JULY 1991 £3.25
DM. 16

Commodore ·DISK·USER·

SCHIZO.....YOU WILL BE AFTER PLAYING THIS GAME

THE DISK

Madix

European Logo Editor

Memory-Transfer

Letter Writer V2

ESP Synth-Version 1

07
9 770953 061014

PROGRAMMERS REQUIRED

-To work on our range of leading edge video frame grabbers.

-You *must* be proficient in asembler and 'C' programming.

-You will need an in depth knoledge of the Amiga hardware and operating system.

-Enthusiasm and a willingness to grasp new concepts are essential.

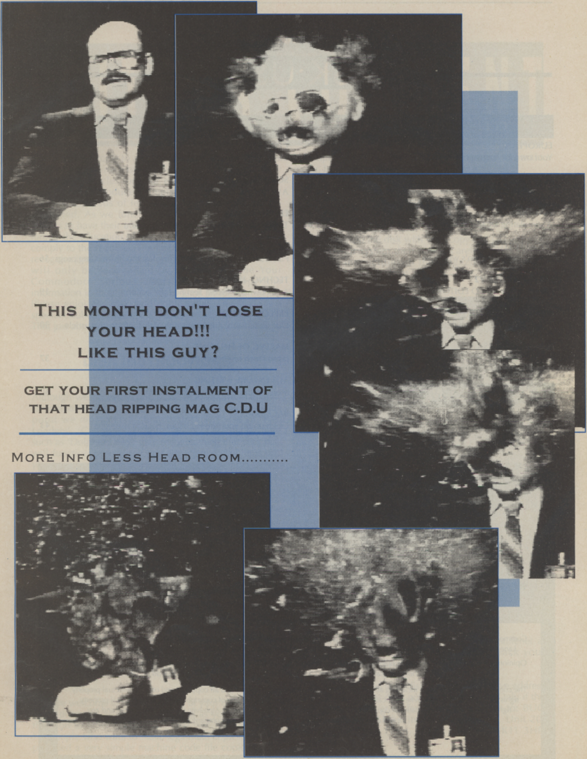
-Salary Cir. £12,000 pa. (full time)

-Please send C.V and sample of your work (if possible) to:

Marcus Sharp
Rombo
6 Fairbairn Road
Livingston
Scotland
Tel: (0506) 414631
Fax: (0506) 414634

ROMBO
Limited

ROMBO - NO.1 IN U.K & EUROPE
LEADING THE WAY FORWARD



THIS MONTH DON'T LOSE YOUR HEAD!!! LIKE THIS GUY?

GET YOUR FIRST INSTALMENT OF THAT HEAD RIPPING MAG C.D.U

MORE INFO LESS HEAD ROOM...........

Commodore DISK·USER·

ON THE DISK

IN THE MAGAZINE

Volume 4 Number 9 JULY 1991

Group Editor: Paul Eves
Designer: Mark Newton
Technical Editor: Jason Finch
Program Evaluator: John Simpson
Publishing Consultant: Paul Crowder
Advertisement Manager: Cass Gilroy
Classified Sales: Deborah Curran
Publisher: Hasnain Walji
Distribution: Seymour Press Distribution
Ltd. Winsor House, 1270 London Road, Norbury, London
SW16 4DH. Tel:081 679 1899. Fax: 081 679 8907
Printed By: Gibbons Barford Print

Subscription Rates

UK	£33.00
Europe	£39.00
Middle East	£39.30
Far East	£41.60
Rest of World	£39.70 or $69.00

Airmail rates on request
Contact: Select Subscriptions. Tel: (0442) 876661

Commodore Disk User is a monthly magazine published on the 3rd Friday of every month. Alphavite Publications Limited, 20, Potters Lane, Kiln Farm, Milton Keynes, MK11 3HF. Telephone: (0908) 569819 FAX: (0908) 260229. For advertising ring (0908) 569819

Opinions expressed in reviews are the opinions of the reviewers and not necessarily those of the magazine. While every effort is made to thoroughly check programs published we cannot be held responsible for any errors that do occur.

EDITORS COMMENT

Hello, and welcome to another issue of CDU.
In the Magazine you will find a couple of very informative articles for your enjoyment. These articles have been re-produced simply because we have had literally hundreds of letters asking for them to be re-published. As we function to be both a platform for readers to have their offerings seen by a, and also to help further the education of using your C64, we have had to comply to the requests. The first is one many of you will recognise immeadiatly "Exploring the 1541." The second will only be recognised by readers of "The Your Commodore Serious Users Guide." I hope the information in these articles are of great benefit to you all. Please enjoy the disk, and don't forget, This issue is a special double-sided disk.
That just about sums it all up. Hope you enjoy the issue.

DISK INSTRUCTIONS

Although we do everything possible to ensure that CDU is compatible with all C64 and C128 computers, one point we must make clear is this. The use of 'Fast Loaders', 'Cartridges' or alternative operating systems such as 'Dolphin DOS', may not guarantee that your disk will function properly. If you experience problems and you have one of the above, then we suggest you disable them and use the computer under normal, standard conditions. Getting the programs up and running should not present you with any difficulties, simply put your disk in the drive and enter the command.

LOAD"MENU",8,1

Once the disk menu has loaded you will be able to start any of the programs simply by selecting the desired one from the list. It is possible for some programs to alter the computers memory so that you will not be able to LOAD programs from the menu correctly until you reset the machine. We therefore suggest that you turn your computer off and then on again, before loading each program.

HOW TO COPY CDU FILES

You are welcome to make as many of your own copies of CDU programs as you want, as long as you do not pass them on to other people, or worse, sell them for profit. For people who want to make legitimate copies, we have provided a very simple machine code file copier. To use it, simply select the item FILE COPIER from the main menu. Instructions are presented on screen.

DISK FAILURE

If for any reason the disk with your copy of CDU will not work on your system then please carefully re-read the operating instructions in the magazine. If you still experience problems then:

1. If you are a subscriber, return it to:
   Select Subscriptions Ltd
   5, River Park Estate
   Berkhamsted
   Herts
   HP4 1HL Telephone; 0442 876661

2. If you bought it from a newsagents,
   then return it to:
   CDU Replacements
   STANLEY PRECISION DATA SYSTEMS LTD
   Unit F
   Cavendish Courtyard
   Sallow Road
   Weldon North Industrial Estate
   Corby
   Northants
   NN17 1JX Telephone; 0536 61787

Within eight weeks of publication date disks are replaced free.

After eight weeks a replacement disk can be supplied from STANLEY PRECISION DATA SYSTEMS LTD for a service charge of £1.00. Return the faulty disk with a cheque or postal order made out to STANLEY PRECISION DATA SYSTEMS LTD and clearly state the issue of CDU that you require. No documentation will be supplied.

Please use appropriate packaging, cardboard stiffener at least, when returning disk. Do not send back your magazine, only the disk please.

NOTE: Do not send your disks back to the above address if its a program that does not appear to work. Only if the DISK is faulty. Program faults should be sent to: BUG FINDERS, CDU, Alphavite Publications Ltd, Unit 20, Potters Lane, Kiln Farm, Milton Keynes, MK11 3HF. Thank you.

EUROPEAN

A C64 language tutorial for all those wishing to learn another tongue - MARK SKINGLE

In DECEMBER 1990, CDU gave us a language tutorial program for all the C128 users amongst us, namely, I.L.S. The German Program. EUROPEAN is my contribution to all the C64 users out there in micro land.

1992 AND ALL THAT

With 1992 quickly approaching, emphasis is being placed on learning a second or third language. Learning a language is much easier if at first you learn how to read or write it, once you have learned the phrases you can then proceed to learn the correct pronunciation without the difficulty in remembering the words you wish to say! European offers invaluable help with the first step, and much more. I have written this article in such a way so as to 'talk' you through the programs many facilities, so load in EUROPEAN by selecting it from the CDU Menu or type LOAD"EUROPEAN",8,1. When the title screen appears, press the SPACEBAR to continue the loading process. When the program has finished loading press RETURN.

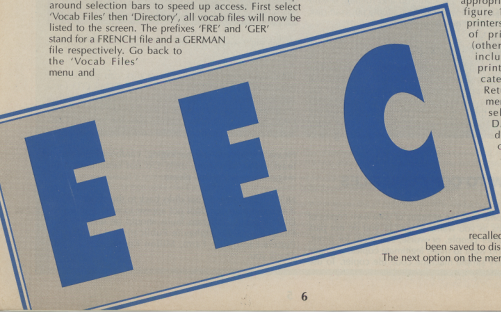
THE PROGRAM

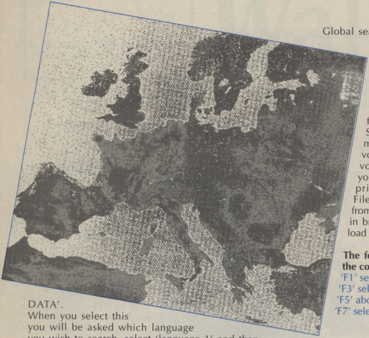
You will now have the main selection menu on screen. To move the selection bar use 'F1' to move up, 'F3' to move down and 'F7' to select. These menus use wrap-around selection bars to speed up access. First select 'Vocab Files' then 'Directory', all vocab files will now be listed to the screen. The prefixes 'FRE' and 'GER' stand for a FRENCH file and a GERMAN file respectively. Go back to the 'Vocab Files' menu and select 'LOAD FILE' it will ask for the language prefix, (as you have not selected which language you will be working with), type in 'GER' in capitals and press return, the program will now consider that you will be using GERMAN files until you change this. Select 'LOAD FILE' and type 'INTRO'. The GERMAN vocabulary in this file will now load in.

Go back to the main menu and select 'Vocabulary' followed by 'Amend Data'. In this case a horizontal selector bar is used. 'F1' will move left, 'F3' right, 'F5' abort (back to menu) and 'F7' select. Over the 'NEXT' option, shift+'F7' can be used to step through the vocabulary data backwards. You can use the delete function to erase the current vocabulary shown. To amend the data select the 'REPLACE' option. To avoid changing the data in one of the two windows just press return when the cursor is in the top left of the appropriate window. Although the new text you type overwrites the text in the window it doesn't keep the old data in memory therefore it will only keep in memory what you type. Using the 'NEXT' function you can examine the contents of a file.

Go back to the VOCABULARY menu (press F5), select 'ADD DATA', this will add vocabulary data onto the end of the vocab in memory. To abort this option you can just press return. You can use the special foreign characters by pressing the appropriate keys (See figure 1), the LC10 printers are capable of printing these (others are not included as the printer does not cater for them). Return to the menu again and select 'DELETE DATA', this is different to the option in the amend data facility as it concerns all the data in memory. This data cannot be recalled unless it has been saved to disk.

The next option on the menu is 'SEARCH





DATA'.
When you select this you will be asked which language you wish to search, select 'language 1' and then type in the search data, ie 'I', it will now, using full wildcard searching, display any data which includes the 'I'. When the program has found a match, press any key to continue the search.

The last option on this menu is 'SORT DATA', select it and then 'Language 1', it is now sorting the data into alphanumeric order (Lower case has priority over Uppercase). You can check this by returning to the amend facility to examine the data.

Go back to the main menu, select 'VOCAB FILES' and then 'UPDATE FILE' this will update the current file on disk. The save option is to save a new file, the same file under a different name or to backup a file onto another disk. Any disk error which occurs during any disk operation will be reported at the top of the screen, use the information along with your disk manual to locate the problem. We now move on to the most important part of the program, the VOCABULARY TEST. You can select this from the main menu. You now need to select either a RANDOM TEST (20 random questions) or a SEQUENTIAL TEST (All questions in order). Now choose the language you wish to have a question in, you will be expected to write the equivalent in the other language. The current score will be noted by 'NUMBER' The final score will be given at the end of the test.

Select the 'DICTIONARY', accessible by the main menu. Now select LOCAL, type in 'HELLO', you will now be given the corresponding word in German (Guten Tag). The local search only checks through the memory. Try Global search and type in 'HELLO' again, this checks all the vocabulary files on disk, the matches will now include 'GUTEN TAG' and 'BONJOUR', the language is indicated in each case. Once again when the border turns red press a key to continue. Selective search enables you to choose which files are to be checked.

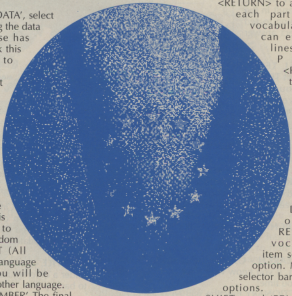
Select **PHRASE BOOK** from the main menu, this is used to print out vocabulary. Print all will printout all the vocabulary whereas Print some allows you to select which vocabulary items to printout (use same keys as in Amend File).The **HELP** files included, accessible from EUROPEAN, include this information in briefer terms. To printout the help files, load in "EUROPEAN PRINTER",8,1

The following is a quick reference guide to the commands in EUROPEAN.

'F1' selector bar up/left
'F3' selector bar down/right
'F5' abort selection
'F7' select option

VOCABULARY

ADD DATA - Use this option to add more vocabulary to the current file. Just press <RETURN> to abort. For each part of the vocabulary you can enter two lines of text. Press <RETURN> to get onto the next line.



AMEND DATA - To DELETE or REPLACE a vocabulary item select this option. Move the selector bar to select options. Pressing <SHIFT> and 'F7' over the NEXT option will do the reverse stepping backwards through the data.

DELETE DATA - If you confirm this option all data IN MEMORY will be deleted that means the current file you are working with unless it has been saved. The prefix will be deleted as well.

SEARCH - First select which language you wish to search. Then input the 'search text' all occurrences of this will be listed. The routine uses FULL wildcat searching.

SORT DATA - Use this to sort the data into alphanumerical order. Select the language to sort by then leave the program to do the rest. NOTE. lowercase has priority over uppercase characters.

VOCAB FILES

See 'VOCAB FILES' menu to select independent helpfile.

VOCAB TEST

RANDOM TEST - Select the language you wish the 'questions' to be in. You will now be asked twenty random questions from the file in memory. The current score is kept alongside 'Number'. A wrong answer will result in the border changing to red and the correct answer given.

SEQUENTIAL TEST - (SEE RANDOM TEST) In this case though you will be given each question in memory in sequential order to answer.

DICTIONARY

LOCAL SEARCH - Use this to enter a word in one language and receive the corresponding word in the other. Local search only searches the data in memory.

GLOBAL SEARCH - Searches every file in every language on disk.

SELECTIVE SEARCH - Use this function to choose the files to search. If you know which file the word appears in will save you time!
NB. The DICTIONARY function will NOT affect data in memory.

PHRASE BOOK

This facility enables you to print out vocabulary listings for easy reference.
It is designed to work in conjunction with the STAR LC 10 printer. However it should work correctly with other printers as well.

PRINT ALL - This will print out all the vocabulary in memory, 18 vocabulary items to a page.

PRINT SOME - This will cycle through the vocabulary with you choosing which items to print. Use F1 F3 and F7 to select. Press F5 to abort.

LANGUAGE

SELECT - Use this function to declare the languages you will be working with.
LANGUAGE1 will generally be English.
LANGUAGE2 will be the language you will be learning.

FILE PREFIX - Use this to identify the disk files by language. The prefix is made up of three characters and is integrated into the file name. You could use the following to identify the files

'GER' for German files.
'FRE' for French files.
'SPA' for Spanish files etc.

DIRECTORY - Use this function to list the vocabulary files which are on the current disk.

LOAD FILE - Use this option to load in vocabulary data from disk. If you have not selected a a file prefix you will be asked to do this first.

UPDATE FILE - Only use this function when during the current session of EUROPEAN use you have either loaded or saved data. It is used to re-save a file after it has been updated.

SAVE FILE - Use this file to save new data or re-save a file under a different name. You could also use this function to backup files.

DISK ERRORS - During disk operation any error which arises will be reported at the top of the screen. Use this information along with your disk manual for further information.

The Weird and Wonderful World of Wally!

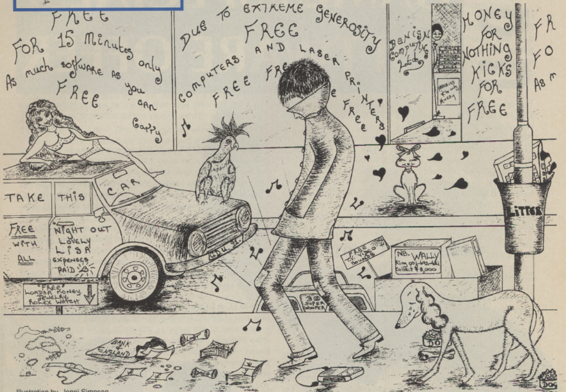

FREE
FOR 15 Minutes only
As much software as you can Carry
FREE
DUE TO EXTREME GENEROSITY
FREE
COMPUTERS AND LASER PRINTERS
FREE FREE
FREE
BENIGN COMPUTING Ltd
MONEY FOR NOTHING KICKS FOR FREE
FR
FO
As m
TAKE THIS CAR
FREE WITH ALL
NIGHT OUT LOVELY LISA EXPENSES PAID
"FREE" LOADSA MONEY JEWELRY ROLEX WATCH
CDU 91
FREE T.V. VIDEO SOUNDS
NB. WALLY Ring 031-642-481 Collect £8,000
3D SUPER WOOFER
LITTER
BANK of ENGLAND
MONEY
DOG
DOG

Illustration by Jenni Simpson

Meet WALLY, the thinking mans answer to Andy Capp. From time to time we will be seeing Wally cropping up in all manner of circumstances. Today we see him deep in thought, musing over all his problems. Wally has decided that 10 readers can have the opportunity of catching up on all those issues of CDU that they have missed. To be in line for this really fantastic prize, you simply have to match the captions below with who you think said what!!. For instance, if you think that the Cat has said caption 1, you simply write on your postcard 1 Cat, and so on!!.

