFLG
```

| COMPUTER | CLK-IN | CLK-OUT | DAT-IN | DAT-OUT |
|---|---|---|---|---|
| C16, Plus/4 | $0001/6 | $0001/1 | $0001/7 | $0001/0 |
| VIC 20 | $911F/0 | $912C/3 | $911F/1 | $912C/7 |
| C64/128 | $DD00/6 | $DD00/4 | $DD00/7 | $DD00/5 |




```
PROGRAM: BUS ROUTE 64

E3  10 BL=26   :LN=50   :SA=3814
    4
C0  20 FOR L=0 TO BL:CX=0:FOR D=
    0 TO 15:READ A:CX=CX+A
82  25 POKE53280,A:POKE SA+L*16+
    D,A:NEXT D
A5  30 READ A:IF A><CX THENPRINT
    "ERROR IN LINE";LN+(L*10):ST
    OP
1D  40 NEXT L:SYS38400
5D  50 DATA 11,8,195,7,158,50,48
    ,54,49,0,0,0,32,113,8,32,765

ED  60 DATA 173,8,32,107,8,173,2
    53,8,208,3,76,181,8,32,228,2
    55,1753
EE  70 DATA 201,0,240,249,141,25
    2,8,72,162,13,142,134,2,32,2
    10,255,2113
11  80 DATA 120,32,145,8,32,163,
    8,176,251,162,8,32,145,8,14,
    252,1556
5F  90 DATA 8,176,6,32,127,8,76,
    77,8,32,136,8,32,154,8,32,92
    0
B2  100 DATA 163,8,48,251,32,163
    ,8,16,251,202,208,223,32,136
    ,8,88,1837
06  110 DATA 104,201,13,208,176,
    206,253,8,240,171,169,1,141,
    253,8,96,2248
55  120 DATA 169,147,32,210,255,
    169,0,141,32,208,141,33,208,
    96,173,0,2014
EA  130 DATA 221,9,32,141,0,221,
    96,173,0,221,41,223,141,0,22
    1,96,1836
CE  140 DATA 173,0,221,9,16,141,
    0,221,96,173,0,221,41,239,14
    1,0,1692
6A  150 DATA 221,96,173,0,221,20
    5,0,221,208,248,10,96,173,0,
    221,41,2134
3A  160 DATA 207,141,0,221,120,3
    2,163,8,48,251,32,127,8,32,2
    46,8,1644
96  170 DATA 32,136,8,162,8,32,1
    63,8,16,251,46,252,8,32,145,
    8,1307
FC  180 DATA 32,246,8,32,154,8,3
    2,246,8,202,208,233,88,169,1
    0,141,1817
AB  190 DATA 134,2,173,252,8,32,
    210,255,173,252,8,201,13,208
    ,197,238,2356
AB  200 DATA 253,8,76,22,8,160,1
    28,136,208,253,96,0,0,0,0,0,
    1348
DF  210 DATA 169,0,133,250,169,1
    49,133,251,169,1,133,174,133
    ,193,169,8,2234
A5  220 DATA 133,175,133,194,169
    ,0,133,252,169,150,133,253,1
    60,0,177,250,2481
59  230 DATA 145,174,230,250,208
    ,2,230,251,230,174,208,2,230
    ,175,165,250,2924
40  240 DATA 197,252,208,234,165
    ,251,197,253,208,228,169,155
    ,133,187,169,150,3156
7F  250 DATA 133,188,169,12,133,
    183,169,0,133,185,160,0,185,
    107,150,240,2147
81  260 DATA 6,32,210,255,200,20
    8,245,32,207,255,240,251,201
    ,49,240,4,2635
2E  270 DATA 201,56,48,230,41,15
    ,133,186,76,234,245,147,17,1
    7,73,78,1797
B9  280 DATA 80,85,84,32,68,69,8
    6,73,67,69,32,78,85,77,66,69
    ,1120
43  290 DATA 82,13,17,67,65,83,6
    1,49,32,47,32,68,73,83,75,61
    ,908
AA  300 DATA 32,56,32,79,82,32,5
    7,58,45,32,0,66,85,83,32,82,
    853
92  310 DATA 79,85,84,69,32,54,5
    2,0,0,0,0,255,255,255,255,0,
    1475
```

