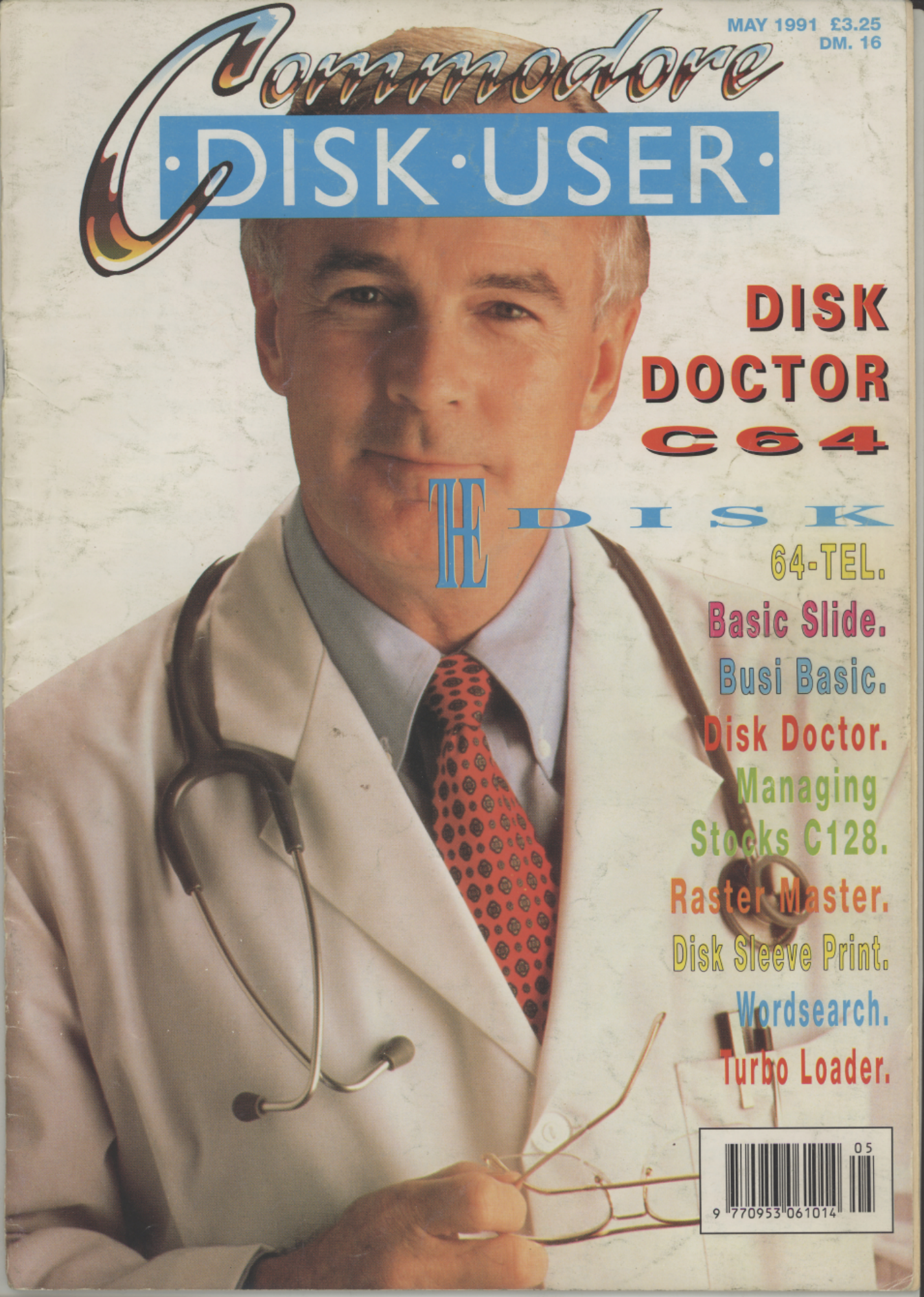
MAY 1991 £3.25
DM. 16

Commodore DISK USER



DISK DOCTOR C64

THE DISK

64-TEL.

Basic Slide.

Busi Basic.

Disk Doctor.

Managing Stocks C128.

Raster Master.

Disk Sleeve Print.

Wordsearch.

Turbo Loader.

05
9 770953 061014

LOGIC

London 081-882 4942 Cheshunt 0992 25323 Peterborough 0733 49696 London W1 071 935 2547

FULL RANGE OF AMIGA, ST, SEGA, 64, SPEC, AMSTRAD, PC, PCW, 2600, XL/XE, BBC

CAULDRON II	FIREBIRD	4.99
MUNSTERS	ALTERNATIVE	4.99
NAM	MINDSCAPE	7.99
SLOT CAR RACER	MINDSCAPE	4.99
BEYOND THE DARK CASTLE	MINDSCAPE	4.99
INDOOR SOCCER	MINDSCAPE	4.99
FINAL ASSAULT	EPYX	4.99
DIVE BOMBER	EPYX	6.99
SPY V SPY III	EPYX	4.99
SPORTS A RONI	EPYX	4.99
SPORTS NEWS BASEBALL	EPYX	5.99
DEATH SWORD	EPYX	7.99
IMPOSSIBLE MISSION	EPYX	2.99
THE GAMES SUMMER EDITION	EPYX	6.99
4*4 OFF ROAD RACING	EPYX	4.99
STREET SPORTS FOOTBA	EPYX	4.99
CALIFORNIA GAMES	EPYX	4.99
WORLD GAMES	EPYX	4.99

STREET SPORTS BASEBALL	EPYX	4.99
SPACE STATION OBLIVION	EPYX	7.99
AXE OF RAGE	EPYX	4.99
SUMMER CHALLENGE	MINDSCAPE	4.99
WINTER CHALLENGE	MINDSCAPE	4.99
TRANSFORMERS	MEDIAGENIC	4.99
CHAMPIONSHIP BASEBALL	MEDIAGENIC	4.99
LEATHER GODDESSES OF PHOBOS	MEDIAGENIC	4.99
EYE OF HORUS	LOGOTRON	3.99
STARRAY	LOGOTRON	3.99
FOOTBALL MANAGER	PRISM	4.99
ADDICTABALL	ALLIGATA	3.99
FOOTBALL MANAGER 2 CONSTRUCTION KIT	PRISM	8.95
NORTH STAR	GREMLIN	2.99