1) "Good thing Wally's no TWITCHER or he'd realise that I'm a rare psychedelic crested warble wobbler and that I've just escaped from the Zoo. Cor! Wot a neat reward for my capture".
2) "Wow, that's my kind of boy! I certainly wouldn't mind sharing a big, juicy marrow bone with that handsome fellow!"
3) "Tut bloomin' heck! If only I had some money and a decent 'puter, 'n' printer, 'n' some half decent utilities, then I'd be able to do all sorts of things."
4) "Sob! that Wally's forgotten to feed me again today! What I wouldn't give to settle down with a cute little bitch and raise a puppy or six."
5) "Wot a Wally!!"
6) "I love that mean, moody, sexy look. I wonder if he'll take me out for a healthy stroll in the countryside?

"Postcard entries only please, to reach the CDU editorial office by 31st August 1991. The winners will be the first 10 with the correct answers that we pull from the hat. Once the draw has taken place, we will contact the winners to find out which issues of CDU you want. Send your entries, on a postcard don't forget, to;

Wonderful world of Wally
CDU Alphavite Publications
20, Potters Lane
Milton Keynes MK11 3HF

The Editors decision is final and no correspondence will be entered into.

GAMES SPECIALS! SOFTWARE OFFER

Fed up of paying huge amounts of dosh for your games??? Let CDU remedy this by offering you these superb games compilations at knock down prices

All of the disks on offer are original, never before seen games. There is something for everybody. (Shoot 'Em Ups, Strategy, Adventure, Mind Benders and straight-forward Platform). No matter what your preference, something, somewhere will take your fancy. To order your choice, simply fill in the coupon below and send it with your Cheque/Postal Order (made out to ALPHAVITE PUBLICATIONS LTD) to:- Alphavite Publications Ltd, 20 Potters Lane, Kiln Farm, Milton Keynes, MK11 3HF. (Please allow 28 days for delivery).

GAMES DISK 1 (1991)

CONFUSION - So you think you are quick witted? Think you are of high IQ? Crosswords don't hold enough interest for you because they are lame ducks for your mind? If you answered yes, or even no, to those questions then Confusion is for you. A two dimensional version of the popular cubic puzzle. Sort out the multicoloured columns - simple? Ha, try it.

TENOGEN - Blast almost everything in sight. By destroying whole waveforms you will increase the amount of extra weaponry to collect later in the level. Eight scrolling levels to destroy takes you to the end of this exciting shoot-em-up, but can you reach the end?

PROJECT X - You play the part of Hank, in this graphic adventure. Hank sole purpose in life is to retrieve the secret documents of project x. There are four very tricky stages to get through in your quest. First you must fly your plane, then land it after which you must run along the beach and climb a cliff, through the jungle, until you find the cave where enemy agents hold the document. Avoiding enemy aircraft, falling boulders, spears, and arrows - phew, can you find the hidden message...?

MEGADOGFIGHT - An aeriel combat game for two players. Guide your plane around the screen and try to shoot down your best friend as he pilots his aircraft around the screen trying to shoot down you... Great game for two people out for a Sunday flyabout.

GAMES DISK 2 (1991)

FAST FUTURE - This is an arcade type game where you take control of your craft and guide it around a circuit a set number of times - oh, if life was as easy as that. Indeed not, there are other craft in the 'race' who plan to give you more than a really hard time. However, being a bit of a b...... yerself, you blast 'em with your twin lasers, as well as bumping them outa existence. Banks, gravity tracks, collecting energy shields, 32 levels, and ....

COLD COMFORT - In this adventure you awake to find yourself alone on an alien space ship, and locked inside a holding cell. Your task, should you accept it, is to escape the cell, learn the alien language, and discover how to pilot the 'ship' back to earth. This text and graphic adventure will keep you pleasantly engrossed for hours. By the way, it is a big ship.

CELLRATOR 11 - The sequel... as you can guess this has the same theme as cellrator but try and beat this one. Scrolling screens of caverns and caves and never ending obstacles as you fly your craft along; heavy foot on the accelerator, getting you into all sorts of collision trouble, making you wonder if it is all worth it. Quite frantically yes it is! Make map??? Ho! Ho! Ho!

ERADICATOR - A very colourful, with beautifully designed graphics, screen scrolling arcade type game. Survival is the name of the game as you try to avoid all contact with other lifeforms - and just what good are your lasers, I'd like to know? Anyway, can you save the earth, yet again! By the way, slimy green aliens are running the world governments and only you know this, but who would believe you anyway - that's why you grabbed your battlecruiser in the first place!

GAMES DISK 3 (1991)

SOLSTICE - This is a three part graphic adventure set deep within the fourth and largest moon of some distant planet. This game will tax your brain with its complexity as you try to reach completion in the third and final part. You will have to kick, punch, dive, roll, and run your way through each screen, all the while keeping your eyes open for clues. Remember, the diamond must be destroyed!

NEW YORK CRISIS - New York has a problem... The computer of NY surface defence missile silo #5 has declared war on the city. As you are Controller, on of the elite trouble shooters in the city, you must assemble a team of three to enter the silo and disable it. No easy task. If you like games of strategy where fast thinking is of utmost importance then this will leave you with weeks, maybe months, of enjoyment.

GAMES DISK 4 (1991)

LIFE - There have been many 'Life' programs created for the computer since John Conway toyed with the idea of a mathematical model of the behavior of living cells in the 1950s.

Here is another version, but this time for the C64, and within which you have the ability to bring to 'life' dead cells. An interesting variation of the theme of life.

WHITEWASH - This is a logic game where the objective is to reduce the counters to white by successive hits before your opponent does the same. The game is based around the C64's ability to show colour on the screen, and the idea is basically to strip off various layers of colour until white is found.

FRUSTRATION - Is a variant of the old hand-held moving tile game. The aim of the game is to arrange all of the tiles in such a way so that they form the picture shown on the right hand side of the screen.

EUCHRE C128 - This C128 game, which works in 80 column mode, is based on the old card game of the same name. You play with a computer partner against two computer opponents.

HYPERSOLVE - Erno Rubik's cube finds its four dimensional equivalent on the C64. Yes, you must solve the problem of the hypercube which is a four dimensional object that consists of 16 corners, 32 edges and 24 faces, making up 8 cubes, each of which is adjacent to 6 of the others - phew! Can you solve this one?

BINGO 128 - Yes, Bingo for the Commodore 128. This rather interesting version of bingo will allow you to print your own bingo cards, and then will produce the bingo numbers either manually, or automatically - what this means is that Manually the time interval between the calling of numbers is controlled by the caller and in Automatic mode you are able to preset the time between each call. This is a must for those family and friends get-togethers.

GAMES DISK 5 (1991)

ORB - Ever heard of living space coral? No? Well let me tell you this is pretty deadly stuff and not for the fainthearted to deal with. However, you are not fainthearted are you, so off to battle with the deadly ORBSTAR, but watch out for the nasty aliens who materialise in the most unpredictable of places - still, with your powerball at the ready, you're sure to be a winner - eventually?LANCE - The island of Brittania has been plunged into the dark ages. The evil witch Morgana has stolen the holy grail. Many brave knights have tried to recover it, now it is your turn.PROBE WARRIOR - Life in deep space is never running smooth. Just when you think all is peaceable and nice, you have to set forth and defend your planet against the dreaded Clax. You must stop him from destroying the lifepod system otherwise all life on the planet will be exterminated.

LIBERATOR - An exciting all action game with ultra-smooth screen scrolling, and where you, as the liberator, and after being sent to Venus, must liberate the people by clearing the lands of all the invading aliens. You can contact the resistance forces, collect credits to gain weapons such as 'smart bombs', and regain your depleting energy from the rejuvinator tree.

GAMES DISK 6 (1991)

OUTBREAK - This is breakout but with a major difference - the screen scrolls. You must break through the massive play area until you reach the ALLMIGHTY wall at the end... On your journey you will meet with aliens, which can be destroyed, life giving blocks, as well as boring, tough, exploding, happy, angry, and deflecting blocks. You will like this one.

THE MYSTERY MAN - Here is a rather snatzee adventure game where you play the down-at-heel private dick with landlord problems and no booze and no customers. Suddenly, into your life comes a man who offers you five-hundred smakeroos just to deliver a cassette recorder to some guy in a downtown hotel. Grabbing the recorder and your gun you head off into the adventure of your life!

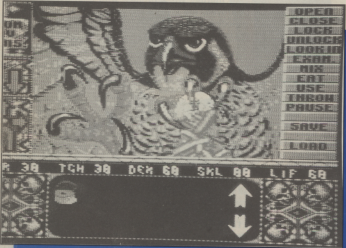
MIRROR IMAGE - Message commences. Dateline 2237. Draconian Empire ships heading through hyperspace towards solar system. Danger scale 100% Multiphase reality track now in operation. Mirror image ERU awaiting pilot. Mission, destroy all Draconian Ships which materialises. Message ends. And of course, you know who the pilot is, don't you?

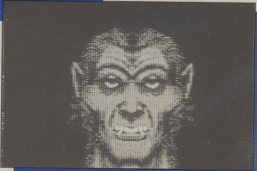
LIBERTE - Here you are, sitting in your hut in the POW camp. You've been there for far too long. A hundred times you have gone over your plan, surely nothing can go wrong. The time as come for you to put your plans into action and escape. It wont be easy though, for a start there are the patrols to avoid, then there is the small matter of the Gestapo HQ to blow up not to mention the rendezvous with the ships Captain. Believe me, I don't envy you in your task.

Please send me........Copies Disk No. 1 @ £5.95 each ....Copies Disk No. 2 @ £5.95 each
........Copies Disk No. 3 @ £5.95 each ....Copies Disk No. 4 @ £5.95 each
........Copies Disk No. 5 @ £5.95 each ....Copies Disk No. 6 @ £5.95 each
I enclose a cheque/postal order for....................................
(Made payable to **ALPHAVITE PUBLICATIONS LTD.**)
NAME..
ADDRESS..
....
....Postcode.....................
Send to; **ALPHAVITE PUBLICATIONS Ltd;** CDU Software Offer, 20 Potters Lane, Kiln Farm,
Milton Keynes, MK11 3HF.

Elvir

MISTRESS OF THE DARK



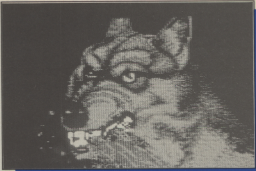


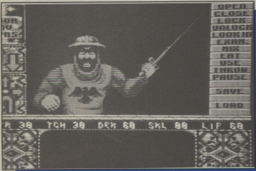
If you have to have a mistress - who better than ELVIRA - JF

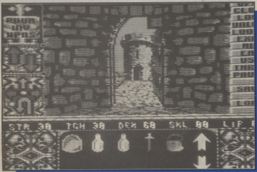
If I had been told a year ago that a team were engaged in the reproduction of that great Amiga game, "ELVIRA - MISTRESS OF THE DARK", for use on the comparatively humble 8-bit Commodore 64, I would have told them that they were out of their minds and I would have been left wondering how anybody could do such a thing. Last week a package arrived - the C64 conversion of "Elvira" - and now I am left wondering how somebody DID do such a thing. For those of you that are unfamiliar with the plot, I shall attempt to explain briefly the background to this excellent fantasy game and how the controls, as it were, operate, followed by my opinions, as the reviewer, on this stunning recreation. To coincide with this review, FLAIR SOFTWARE LTD and ALPHAVITE PUBLICATIONS have joined forces to bring you an exclusive PLAYABLE DEMO which you will find on the reverse side of this months disk.

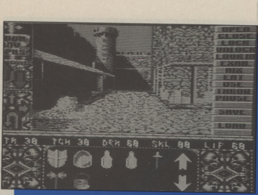
INTO THE CASTLE. The game takes place at Killbragant Castle, surrounded by beautiful English countryside, where you are to help out a rather well-endowed young lady with the task of eliminating evil spirits from the castle. She has inherited the fortress and its grounds and has plans to turn it into some sort of tourist attraction. Her great-great grandmother was Lady Emelda, who was married to Sir Elric, a rather dull gentleman. So when he wasn't around, Emelda had an affair with a Lord Beremond. Unfortunately this was rather short-lived as Beremond was killed accidentally on a hunting trip. When Elric returned, he was none too pleased to find that, due to this affair, everything else had gone to pot, but his life was soon over when Emelda found the old family sword! Sad isn't it, but Emelda also died a few years later. The directions for starting (and stopping) her subsequent resurrection are reputedly hidden somewhere in Killbragant Castle, in an old chest. The only problem is that this is some chest, and it takes six keys to unlock. These were given to Emelda's pals so that they could hang on to them and come back with her for the second attempt at living. This gang of dead geeks still haunt the place and beasties adorn the castle by the coach load. In trying to redecorate the castle, your lady friend has upset the memories and awoken the dead. The six keys to the chest, and the chest itself, have to be found so that Emelda's imminent return can be prevented. That basically then is your task! When you purchase your copy of this game, and purchase it you will, you'll be presented with an instruction booklet, a book of magic spells (a pretty blue on blue combination preventing those horrible pirates from photocopying the means of protection!), and no less than three double sided disks on which can be found the staggering 700K worth of code and graphics. The spell book will help you to decide which of the spells you need to concoct in order to defeat certain ghosties and overcome certain problems. For all of these you must collect ingredients and then present them in the kitchen for mixing. Flopping disk one in the drive and loading it up results in you being confronted by the intro. A stirring, sombre piece of music plays and grabs your attention immediately whilst you are given a taste of the superb graphical animation sequences that are to come. From within the game, pictures of which you will find dotted around this review, everything is controlled by a joystick.

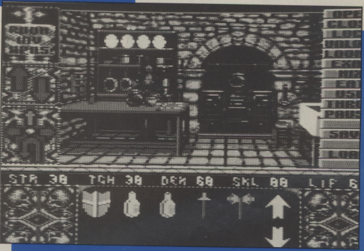
ON WITH THE SHOWOn the left of the main screen are three options: ROOM, INV and WEAPONS. By pointing the arrow and clicking on these, you will bring up a display of either what objects are in the room, in your possession, or in your armament. These appear in the box under the main window which also serves as a dialogue box. Again to the left are direction arrows. No matter where you are, the directions that you can take are highlighted in green on the left. If you are near a staircase, the up and down icons may light up and you simply point and click, and you are away. By going up and down some staircases you will be greeted by an animation sequence, showing the view that you would have were you really to be climbing a spiral staircase - the rotational effect simply has to be seen to be appreciated. On the right of the display you have a multitude of other options such as UNLOCK, EXAMINE, LOOK IN, USE and so forth. These are all self-explanatory and when one or more are highlighted in green, you can click on them to use a certain object, or to examine it and so on. Between all this and the dialogue box is the status bar, telling you how much life you've got left in you, and also, for example, how resilient you are. The main window is where the scenes are depicted. Every single location throughout the adventure has its own highly detailed graphical representation. These were created by four artists who have left nothing out. It is hard for me to describe in words just how excellent these graphics have turned out,











considering that they are produced on a computer that allows only sixteen different colours within so many different constraints - compare this to the Amiga's 4096 colours and you will be amazed at how similar the two versions are. Should you want to open a door you simply point to it in the main window and press fire. To pick objects up you just point at them, press fire, and move the 'hand' to over the INV command. It really is as simple as that. Everything is described in the comprehensive "manual" which gives you all the information that you need.

On your travels you will meet plenty of "things" that have staked out their territory and are prepared to fight for what is theirs. The combat scenes are very well animated, producing as usual your eye-view of the situation. The more strength and resilience you have, the easier it will be to fend off the attackers. But like them, you can only sustain a certain level of injury - then it's cheerio.

I've mentioned that the game is on three disks, and you do have to swap them during play. You are prompted as to which to insert next and when you have become engrossed in the gameplay, these disk changes seem to merge in with the action very well. There is, after all, no way that these could be eliminated - the group could have compromised on the graphics, but then what is the point of ruining an otherwise superb game for the sake of a couple of seconds here and there. Disk access has been speeded up considerably by a special disk turbo written specially for "Elvira" and all the different zones have been concentrated on specific disks so that you can, for instance, traipse about the battlements for hours without having to do one single disk swap. Sound effects are produced as and when required - there is no wish to turn the volume down to kill some awful background soundtrack as there is in some other games.

WHO NEEDS AN AMIGA? The artists and the sole programmer, Bruce Le Feaux, alike have put in over eighteen months of work and the result is almost a carbon copy of the Amiga version. None of the magic has been lost and the playability is still there in all its glory. The animation frames are as equally well drawn as the locations and the programmer has made them sufficiently fast and totally flicker free. So far I have been hacked to death by a mad cook and.. erm.. devoured by a werewolf. Both portrayals went to just the right level so don't worry about the realism getting to be too much if you are killed! The elimination of keyboard commands makes the game run smoothly and the save game option means that you can start again where you left off if the tension becomes too much for you. If you prefer just a short coffee break then the pause mode will suffice.