```
600             BEQ CHKFLG
610 INTVAR      LDA #$01
620             STA TLKFLG      ;0=LISTENER / 1=TALKER
630             RTS
640             ;
650 INTVID      LDA #$93
660             JSR CHROUT      ;CLEAR SCREEN
670             LDA #$00
680             STA EXTCOL
690             STA BGCOL0      ;BLACK SCREEN AND BORDER
700             RTS
710             ;
720 DATLO       LDA CI2PRA
730             ORA #$20
740             STA CI2PRA      ;MAKE SERIAL LINE DATA LOW
750             RTS
760             ;
770 DATHI       LDA CI2PRA
780             AND #$DF
790             STA CI2PRA      ;MAKE SERIAL DATA LINE HI
800             RTS
810             ;
820 CLKLO       LDA CI2PRA
830             ORA #$10
840             STA CI2PRA      ;MAKE SERIAL CLOCK LINE LOW
850             RTS
860             ;
870 CLKHI       LDA CI2PRA
880             AND #$EF
890             STA CI2PRA      ;MAKE SERIAL CLOCK LINE HIGH
900             RTS
910             ;
920 GETBIT      LDA CI2PRA
930             CMP CI2PRA      ;AWAIT STABLE CLOCK & DATA LINES
940             BNE GETBIT
950             ASL A           ;GET DATA IN .C & CLOCK IN .N
960             RTS
970             ;
980 LINREL      LDA CI2PRA
990             AND #$CF
1000            STA CI2PRA      ;RELEASE DATA & CLOCK (BOTH HI)
1010 LISTEN     SEI
1020 WLOCLK     JSR GETBIT
1030            BMI WLOCLK      ;WAIT FOR ATTENTION
1040            JSR DATLO       ;SEND ACKNOWLEDGE
1050            JSR DELAY
1060            JSR DATHI
1070            LDX #$08
1080 WHICLK     JSR GETBIT
1090            BPL WHICLK      ;WAIT FOR DATA VALID
1100            ROL BYTE        ;GET BIT
1110            JSR CLKLO       ;THEN ACKNOWLEDGE IT
1120            JSR DELAY
1130            JSR CLKHI       ;RELEASE CLOCK LINE
1140            JSR DELAY
1150            DEX
1160            BNE WHICLK      ;DO EIGHT BITS
1170            CLI
1180            LDA #$0A        ;LIGHT RED TEXT
1190            STA COLOUR
1200            LDA BYTE
1210            JSR CHROUT      ;PRINT CHAR TO SCREEN
1220            LDA BYTE
1230            CMP #$0D
1240            BNE LISTEN      ;IF RETURN KEY THEN TALK
1250            INC TLKFLG
1260            JMP CHKFLG
1270            ;
1280 DELAY      LDY #$80
1290 WAIT       DEY
1300            BNE WAIT
1310            RTS
1320            ;
1330 BYTE       .BYTE   $00
1340 TLKFLG     .BYTE   $00
1350 ENDCOM     .END
```

# DON'T GET LEFT OUT... GET IN ON THE ACTION

**COMMODORE DISK USER** is a lot more than just another computer magazine. Every issue carries a diskette containing more than £30 worth of software ranging from serious programming utilities to arcade games. There are plenty of Commodore magazines on the market, but we believe that this is the *first* to cater for disk users of all ages and tastes.


*COMMODORE DISK USER is what you have been waiting for – take out a subscription TODAY!*

Please begin my subscription(s) to COMMODORE DISK USER with the ..... issue
I enclose my cheque/money order for £..... made payable to Argus Specialist Publications Ltd.
or debit £..... from my Access/Barclaycard No. .....
valid from ..... to .....
NAME (Mr/Mrs/Miss) .....
ADDRESS .....
.....
Postcode .....
Signature .....
Date .....

Please use BLOCK CAPITALS and use post codes.

**SUBSCRIPTION RATES**

**£15.00 for 6 issues U.K.**
**£18.00 for 6 issues EUROPE**
**£18.20 for 6 issues MIDDLE EAST**
**£19.30 for 6 issues FAR EAST**
**£18.40 for 6 issues REST OF WORLD**
**Airmail Subscription Rates on Request**

**Send your remittance to:**
***INFONET LTD., 5 River Park Estate, Berkhamsted, Herts. HP4 1HL.***


commodore 128D


commodore 64


commodore 64