INTERNATIONAL SOCCER	CBM	4.99
VENOM STRIKES BACK	GREMLIN	2.99
GARY LINEKARS SUPERSKILLS		3.99
STARCROSS	CBM	4.99
SUSPENDED	CBM	4.99
DEADLINE	CBM	4.99
PARADROID	HEWSON	2.99
MURDER OFF MIAMI	CRL	4.99
UP PERISCOPE	SUB LOGIC	9.95
RUNNING MAN	GRANDSLAM	4.99
SUPER SCRAMBLE SIMULATOR		4.99
GLIDER PILOT	CRL	4.99
HERCULES SLAYER OF THE DAMNED	GREMLIN	2.99
BAAL	PSYGNOSIS	4.99
EXOLON	HEWSON	2.99
SHIRLY MULDOWNEYS TOP FUEL CHALLENGE	US GOLD	4.99
BLASTEROIDS	IMAGE WORKS	3.99

SOUND STUDIO	
HOME RECORDING STUDIO	4.95
GEOS INC GEO WRITE GRAPHICS OPERATING SYSTEM WITH WORD PROCESSOR	9.95
SOUND EXPANDER FM SOUND MODULE	9.95
ASSEMBLER DEVELOPER	9.95
PROGRAMMERS UTILITIES	9.95
LOGO	9.95
PROGRAMMERS TOOL BOX	4.95
INTRO TO BASIC PART 2	4.95
PACK OF ACES	4.99
INTERNATIONAL KARATE, BOULDERDASH WHO DARES WINS, NEXUS BEST OF ELITE	4.99
BOMB JACK, FRANK BRUNOS BOXING SPACE HARRIER, AIRWOLF POWER HOUSE	2.99
FIGHT NIGHT, OSMIUM GOLDEN OLDIES	6.99
BIATHLON, MOONSWEEPER, LAWN TENNIS SPACE GALLERY, SLALOM, INTRUDER, SQUASH KO BOXING ETC ETC ETC ETC KICK BUT SLAM	4.99
FIST, RAMBO, UCHI MATA, BOP N WRESTLE PERSONAL MONEY MANAGER	9.95

BALLISTIX	PSYGNOSIS	4.99
NINJA HAMPSTER	CRL	3.99
ARTURA	GREMLIN	4.99
APACHE STRIKE	ACTIVISION	4.99
PAZZAZ	CBM	4.99
SCRAMBLE SPIRITS	GRANDSLAM	5.99
DEJA VU	MIRRORSOFT	6.99
SOKO BAN	SPECTRUM HOL	4.99
MURDER BY THE DOZEN	CBS	7.99
PARRALAX	OCEAN	4.99
INFILTRATOR II	MINDSCAPE	5.99
THE FLINTSTONES	GRANDSLAM	4.99
MS PACMAN		4.99
THUNDER CHOPPER	SUB LOGIC	9.95
UNINVITED	MINDSCAPE	7.99
PACLAND	GRANDSLAM	7.99
TERRYS BIG ADVENTURE	GRANDSLAM	5.99
BLOODWYCH	IMAGE WORKS	6.99
FIGHTING SOCCER	ACTIVISION	4.99
NAVY SEAL	COSMI	7.99
INTRIQUE	MIRRORSOFT	6.99
BATALLION COMMANDER	SSI	6.99
THE THREE STOOGES	MIRRORSOFT	5.99

SWIFT SPREADSHEET 9.95

THE MOST POWERFULL SPREADSHEET PROGRAM
AVAILABLE ON THE COMMODORE 64, WITH POP
UP MENU CONTROL, CELL MATRIX FROM A1 TO
Z254 OR LARGER ON THE C128.

SUB BATTLE SIMULATOR 9.95

THIS IS YOUR CHANCE TO EMBARK ON WHAT IS
UNQUESTIONABLY THE MOST DETAILED, REALISTIC, ALL ENCOMPASSING WAR SIMULATION EVER CREATED.

TRACK AND FIELD 9.95

THE CLASSIC SPORTS ARCADE GAME BY ATARI
SOFT INCLUDING A FREE ARCADE STYLE CONTROL PAD.

JET 11.95

BY SUBLOGIC. FLY AN F16 FIGHTING FALCON
THE MOST ADVANCED TACTICAL FIGHTER IN
THE WORLD, OR PUT YOURSELF AT THE CONTROLS OF A CARRIER BASED F1`8 HORNET
THE US NAVYS NEWEST MULTI ROLE FIGHTER.

STEALTH MISSION 14.95

STEALTH MISSION PUTS YOU IN THE PILOTS SEAT OF THREE DIFFERENT JETS FROM THEF19, X29, OR NAVY TOMCAT,
NEW 3D ANIMATION TECHNIQUES PROVIDE DRAMATICALLYFASTER FRAME RATES FOR ALL COCKPIT WINDOW VIEWS.
INCLUDES COMPLETE COR, ILS, ADF AND DME AVIONICS FOR CROSS COUNTRY NAVIGATION.
COMPATIBLE WITH SUBLOGIC SCENARY DISCS.