If you think that you could never become addicted to a role-playing game then think again, because this will prove the exception to any rule. The first session I had at this game lasted throughout an afternoon and an evening - both the ease-of-use and speed at which you pick up how to do things are a real boon and you could find yourself engulfed in trying to solve the puzzles within this great game for hours. Reviewers usually have the odd quibble about a game or utility - perhaps that little feature that could have been implemented but wasn't. I really don't have anything to say against this game - even small things like separating out the SAVE and LOAD options so that you don't accidentally click on the wrong one have been seen to. My congratulations go to all the people involved in creating this masterpiece which really does have to be seen in action to be believed.

The game retails for £24.99, the distributors being Flair Software Ltd., The Smithy Side, Ponteland, Newcastle Upon Tyne, NE20 9BD.

...it's dynamite!

POWER CARTRIDGE

FOR YOUR COMMODORE

64/128

SO MUCH FOR SO LITTLE

"...unbelievable value for money" ZZAPP! Dec 89

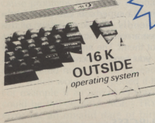
- ✱ POWER TOOLKIT
- ✱ POWER MONITOR
- ✱ TAPE & DISK TURBO
- ✱ PRINTERTOOL
- ✱ POWER RESET
- ✱ TOTAL BACKUP

..."Money well spent" YC/CDU Jan 90

42 Pg Manual - "Damned Good Handbook" CCI Jan 90

AVAILABLE FROM ALL GOOD COMPUTER RETAILERS

TRIED AND TESTED - OVER 100,000 SOLD IN EUROPE



..."highly recommended for C64 users" CCI Jan 90

YOU WILL WONDER HOW YOU EVER MANAGED WITHOUT IT

POWER TOOLKIT

A powerful BASIC-Toolkit (Additional helpful commands) that considerably simplifies programming and debugging.

AUTO
AUDIO
COLOR
DEEK
DELETE
DOKE
DUMP
FIND
HARDCAT
HARDCOPY
HEX$
INFO
KEY
PAUSE
PLIST
ILOAD
RENUMBER
REPEAT
SAFE
TRACE
UNNEW
QUIT
MONITOR
BLOAD

RENUMBER : *Also modifies all the GOTO's GOSUB's etc. Allows part of a program to be renumbered or displaced.*
PSET : *Set up of printer type.*
HARDCAT : *Prints out Directory.*

The toolkit commands can be used in your programs.

DISK TOOL

Using POWER CARTRIDGE you can load up to 6 times faster from disk.
The Disk commands can be used in your own programs.

DLOAD
DSAVE
DISK
DVERIFY
MERGE
DIR
DEVICE

MERGE : *Two BASIC programs can be merged into one.*
DISK : *With DISK you can send commands directly to your disk.*

TAPE TOOL

Using POWER CARTRIDGE you can work up to 10 times faster with your data recorder. The Tape commands can be used in your own programs.

LOAD
MERGE
SAVE
AUDIO
VERIFY

POWERMON

A powerful machine language monitor that is readily available and leaves all of your Commodore memory available for programming.
Also works in BASIC-ROM, KERNAL and I/O areas.

A	ASSEMBLE	I	INTERPRET	S	SAVE
C	COMPARE	J	JUMP	T	TRANSFER
D	DIS-ASSEMBLE	L	LOAD	V	VERIFY
		M	MEMORY	W	WALK
F	FILL	P	PRINT	X	EXIT
G	GO	R	REGISTER	$	DIRECTORY DOS Commands
H	HUNT				

PRINTERTOOL

The POWER CARTRIDGE contains a very effective Printer-Interface, that self detects if a printer is connected to the Serial Bus or User-Port.
It will print all Commodore characters on Epson and compatible printers.
The printer-interface has a variety of set-up possibilities. It can produce HARDCOPY of screens not only on Serial printers (MPS801, 802, 803 etc) but also on Centronic printers (EPSON, STAR, CITIZEN, PANASONIC, etc).
The HARDCOPY function automatically distingishes between HIRES and LORES. Multi-colour graphics are converted into shades of grey. The PSET functions allow you to decide on Large/Small and Normal/Inverse printing.
The printer PSET functions are:

PSET 0 - *Self detection Serial/Centronics.*
PSET 1 - *EPSON mode only.*
PSET 2 - *SMITH-CORONA mode only.*
PSET 3 - *Turns the printing 90 degrees!!*
PSET 4 - *HARDCOPY setting for MPS802/1526.*

PSET B - *Bit-image mode.*
PSET C - *Setting Lower/Upper case and sending Control Codes.*
PSET T - *All characters are printed in an unmodified state.*
PSET U - *Runs a Serial printer and leaves the User-port available.*
PSET Sx - *Sets the Secondary address for HARDCOPY with Serial Bus.*
PSET L1 - *Adds a line-feed, CHR$ (10), after every line.*
PSET L0 - *Switches PSET L1 off*

Bitcon Devices Ltd does not authorise or purport to authorise the making by any means or for any purpose whatsoever of copies or adaptions of copyright works or other protected material, and users of the Power Cartridge must obtain the necessary prior consent for the making of such copies or adaptions from all copyright and other right owners concerned. See UK Copyright, Designs & Patents Act 1988.

POWER RESET

On the back of the POWER CARTRIDGE there is a Reset Button. Pressing this button makes a SPECIAL MENU appear on the screen.
This function will work with many programmes.

CONTINUE - *Allows you to return to your program.*
BASIC - *Return to BASIC.*
RESET - *Normal RESET.*
TOTAL BACKUP DISK - *Saves the contents of the memory onto a Disk. The program can be reloaded later with BLOAD followed by CONTINUE.*
RESET ALL - *RESET of any program.*
TOTAL BACKUP TAPE - *As BACKUP DISK but to TAPE.*
HARDCOPY - *At any moment, prints out a Hardcopy of the screen. Using CONTINUE afterwards you can return to the program.*
MONITOR - *Takes you into the Machine language Monitor.*

BDL
Bitcon Devices Ltd

88 BEWICK ROAD
GATESHEAD
TYNE AND WEAR
NE8 1RS
ENGLAND

Tel: 091 490 1975 and 490 1919 Fax 091 490 1918
To order: Access/Visa welcome - Cheques or P/O payable to BDL
Price: £17.30 incl. VAT.
UK orders add £1.20 post/pack total - £18.50 incl. VAT.
Europe orders add £2.50. Overseas add £3.50
Scandinavian Mail Order and Trade enquiries to: Bhiab Elektronik, Box 216, Norrtalje 76123, SWEDEN. Tel: + 46 176 18425 Fax: 176 18401
TRADE AND EXPORT ENQUIRIES WELCOME

PROGRAM

We look at DIY PROGRAMMING and in particular a DATABASE **Steven Burgess**

Last month, I started to discuss the possibilities of designing our own Database program. On face value, this would seem like an impossible task to most people. However, with a little thought and careful planning, you will discover that the task is not that impossible at all. (Please re-read last months article to recap on what has already been said).

ON WITH THE SHOW

If that all sounded rather heavy and difficult to program - which it is - then I wouldn't bother with it. Very few of the database titles floating around actually use it, as it is hard to devise an equation to fit all situations. Anyway, for your own use you will probably not need it and ordinary storage is much more versatile, if quite a bit slower.

Now we had all of those grass roots options detailed before didn't we? Well now we are going to think about a few more which will make using the program altogether a more pleasurable experience, and also about putting them together in menus.

MENUing

It is a good idea to include options which relate to one another on the same menu. In my view all matters regarding the manipulation or viewing of the database should be stored in the same menu. This may be called the DATABASE menu or the DATA MANIPULATION menu or whatever. All matters regarding LOADING and SAVING should be stored on the same menu, together with a directory command and, maybe, a scratch command. And so on. So you end up with a LOAD/SAVE menu, a DATABASE menu and a PRINTER menu and any other less necessary ones such as PREFERENCES and DISK UTILITIES and what have you.

As far as possible it is more desirable to use numbers as the keys to be pressed than letters. The numbers are situated altogether in a line across thȩ top of the keyboard; they are very easy to find. The letters, however, are rather higgedly piggedly and to someone who is used to the ABCDE... type format of children's typewriters, it could be very confusing indeed.

MAKING A DATABASE A SUPERBASE

If you include all of the grass roots options then you will have a pretty plain, but functional, database. But here we are not interested in plain databases. In this magazine we are only interested in SUPERBASES!!!

To make a database into a superbase you must firstly make it more user-friendly. Think of a few of the databases you have seen around. What's the single most unattractive thing about them? The answer is the record display screen. Don't you agree? A common output is this

```
RECORD 1
_____________

NAME : STEVEN BURGESS
AGE : 19
SEX : MALE
```

all clumped up together and if you've only got three fields then it is going to look a bit insignificant on screen, stuck in the top left hand corner.

So what we want in our database is a RECORD CARD DESIGN option. Where the user can choose where each field should be put on screen. For example:

```
RECORD 1
_____________


 NAME : STEVEN BURGESS

  AGE : 19

  SEX : MALE
```

simply by putting a space between each field and lining up the colons, the display looks altogether better.

So once the positions had been set they could be used for all output of records and even for input of records. It could be used as the template for searches as well.

PLANNING

VARIABLE TYPES

In an ideal database, the user should be able to assign specific variable types to specific fields. So AGE would be an integer, NAME a string and so on. The length of strings should also be setable (is that a word, Ed?) - this is essential when using relative files as it is necessary to know the record length as a whole.

Note it is more economical to store numeric data in numeric variables as they occupy less memory than a string containing the same number, however this may cause problems with array databases. In this instance it might be a good idea to store the number in a string and to take it out when sorting is in process so that the correct order is achieved. Sorts with strings containing numbers are prone to error.

Another useful feature would be to have ranges which data entered must fit into for each field and a specific error which would be reported if the range was violated. For example if an age of -5 was entered then an error could be IMPOSSIBLE AGE - TRY AGAIN. Whereas an error for an invalid date of birth could be given as INVALID DATE OF BIRTH - TRY AGAIN.

This user friendliness gives the user more of an idea as to what is going on and he knows then that he has made an error which many databases would not have reported.

Talking about the input of the data there is one thing that needs to be designed straight away: a more friendly input command. The built in version is okay for very simple programs which only you are to use, but it just isn't on for programs to be published which other people are expected to use. How can they know what they are allowed to type? The answer is to design your own input command which should have a limited number of allowable characters. The allowable characters could change for each field - C128 owners are lucky in this regard as they simply need to store the character set permitted into a variable and then use the INSTR(va$,v1$) command to see if v1$ is inside va$. So you could have several permitted sets - one for numbers only, one for letters only, one for letters and numbers, one for pound/dollar signs and numbers etc. Then the user could choose which one should be used by each field.

SORTS

With sorts it is handy for the user to be able to dictate which way the sort should go - in ascending or descending order. Also it should be as quick as possible - everybody loves a quick sort. The user should also be able to say which field the sort should run by.

SEARCHES

Searches should be as versatile as possible so that records which the user may have thought would turn up, turn up. You should incorporate wildcards (? and *) so that unknown characters or fields will not hinder searching. The wildcard format which I use is as follows:

? is used for single characters and will match with any character. E.G: ST???? will match with STEVEN, STRIKE and STRENT, but not with STRIKER and SEQUIN.

* is used for all characters from the asterix and matches for all of them. E.G: S* matches with anything beginning with S. * matches with anything. SPA* will match for anything which has the first three words SPA (SPADE, SPARSE etc)

If the user enters nothing for a particular record then it should be regarded as a *. If he enters something without any wildcards then it is an absolute entry - it will only match with things which it is identical to. The user should be able to enter something in all fields - but should not be forced to do so.

MISCELLANEOUS

If you include all of what is detailed above then you will certainly have a SUPERBASE. But there are extras which can make it a little bit better.

A DIRECTORY function is a godsend with databases as, unless you can remember which disk you stored your database on, you have to keep LOAD"$",8...ing all the time before loading the program.

DATE/TIME stamping may be helpful to some users, too. Then they can make sure that they have loaded the correct version of the database they have created. This leads onto a permanent DATE/TIME fixture which may be a menu in its own right and may incorporate such things as alarm clocks.

It is also useful to be able to change screen colours so that black & white t.v. owners can optimise the output and colour t.v. owners can choose colours which are gentle on the eyes.

The more you delve into application programming, the more you can find to stick in. I hope what is laid out above gives you a few ideas and, maybe, a few good programs which, indeed, CDU may be interested in seeing. Good Luck.

SchizO!

```
2/PEASY
```

It's a Mad, Mad, Mad, Mad, Mad World (as the film said), and this game proves it - STEVEN BURGESS

This week we had a letter from a Dr Madman from Lyme Regis. Dr Madman says, "I am Dr Madman and I am completely idiotic. I have written a program which I would like you to publish and if you don't I shall blow up your office. The programme is designed to make who-so-ever plays it madder than even me. Thus, I intend to make the entire world completely bonkers."

Well, how could we say to him nay?

At the point of a gun, Dr Madman forced me to play the game 100 times thus rendering me mentally mad, so that I could write for him the instructions to the game.

THE GAME

Once loading has completed, either by using the C.D.U menu or by typing **LOAD"SCHIZO"**, then you are presented with the title screen.

If you really want to play the game, and I really wouldn't advise it if you wish to remain sane, then press the fire button on a joystick in port one or press space.

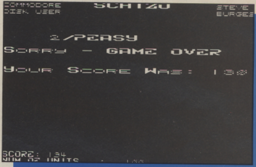
You will then be presented with the game screen. In the centre of the game screen is a sprite which, in his infinite madness, Dr Madman made in his own form. It is this that you control.

The idea of the game, apart from making the earth into a planet of mad people, is to keep the Dr Madman on the screen. Easy, I hear you cry. And so it is, at first.

You see the fiendish and irreversibly mad Dr Madman has incorporated into his fiendish and irreversibly mad program a number of fiendish and irreversibly mad features which make the program so much harder to play. Firstly, on some levels, there is a very strong gravity field which pulls you to the bottom of the screen. On some there are magnets which pull you to the left, or the right, or up, or any combination of the three. Then there is a level where all of these, left, right up and gravity are all used at different times so you never know which way you are being pulled. There is also a fiendish skull which appears quite maddeningly on some levels, then disappears and reappears in a maddeningly different and unpredictable place.

But Dr Madman has a rather more pleasant side to his madness which your first, second and seventy-eighth glance will not make you aware of. For your trouble, if you play the game, you are awarded points. The faster you move around on a screen, the more points you get. On some levels a BONUS block appears which, if you touch it, gives you 1000 points. These BONUS blocks are situated in rather precarious locations on the screen.

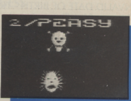
The points that you achieve from each screen all add up and when you finish the game, if you have achieved a score high enough, you will be entered into the high score table.



```
COMMODORE          SCHIZO          STEVE
DISK USER                          BURGESS

        2/PEASY
SORRY  -  GAME  OVER

YOUR  SCORE  WAS:  138




SCORE: 134
NUM OF UNITS
```

All in all there are twenty devilishly fiendish levels. If, and only if, you finish these, then you are returned to the first level so that you can amass a huge score.

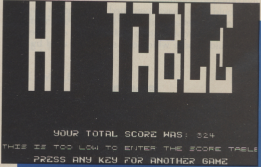
That is all I have to say about the program. Now I have finished, I am going into a dark room to stand on my head and read a famous five adventure from back to front.



```
2/PEASY
```

If you have not been put off by this article, then I would say that you are quite mad already and the game is unlikely to have any effect on you. Goodbye.