WHERE TO BUY:
EITHER AT

19 THE BROADWAY
THE BOURNE
SOUTHGATE
LONDON
N14 6PH

UNIT 6
MIDGATE
PETERBOROUGH
CAMBS
PE1 1TN

5 LYNTON PARADE
CHESHUNT
HERTS
EN8 8LF

MAIL ORDER TO:
5 LYNTON PARADE
CHESHUNT
HERTS
EN8 8LF
Tel: 0992 25323

POSTAGE AND PACKING: 1-3 ITEMS 75P, 4 OR MORE £1.00

CONTENTS

Commodore DISK·USER·

Volume 4 Number 7 MAY 1991

ON THE DISK

DISK DOCTOR
A useful utility for Disk Drive Users 5

MANAGING STOCKS
A Stocks management program for C128 users 7

DISK SLEEVE PRINT
Another sleeve printer for normal printers 8

RASTER MASTER
Yet another Raster Utility for Graphic freaks 20

64-TEL
Transform your C64 into a Teletext service 21

WORDSEARCH
A puzzle compilers dream come true 31

BUSI BASIC
Business program makers get a helping hand 39

TURBO LOADER
Fast load all your favourite programs

IN THE MAGAZINE

WELCOME
Instructions and Editors comment 4

ADVENTURE HELPLINE
More help with THE ASTRODUS AFFAIR 9

BASICS OF BASIC
Basic programmers get more assistance 10

INTRODUCTION TO MACHINE CODE
Our new series is now well underway 24

TECHNO-INFO
It's that man again with all the answers 32

NUMBYTES
Finally the last in the series for math freaks 37

ADVENTURE WRITING
Jason Finch continues with his tutorial 49

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

Printed by Garnett Dickinson Print Limited, Rotherham and London

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.

The contents of this publication including all articles, designs, drawings and programs and all copyright and other intellectual property rights therein belong to Alphavite Publications Limited. All rights conferred by the law of copyright and other intellectual property rights and by virtue of international copyright conventions are specifically reserved to Alphavite Publications Limited and any reproduction requires the prior written consent of the company

© 1991 ISSN 0953-0614

EDITORS COMMENT

Most of you will by now have discovered the new price for CDU. We are very sorry that we have had to make this increase to the cost of your favourite C64 magazine. Like everyone else, we are constantly at the mercy of the general economic climate of the country. However, we do feel that even at the new price of £3.25, CDU still gives outstanding value for money and we will endeavour to continue to keep up the standards.

I hope you like the new style disk sleeve and disk logo. Many of you have stated your preference to having the VOLUME and DISK number on the disk, instead of on the sleeve as in the past. I must say that I agree.

This months issue is full of interesting features which I trust you will find will be of some benefit to you. One of the programs on the disk, namely "Turbo Loader" does not have any text associated with it. This is due to the fact that we received it too late for printing. However, the program is self explanatory so it should not present you with any problems. Please enjoy this 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 be 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.

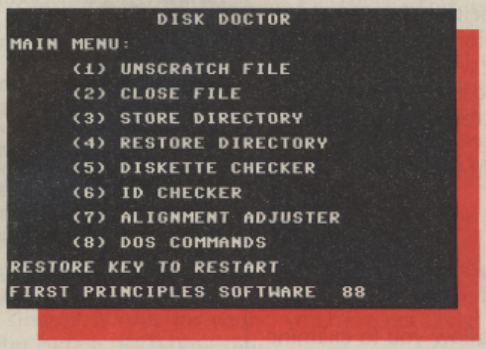
After eight weeks a replacement disk can be supplied from STANLEY PRECISION DATA SYSTEMS LTD for a service charge of £1.00. Return the faulty disk with a cheque or postal order made out to STANLEY PRECISION DATA SYSTEMS LTD and clearly state the issue of CDU that you require. No documentation will be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

DISK DOCTOR

A simple utility to aid Disk Drive users brought to you by FERGAL MOANE



Although the Commodore Disk format is very reliable, it is not immune to human mistakes! It is the aim of this program to try and prevent some of these mishaps which can be potentially very costly. DISK DOCTOR is essentially a simple program, but could save some angry words. An explanation of each item in the main menu is given below.

UNSCRATCH FILE

When you delete a file using the DOS SCRATCH command, it is luckily not erased from the disk. The entry in the directory track corresponding to that file is simply marked as deleted and as such it is ignored by the DOS. Using this option allows you to recover a deleted file, provided you give the correct name. Unscratch file is most effective when used immediately after the delete command has been issued. This is because the undeleted file might be overwritten by other material, saved in the space occupied by the old file.

CLOSE FILE

If you use your C64 for wordprocessing, you will no doubt have experienced the loss of a file, usually that huge 10,000 word masterpiece! Perhaps a disk error or power cut happened when you were saving the file, and only part of the file was saved. When you try to load the partially complete file you get error 60: WRITE FILE OPEN. This option allows the file to be closed properly and to be loaded back into the wordprocessor. It should work with all file types, but it is most effective with **SEQUENTIAL FILES.**

STORE/RESTORE DIRECTORY

You will probably be aware of the quick format command available on Commodore drives. It takes the form OPEN15,8,15,"NEW:diskname". Note how the ID is left out. Its function is to clear out the directory of an already formatted disk and is far faster than reformatting the entire disk. But what if you have cleared a disk with valuable files on it? The directory has been cleared so the disk is effectively blank. These two options provide a limited but useful archival system for your disks. Store Directory will read the directory track of a disk and allow you to give it a name for storage on another disk. When you accidentally blank a disk, it is a simple matter of using RESTORE DIRECTORY to put the correct directory track back on the blank disk. Your files are resurrected! My advice is to use one disk for storage of all the directory tracks of your most important disks. Then when an accidental format occurs, you can refer to the archive disk for their directory tracks. Remember to note the names that you give the files for each disk. This method will also recover from corruptions of the directory track (error 71:Directory error). Restoring an incorrect directory would destroy the contents of the disk. Please note however that this option will NOT recover from a full format (including ID). With the long format command, the disk is actually erased and cannot be recovered.

DISKETTE CHECKER

This option allows you to examine the performance of the drive, the condition of a whole disk or track or a disk. DISK DOCTOR will check the disk and report back the number of any error that occurs. The main use of this option is to check to see if disks are actually corrupt and to find the location of the error. DISK DOCTOR will try and allocate any bad blocks, making sure that valuable data is not put into these dodgy sectors. Note that in a performance test the disk will be destroyed, so please be careful.

ID CHECKER

When a disk is ID formatted, the ID number that you give in the format command is written on every sector of the disk. If this track ID number is different from the disk ID, untold problems can occur. This is usually caused by swapping disks between a write to the disk and without initialising the new disk. This command will show up any incorrect IDs. Skilled people could use a **DISK**

EDITOR to change the track ID.

ALIGNMENT ADJUSTER

This is potentially the most useful command that DISK DOCTOR provides. It makes adjustments to how the disk drive handles read errors. It will seek using half tracks, so if the disk is out of alignment with the disk head, chances of finding the correct track are immediately doubled. You may also specify the number of times that the disk drive will try and read a particular track. Again, the more attempts the drive makes, the better the chances are of reading the disk correctly. This option will also remove the irritating 'WOODPECKER' noise when the drive head hits the stop after a read error. Choosing this option will reset the 64, allowing you to attempt to load your program. I have had considerable success in loading misaligned or corrupted disks but this is not a miracle cure for clinically dead disks!

DOS COMMANDS

Finally, this was simply included to allow you to examine the directory, issue DOS commands and to check the error channel of the drive. The RESTORE key will return you to the main menu of DISK DOCTOR at any time. Good luck in dealing with bad-disk blues!

TIMEWORKS SOFTWARE SALE

	RRP	OFFER
PARTNER 128 (accessory pack cartridge)	69.95	24.95
PARTNER 64 (accessory pack cartridge)	59.95	19.95
SWIFTCALC 128 (spreadsheet with Sideways)	69.95	24.95
SWIFTCALC 64 (spreadsheet with Sideways)	49.95	19.95
WORDWRITER 128 (word processor with spell checker)	69.95	24.95
WORDWRITER 64 (word processor with spell checker)	39.95	24.95
SIDEWAYS 64/128 (Sideways print utility)	29.95	9.95

VALUETIME PRODUCTS

These C64 disk software products normally retail at £9.95.
Offer price £4.95 each or any 3 for £12.

The Musician (3 different programs to compose and play music)
The Artist (3 programs to create multicolour graphics, print your artwork on most popular printers, design sprites)
Disk Drive Manager (load your programs up to 5 times faster, add new disk commands, backup your disks)
The Entertainer (3 fast action arcade games)
The Entertainer 2 (3 challenging strategy games)
The Entertainer 3 (3 fast-paced action/adventure games)
The Educator (improve your typing skills, learn algebra and improve your numeric skills by playing games)
The Educator 2 (for children aged 5-10. An entertaining way to learn to tell the time, spell and use the traffic lights)
Electronic Cheque Book (organise, classify and record your chequebook transactions)
The Home Manager (Wordprocessor, Database, Calculator and Memo Pad)
Word Wizard 64/128 (spelling game with speech)

DTBS, 18 Norwich Avenue, Rochdale, Lancs. OL11 5JZ
Tel: 0706 524304
VISA
Send SAE for descriptive list

Note: DT Books and Software (DTBS) is the new trading name of Adamsoft.

OFFICIAL DEALER FOR ☆ COMMODORE ☆ STAR ☆

COMPUTERS

Amiga 1500 comprising : B2000+1084S Monitor+Twin Floppies+ The Works!+Deluxe Paint 3+3 Games	£938.00
Amiga 500 SCREEN GEMS PACK comprising : TV Modulator, 4 Games & Deluxe Paint 2	£379.00
Amiga 500 Basic Pack comprising : TV Modulator, Mouse, Workbench 1.3, Extras Disk, Tutorial, Manuals	£319.00
Commodore 64C NIGHT MOVES Pack comprising : Cassette Recorder, 8 Games & 2 Joysticks	£149.95
Commodore PC Starter Packs(AS SEEN ON TV)	PHONE.

PRINTERS

Citizen 120D+Parallel or Commodore	£135.00
Star LC-10 Parallel	£159.00
Star LC-200 Colour Parallel	£209.00
Star LC 24-10 24 pin Multi-font170/57cps	£209.00
Star LC 24-200 24 pin Multi-font 200/67 cps	£259.00
Star LC 24-200 Colour Version of above	£299.00
Okimate 20 Print heads/Ribbons/Paper	PHONE.

MONITORS

Commodore 1084S Stereo Colour Mon	£259.00
Philips 8833-II Stereo Colour Monitor	£249.00
Philips 7502 Green Screen Monitor	£ 99.00

MISCELLANEOUS

Amiga 512K RAM/Clock Exp for A500.	Free p&p.	£ 39.00
Commodore 1541-II Disk Drive		£129.00
Commodore C2N Data Recorder.	p&p. £2	£ 29.95
External 3.5" Disk Drive for Amiga	p&p. £2	£ 59.95
Power Supply for C64	p&p. £1	£ 26.95
Super-G Parallel Interface for C64/128. (Allows use of standard parallel printers on C 64/128)	p&p. £1	£ 34.95
User Port Parallel Printer Cable	p&p, £1	£ 15.99

SOFTWARE

Superbase 64 or 128	Free p&p.	£ 39.95
Superscript 64 or 128	Free p&p.	£ 39.95
Tasword 64 40/80 Col WP-Tape or Disk.	Free p&p.	£ 24.95
Mini Office II for 64 - Disk	Free p&p.	£ 19.95

ALL PRICES ARE INCLUSIVE OF VAT AT 17.5% CARRIAGE £5 (EXPRESS £10)
Prices subject to change without notice

Delta Pi Software Ltd

VISA

8 Ruswarp Lane, WHITBY, N. Yorks YO21 1ND
TEL/FAX: 0947 600065 (9am - 7pm)

MasterCard

MANAGING STOCKS

A stock management program for C128 users

F.A.LATEGAN

Every serious Commodore Disk User has some stock. Even the others have some experience of the stock market if they have played "THE FIRST MILLION", featured on the February 1990 disk of CDU. If you own more than one stock in your portfolio, it becomes soon impossible to really keep track of what is happening to your stocks.

This program, SOLSTOCK.128 was developed for people like you. Suffer or Learn (SOL) Stocks is for the more serious people.