One last thing, (I'm sorry to be adding all of these annoying post-scripts, but I am mad, so what do you expect?). One last thing. The game was written and developed with LASER BASIC and LASER COMPILER from the OCEAN IQ range of utilities. Right, I've got my Enid Blyton and my head cushion. Switch off that light and shut that door! Cheerio.)



```
HI TABLE

YOUR TOTAL SCORE WAS: 324
THIS IS TOO LOW TO ENTER THE SCORE TABLE
PRESS ANY KEY FOR ANOTHER GAME
```

Techno Info

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

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	$0022-0025	34-37	Utility pointer area
RESHO	$0026-002A	38-42	Product area for multiplication
TXTTAB	$002B-002C	43-44	Pointer start of BASIC ($0801)
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	$0037-0038	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	$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)
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	$0200-0258	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
LSTHF	$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	$0300-0301	768-769	Vector: BASIC error messages ($E388)
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	8502 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
MDFLAG	$0076	118	Graphics area set flag (0-no)
NOZE	$0077	119	Sprite temp/zero counter for USING
HULP	$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)
SALH	$00AC-00AD	172-173	Pointer: tape buffer/screen scrolling
EALH	$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
STALH	$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	$00D8	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
VN	$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
SNO	$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
INDTXT	$03C9-03D1	969-977	Subroutine: fetch TXTPTR 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
M51CTR	$0A10	2576	RS232 control register image
M51CDR	$0A11	2577	RS232 command register image
M51AJB	$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
SAV80	$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
PKYBUF	$1000-1009	4096-4105	Function key string lengths table
PKYDEF	$100A-10FF	4106-4351	Function key definition table
	$1100-1108	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
VTEMP	$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
PUFILL	$1204	4612	Fill symbol for USING
PUCOMA	$1205	4613	Comma symbol for USING
PUDOT	$1206	4614	Decimal point symbol for USING
PUMONY	$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
TRPFLG	$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
POTTMP	$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	16384-44909	BASIC ROM
$AA6E-AEFF	44910-44799	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	$484B	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	$4874	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
$A94B $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
$AD8A $77D7 Evaluate numeric expression
$AD8D $77DA Checks on numeric
$AD8F $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 $7978 Gets variable
$AFA7 $48F7 Set up references
$AFE6 $4C86 BASIC-command OR
$AFE9 $4C89 BASIC-command AND
$B016 $4C86 Comparison operations
$B081 $5878 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
$B3A8       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 $9383 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 $882E Minus FAC - constant (A/Y) - FAC
$B853 $8831 Minus FAC - ARG - FAC
$B867 $8A45 Plus FAC - constant (A/Y) - FAC
$B86A $8848 Plus FAC - ARG + FAC
$B947 $8926 Complement FAC
$B97E $895D Output of 'Overflow'
$B983 $8962 Single byte multiply
$B9BC $899C Floating point constant for LOG
$B9EA $89CA BASIC-function LOG
$BA28 $8A08 Multiplication FAC - constant (A/Y) * FAC
$BA2B $8A0B Multiplication FAC - ARG * FAC
$BA8C $8A89 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 $8C28 FAC - ARG
$BC0C $8C38 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 $8C5B 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
$BFBF $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:0-39 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	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
			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
			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
			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	C128	Description of Routine
HEX	HEX	
$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
$EA87		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	$E38B	Send TALK
$ED0C	$E343	Send LISTEN
$ED40	$E3E2	Output of byte on IEC-bus
$EDB9		Send secondary address for LISTEN
$EDC7		Send secondary address for TALK
$EDEF	$E515	Send UNTALX
$EDFE	$E526	Send UNLISTEN
$EE13	$EF5C	Get a byte from the IEC-bus
$EEB3		One milliesecond delay
$EEBB	$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
$F12B	$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	$F188	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	$F65E	Get time
$F6E4	$F665	Set time
$F6ED	$EABF	Test stop-key
$F6FB		Put out error messages of the operating system
$F72C	$E8D0	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	$E98E	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
$F86B	$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
$FB97	$ED5A	Set bit counter for serial output
$FBA6	$ED69	Write one bit to tape
$FBCD	$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 ($02D5)
$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 sape 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		HEX	DECIMAL		HEX	DECIMAL	
	LOW	HIGH		LOW	HIGH		LOW	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			

EXPLORING THE 1541

TO COMPLIMENT THE SERIES ON BASIC PROGRAMMING WE ARE REPRINTING THE ARTICLE ON USING THE 1541 DISK DRIVE. WE APOLOGISE IF YOU ALREADY HAVE THIS ARTICLE BUT WE HAVE HAD LITERALLY HUNDREDS OF LETTERS REQUESTING THAT WE REPUBLISH THIS PARTICULAR ARTICLE!!!

Now that you have purchased your 1541/1570 disk drive,what can you do with it? Well the simple answer is, nothing, until you understand how and why it works. By the end of this article, you should have grasped some knowledge into the inner workings of this 'Rectangular Box'. Hopefully, your usage of the drive will benefit from what you are about to read.....

Newcomers to the world of the 1541 will probably only use the drive for storing programs, perhaps they are not aware that you can use the drive for a lot more. The more experienced users will by now be saying to themselves: 'Here we go again, heard it all before'. Before you go rushing off to make a cup of Coffee though, read on....It's never too late to learn new things.

This article is MAINLY for the 1541/1570 users, although much of the info is also pertinent to the 1571. Where possible, I will give examples for both units. (For example, everyone is aware that to communicate with the 1541 you use BASIC 2.0 commands, but for the 1571 you can also use BASIC 7.0 commands.) How do you go about learning about something like the 1541, the first thing you should know is how the information is stored on the diskettes that you spend your well earned money on. To be able to understand that, you need to know how a diskette is made up.

Information is stored on the diskette on TRACKS. On a standard 1541 disk there are 35 of these tracks. Each track is made up of a number of SECTORS. The sectors are the areas that contain the bytes of data. Each sector holds 256 bytes. The tracks are numbered from the outside to the centre. Therefore, as you get nearer the centre of the diskette, the less number of sectors each track holds. (See 1541 layout). Of these 35 tracks, there's one very important one, this is track 18. Track 18 is known as the BAM(Block allocation map) and and the DIRECTORY track. The BAM shows us what tracks and sectors contain information and which do not, and the Directory track tells us about each file that is stored on the disk. (See 1541 layout). Before we go into more detail, below is the layout of the tracks, and the sectors of the 1541, together with the sort of information that they contain.

PROGRAM FILE FORMAT

BYTE DEFINITION

FIRST SECTOR

0,1 Track and sector of next block in program file 1
2,3 Load address of program
4-255 Next 252 bytes of prg info stored as in comp mem.(keywords tokenized)

REMAINING FULL SECTORS

0,1 Track and sector of next block in program file1
2-255 Next 254 bytes of prg info stored as in comp mem.(keywords tokenized)

FINAL SECTOR

0,1 Null ($00), followed by number of valid data bytes in sector
2-??? Last bytes of prg info stored as in comp mem.(keywords tokenized).

The end of a BASIC file is marked by three zero bytes in a row. Any remaining bytes in the sector are garbage and may be ignored.

SEQUENTIAL FILE FORMAT

BYTE DEFINITION

ALL BUT FINAL SECTOR

0,1 Track and sector of next sequential data block
2-255 254 bytes of data

FINAL SECTOR

0,1 Null ($00), followed by number of valid data bytes in sector
2-??? Last bytes of data. Any remaining bytes are garbage & can be ignored

RELATIVE FILE FORMAT

BYTE DEFINITION

DATA BLOCK

0,1 Track and sector of next data block
2-255 254 bytes of data. Empty records contain $FF (all binary ones) in the first byte followed by $00 (all binary zero's) to the end of the record. Partially filled records are padded with nulls ($00)

SIDE SECTOR BLOCK

0-1	Track and sector of next side sector block
2	Side sector number (0-5)
3	Record length
4-5	Track and sector of first side sector (number 0)
6-7	Track and sector of third side sector (number 2)
10-11	Track and sector of fourth side sector (number 3)
12-13	Track and sector of fifth side sector (number 4)
14-15	Track and sector of sixth side sector (number 5)
16-255	Track and sector pointers to 120 data blocks

DIR FILE FORMAT TRACK 18

SECTORS 1-19

BYTE	DEFINITION
0,1	Track and sector of next directory block
2-31	File entry 1
34-63	File entry 2
66-95	File entry 3
98-127	File entry 4
130-159	File entry 5
162-191	File entry 6
194-223	File entry 7
226-255	File entry 8

STRUCTURE OF EACH

INDIVIDUAL DIRECTORY ENTRY

BYTE CONTENTS DEFINITION

0 128+type File type OR'ed with $80 to indicate properly closed file. (if OR'ed with $C0 instead, file is locked)

TYPES:
0 = DELeted
1 = SEQuential
2 = PROGram
3 = USER
4 = RELative

1-2 Track and sector of first data block
3-18 File name padded with shifted spaces
19-20 Rel file only. Track/ sector of first side sector
21 Rel file only. Record length
22-25 UNUSED
26-27 Track and sector of replacement file during an @SAVEor@OPEN
28-29 Number of blocks in file, stored as a two-byte integer in normal lo-byte hi-byte format

The above information tells you how each track and sector is made up, and what information is contained therein. Later in the article, I will explain just HOW the information is written to the disk. Before we get too technical though, I want to show you some of the commands available to you and how we use them. The table below shows you the various commands available, (Using BASIC), both for the 1541/1570 and for the later version 1571. After the table I will demonstrate exactly how to use each one in turn. Using BASIC 2.0 the general format is:- OPEN15,8,15:PRINT#15,"command":CLOSE15 or OPEN15,8,15,"command letter0:information":CLOSE15. (NOTE:- The first 15 in the OPEN/CLOSE command is not mandatory. This is just the file number we allocate to the command. (Normally though 15 is most widely used).

HOUSEKEEPING COMMANDS

BASIC 2.0

NEW	**"N0:disk name,disk id"**
COPY	**"C0:new file=old file"**
RENAME	**"R0:new nam=old name"**
SCRATCH	**"S0:file name"**
VALIDATE	**"V0"**
INITIALISE	**"I0"**

BASIC 7.0

NEW	**HEADER"disk name",id,dv**
COPY	**COPY"old file"TO"new file"**
RENAME	**RENAME"old name"TO"new name"**
SCRATCH	**SCRATCH"file name"**
VALIDATE	**COLLECT**
INITIALISE	**"I0"**

FILE COMMANDS

BASIC 2.0

LOAD	LOAD"filename",8 or LOAD"filename",8,1
SAVE	SAVE"filename",8
VERIFY	VERIFY"filename",8
OPEN	OPENfn,8,channel,"0:filename,file type,direction"
CLOSE	CLOSEfn
PRINT#	PRINT#fn,data list
GET#	GET#fn,variable list
INPUT#	INPUTfn,variable list

BASIC 7.0

BLOAD	BLOAD"filename"Bank#,Start address
BSAVE	BSAVE"filename"Bank#,Start address TO end address
BOOT	BOOT"filename"
OPEN	DOPEN#fn,"filename"{record length},{W}
CLOSE	DCLOSE#fn
RECORD	RECORD#fn,record number{,offset}
PRINT#	PRINT#fn,data list
GET#	GET#fn,variable list
INPUT#	INPUT#fn,variable list

DIRECT ACCESS COMMANDS

BLOCK-ALLOCATE	"B-A";0;track;sector
BLOCK-EXECUTE	"B-E";channel,0;track;sector
BLOCK-FREE	"B-F";0;track;sector
BUFFER-POINTER	"B-P";channel;byte
BLOCK-READ	"U1";channel;0;track;sector
BLOCK-WRITE	"U2";channel;0;track;sector
MEMORY-EXECUTE	"M-E"CHR$(<address)CHR$(>address)
MEMORY-READ	"M-R"CHR$(<address)CHR$(>address)CHR$(number of bytes)
MEMORY-WRITE	"M-W"CHR$(<address)CHR$(>address)CHR$(number of bytes)CHR$(data byte)CHR$(data byte).......etc
USER	"Uchar"
UTILITY LOADER	"&0:file name"
BURST (1571 only)	"U char"+character(s)

Commands intended for the drive are sent over a CHANNEL. Communication with the disk drive can be achieved over any 1 of 15 channels. Channel 15 however is reserved as the COMMAND channel. Data transfer over this channel is as follows:- Opening the channel (OPEN)

Data transfer (PRINT)
Close the channel (CLOSE)

When you initially open the channel, you specify a logical file number, this number must be in the range of 1 to 127, the device number of the drive, (this is normally 8 for single units), and a secondary address. (15 for the command channel. The logical file number is used in any subsequent commands, any number of commands can be sent until the channel is closed. These commands must be referenced by the logical file number first used in the OPEN statement

NEW - Formatting a diskette

The command NEW formats a diskette, that is to say, it prepares a new diskette for receiving data. As in all commands, the command word NEW can be reduced to a single letter. EG N=NEW. R=RENAME. For clarity, I will show all commands in their condensed format. That is to say that instead of OPEN 15,8,15:PRINT#15,"NEW:name,id". I will use the much shorter method of OPEN15,8,15,"n:name,id". Therefore to Format a new diskette we use the command:-

OPEN15,8,15,"N:name,id"

COPY - Copying files

This command allows the user to copy a file already present on the diskette The command is however seldom used, it's only real benefit is in the ability to combine several SEQUENTIAL files together to make one larger file. This method cannot be employed on PROGRAM files though.

OPEN15,8,15,"C:new file=old file1,old file2 "

RENAME - Renames a file with a new name

This command allows the user to change the name of a file on disk. It works on all file types.

OPEN15,8,15,"R:new name=old name"

SCRATCH - Scratch a file

This command allows you to get rid of any redundant files. It has the added advantage that you may scratch more than one file at a time.

OPEN15,8,15,"S:prog 1" - this would get rid of prog1 only
OPEN15,8,15,"S:prog 1,prog 2,prog 3" - this would scratch all 3 files.

(Later on you will learn how you can RECOVER files that have been scratched by mistake).

VALIDATE - Validate diskette

This command allows you to 'Clean up' or Validate your diskette. Whenever you Scratch a program, the program itself is still on the disk. All that happens is that the entry for that program is removed from the directory Vaildating your diskette makes the space of scratche'd files re-usable.

```
OPEN15,8,15,"V"
```

INITIALISE -

Initialising the disketThe DOS, or Disk operating system, requires a BAM, (Block allocation map), to be present on each disk. If you should change disks in the drive when using it, the DOS will not know that you have a different disk in the drive. Therefore it will be working on the old BAM. To combat this, you can initialise the drive. This forces the DOS to read the new BAM.

```
OPEN15,8,15,"I"
```

Now that we have dealt with the basic commands for talking to the drive, lets go on to the more exciting commands. These commands are known as the 'Direct Access' commands. Once you understand the concept behind these commands, and what they are capable of, then programming the drive in BASIC is far more entertaining. However, before I go into more detail about these commands, I feel it is time we had a look at the 'Memory Map' of the 1541. To be able to program the drive efficiently, you will need to know it's inner workings better. This is very important once you begin to experiment with M/C programs.

1541 MEMORY MAP

DRIVE ADDRESSES

HEX	DEC	DESCRIPTION
$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	ID
$0020-0021	32-33	Flag for head transport
$0030-0031	48-49	Buffer pter for disk controller
$0039	57	Constant 8, mark for begining of data block header
$003A	58	Parity for data buffer
$003D	61	Drive no. for disk controller
$003F	63	Buffer no. for disk controller
$0043	67	No. 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 no. for formatting
$0069	105	Step size for sector division (10)
$006A	106	No. of read attempts (5)
$006F-0070	111-112	Pointer to address for M and B commands
$0077	119	Dev no. + $20 (32 dec) for Listen
$0078	120	Dev no. + $40 (64 dec) 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
$0080	128	Track number
$0081	129	Sector number
$0082	130	Channel number
$0083	131	Secondary address
$0084	132	Secondary address
$0085	133	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	Pter to input buffer $0200
$00A5-00A6	165-166	Pointer to buffer error message $02D5
$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 free blocks
$0300-03FF	768-1023Buffer 0	
$0400-04FF	1024-1279	Buffer 1
$0500-05FF	1280-1535	Buffer 2
$0600-06FF	1536-1791	Buffer 3
$0700-07FF	1792-2047	Buffer 4

Right now, let's go on to the 'Direct Access Commands'. These commands will all be in BASIC, (Machine Coder's be patient).

Looking at the memory map, you can see that there are 5 buffers. However, only 4 are free for your use. (Buffer 4 is normally used for the BAM). Also please note that when using Seq and Rel files at the same time, buffer 3 is also not available because the Directory uses it. When you wish to use a buffer, you first have to OPEN a channel and specify which buffer you wish to use. For example OPEN 1,8,2,"#2" would open the channel to Buffer number 2. However it is good practice to not specify the actual buffer number but let the DOS select it for you. You achieve this by OPENing x,x,x,"#". If your selected buffer contains Alphanumeric Data, and is not over 88 chars in length. You can use the INPUT# command. (Providing the data is separated by a carriage return). Otherwise you have to use the GET# command. Remember though, that when using GET# it does not allow for null values, therefore we have to check for it via IFA$=""THENA$=CHR$(0).

Before we go any further there are 4 things you must remember:-

1. The PRINT# statement sent to the command channel 15, a direct. access command to the DOS
2. A PRINT# statement to channels 2 through to 14 sends data to a buffer.
3. An INPUT# or GET# statement to channel 15 returns any error messages.
4. An INPUT# or GET# statement to channels 2 through 14 reads data from a buffer.

The Block-read command tells the 1541 to read a sector from the disk into your openend buffer. (Strictly speaking this is known as a DIRECT ACCESS FILE). Because the first byte of the block does not get read with the Block-read command this command can be shortened to U1 or B-R. The Block-write command allows us to copy the buffer contents onto the desired sector on the disk. Block-read can be shortened to B-W or U2. Therefore, the obvious advantage to this command is to READ data into a buffer, alter it, then re-write it back to the disk. The Block-Allocate, or B-A command allows the user to reserve blocks on a disk The main purpose of this command is to prevent data from being overwritten. The Block-free or B-F command is the opposite to the B-A command. It tells the the BAM which blocks to make available. The Buffer-pointer command, shortened to B-P is to tell the DOS just where you wish to start reading or writing data to/from.

The Block-execute, shortened to B-E is quite a powerful command. In essence, you read a sector from the disk into your previously opened buffer. The contents are then executed as a machine code program from within the buffer. In practice when using this command, you specify the buffer number in the OPEN command

Along with the Direct access commands above, you have a few commands that allow you to access the DOS. (Disk Operating System). These are: A.Memory-read B.Memory-write and Memory-execute, shortened to M-R,M-W and M-E respectively.

I will now give a few examples of the Direct Access commands in operation. Feel free to experiment, but always make sure that you work on disk with no important data on it. (Mistakes DO happen).