To Load, switch on the computer with the 40/80 display key in the down position. Type RUN"SOLSTOCK.128" and press RETURN. When prompted for the date, type it in any format which suits you, since it is only for your information on the print-out. After hitting 'RETURN' the computer then loads "SOLSTOCKS.12.MC", the machine code for the pull-down menus. I used MENU MAKER 128 by NICK GREGORY, published in a previous issue of CDU. Now that the system is booted, it is time to set up the stocks.

TUTORIAL

Use the cursor keys to move to NEW INFO. Press the down arrow or RETURN to select it. Press RETURN again to select NEW STOCK DATA. For our example, let's use the following data.

JJ ENTERPRISES INC	**300 Stocks**
ABC INSURANCE	**1000 Stocks**
COMMODORE INC	**750 Stocks**
CDU	**1500 Stocks**

Type XXX to end the input.

Select first the CHANGE menu, then NEW PRICES. You will have to retype the date because, after the first time you don't have to retype all the stock info. Just reload the last stock data, and change the prices. (By loading the data, you also load the old date). Type the following prices;

JJ ENTERPRISES INC	**7.50**
ABC INSURANCE	**0.75**
COMMODORE INC	**25.78**
CDU	**57.98**

To see what the stocks are worth in total, select OUTPUT and then SCREEN to see it on the screen, or PRINTER to have a hardcopy. (Your printer must be device number 4. I have tested it on the CBM801,803 and the Seikosha SP-180VC). Now for the other options.

DISK - Lets you Save, Load or Replace your stock.

OUTPUT - This is used to view the portfolio.

CHANGE - This is used to type in new prices, to change a price (to correct a typing mistake for instance without retyping all the prices), to change the number of stocks, to change a stock name, to change the date, to add another stock name and to delete a stock name.

NEW INFO - This is used to compile a new stock portfolio.

REORDER - This is used to order the stock names in alphabetical order.

HELP - Gives a short help on all of the menu options.

EXIT - This is used to leave the program. WARNING: if you have not saved you stock data, it will be lost.

ADVANCED TUTORIAL

Continue from the tutorial with the same data. If you have erased it, enter it again as described in TUTORIAL.

First of all, you have bought stock in a new company, ROBOTICS ALL. Select CHANGE, and then ADD STOCK. Enter the name, and the number of stock you have decided to buy (250 in this example).

Next, you have sold all stock of ABC INSURANCE. Select CHANGE, and then DELETE STOCK. When ABC INSURANCE is displayed, type 'D' to delete it. You have also bought an additional 500 stocks of CDU. Select CHANGE, and then CHANGE NUMBER. When CDU appears, type 2000, otherwise just type RETURN.

At this moment, ROBOTICS ALL's price is still 0 since you have not yet typed in anything else. To repair this, select CHANGE, followed by CHANGE PRICE. Type RETURN until you get to ROBOTICS ALL. Enter the price as 18.3.

Now select REORDER, and then ALPHABETICAL ORDER to order your stock alphabetically. Select OUTPUT, followed by SCREEN.

Now select DISK, and then SAVE DATA to save the configuration. Reload it using the LOAD DATA on the same menu. Select CHANGE, followed by NEW PRICES. Type the following prices;

ABC INSURANCE	**0.5**
CDU	**63.03**
COMMODORE INC	**27.33**
ROBOTICS ALL	**12.81**

You are now ready to attack the LONDON STOCK EXCHANGE, and remember to look out for future SUFFER or LEARN programs.

DISK SLEEVE PRINT

A disk sleeve utility for those of us with NORMAL printers by STEVEN BURGESS

If your disk-boxes are as disorganised as mine are then, like me, every time you want to find a program you probably have to wade through every one of your disks and directory each one until you find your program which, almost invariably, is on the very last disk you come to. Enter the answer.

This program will let you make a brand spanking new

```
DISK SLEEVE MAKER

by STEVEN BURGESS

for COMMODORE DISK USER

MENU

1..ENTER PROGRAMS
2..PRINT SLEEVE
3..END

SELECT OPTION
```

disk sleeve for your disks. But unlike most ordinary plain white sleeves, this one is going to be different. On the front of it is going to be a list of the main programs on your disks. So, when you want to find the disk with your latest blockbusting computer program on it you simply flick through your box and, hey presto, you have your disk. The advert over, let's get down to business.

THE PROGRAM

When the program has loaded, and it is not terribly long so it should only take about half a minute, you are presented with a menu.

The only option you can select at this moment is the option ENTER PROGRAM NAMES. Before selecting it, however, you must make sure that you have the disk, which you wish to create a sleeve for, in the drive. Once done and drive switched on select the option.

The screen will clear and then the name of the disk will appear on the screen. Then, one by one, the names of all the files will be printed and you will be asked which ones you wish to be printed on the sleeve. Be careful with which ones you chose, though, because even files that have been deleted will appear.

Once the entire disk directory has been on the screen and you have selected which files will be on the sleeve you are asked to wait while the computer formats the data. This is so that it will fit into the sleeve design without over- or underflowing. You are then returned to the menu.

Selecting the option PRINT OUT SLEEVE will cause the sleeve to printout straight away so be sure to have the printer set up and some paper in it. The design uses one full piece of printer paper.

ALONG THE DOTTED LINE

Once the sleeve has been printed and you have removed the paper from the printer and marvelled sufficiently at the sheer brilliance of this program, it is time to cut it out. You cut along the outer edge of the design. And then you fold along the remaining lines.

Then all that remains is for you to glue it together and pop in your disk and then start all over again.

STRATEGY ADVENTURE

C64 disks only

INFOCOM		HILLS FAR	£19.95
BALLYHOO	£14.95	OVERRUN	£24.95
BUREAU CRAZY C128	£14.95	PANZER STRIKE	£24.95
HITCHIKERS GUIDE	£9.95	PHANTASIE III	£24.95
LEATHER GODDESS	£9.95	POOL OF RADIANCE	£24.95
		PRESIDENT ELECT	£14.95
INTERSTEL		QUESTRON I	£19.95
EMPIRE	£29.95	QUESTRON II	£19.95
		ROADWAR EUROPA	£19.95
LUCASFILM		SECRET OF SILVERBLADES	£24.95
ZAK MC KRACKEN	£14.95	SIX-GUN SHOOTOUT	£12.95
		STORM ACROSS EUROPE	£24.95
MICROLEAGUE		TYPHOON OF STEEL	£24.95
MICROLEAGUE BASEBALL	£24.95	WAR OF THE LANCE	£24.95
MICROLEAGUE FOOTBALL	£24.95	WARGAME CONSTR SET	£19.95
MICROLEAGUE WRESTLING	£19.95		
		SUBLOGIC	
SSI		FLIGHT SIMULATOR II	£24.95
50 MISSION CRUSH	£12.95	NIGHTMISSION PINBALL	£12.95
BATTLES OF NAPOLEON	£24.95	STEALTH MISSION	£24.95
BUCK ROGERS	£24.95		
CHAMPIONS OF KRYNN	£24.95	**TELARIUM**	
CURSE OF AZURE BONDS	£24.95	DRAGON WORLD	£9.95
DRAGON STRIKE	£24.95		
FORTRESS	£12.95	**WIZARD**	
GEOPOLITIQUE 1990	£12.95	SUPERSTAR ICE HOCKEY	£14.95

Hint Books £7.95 Each:

BARDS TALE I, II or III, BUCK ROGERS, CHAMPIONS OF KRYNN, CHAOS STRIKES BACK, CURSE OF AZURE BONDS, DRAGON WARS, DRAKKHEN, DUNGEON MASTER, ELITE, ELVIRA, INDIANA JONES L.C. ADV, MANIAC MANSION, MIGHT & MAGIC I OR II, NEUROMANCER, POOL OF RADIANCE, SECRET OF SILVER BLADES, STAR FLIGHT, ULTIMA III, IV, V or VI, WASTELAND, ZAK MC KRACKEN.

Mail order only. Please allow 28 days for delivery.
Please make cheques and postal orders payable to CINTRONICS LTD.
Free post and packaging within the UK. Europe add £2 per item. Overseas add £4 per item.

CINTRONICS LTD.
16 Connaught Street,
London W2 2AG

ADVENTURE HELPLINE

The ASTRODUS AFFAIR gets another airing by JASON FINCH

So the merry month of May is upon us and I will be giving nothing away if I tell you that by the end of this article you will be on the very edge of being able to complete the excellent adventure published by CDU last year, THE ASTRODUS AFFAIR. Last month I finished by explaining how to repair the hole that was causing a few problems and this month I shall explain where you go from there, including the problem of the stabalization chamber and how to get from the top level to the bottom. I shall leave telling you how to replace the fuses in the crafts drive so that you can fulfil all of the conditions that allow you to finish the adventure until next month. Those being having the generator on, the fuses in the engine room repaired/replaced, the stabalization chamber 'active', the hole in the corridor repaired, the button pressed for navigation and finally the correct use of the code 'XX2V'. The format is the same as that for previous articles, a general outline of the method for obtaining the solution to a certain problem, followed by a section detailing the systematic way about things. Also in June I shall provide you with a full map of the Astrodus - Editor permitting, that is! So, let the Astrodus slide once again under the microscope...

THE WALL OF OXYGEN

The way into the stabalization chamber is blocked by a wall of oxygen that is leaking from a ruptured pipe. Of course you won't have discovered the chamber yet if you haven't solved this problem, so it is probably wise to cover this first. Once you have repaired the hole, though, you need to set up some situations ready for later on. First you must drop the globe again and go and switch the power off. This will enable you to return and move the globe to the location immediately east of where it was. Then go and put the power back on. The globe will be free to move around so long as you keep out of locations 12 and 15. If you don't then not only will the globe be immobilised, so will your attempts at finishing the adventure! You need to then go to cross-section B and collect the infra-red grenade. It is not intended for consumption by you or drygars so don't try feeding it to one of them! It will only result in you being covered by "steaming drygar meat" with the rest of the bunch soon making short work of you. The grenade is intended to be thrown at the wall of oxygen in location 4. Then you can return to where you deposited the globe. Skip the next section if you don't require further assistance with that problem.

BREAKING THE BARRIER

Once the hole is repaired, DROP the GLOBE and go WEST and then SOUTH. Turn the power off by PRESSing A51X. Return NORTH and then EAST. GET the GLOBE and go EAST again. Now DROP the GLOBE and go WEST, WEST and SOUTH. PRESS A51X a final time to switch the power back on and go NORTH, EAST and EAST again. The globe is free to move elsewhere. Now go SOUTH and DOWN the staircase. GET the GRENADE and be careful! Go UP, NORTH, EAST, UP another flight of stairs and then SOUTH, and SOUTH again to location 4. THROW the GRENADE and the oxygen will be ignited. You are then free to go east to the stabalization chamber. However, before you do that, return to the globe by moving, NORTH, NORTH, DOWN and WEST.

THE CHAMBER PROBLEM

I bet you haven't got the rope, have you? Well first things first - you must get the rope which is probably still in the supplies room, return to the globe, take it and then tie the rope to it. You can then go the stabalization chamber. The globe will be attracted to a point above the silver disc at the base of the chamber. Because the rope is tied to it, and the rope dangles against the side of the platform on which you are standing, you can simply climb down it to the bottom. That way you avoid such pleasant messages as "the floor ... races cheerfully towards you with a squelchy thud"! To find out exactly what to do, read on; otherwise skip to the section labelled CONSOLE QUERIES.

GETTING TO THE BOTTOM

If you don't have the rope, go to the supplies room by moving SOUTH, DOWN and WEST. GET the ROPE and go back EAST, UP and NORTH. Now GET the GLOBE and TIE the ROPE TO the

GLOBE. Now make your way EAST, UP the staircase and then SOUTH and SOUTH again. Seeing as how you are free to go east now, do just that - GO EAST. The globe should be attracted over the disc and you will be told that rather than dangle precariously from the roof of the chamber, you drop off onto the platform. The observant will notice that "the rope dangles down..., swinging idly against the platform". If that isn't a cue to enter DOWN, then I don't know what is!