NOTE:- When using the D/A commands, there are two methods available. Either may be used depending upon your own preferance:-

Method A is PRINT#15,"U1:"channel number;drive
Method B is PRINT#15,"U1 channel number drive"

If using method B remember to leave a space between each item inside the quotation marks.

BLOCK READ:

Suppose you wished to follow a program through on the disk by track and sector without actually reading the data. To do this you need to follow the path of the 'Link' bytes. That is the 2 bytes at the start of each block that tells you the track and sector of the next block.

```
1 OPEN8,8,15      ;Opens the command channel
2 OPEN4,8,4,"#"   ;Opens the direct access
                   file,(no specific buffer)
3 INPUT"Track and sector";TR,SE
4 PRINT#8,"U1:"4;0;TR;SE  ;Reads contents of
desired Track/Sector into buffer
5 GET#4,T$,S$             ;Reads the first two bytes of
the buffer
6 TR=ASC(T$+CHR$(0)):SE=ASC(S$+CHR$(0))
;Converts string variable to integer,
                          ;allowing for null string
7 IFTR=0THENCLOSE4:CLOSE8:END ;If last track
then finish
```

Continued on page 48........

MADDIX

An unusual concept in games play makes this game somewhat different - MARK JUDGE

What does the average computer game have? Yes, that's right, an aim. An ending in which you complete the game and think 'Oh good! I've completed it, now for something else more useful, like eating or sleeping. Well, MADDIX doesn't have an ending. However, before declaring that the game must be pretty pointless, it is worth stating that there is one purpose of playing the game, that is to get as high a score as is humanly (or otherwise) possible.

THE BASIC CONCEPT

The game is very simple, all you have to do is direct the blocks out of the bottom of the screen, where there is a small passage indicated by two white arrows pointing towards each other. Here they will be blown up. You get points for practically everything, from just moving a block, (achieved by using the fire button to pick up a block), to exploding a bonus block. A bonus block will start flashing when it is ready to be moved out of the screen, this will happen every three times you get a block out. (Indicated by the three lights at the top left of the screen). The score also varies depending upon which level you are on.

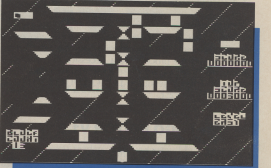
TIME IS THE ENEMY

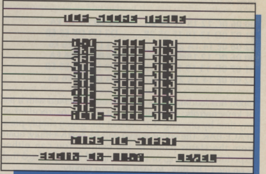
Your only enemy is time, when time runs out, a new block will appear on the screen, and a light will come on under the clock (top-right). When the time runs out three times in a row, without a block being blown up, or if more than twenty-five blocks appear on the screen then GAME OVER will occur.

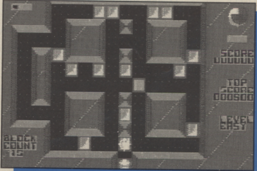
HINT TIME

A handy hint for all; the chute at the left hand side of the screen can be very useful for a speedy descent. To pick a level of play, pull the joystick left and right while on the high score screen, this will change from DODDLE (the easiest level), through to EASY, WORRIED, INSANE, SERIOUS, FIERCE, GIFTED and then MADDIX (the most difficult level).

For those that are interested, this was written in Basic and then converted to Machine Code using a compiler, obviously to speed up running time. So, there you go, Basic is not as useless as some people may lead you to believe. By the way, my highest score is 50,000, beat that!!







LOGO EDITOR V1.0 and LETTER MAKER V2.1

Graphics utilities are becoming more and more widely used. Here's two you can add to your library - ROBERT TROUGHTON

As more and more computer users are becoming increasingly interested in programming their machines, utilities to aid the process are a necessity. Graphics and Visual effects are a must these days, and to help you on your way I have designed LOGO EDITOR V1.0 and LETTER MAKER V2.1.

LOGO EDITOR V1.0

This extremely useful (!) utility was made for the sole intention of being used for displaying LOGO's to be used on DEMOS, GAMES and LETTER-PAGES. The logo-size is FIXED at 40 characters horizontally and 6 characters vertically. The character-values are structured within the logo as follows:-

```
00 06 0C 12 18 1E 24 2A.........02 08 DE E4 EA
01 07 0D 13 19 1F 25 2B.........D3 D9 DF E5 EB
02 08 0E 14 1A 20 26 27.........D4 DA E0 E6 E6
00 06 0C 12 18 1E 24 2A.........02 08 DE E4 EA
01 07 0D 13 19 1F 25 2B.........D3 D9 DF E5 EB
02 08 0E 14 1A 20 26 27.........D4 DA E0 E6 E6
```

Upon first loading the utility, you are presented with a list of key-controls. This HELP-SCREEN can be recalled at any time by pressing "F3". To exit the screen simply press SPACE-BAR. The editor-screen will be nearly empty, apart from the status panel in the centre. You can either experiment drawing, or try loading the example-logo that is on the CDU disk. To load the logo simply;

Press F1 -	to enter the disk menu
Press L -	to select 'load logo'.
Enter -	"Example logo 1" and press RETURN.
Press-	SPACE-BAR after menu appears.

CONTROLS IN EDITOR

Use CURSOR/JOYSTICK to move cursor.

FIRE/*	Set pixel under cursor
SPACE	Clear pixel under cursor
1-3	Select colour 1-3
SHIFT 1-3	Change colour 1-3
RETURN	Carriage return
F1	Disk menu
F3	Help screen
CLR	Clear whole logo
HOME	Home cursor

DISK MENU

D	**Directory**
L	**Load logo**
S	**Save logo**
SPACE	**Return to editor**

The second utility is LETTER MAKER V2.1 and is intended for use with LOGO EDITOR V1.0. You can incorporate logos designed with the LOGO EDITOR into your letters. The controls are simple and follow the format of LETTER WRITER V1, published earlier in CDU.

KEY CONTROLS

F1	Page forward
F2	Page backward
F3	Centralise line
F5	Options menu
DEL	Delete character
INST	Insert character
CLR	Clear screen
HOME	Home cursor
RETURN	Carriage return
CBM I	Insert line
CBM D	Delete line

Cursor keys move the cursor

OPTIONS MENU

+/-	Change number of pages
V	View letter
E	Edit letter
L	Save text
M	Load new music
D	Directory
C	Change logo colours
G	Load new logo
X	Save finished letter

Finally, if anyone experiences problems using any of the utilities, you can write to me (Care of) CDU editorial office and I will get you sorted out.

THE MAKING OF HELPLINE

Jason Finch discloses some of his secrets for cracking CDU Adventures

The first Adventure Helpline article appeared in the June 1990 issue of CDU and was designed to help those many people that had written to us with questions about how to overcome certain obstacles in the different adventures that the magazine had published. The first six articles covered KRON by TONY ROME and last month we finished dealing with THE ASTRODUS AFFAIR by MARK TURNER. This month we are having a break for something different, because not only do we receive letters about problems with adventures, we also receive letters asking how I know all the detailed information that I offer at monthly intervals. Questions like: Are you given the solution by the author?, Do you burn the midnight oils for weeks at a time until you finish it?, and how do you appear to know even the most obscure messages? All of these questions, and more, will be revealed in this, what I hope will be an entertaining and informative article - The Making of Helpline.

THE BURNING QUESTION

So how exactly do I find out everything about the adventures? The answer is simple: I use the same tool that the authors have used - the Graphic Adventure Creator (GAC). Once an adventure is saved off as a "runnable" file from GAC, it can actually be converted back into a data file, and then reloaded back into the GAC system. The adventure then appears in its raw format. The vocabulary is easily accessible, the room descriptions are all intact, as are the graphics and those infamous messages. The complicated conversion process (which relies on a rather nifty piece of machine code) must, I'm afraid, remain a secret - that is one thing that I will not reveal. Anyway, the whole truth is that I do not play the adventures in order to find out how to solve them, I glean all my information from the author's final version in GAC. Sorry to disappoint you!! However, that is only the beginning - the tasks involved in converting the information into something that I, and more importantly you readers, can understand have not even been touched upon yet. The next adventure we shall be covering is THE CRANMORE DIAMOND CAPER by that great adventure writer TONY ROME. That particular adventure was quite a challenge to "crack" because of the many complicated aspects involved in the programming of it. Throughout the rest of this article, it is to that adventure I shall be referring.

VOCAB COPYING

The first things that are copied out onto sheets of paper are the lists of nouns, verbs, adverbs and objects. The typical sort of end result then is shown in part below:

```
1  N,NORTH
2  S,SOUTH
3  E,EAST
4  W,WEST
5  U,UP
6  D,DOWN
7  GET,TAKE
```

and so on, with the nouns and adverbs being recorded in a similar fashion.

OBJECTS AND MESSAGES

For the objects, it is the number, the description, the start location and the weight that must be noted. Some of the ones from Cranmore are shown as examples:

```
1, a knife, 60, 4
2, a torch, 54, 4
8, a key, 60, 4
54, the locksmith, 2, 4
55, a guard, 14, 4
```

When all that has been done, the next stage is to write out all of the 255 messages that are involved in the adventure. To save on pencil leads, these are entered on a word-processor and then printed out. A booklet of some seven or eight pages is produced with entries like:

```
1:In a drawer are the numbers 29...

2:Stuck on the floor is a piece of paper. On the paper are the numbers 053...

3:The commissionaire leaves.

4:He isn't here.

5:You like your whiskey don't you!
```

THE LOCATIONS

Now the room descriptions are entered into the word-processor and printed out, two to a sheet of paper. There is then a suitably large gap in which all information about that room can be written. In case you are unfamiliar with GAC, the system requires that a set of high-priority conditions are set up, these being scanned before each input; also a set of low-priority conditions that are read after each input; and finally a set of local conditions that correspond to individual locations. The GAC system employs a whole new language to construct these conditions and it is these that are the heart of the adventure. I'll show below just one of the locations as it would appear on my sheets of paper.

2: S9
Inside a locksmith's shop. The door is to the south.

```
IF (VERB17 AND NOUN10 and CARR10 and SET?20)
MESS82 DROP10 10 TO 0 CTR(0)+7 CSET 0 SET21
WAIT END

IF (VERB75 and NOUN54 and ADVE1) MESS89 WAIT
END

*INCR(54) END

*IF (CTR(54)=1) LF MESS63 END

*IF (NOT(AT2)) 0 CSET 54 END
```

Unless you are familiar with GAC, most of that will have meant absolutely nothing to you. By the end of this article you will see how that sort of thing is converted into perfectly understandable English sentences! Let's look at the components. The number '2' is simply the location number and the 'S9' afterwards is called a connection. It means that by going SOUTH you will arrive at location number nine. The next bit is simply the description as it appears on the screen. It is the next lines that take time.

A QUICK OVERVIEW

GAC uses a system of "flags" to detect whether certain things have been done or not, such as whether the guard is awake or whether he has fallen asleep. The language involved can be rather complicated but things like DROP10 mean 'drop object number ten', and GET10 would do the opposite. 10 TO 0 means put object ten in location zero, CTR(0) is the score. The counters (CTR) act exactly the same as variables. You can add or subtract values to them and from them. WAIT is just a command to tell GAC that it should then wait for the next input. If you are unfamiliar with GAC then you may find some aspects of this article confusing, although I shall do my best to keep it straightforward. It just isn't possible for me to duplicate the GAC manual here for you.

ALL DONE

When all of the location information has been entered, the high- and low-priority conditions are copied out. These look the same as above and any that correspond to certain locations are copied to the relevant location info sheet. Hopefully you can appreciate that quite a lot of paperwork has been amassed by now.

SET WHAT?

The next job is to go through the text that I have written out and highlight every reference to a counter or a flag. The laborious process of finding out exactly what each does then begins. In the last example you saw a command SET21. In Cranmore this has the effect of telling the computer that the locksmith has been given the wax. Similar situations warrant the use of other flags - is the torch on? Is the tablet in the bottle? Has the glass been cut? And so on. Counters in Cranmore are used to count the number of turns that you have spent in Ricos, to calculate how long the torch batteries will last, to keep note of the floor number that you are on, etc.. Once that is done, I have a list of vocabulary, objects, messages, what each flag/counter does, all of the conditional checks that the adventure makes and usually also a roughly drawn map of what I think the adventure looks like. You will have seen one of these last month in the Adventure Helpline section. For Cranmore it was also necessary to draw up a chart of different times, and to work out exactly what had to be done by certain times, or within certain time restrictions.

INTO ENGLISH

The next stage is to convert the conditions into a plain English format. Commands from GAC such as IF (VERB34 and NOUN3 and CARR3) MESS142 EXIT can be converted into statements like: 'If "EAT/SWALLOW TABLET" typed and player has tablet, then print "You start to feel drowsy and fall into a deep sleep....", end game.' This process is carried out on EVERY high- and low-priority condition that is independent of any specific location. I have listed a few examples directly from my paperwork below:

If "GIVE MONEY" typed and not carrying MONEY: Print"You have no money", (WAIT)

If "SWITCH TORCH OFF" typed and torch is on: Print"You switch the torch off", flag torch as off, (WAIT)

If "ASK LOCKSMITH + something" and he's NOT present: Print"He isn't here", (WAIT)

The above are all low-priority commands that are based on what the player has input. The high-priority commands, as I have said before, are assessed before the player has entered any command. Such lines become, in plain enough English:

If TURN=83 (Time=7.50pm): Move guard out of adventure

If TURN=149 and locksmith has wax (Time=10.00pm): Put locksmith in Rico's bar and flag that he is there.

However, there are occasional lines where the "jargon" remains. One of the ones in Cranmore that relates to displaying the time has ended up as:

If (TURN>248 and FLAG 28 IS SET but FLAG 34 IS RESET) (1.20am or later): "A guard grabs you!....", EXIT

JUST THE ROOMS

When all that is done, only the rooms remain. Near the start we saw a small example of one location - it was location number two. Knowing what the VERBs and NOUNs are, and what the different flags and counters do, we can translate all of that into very plain sentences:

Location 2: South to 9.
Inside the locksmith's shop. The door is to the south.

*If you have just entered the locksmith's shop he will ask if he can help you.

If you are carrying the wax in which you have made an impression of the key, and you give the wax to the locksmith then he will agree to meet you at Rico's at exactly 10pm.

If you ask him anything else, he will just shrug his shoulders.

The asterisked entry corresponds to a high-priority command that is directly related to this location. You will notice that now we have only three entries and not the five we had before. The first line corresponds to "IF (CTR(54)=1) LF MESS63 END". Counter 54 keeps track of how many turns you have had in the shop. If it is one then you have just entered. MESS63 displays message number 63 which is the greeting. The two high-priority commands that are missing are "INCR(54) END" and "IF (NOT(AT2)) 0CSET54 END". They are left out of the English translation because in simple terms there is no need to translate them. The first would be "add one onto the number of turns in the shop" and the second would be "as soon as you leave the shop tell the computer you are not in it". There is no point in putting them in the literal translations of the raw code.

ALL THERE IS TO IT

Now that is done for every single location in the adventure, some having no associated sentences and some having ten to fifteen. I hope that you have understood everything that I have said and that I have put an end to your curiosity as to how I am able to give you hints and tips. The very last thing that I do before embarking on a series about one adventure is to draw up a sequential list of location numbers. You will probably have noticed that in the past articles, no location numbers are missing - it starts at number one, and runs on to two, three, four, all the way to the final one. However, in the "raw" form of the adventure, many numbers are missed out. For example, Cranmore uses locations 1 to 18, but then skips to 20, then 24, 25, 26 and 27, then 30 and so on. My last job is to make sure that the order in the final series that appears in the magazine is correct, running from one, through every number to the maximum.

So now you know the secrets. I have taken you on a very quick guided tour of the methods involved. The final booklet that tells me everything about Cranmore is fourteen pages thick and contains information about every location. The low- and high-priority information is mingled in where necessary. From start to finish, working on an adventure non-stop, the process takes what may appear to be a long time - seven days. Bear in mind there is a lot of typing to be done!! Now then, where did I put that February disk? Perhaps now I'll be able to sit down and actually play through the Cranmore Diamond Caper!

ADVENTURE WRITING

Jason Finch continues his tutorial for all you budding Adventure Writers

This month we are going to discuss possible programming techniques for the main body of the adventure. You will find out what the basic methods for recognising and acting upon commands are, and you will discover how you can get the computer to react quite simply by displaying various fixed reports. On this month's disk you should find two more picture files for the final adventure that we are working towards - they are prefixed with the word PIC. As always these have been done by my graphical artist friend, Doug Sneddon, down there near Salisbury. Many thanks to him for them. If you would like to see these two pictures then you can use the MODULES program that I presented a few months back. You will first have to change the number of files accepted by the BASIC program which shouldn't cause too many hassles.

Right then, how many of you have used the Graphic Adventure Creator from Incentive Software? The method used for designing adventures in that is a pretty standard method and is similar to the one that I shall be explaining here. It relies on you having your adventure split up into locations. You then have a group of things that are done before an input is requested from the player, a group of things that are done immediately after the input is received, and a group of things that are specific to the location that you are in, which are also done after the player's input has been received. There are different methods though and I shall discuss both the above and one of the latter below.