CONSOLE QUERIES

Once at the bottom you should be informed that there is a black console set into the wall. Upon examination you should be told that all four sides are raised away from the wall. The perfectly natural thing to do (?) would be to pull the edges and wait for it to plop into your hand. So do it! But the writer of the adventure has played on your curiosity here because everyone will want to know what the little red button does. If you press it then you will find out just why being pressed against a wall with the floor rising towards the ceiling isn't really a lot of fun. Make sure that you are out of the chamber and back in location 4 before you press the button. You will then notice that the chamber door closes and if you were in location 24 you would be told that the chamber was 'active'. One more piece of the puzzle falls into place. Ignore the next section if you don't want to spoil the fun!

P-U-L-L-!!

EXAMINE the small black CONSOLE and then PULL its EDGES. When it has dropped into your hand, go UP and then WEST. Now PRESS the BUTTON. That was a bit simple wasn't it.

THE END ALREADY

No, not the end of the adventure - the end of this month's offering of help. You have learned a number of things hopefully and next month we will look at the final problems. Incidentally, don't enter location 23 unless you have the laser! More about that in June. All we now have to do is open the safe, retrieve the fuses and put them in the engine room. Then it's hey presto and onto another adventure in this long running quest to provide help. Which would you prefer - Runaway or the Cranmore Diamond Caper? Do please let me know! Have fun trying to solve this month's adventures and until thirty odd days have elapsed, see yer!

BASICS OF BASIC

A series of Basic tutorials designed to make the beginner an expert
JOHN SIMPSON

IMPORTANT ANNOUNCEMENT - Due to unforeseen technical problems, and the fact the your illustrious Editor is a bit of a wally, we inadvertently missed out the start of section two of this series in the MARCH issue of the magazine. Therefore, before this months offering gets underway, we present the missing piece for your enjoyment.

COMMENCEMENT OF PART TWO

Let us begin this session by first writing a small program which will allow us to get the computer to request some input of numbers which we can then manipulate and finally display a result.
Remember to press the return key <CR> after you have typed in each line.

```
10 INPUT"FIRST NUMBER";FIRST <CR>
20 INPUT"SECOND NUMBER";SEC  <CR>
30 LET SUM = FIRST + SEC     <CR>
40 PRINT"THE RESULT IS"SUM   <CR>
50 END
```

Once you have typed in the program, let's refresh our memory of clearing the screen and listing a program from the last session (part 1).

Perform <CLR>... i.e. hold down the shift key and tap the CLR/HOME key. Now type, without a line number, LIST <CR>.

We shall now make the computer execute our program by typing, RUN <CR>.

As soon as you do this the computer will respond by printing on a new line below RUN,

FIRST NUMBER? (and a flashing cursor)

You are being prompted to type in a number. Should you type in a letter, or a series of letters, or even numbers and letters, then the computer will respond with:

?REDO FROM THE START

This shows you that it is able to distinguish between a number and something else (we'll discuss this more closely in a short while).

Should you perform a <CR> without typing a number the computer will fill the variable, FIRST, with the default value of zero, and then continue to the next line of the program, the screen will now display:

SECOND NUMBER? [Cursor]

Type in the second number <CR>, and the computer will respond by displaying on the screen:

THE RESULT IS [first + sec]

READY

Let us examine more closely what has taken place. In line 10 we used the Keyword INPUT. This is a command to tell the computer to expect the user to enter something from the keyboard, and to store it in the variable labelled FIRST.

You will have noticed that the keyword INPUT can also be used to print information to the screen. The information, like the PRINT command's output, being enclosed within quotes "...". Make note of the semicolon (;) following the final quote, and before the label, FIRST. This is important. Without it the computer will display a syntax error.

THE SCREEN EDITOR

The cursor can be directed around the screen by using a combination of the SHIFT key and the cursor keys. The cursor keys are the two keys next to the right SHIFT key and marked CRSR with up/down left/right arrows. To move the cursor down the screen use the cursor 'up/down' key, and to move up the screen hold down a SHIFT key together with the same key. To move the cursor right use the cursor 'left/right' key, and to move left hold down a SHIFT key together with the same key.

To edit a part of the program move the cursor up to line 10, and along the line to the F in FIRST. Now tap the INST/DEL key at top right, and the label FIRST will shift to the left, deleting the semicolon. Press <CR>.

You have just used the screen editor. Whenever you do use this editor to make changes to a line, always remember to press return to enter the changes into the program.

We have just deleted a piece of program code, namely the semicolon, but we can just as easily insert code by holding down the SHIFT Key and tapping the INST/DEL key which will shift the code to the right opening up a space for you to enter new code. Move back to the F in FIRST and try this, then type the semicolon back in and press return.

If you run through the deletion of the semicolon once more, then RUN the program (first ensuring that you move the cursor back down the screen to a clear line). The computer will respond by displaying a syntax error message, however, unlike the message which was printed in part one, where we used a DIRECT COMMAND, this error will actually print the line number in which the error occurs. In our case the message will read:

? SYNTAX ERROR IN 10
READY

You can use the screen editor quite freely to edit a program, i.e. add code, and delete code. If you find the need to delete a whole line, all that is required is to type the line number, on a new line, followed by <CR>. Let's try this by deleting line 50 in our program:

50 <CR>

Now LIST the program and hey presto! line 50 has gone. If you decide that you want to get rid of the entire program, this can be done by typing a Direct Command (i.e. no line number), namely:

NEW <CR>

Try this. Now we have just deleted our program. This is a useful command, but one fraught with danger for when you do NEW a program, it has gone for good! Try LIST and you will just receive: READY. The program has gone - with no way to retrieve it.

STRING INPUT

We have just seen that we can easily force the user to input numbers, albeit only two at a time in the foregoing program, although if you wanted to place more INPUT commands plus variable labels you can do so.