GETTING YOUR PRIORITIES RIGHT

If there is to be a witch in your adventure that looks at you as soon as you enter her cave, you will need a comment such as "The witch turns and stares at you with an evil glance". This would need to be displayed BEFORE the prompt "What now?" or similar appears. However, something like "The witch follows you" would want to be displayed AFTER the input has been received. These two types of situation need to be distinguished and you would use a GOSUB command to jump to the routines that do the HIGH priority commands - those that are issued before you enter any command, and then one to jump to the LOW priority commands - those checked after you enter a command. Whatever method you use for the other bits, these routines are vital.

METHOD ONE

For the rest of the adventure, there are, as mentioned, two methods that you can use for distinguishing what can be done. The first one is as follows. Each location can have its own conditions and checks that are contained in one subroutine. You can use an ON L GOSUB xxx,xxx,xxx... command to jump to the different ones. Each location can have any number of checks and these are often based on what has been entered. For example, you may want to see whether the player has entered "TOUCH CAULDRON" so that you can display the message "The cauldron contains boiling liquid and burns you instantly". It would be pointless doing this check as a LOW priority condition because it is only concerned with the one location - the one in which the cauldron is placed. Other things specific to certain locations can be counters. For example, each time you are in the cave, you may want to increment a counter, and when it reaches a certain value have the witch grab you. Again, this counter and its appropriate messages only apply to the one location. Each location has a subroutine to check the player's INPUT and the response that is required, as opposed to method two which....

METHOD TWO

Is the opposite way around. Each VERB in your adventure has its own subroutine. After a verb has been recognised, you jump to the subroutine with something like ON V GOSUB xxx,xxx.... The "TOUCH CAULDRON" example would then be handled as follows. TOUCH would be detected as a verb and the computer would jump to the appropriate section of the program. You then check to see whether the location is equal to that of the cave, and if it is you do a further check to see whether you have used CAULDRON as the NOUN. If you have, it prints the appropriate retort. You see then that with this method, each verb has a subroutine to check the player's LOCATION and the response that is required.

THE BRAIN

Whichever method you decide to use, it all needs linking together into a section of the program that I am going to call the brains of the operation. Forget the patser for a moment - that just works out what you are saying. The brain has to work out exactly what you mean, and exactly how to react. The structure of the brain is shown below as a rough sort of English

BASIC section:

```
(start)
GOSUB high

IF dead=1 THEN do death
GOSUB input
GOSUB parser
GOSUB low

IF dead=1 THEN do death
ON L GOSUB x,x,x...

IF dead=1 THEN do death
GOTO start
```

This may seem to be a bit over simplistic and a bit morbid with all the comments about death, but they are just checks to see whether the adventure is over, either by the player having been killed, or by him quitting (which will have been detected by the general low priority commands in "GOSUB low"). You can see how the structure of the brain is put together and in what order the routines should be called. I have used above method one whereby each location has its own subroutine. It is not vital that it is done that way, but it is a lot easier.

That really is all there is to programming an adventure in theory. What a bold statement I have just made. Of course the reality is much more difficult because we can't just say "GOSUB input" and have the computer know what we mean, we need to program an input section, and you will find one in the MODULES program that was provided a few issues ago. That is a rather decent subroutine that you should find satisfies your needs. The next important thing to discuss are reports of what is going on in the adventure. These take the form of text that the program displays either BEFORE or AFTER the player has entered his input. For example, "You examine the chest and find that it is locked" is a report, as is "The cave is dark with water dripping from various areas of the rock roof. To the east the tunnel continues". The latter report is just a special one - a location description. The easiest way to store these reports in BASIC is to have them as string variables. You can READ them in with DATA statements if you like but you will need some way of connecting them together to form long strings. Next time I'll provide you with some example messages and show how they would be displayed and used to the best effect. To display a report, you simply have to do something like PRINT RP$(3). If RP$(3) was "It is locked." then this can be used each time that you try a locked door, or attempt to open a locked chest.

IS THAT ENOUGH?

Yes, I think it is. I have given you plenty to be going on with, although it may not seem like it. You can now start writing down on paper what conditions are required in certain circumstances and what sort of messages need displaying. If you are having difficulties in programming the commands successfully, then be patient and next time I'll give you a chance to see how I have done it. Until then, which due to this series being bimonthly, will be September, good luck with your designing. I look forward to seeing some of your creations when you have finished them.

If you have any Ideas, Hints, Tips or Suggestions that will he of interest to all the other readers, put it in a leter (or on a postcard if you don't feel like writing too much) and pop it into one of the reepticals below to;

ADVENTURE WRITING
CDU
Alphavite Publications Ltd
20, Potters lane, Kiln Farm
Milton Keynes, Bucks
MK11 3HF

To enter the wacky world of THE WHEELER DEALER GUIDE ring 0908 569819

MEMORY TRANSFER

A simple Memory Transfer program for novices wishing to learn more about memory management - LEE BAMBER

The **MEMORY TRANSFER** program is a very useful utility to keen programmers and novices, for it does more than just transfer memory. It explains what it is, why it's used and how. By the time you have used this simply utility you will have climbed another rung up the ladder of memory management.

Programmers move memory around to suit their programs. If not, they could end up with a major problem, no room left for their code, for example. Screens can also be found and moved around to suit your purposes, be it business or pleasure.

All relevant information is on the disk but I will give you a quick explanation here to show you the workings of the program. The **MEMORY TRANSFER** has three **OPTIONS/COMMANDS.** (Two of significance, and one for quitting the utility). The first of the options is **MEMORY TRANSFER,** this transfers selected memory locations around the computers memory. It uses questions to gather the relevant information needed to carry out the operation. The second is a **MEMORY VIEWER,** which enables you to see what you are transferring, and where you have transferred to.

TO BEGIN

On the disk, along with the main utility, is a short Basic introduction to the program. Select it from the main **CDU** menu, or alternatively, load it directly by the command **LOAD"MEMORY TRANSFER",8** when the **READY** prompt appears type RUN. After the introduction has finished, you will be prompted to load in the main **MEMORY TRANSFER** utility.

SAVEing BLOCKS

If for any reason you would like to save a specified block of memory, use the following formula;

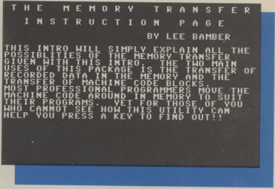
```
PRINT (start address)/256      <RETURN>
XX                             <XX=High byte start address>
PRINT ((start address)-XX*256) <RETURN>
YY                             <YY=Low byte start address>
```

Now do the same but replace (start address) with (end address) to give the HIGH and LOW bytes of both the start and end addresses needed to operate the save program. Use the following formula to save the specified block of memory.

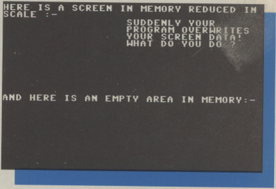
```
SYS 57812"(filename)",8,1
POKE193,(HB SA):POKE194,(LB SA)
POKE174,(HB EA):POKE175,(LB EA)
SYS 62957
```

(Where HB = High Byte, LB = Low Byte, SA = Start Address, EA = End Address).

You should now have a file on the disk which contains the memory block between the two addresses.



```
THE MEMORY TRANSFER
   INSTRUCTION PAGE
                BY LEE BAMBER

THIS INTRO WILL SIMPLY EXPLAIN ALL THE
POSSIBLITIES OF THE MEMORY TRANSFER
GIVEN WITH THIS INTRO.  THE TWO MAIN
USES OF THIS PACKAGE IS THE TRANSFER OF
RECORDED DATA IN THE MEMORY AND THE
TRANSFER OF MACHINE CODE BLOCKS.
MOST PROFESSIONAL PROGRAMMERS MOVE THE
MACHINE CODE AROUND IN MEMORY TO SUIT
THEIR PROGRAMS.  YET FOR THOSE OF YOU
WHO CANNOT SEE HOW THIS UTILITY CAN
HELP YOU PRESS A KEY TO FIND OUT!!
```



```
HERE IS A SCREEN IN MEMORY REDUCED IN
SCALE :-
                SUDDENLY YOUR
                PROGRAM OVERWRITES
                YOUR SCREEN DATA!
                WHAT DO YOU DO ?

AND HERE IS AN EMPTY AREA IN MEMORY:-
```

CAUTION

Do not transfer memory blocks between locations 2043-4010 for the **MEMORY TRANSFER** program resides there. I hope you enjoy using this simple utility, and that it gives you a better insight into the art of memory management.

NOW IS THE TIME TO CATCH UP ON ISSUES YOU HAVE MISSED

The following back issues of CDU are still available direct from ALPHAVITE PUBLICATIONS LTD. Please note that if ordering one of the following back issues, you will receive a copy of the disk, along with photostat copies of instructions for the relevant disk programs ONLY. These back issues cost £4.50 each which includes Post/Packing. Please make cheques/Postal Orders out to:- ALPHAVITE PUBLICATIONS LTD (Allow 28 days for delivery).

VOL 1 No.1 NOV/DEC '87

DIRECTORY DESIGNER - Tidy up your disks with this Editor/Designer.
TEXT ENHANCER - Improve your text displays.
MOBSTER - Have you got what it takes to be a gangster.
3 INTO 1 PLUS - A superb Character, Sprite and Background Editor.
SKI RUN - All the thrills of the slopes with this game.
SPRITE PRINTER - Dump your favourite sprites onto your CBM printer.

VOL 1 No.2 JAN/FEB '88

DISK LIBRARIAN - Keep track of what's on what disk.
DISK MATE - Handy pop-up disk functions.
NOLUXE PAINT - A superb low-res drawing package.
TEXT CRACKER - Grab those character sets you like for your own use.
QUAD - New life for the brick/bat game.
FIVE-UP - Can you win at this dice game?
RAM DISK C128 - Our first program for the C128.

VOL 1 No.3 MAR/APR '88

SUPER-TACT - Tactics are the essence of this game.
CHAOS IN SPACE - A shoot-em-up that's deceptively different.
C-ZAP - Speed is the name of the game with this compiler.
BASIC+ - A comprehensive Basic extension.
TAPE ARCHIVE - Be safe and back up your disks.
LINK & CRUNCH - Running out of memory/disk space? Not anymore.
PSYMON - A full-facility machine code monitor.
DISK LIBRARIAN II - An updated version of DL.
C128 AUTOBOOT - For C128 owners - load C64 progs at C128 speed.

VOL 1 No.4 MAY/JUN '88

DRUMSYNTH - Percussive programming.
C128 PULL DOWN WINDOWS - Windowing for the C128.
TOKENISER - Word-process your Basic programs.
C-CAD - Enter the world of Computer-Aided-Design.
BASIC COMPACTOR - Squeeze up your Basic programs.
SANTOLUS - A demanding smooth-scrolling maze.
ATLANTIS - Explore the lost continent.

VOL 1 No.5 JUL/AUG '88

DISK TOOLKIT - The Editors very own comprehensive utility for disk users.
RELOCATOR - How to move your machine code.
MIND GAMES - Unscramble the Presidents brain.
3-D BREAKOUT - Bash those bricks in 3 dimensions.
PEGGY 128 - An amusement for C128 owners.
ORRERY - Planetary positions computed.
MESSAGE CONSTRUCTION KIT - Full-screen scrolling messages.

VOL 1 No.6 SEP/OCT '88

SCORPION - If it moves, kill it.
COLOUR MATCH - Tailor your 64 screen colours to your own taste.
C128 SPREADSHEET - Accounts can be simple.
ESCAPE - Can you find a way to escape the Nazis?
STARBURST - Your chance to save the galaxy.
SCORE KEEPER - Using Sprites for your game scores.
ADDIT - A tactical numbers game.
LOCATION FINDER - Find out what that bit of code's up to.
FRACTAL FROLICS - Fun with the Mandelbrot set.

VOL 2 No.1 NOV/DEC '88

CDU FORTH - Escape from Basic with our compiler.
TEXTED - Word-processing made easy.
EXTRACTOR - Build up your sprite library.
WINDOWS 64 - Generate windows painlessly.
ZMON - Program your C28's Z80 chip.
CRIBBAGE MASTER - A C64 first, this program plays a mean game.
OBLIVION! - Fight off the deadly Janoids.

VOL 2 NO.2 JAN/FEB '89

DISK TURBO - Speed up your disk access.
BLASTBALL - A bat 'n ball extravaganza.
COLOUR BIND - A sliding block problem with a difference.
BORDER SPRITE - Make use of those screen edges.

DATA MAKER - The easy way to get machine code into Basic.
LIFE - The bizarre world of cellular automata.
MENU MAKER 128 - Make your 128 disks easy to use.
MICRODOT - Save the world from radiation sickness. (Superb game)
SPOTS - Can you beat the machine's dice rolls?
EASY SCROLLER - Scrolling made simple.
RUNAWAY - Can you escape from a dreary domestic existence?
LOGIC - The puzzle that will have you tearing your hair out.

VOL 2 No.3 MAR/APR '89

DARTS - Pub fun - in the comfort of your own home.
CDU PAINT - The ultimate C64 graphics package.
DEVAID - The Editors very own extended Basic with 41 new commands.
BAZAIR - Can you survive the maze of death?
ARAKNIFOE - Get your own back on those creepy-crawlies.
C128 GRAPHICS PRIMER - Make use of those 80 columns.
DOMINOES - Pit your wits against Max and Joe.
PHANTOM - Strike a blow for world revolution.

VOL 2 No.4 MAY/JUN '89

BASE ED - Get organised with this C64 database.
DBASE 128 - 40 or 80 column storage for C128 owners.
6510+ - The ultimate in C64 assemblers.
SID SEQUENCER - Make commodore music with ease.
LIBERTE - Escape the POW camp in this 1940's style adventure.
FX KIT - Bangs, Pows and Zaps made easy.

VOL 2 No.5 JUL/AUG '89

FONT FACTORY - Create your own characters.
HIRES DEMO KIT - Add music to your favourite picture.
ANIMATOR - Get those sprites moving
BORDER MESSAGE SCROLL - Say what you like along the bottom of the screen.
TYPIT-128 - Create professional text layout on your C128
SCREEN COPIES UTILITY - Download your favourite screens, including CDU paint files.
VIDI-BASIC - Graphic based extension to Basic.
64 NEWS DESK - Become a C64 reporter.

VOL 2 No.6 SEP/OCT '89

MICKMON - An extensive M/C monitor.
SCRAPBOOK - Collectors and hobbyists database.
CELLRATOR - Enter the caves if you dare.
RAINBOW CHASER - Rainbows means points in this unusual game.
HIDDEN GRAPHICS - Utilise those graphic screens.
FORTRESS - Save the world! Yet again.
DISK HUNTER - Keep tabs on your disk library.
SUPERFILE - One more for the record keepers.

VOL 3 No.1 NOV '89

BASIC EXTENSION - Windows and Icons the easy way.
B-RAID - Vertical scrolling shoot'em up.
DISKONOMISER - Prudent disk block saving.
HELP - Design your own information help screens.
ORSITAL - An arcade style game with a difference.
PROGRAM COMPARE - Basic program development has never been easier.
RASTER ROUTINES - A few colourful demos.
SPRITE EDITOR 1 - A no nonesense basic sprite editor.
WABBIT - Help the rabbit collect his carrots.

VOL 3 No.2 DEC '89

KRON - Can you meet the challenge?
CDU MENU KIT - Design your own menus with ease.
PHOBOS - Break out of jail and gain your freedom.
LIMBO - Collect the cells off the blocks.
MUSIBASIC - Sound and music made easy.
PANIC - Is your eye as quick as your joystick.
TEMPLATE DESIGN - Backgrounds the easy way.
QUIKWORD - You very own expandable word processor.

VOL 3 No.3 JAN '90

4 IN A ROW - Connect a row of counter.
FROGS IN SPACE - Leap to safety across the space lanes.
BLACKJACK - Don't loose your shirt.
LORD OF DARKNESS - Defeat the evil lord in true adventure style.
MARGO - Fly around and collect jewels and fuel.
JETRACE 2000 - Have you got what it takes to be best?
ULTIMATE FONT EDITOR - Create your own screens, layouts and characters.
SELECTIVE COLOUR RESTORER - Design your own system start up colours.
6510+ UNASSEMBLER - Transform 6510+ M/C into source with labels.
TRIVIA CHALLENGE - The first of 3 files for this superb game.

VOL 3 No.4 FEB '90

COLOUR PICTURE PRINT - Download your favourite colour screens.
BASE-ED2 - An update to our popular database system.
1ST MILLION - Play the market in this strategy game.
FM-DOS - Enhance your drives operating system.
GEOS FONTS - A further 4 fonts for Geos users.
HASHING IT - Reletive file programming made easy.
MULTI-SPRITE - Make full use of up to 24 sprites.
DIRECTORIES EXPLAINED - Find your way round the directory jungle.
TRIVIA CHALLENGE - The 2nd part.

VOL 3 No.5 MAR '90

PLAGUE - Become your planets Guardian and Defender.
SURROUND - Reversi on the C64
GEOS FONTS - The last of 12 new Geos fonts.
SCREEN SLIDE - Create your own slideshows.
JOYSTICK TESTER - Put your stick(s) through the mill.
COLOUR MATCHER - Mastermind for the younger players.
SCREEN MANIPULATOR - Full use of the screen now obtainable.
VIDEO RECORD PLANNER - Keep tab on your home recordings.
TRIVIA CHALLENGE - The 3rd and final part of the game.

VOL 3 No.6 APR '90

BAR PROMPTS - M/C input routine.
HI-LITE BARS - Input routine but in Basic.
TEXAS DEMO - Example of using Basic in demos.
CHARS TO SPRITES - Convert UDG's to sprites.
FONT FACTORY - Complimentary program to the above.
3D-TEXT MACHINE - Impressive 3D text screens the easy way.
SCREEN ENHANCER - Makes full use of the screen easy to achieve.
SPREADSHEET 64 - An excellent, easy to use spreadsheet.
MINI-AID - 3 short utilities to aid the Basic programmer.
C128 COLLECTION - 3 very useful C128 programs.

NOW IS THE TIME TO CATCH UP ON ISSUES YOU HAVE MISSED

The following back issues of CDU are still available direct from ALPHAVITE PUBLICATIONS LTD. Please note that if ordering one of the following back issues, you will receive a copy of the disk, along with photostat copies of instructions for the relevant disk programs ONLY. These back issues cost £4.50 each which includes Post/Packing. Please make cheques/Postal Orders out to:- ALPHAVITE PUBLICATIONS LTD (Allow 28 days for delivery).

VOL 1 No.1 NOV/DEC '87

DIRECTORY DESIGNER - Tidy up your disks with this Editor/Designer.
TEXT ENHANCER - Improve your text displays.
MOBSTER - Have you got what it takes to be a gangster.
3 INTO 1 PLUS - A superb Character, Sprite and Background Editor.
SKI RUN - All the thrills of the slopes with this game.
SPRITE PRINTER - Dump your favourite sprites onto your CBM printer.

VOL 1 No.2 JAN/FEB '88

DISK LIBRARIAN - Keep track of what's on what disk.
DISK MATE - Handy pop-up disk functions.
NOLUXE PAINT - A superb low-res drawing package.
TEXT CRACKER - Grab those character sets you like for your own use.
QUAD - New life for the brick/bat game.
FIVE-UP - Can you win at this dice game?
RAM DISK C128 - Our first program for the C128.

VOL 1 No.3 MAR/APR '88

SUPER-TACT - Tactics are the essence of this game.
CHAOS IN SPACE - A shoot-em-up that's deceptively different.
C-ZAP - Speed is the name of the game with this compiler.
BASIC+ - A comprehensive Basic extension.
TAPE ARCHIVE - Be safe and back up your disks.
LINK & CRUNCH - Running out of memory/disk space? Not anymore.
PSYMON - A full-facility machine code monitor.
DISK LIBRARIAN II - An updated version of DL.
C128 AUTOBOOT - For C128 owners - load C64 progs at C128 speed.

VOL 1 No.4 MAY/JUN '88

DRUMSYNTH - Percussive programming.
C128 PULL DOWN WINDOWS - Windowing for the C128.
TOKENISER - Word-process your Basic programs.
C-CAD - Enter the world of Computer-Aided-Design.
BASIC COMPACTOR - Squeeze up your Basic programs.
SANTOLUS - A demanding smooth-scrolling maze.
ATLANTIS - Explore the lost continent.

VOL 1 No.5 JUL/AUG '88

DISK TOOLKIT - The Editors very own comprehensive utility for disk users.
RELOCATOR - How to move your machine code.
MIND GAMES - Unscramble the Presidents brain.
3-D BREAKOUT - Bash those bricks in 3 dimensions.
PEGGY 128 - An amusement for C128 owners.
ORRERY - Planetary positions computed.
MESSAGE CONSTRUCTION KIT - Full-screen scrolling messages.

VOL 1 No.6 SEP/OCT '88

SCORPION - If it moves, kill it.
COLOUR MATCH - Tailor your 64 screen colours to your own taste.
C128 SPREADSHEET - Accounts can be simple.
ESCAPE - Can you find a way to escape the Nazis?
STARBURST - Your chance to save the galaxy.
SCORE KEEPER - Using Sprites for your game scores.
ADDIT - A tactical numbers game.
LOCATION FINDER - Find out what that bit of code's up to.
FRACTAL FROLICS - Fun with the Mandelbrot set.

VOL 2 No.1 NOV/DEC '88

CDU FORTH - Escape from Basic with our compiler.
TEXTED - Word-processing made easy.
EXTRACTOR - Build up your sprite library.
WINDOWS 64 - Generate windows painlessly.
ZMON - Program your C28's Z80 chip.
CRIBBAGE MASTER - A C64 first, this program plays a mean game.
OBLIVION! - Fight off the deadly Janoids.

VOL 2 NO.2 JAN/FEB '89

DISK TURBO - Speed up your disk access.
BLASTBALL - A bat 'n ball extravaganza.
COLOUR BIND - A sliding block problem with a difference.
BORDER SPRITE - Make use of those screen edges.

DATA MAKER - The easy way to get machine code into Basic.
LIFE - The bizarre world of cellular automata.
MENU MAKER 128 - Make your 128 disks easy to use.
MICRODOT - Save the world from radiation sickness. (Superb game)
SPOTS - Can you beat the machine's dice rolls?
EASY SCROLLER - Scrolling made simple.
RUNAWAY - Can you escape from a dreary domestic existence?
LOGIC - The puzzle that will have you tearing your hair out.

VOL 2 No.3 MAR/APR '89

DARTS - Pub fun - in the comfort of your own home.
CDU PAINT - The ultimate C64 graphics package.
DEVAID - The Editors very own extended Basic with 41 new commands.
BAZAIR - Can you survive the maze of death?
ARAKNIFOE - Get your own back on those creepy-crawlies.
C128 GRAPHICS PRIMER - Make use of those 80 columns.
DOMINOES - Pit your wits against Max and Joe.
PHANTOM - Strike a blow for world revolution.

VOL 2 No.4 MAY/JUN '89

BASE ED - Get organised with this C64 database.
DBASE 128 - 40 or 80 column storage for C128 owners.
6510+ - The ultimate in C64 assemblers.
SID SEQUENCER - Make commodore music with ease.
LIBERTE - Escape the POW camp in this 1940's style adventure.
FX KIT - Bangs, Pows and Zaps made easy.

VOL 2 No.5 JUL/AUG '89

FONT FACTORY - Create your own characters.
HIRES DEMO KIT - Add music to your favourite picture.
ANIMATOR - Get those sprites moving
BORDER MESSAGE SCROLL - Say what you like along the bottom of the screen.
TYPIT-128 - Create professional text layout on your C128
SCREEN COPIES UTILITY - Download your favourite screens, including CDU paint files.
VIDI-BASIC - Graphic based extension to Basic.
64 NEWS DESK - Become a C64 reporter.

VOL 2 No.6 SEP/OCT '89

MICKMON - An extensive M/C monitor.
SCRAPBOOK - Collectors and hobbyists database.
CELLRATOR - Enter the caves if you dare.
RAINBOW CHASER - Rainbows means points in this unusual game.
HIDDEN GRAPHICS - Utilise those graphic screens.
FORTRESS - Save the world! Yet again.
DISK HUNTER - Keep tabs on your disk library.
SUPERFILE - One more for the record keepers.

VOL 3 No.1 NOV '89

BASIC EXTENSION - Windows and Icons the easy way.
B-RAID - Vertical scrolling shoot'em up.
DISKONOMISER - Prudent disk block saving.
HELP - Design your own information help screens.
ORSITAL - An arcade style game with a difference.
PROGRAM COMPARE - Basic program development has never been easier.
RASTER ROUTINES - A few colourful demos.
SPRITE EDITOR 1 - A no nonesense basic sprite editor.
WABBIT - Help the rabbit collect his carrots.

VOL 3 No.2 DEC '89

KRON - Can you meet the challenge?
CDU MENU KIT - Design your own menus with ease.
PHOBOS - Break out of jail and gain your freedom.
LIMBO - Collect the cells off the blocks.
MUSIBASIC - Sound and music made easy.
PANIC - Is your eye as quick as your joystick.
TEMPLATE DESIGN - Backgrounds the easy way.
QUIKWORD - You very own expandable word processor.

VOL 3 No.3 JAN '90

4 IN A ROW - Connect a row of counter.
FROGS IN SPACE - Leap to safety across the space lanes.
BLACKJACK - Don't loose your shirt.
LORD OF DARKNESS - Defeat the evil lord in true adventure style.
MARGO - Fly around and collect jewels and fuel.
JETRACE 2000 - Have you got what it takes to be best?
ULTIMATE FONT EDITOR - Create your own screens, layouts and characters.
SELECTIVE COLOUR RESTORER - Design your own system start up colours.
6510+ UNASSEMBLER - Transform 6510+ M/C into source with labels.
TRIVIA CHALLENGE - The first of 3 files for this superb game.

VOL 3 No.4 FEB '90

COLOUR PICTURE PRINT - Download your favourite colour screens.
BASE-ED2 - An update to our popular database system.
1ST MILLION - Play the market in this strategy game.
FM-DOS - Enhance your drives operating system.
GEOS FONTS - A further 4 fonts for Geos users.
HASHING IT - Reletive file programming made easy.
MULTI-SPRITE - Make full use of up to 24 sprites.
DIRECTORIES EXPLAINED - Find your way round the directory jungle.
TRIVIA CHALLENGE - The 2nd part.

VOL 3 No.5 MAR '90

PLAGUE - Become your planets Guardian and Defender.
SURROUND - Reversi on the C64
GEOS FONTS - The last of 12 new Geos fonts.
SCREEN SLIDE - Create your own slideshows.
JOYSTICK TESTER - Put your stick(s) through the mill.
COLOUR MATCHER - Mastermind for the younger players.
SCREEN MANIPULATOR - Full use of the screen now obtainable.
VIDEO RECORD PLANNER - Keep tab on your home recordings.
TRIVIA CHALLENGE - The 3rd and final part of the game.

VOL 3 No.6 APR '90

BAR PROMPTS - M/C input routine.
HI-LITE BARS - Input routine but in Basic.
TEXAS DEMO - Example of using Basic in demos.
CHARS TO SPRITES - Convert UDG's to sprites.
FONT FACTORY - Complimentary program to the above.
3D-TEXT MACHINE - Impressive 3D text screens the easy way.
SCREEN ENHANCER - Makes full use of the screen easy to achieve.
SPREADSHEET 64 - An excellent, easy to use spreadsheet.
MINI-AID - 3 short utilities to aid the Basic programmer.
C128 COLLECTION - 3 very useful C128 programs.

BACK ISSUES

VOL 3 No.7 MAY '90

NUDGE - FLD explained in laymans terms.
WINDOW WIPER - An alternative screen wipe system.
CHARACTER EXTRACTOR - Borrow those nice character sets you see.
MAZE GENERATOR - Create your own fun.
HIRES ANIMATOR - This difficult subject made easier.
SPRITE DRIVER - Platform game designing without the fuss.
ROTOTRON - Demonstration of Sprites and Sound.
TEXT COMPRESSION - How to squeeze a gallon into a pint.
SCREENS - Make up your own help screens and keep them in memory.
INTERRUPT POINTERS - Geos style windows and pointers for you.

VOL 3 No.8 JUN '90

ALEATORY MUSIC - An alternative music system.
SPRITE BASIC - Efficient sprite handling through Basic.
SPRITE GENERATOR - Another sprite editor for your library.
MUNCHER - Pacman returns with a vengeance.
ASTRODUS - Escape the spaceship Astrodus in this adventure.
1581 DIRECT ACCESS - Find your way around the 1581 disk drive.
PERSONAL ORGANISER - Design your own organiser pages.
128 CONVERTOR and MATHS AID - 2 more for C128 users.

VOL 3 No.9 JUL '90

QUICK MERGE 64/128 - Another useful routine for your archives.
THE GAME PLAN - An aid to knowing what where in your games.
CHARACTER dESIGNER - Another designer for those without.
HASHBASE 128 - A powerful database for C128 users.
REVASM 64/128 - Two unassemblers for non Speedy Assembler owners.
SPEEDY UNASSEMBLER - An unassembler specific to Speedy Assembler users.
BANKS AND MEMORY - An aid to redifining screen and graphic memory.
GRAPHICS FACTORY - A novel way of getting in graphic design.
POT POURRI - A selection of useful routines for all users.

VOL 3 No.12 OCTOBER '91

ROLL'EM - An example of using Graphics Factory.
COLOUR MATCH - A short utility for C128 users.
SPREAD-ED - The third in the 'ED series.
RASTER EDITOR - Put those raster lessons to good use.
ADDRESS BOOK - A somewhat unusual address base.
SUPERSORT 64/128 - Sorts have never been easier.
SPRITE EDITOR C128 - Another utility for C128 users.
GRAPH-ED - The last in the 'ED series.
BACKGAMMON - That popular board game gets an airing.

VOL 4 No.4 FEBRUARY '91

CRANMORE DIAMOND - Another Tony Rome adventure.
COMPACTOR - Program protection made easy.
ADVISOR - Artificial intelligence on the 64.
GALACTIC ENCOUNTER - Battleships played out in space.
IRQ64 - Interrupts on the C64 explained.
2 FOR THE C128 - Check your 128's RAM and PLAY command.
CHEQUE BOOK ORGANISER - Organise your cheque books and statements.

The following back issues can be obtained from Select Subscriptions Ltd, River Park Estate, Berkhamsted, Herts, HP4 1HL. These back issues cost £3.75 each which includes Post/packing. Please make cheques/postal orders out to:- SELECT SUBSCRIPTIONS LTD (Allow 28 days for delivery).

VOL 3 No.10 AUG '90

LIMBO 2 - The sequel to Limbo
SCREEN DESIGNER 128 - Screen designing made easy.
DATABASE78 - A database full of features.
LETTER MAKER - Text Screens made decidedly pleasing.
FUNCTIONS - Make full use of those function keys.
GAMES LIST CREATOR - Keep tabs on your games disks.
DUAL DISKCOPY - At last an intelligent disk copy program.
SEQUENCER64 - Musicians have a field day.
SECURITY - Put all those broken joysticks to good use.
SUPERBOOT! - Auto load your programs.

VOL 3 No.11 SEP '90

BANKING 128 - A simple way of keeping your money straight.
DISK DRIVER V4 - A simple disk utility.
AUTOBOOT 128 - C128 users get easy access to CDU programs.
READING BETWEEN THE LINES - Build your own adventure parser.
I.D.O.S. - A comprehensive drive utility.
PRICE CALCULATOR - Keep tabs on inflation.
B.O.S.S. - Yet another alternative to the standard Basic.
SCREEN DESIGNER/COMPILER - Impressive screen layouts for all.
LANDSCAPE ROUTINE - Beginners guide to scrolling backdrops.
SAMPLE KIT 64 - More sampling for all you musicians.

VOL 4 No.1 NOV '90

WAR AT SEA - C128 version of battleships.
FULL DISK JACKET - Keep track of your disks.
NEAGOX - Blast everything that moves.
NUMDEF - A basic game to test the reflexes.
MEMORY SCANNER - Look through memory the easy way.
MONEY 64 - Budget planning for the 90's.
XINOUT - An alternative input routine.
CALENDAR C128 - No more buying of calendars.
GOMOKU - An interesting variation of 'GO'.
SMOOTH SCROLL DEMO - Create your own scrolls.

VOL 4 No.2 DEC '90

ILS (German Program) - A C128 language tutorial.
SCREEN DESIGN CORRECTION - An update to this excellent utility.
BETTER BACKUPS - Help for ARC users.
MACHINE CODE GEMS - A suite of MC routines to aid you.
COLOUR TABLE EDITOR - Get those raster bars right.
MULTITASKING C128 - An implementation for the C128.

VOL 4 No.3 JAN '91

2X2 CHARACTER EDITOR - Design large fonts easily.
ENCRYPTION - Password and program protection.
SECURE - Another against would be hackers. .
KA43/5 OPEN SYSTEM - Expand the C64's op system.
32 SPRITES - The world of sprites opened up.
DISPLAY.ASM - Multitasking source file displayer.
TERMINUS - A demo that is somewhat unusual.

Subscribe now . . . And Save £9

OR KICK YOURSELF FOR THE REST OF THE YEAR . . .

We've gone mad and are offering you the opportunity of receiving Commodore Disk User's next twelve issues for the staggeringly low price of *£30 if you live in the U.K. We will even post it to you free as well.**

The current price of Commodore disk user for 12 issues is £39,
however if you are smart you can save £9 by subscribing to this offer, so don't delay the offer ends June 28th 1991
CDU is the only magazine worth considering in the world of the serious C64 Commodore user.

Published monthly –

COMMODORE DISK USER is the answer to every Commodore computer owner's dream. The disk supplied with the magazine contains a variety of ready to use, high quality computer programs – no more lengthy typing in of listings. The scope of the programs is wide, varying from games to business software and high-powered disk utilities – and the disk would retail for at least £50.00 if bought independently.

Of course, that isn't all. The magazine, besides containing full and comprehensive instructions for using the disk, is a complete computer journal in its own right, with news, reviews, programming, competitions and general interest features.

PRIORITY ORDER FORM

Please commence my subscription to Commodore Disk User with the ....issue.
I enclose a cheque/postal order for £.. made payable to **ALPHAVITE PUBLICATIONS LTD.**
or debit £..................from my Access/Visa Card No:
Valid from.......................to............................ □□□□□□□□□□□□□□□□
Signature...Name...
Address...
....Post code ....

Cut out and send this form with your remittance to: **SELECT SUBSCRIPTIONS LTD.**
5 RIVER PARK ESTATE, BERKHAMSTEAD, HERTS HP41ML

...............Offer ends JUNE 28th 1991..............

* Rates refer to subscriptions sent post free to UK addresses. Overseas rates on request.

```
 8 PRINT"Track number is: "TR,"Sector number is: "SE ;print them out
 9 GOTO4 ;Repeat process
```

BUFFER POINTER:

Suppose you wish to read the diskette name from within a program. As you know the name starts at position 144 of track 18, sector 0. Normally you would have to read the first 143 bytes and ignore them. However the DOS has an easier way. You can point to any position within the buffer by the B-P command The bytes are numbered 0-255 in the buffer, the buffer pointer can be set to zero automatically by the use of the U1 command though.

```
 1 OPEN8,8,15 ;Open command channel
 2 OPEN4,8,4,"#" ;Open direct access file
 3 PRINT#8,"U1:"4;0;18;0 ;Read contents of desired Track/sector into buffer
 4 PRINT#8,"B-P:"4;144 ;Point to where we want to start reading from
 5 FORX=1TO16 ;Length of disk name
 6 GET#4,X$:IFX$=CHR$(160)THEN8 ;If shifted space end
 7 PRINTX$:NEXT ;print out,read next letter
 8 CLOSE4:CLOSE8:END
```

BLOCK-WRITE:

Block-write, is used in conjunction with the block-read command. It allows one to write the contents of a buffer onto the disk at any desired position. The command does NOT alter the contents of the buffer.(You do this task yourself). In the following example we will be changing the disk name that we read with the previous example.

```
 1 OPEN8,8,15
 2 OPEN4,8,4,"#"
 3 PRINT#8,"U1:"4;0;18;0
 4 PRINT#8,"B-p:"4;144
 5 X$="NEW DISK NAME"
 6 IFLEN(X$)<16THENX$=X$+CHR$(160):GOTO6
 7 PRINT#4,X$; ;Change the contents of the buffer
 8 PRINT#8,"U2:"4;0;18;0 ;Write contents back to disk
 9 PRINT#8,"I":CLOSE4:CLOSE8:END ;Re-intialize drive and finish
```

BLOCK-ALLOCATE:

When using Program, Sequential or Relative files on a disk, the BAM is being constantly updated as to blocks that are allocated. This prevents blocks from being overwritten. However, when we use Direct Access files, these are NOT allocated in the BAM, therefore there is a danger that they could be overwritten. To prevent this from happening we can use the Block-Allocate command If we try to Allocate a block that has already been allocated, we will be given the error message 65,NO BLOCK,T,S (T and S are the next higher numbered free blocks available).

The syntax for using the Block allocate command is:- B-A drive track sector The following example would mark track 17 sector 5 as being allocated in the BAM

```
 1 OPEN8,8,15
 2 PRINT#8,"B-A:"0;17;5
```

BLOCK-FREE:

As indicated by it's name, this command frees any allocated blocks and marks them in the BAM as being free to use

If you wished to make the above track and sector free to use you would use the following

```
 OPEN8,8,15
 PRINT#8,"B-F:"0;17;5
```

NOTE: Allocating and freeing blocks has an effect only on blocks that are used by Prg,seq and rel files by the DOS. The B-W and B-R commands do not check the BAM before overwriting blocks. Using these commands you can write to blocks marked as allocated in the BAM. If, for instance, you have a disk that contains only Direct access files, it is unnecessary to allocate written blocks because no other files will be written on the diskette. Therefore in this case you could use the directory blocks in track 18 and therefore have 672 blocks available on the diskette.

To give you an example of the use of this. One could store a menu program onto track 18, thus space on the diskette is not wasted by the menu.

BLOCK-EXECUTE:

Block-execute is used when you wish to read a block from the disk into a buffer then execute the contents as a machine code program. The syntax for the command is: B-E channel drive track sector. When using the B-E command, the buffer number is usually given in the OPEN command, just in case the M/C prog is not relocatable. IE: OPEN4,8,4,"#2".

```
1 OPEN8,8,15
2 OPEN4,8,4,"#2"
3 PRINT#8,"B-E:"4;0;14;6
```

This would read the contents of track 14, sector 6 The B-E command is used in conjunction with the B-R and Memory Execute commands that follow.

MEMORY COMMANDS

There are three memory commands that we will deal with. They are Memory Read, (M-R) Memory write, (M-W) and Memory execute, (M-E). All these commands pre-supposes are knowledge of the inner workings of the DOS and a knowledge of 6502/6510 code.

The syntax for the Memory read command is:-

```
M-R CHR$(LO) CHR$(HI) {(CHR$(number)}
```

CHR$(LO) is the low byte of the address in DOS that is to be read

CHR$(HI) is the hight byte of the address in DOS that is to be read

CHR$(number) is the OPTIONAL extra parameter indicating how many bytes to read

In the following two examples, example 1 shows how to read how many free blocks are remaining on the disk. Example 2 shows how to read the disk name.

```
1 OPEN8,8,15
2 PRINT#8,"M-R"CHR$(250)CHR$(2)
3 GET#8,X$:IFX$=""THENX$=CHR$(0)
4 PRINT#8,"M-R"CHR$(252)CHR$(2)
5 GET#8,Y$:IFY$=""THENY$=CHR$(0)
6 PRINTASC(X$)+256*ASC(Y$)
7 CLOSE8
```

```
1 OPEN8,8,15
2 PRINT#8,"M-R"CHR$(144)CHR$(7)CHR$(16)
3 INPUT#8,X$
4 PRINTX$
5 CLOSE8
```

Memory write is the complimentary command to Memory read. Writing can only be accomplished to DOS Ram, page zero, stack and the buffers. It is possible to send more than 1 byte with this command. The command syntax is as follows:

```
M-W CHR$(LO) CHR$(HI) CHR$(NUMBER) CHR$(DATA) CHR$(DATA) etc etc...
```

Finally, the Memory execute command will call up and execute a machine code program that resides in DOS memory. The routine MUST end with an RTS. The syntax for the command is as follows:-

```
M-E CHR$(LO) CHR$(HI)
```

You can not only execute your own routines written with the use of the M-W command, but also the DOS ROM routines.

So now that we have skirted the subject of Direct Access and Memory commands, just what exactly is possible. The following table list just a few ideas that readily spring to mind:-

A. You can manipulate the sectors and change the BAM
B. You can make changes to the Directory
C. You can make changes to files
D. You can protect files from accidental erasure
E. You can CLOSE files that are still OPENed
F. You can read and alter any sector that you desire
G. You can prevent directories from being viewed
H. You can prevent directories from being loaded into memory
I. You can recover lost or damaged files
J. You can create data structures that the DOS would not normally recognise
K. You could place a menu program within the directory track,thus saving space
L. You could put a simple form of 'Protection' on the disk to prevent
illegal pirating of a file.

Really the list is boundless. Only your own imagination will set the limits of what can be achieved by the use of these commands. I cannot stress the importance of making sure you do not use important disks for your experiments.

As you are no doubt aware, the 1541 uses the GCR, (Group Coded Recording), method of storing data onto the disk. If you want to know more about this method, I refer you to 'Your Commodore', issue JUNE 1986, page 75-77. All I will say on the subject is that by using this method, more information can be stored on the disk than you think is possible.

I hope that this article as given you a better understanding of the 1541, and of how to use it. There are many things that I have left out, but these are all covered by the many publications that you can buy. There is not enough space here to explain everything in detail. Study the listings of some of the programs in this issued, and of previous issues. Practice, Experiment but above all else............

Have fun!!!

Commodore
·DISK·USER·

Lineage: 58p per word. (+VAT)
Semi display: £11.50 plus VAT per single column centimetre minimum 2cm. Ring for information on series bookings/discounts.

All advertisements in this section must be prepaid.
Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request).

0908 569819

Send your requirements to:
CDU CLASSIFIED DEPARTMENT, ALPHAVITE PUBLICATIONS LTD.,
20 POTTERS LANE, KILN FARM, MILTON KEYNES, MK11 3HF.

PUBLIC DOMAIN SOFTWARE

for the C64/128.
We have 150 disks full of all types of programs
All disks are £2.95. each

NOW FOR THE+4
From £3.75 per Disk.
Send SAE or phone for free catalogue (state model)

Kingsway Computers Services
72 Glencoe Road,
Sheffield S2 2SR
Tel:
0742 750623 / 0742 780222

HOME WORKERS WANTED
.(Mailing Enverlopes) £3 per Hr.
STAMPED S.A.E. TO
MAT KIZA
27 WOODSIDE PLACE
GLASGOW G37QL.

REPAIRS

C64, C+4............................£30. inc.
C128. 1541.........................£40. inc.
Three month warranty, 1 week turnrounnd

SPARES

906114....	£9.30	8501.......	£11.75
6510.......	£11.50	C64 Power Pack	£22.00
6526.......	£11.95	C64 Cassette unit	£22.00
6581.......	£14.95		
6569.......	£21.50		

OASIS COMPUTER SERVICES
Dept 4C. 14 Ridgeway Road,
Salisbury, Wiltshire, SP1 3BU
Tel:(0722) 335061

COMMODORE 64/128 SOFTWARE LIBRARY

☆Life Membership
☆Updates
☆Teenage Mutant Hero Turtle Tape £1.75
☆ 7 Day Hire
☆Tapes From £1.50 to £1.75
Also Disks.

FOR FREE CATALOGUE S.A.E TO -
ACE SOFTWARE LIBRARY,
14 CHIPENDALE COURT, BELFAST, BT10 0DU

CLASSIFIED COUPON

ALL CLASSIFIED ADVERTISEMENTS MUST BE PRE-PAID.
THERE ARE NO REIMBURSEMENTS FOR CANCELLATIONS.

CDU CLASSIFIED DEPARTMENT, ALPHAVITE PUBLICATIONS,
20 POTTERS LANE, KILN FARM, MILTON KEYNES, MK11 3HF.

RATES:Lineage 58p per word (+VAT). Semi-display: £11.50 (+VAT) per single column cm minimum size 2cm. Series discounts available.

I enclose my Cheque/Postal Order for £.................for ....insertions.
made payable to Alphavite Publications.
(Delete as necessary)

□□□□□□□□□□□□□□□□ PLEASE DEBIT MY ACCESS/VISA

CARD NO.:

EXP. DATE:...
£...............................FOR..........................INSERTIONS
Name:...
Address:...
....Post Code:.......................
Daytime Tel No.: ....

Signature: ....Date:...
❑ FOR SALE ❑ SOFTWARE ❑ SPECIAL OFFERS ❑ OTHER

ONLY POOLS AND HORSES

Every program written by a mathematician who has spent many years in the betting industry. Programs that utilise the tried and trusted methods of the professional, not pie in the sky theories that fail to pass the test of time.

FOOTBALL BOXFORM Written by a former pools expert for Littlewoods. The program has forecast over 50% more draws than would be expected by chance.

Homes, Aways and draws shown in order of merit and true odds given for every match. Merit tables show at a glance the teams currently in form and those currently having a lean spell. Australian pools program included in the price.

POOLS PLANNER by the same author. Full details given of 369 easily entered block perms ranging from 9 to 73960 lines and from 12 to 56 selections. All are accepted by the pools firms and are checked in seconds by your computer.

RACING BOXFORM Course characteristics (built in to the program) as well as the form of the horses are considered to speedily produce an order of merit for each race. Designed for flexibility allowing users to amend the program if they wish. **Price** still includes the highly acclaimed HANDI CAP WINNER - more than 1000 winners every year - over 25% of them at 5/1 or better. Order two or more and receive **FREE** a program
to work out almost any bet. So good its used by bookies,

Prices (Tape) £15.95 each. £25.95 any two. £35.95 all three. For discs please add £2. per program.

Advertised for Six years in the sporting and computing press.
BOXoft CLEVER.. GET THE BEST

BOXOFT (CDU), 65 Allans Meadow,
Neston, South Wirral L64 9SQ
Chque/ P.O./Access/Visa Tel:051-336-2668

VISA

BBC B & MASTER, AMSTRAD CPC & PCW COMMODORE 64/128, SPECTRUMS.

TO ADVERTISE ON THIS PAGE OR OTHERS TEL: 0908 569819

Elvira®

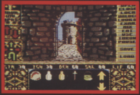
mistress of the dark™

- Hour upon hour of addictive entertainment
- Totally icon-driven for ease of play
- Turbo disk loading guarantees fast graphical access
- An Award Winning game

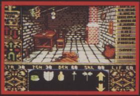
A SUPERB FANTASY ROLE-PLAYING GAME WITH OUTSTANDING GRAPHICS AND GAMEPLAY

- Over 1 Megabyte of game code supplied on 3 double-sided disks

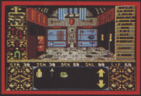
- Hundreds of locations in and around Elviras Haunted Castle
- Hand to hand real time combat with Knights and Monsters
- Includes instructions and secret spell book for mixing spells and potions



CASTLE RAMPARTS
CBM64



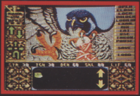
GARDENERS SHED
CBM64



CASTLE ARMOURY
CBM64

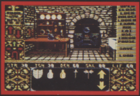
AWESOME QUALITY: THE BEST FRP EVER SEEN ON THE CBM 64

ELVIRA rescues you from the Jailers cell and sends you on an unforgettable mission to dispose of the spirit of her great great grandmother **EMELDA.** On your travels in and around the **HAUNTED CASTLE** you will come across some of the most weird and...

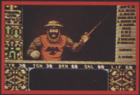


FALCON ATTACK
CBM64

....**EVIL** monsters and beings you have ever encountered. From **UNDEAD KNIGHTS** to the **CATACOMB BEAST, WEREWOLVES** to a **VAMPIRE,** and of course wicked **EMELDA** herself, this is the challenge that you will play **AGAIN AND AGAIN!**

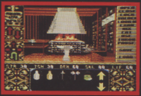


ELVIRA IN THE KITCHEN
CBM64



A KNIGHT FIGHTING

ALL SCREENSHOTS FROM CBM64



THE BLACKSMITHS FORGE
CBM64

For information about the Elvira Fan Club:
14755 Ventura Blvd.,
#1-710, Sherman Oaks, CA 91403, USA

DESIGNED BY HORROR SOFT LTD

FLAIR
S·O·F·T·W·A·R·E

Available from your local dealer or direct from Flair Software Ltd, The Smithy Side, 7 Belle Villas, Ponteland, Newcastle Upon Tyne. NE20 9BD Priced £24.99

NEW

THE COMPLETE COLOUR SOLUTION

£179

Vidi ... No 1 in UK & Europe (Leading the way forward)

Get the most out of your Amiga by adding:

"The Complete Colour Solution"

The Worlds ultimate creative leisure product for your Amiga. Capture dynamic high resolution images into your Amiga in less than one second.

And Look No Filters

Images can now be grabbed from either colour video camera, home VCR or in fact any still video source. The traditional method of holding three colour filters in front of your video camera is certainly a thing of the past. Because Vidi splits the RGB colours electronically there are no focussing or movement problems experienced by some of our slower competitors. Lighting is also less of an issue as light is not being shut out by lens filters. Put all this together with an already proven Vidi-Amiga/VidiChrome combination and achieve what is probably the most consistant and accurate high quality 4096 colour images ever seen on the Amiga.

The colour solution is fully compatible with all Amiga's from a standard A500 to the ultimate A3000. No additional RAM is required to get up and running.

You will see from independant review comments that we are undoubtedly their first choice and that was before the complete solution was launched. If you have just purchased your Amiga and are not sure what to buy next, then just read the comments or send for full review and demo disk.

Actual unretouched digitised screenshot

Features ...

- Grab mono images from any video source
- Capture colour images from any still video source.
- Digitise up to 16 mono frames on a 1meg Amiga.
- Animate 16 shade images at different speeds.
- Create windows in both mono & colour.
- Cut & Paste areas from one frame to another.
- Hardware and software brightness & contrast control.
- Choice of capture resolutions standard & Dynamic interlace.
- Full Palette control.
- Add text or draw within art package.

Amiga Computing: The best Amiga digitiser has had the technicolour treatment. Vidi must be one of the most exciting peripherals you can buy for your Amiga.

Micro Mart: When I first saw Vidi "in the flesh" as it were, at the CES show last September it looked to be the answer to a frustrated Digi View owner's dreams – in fact to see pictures appearing on screen without the customary two minutes wait seemed almost too good to be true. I have consistently produced more good quality pictures in the short time I have had Vidi than I ever did with Digiview.

Zero: Now under normal circumstances cheap usually means poor quality but this is not the case with Rombo. Why? cos Vidi-Amiga is the best digitiser for under £500 and I've tried them all.

Amiga Format: Where quality is concerned, Vidi produces some of the best results I've seen on any digitiser at any price.

Amiga User International: The latest addition to the Rombokit is called Vidi-RGB and brings this already impressive package to the realms of totally amazing. CONCLUSION: Who will find Vidi-Amiga useful? The answer to this is almost anyone with a video recorder or camera and a passing interest in graphics.

ROMBO
Limited

Full colour demonstration disk available for only £1.95 to cover P&P.

6 Fairbairn Road, Livingston, EH54 6TS. Tel: 0506-414631 Fax: 0506-414634

Visit us at the 16 Bit Computer Show Novotel hotel, Hammersmith 12th-14th july 1991 Stand No. 101