We touched briefly on the term strings, earlier in Part 1, but now we shall look into them in more detail. A string is a group of alphanumeric characters contained within a pair of quotes:

"THIS IS A STRING"
"AND SO TO IS THIS ABCD1234"
"1234567890" is also a string

How long can a string be? Well CBM basic does allow for up to 256 characters, but their is a limitation on a basic line length of 80 characters, and this will include the line number and statements, so for the time being we will restrict ourselves to 40 characters. Later we will

discuss how to overcome this restriction.

To distinguish between a numerical input, as used earlier, and a string input the dollar sign ($) is utilised. For the numerical input we simply type in a label, such as we did earlier with FIRST, SEC, and SUM. Simply by having no 'prefix', the computer understands that it is dealing with numbers. The string input, however, is prefixed with a dollar ($) sign. Right, let's write a small program which will demonstrate this by imputing string information:

```
10 INPUT"YOUR NAME PLEASE"; N$
20 INPUT"YOUR ADDRESS PLEASE";A$
30 INPUT"YOUR TELEPHONE NUMBER PLEASE";T$
40 PRINT CHR$(147)
50 PRINT N$
60 PRINT A$
70 PRINT T$
80 END
```

When you RUN this program you will be asked to type in your name, spaces are allowed, and as before when you have typed in your input this is entered by pressing <CR>. Next you will be asked to type in your address, followed by your telephone number.

Before we examine line 40 you may have received a message which stated:

```
?EXTRA IGNORED
```

If you did, then by examining the printed output you will see that everything that followed the character which preceded the comma which you typed has not been printed. What this tells us, loosely, is that CBM Basic will not accept a comma emplaced within a string, and will thus terminate any output including the comma onwards.

Line 40 is a special command, CHR$(147), which tells the computer to clear the screen - the same as you did when you used the SHIFT key and INST/DEL key. There are a lot of numbers we can use within the brackets of CHR$(n) to tell the computer to do many things, however, that is for later.

The computer will create two storage areas, one is used for variables and constants, and the other is used for holding strings.

With strings, the same as variables, the maximum length of a label is two characters although more can be used for information purposes. I used N$ for the name, this could just as easily have been NAME$, however, with NA being the significant portion of the label. Note also the prefix of $.
to let the computer know it is dealing with a string of characters.

ANOTHER GLANCE AT VARIABLES AND STRINGS

As well as using INPUT statements to assign a variable or a string to a label, they can also be assigned using the LET statement:

```
10 LET A = 10
20 LET B = 1000
30 LET A$ = "THIS IS A STRING"
40 LET B$ = "ABC 123X": REM - A CAR NUMBER PLATE.
```

If somewhere else in the program you state:

```
100 PRINT B / A
110 PRINT A$
120 PRINT B$
```

I'm sure you can work out the result! (Remember that the (/) symbol = division).

MULTIPLE STATEMENTS

On line 40 in the program lines above and after the string "ABC 123X", and before the REMark I placed a colon (:). The purpose of the colon is to separate statements which are on the same line, it's as though you have started a new line, but without a line number. It means the same as this:

```
40 LET B$ = "ABC 123X"
50 REM  - A CAR NUMBER PLATE
```

We are allowed to place as many statements as we can upon each line so long as they are separated by colons. This does have the advantage of freeing up some memory, but it often has the disadvantage of making programs more difficult to follow, and, more importantly, there are some statements which do require a line all to themselves. We will discover these particular instances as we come to them in the lessons.
However, to round off multiple statements here is an example of multiple statement lines of code:

```
10 LET X = 34 : LET Y = 72 : PRINT X : PRINT Y
20 LET N$ = "YOUR NAME" : LET A$ = "YOUR ADDRESS"
30 INPUT N1$ : INPUT A1$ : PRINT N1$ : PRINT A1$
```

OUR FIRST LOOP

Ofttimes it is extremely useful to get the program to run around a loop - as an example of a need for this is a delay loop, where you wish to slow down the program execution for some reason. There are two basic methods available and the first of these is called a FOR...NEXT loop.

Type NEW then <CLR> (i.e. clear the screen), then type in the following program:

```
10 PRINT "HELLO"
20 FOR X = 1 TO 1000
25 REM - A DUMMY LINE TO DEMONSTRATE THE
```

```
LOOP
    30 NEXT X
    40 PRINT "WORLD"
    50 END
```

When you run this program, first HELLO is printed, then a lengthy delay where, on the following line, WORLD is printed.

What we have done here is to set up a variable called X and have given it an initial value of 1. FOR X = 1... We have also established that the value of X will not increment above 1000. ...TO 1000. When the program steps through to the line 30, NEXT X, it will check the value of X and if it is not greater than 1000 then it will jump backwards to the statement following the FOR ... statement, in our case, line 25 which was a REM placed there to create a line which actually does nothing of importance. In a more complex program the distance between the FOR and the NEXT could be quite large with several statements being executed in between. Let's have a look at a simple example of this:

```
10 LET A$ = "THE VALUE OF X IS"
20 FOR X = 1 TO 10
30 PRINT A$ X
40 FOR Y = 1 TO 1000:NEXT Y
50 NEXT X
60 END
```

Now if you RUN this program it will print ten lines, each one telling you what the current value of X is. You will also observe that we have place one FOR...NEXT loop inside another, creating two statements on one line. This is known as 'nesting'. The outer X loop, iterates ten times printing the message of A$, and the inner Y loop acts as a delay, slowing things down. The outer loop will iterate once, then the inner loop will iterate 1000 times before the outer loop iterates again. So the inner loop will iterate 10 x 1000 times.

The other basic method for a controlled loop is to set up a variable, increment the variable and test it to find out if it has reached a predetermined level, if it has then exit from the loop. Now if this does sound similar to the FOR...NEXT loop, then in essence it is, except with this loop we must determine what action to take upon each iteration. This will require decision making.

MAKING DECISIONS

The more common method of decision making is to use the command statements IF...THEN. In other words, IF this is true, THEN do something, otherwise do something else. For example, START - IF X = 10 THEN X = 0 AND END, OTHERWISE X = X + 1 GO BACK TO START.

We will now create another, but smaller, delay loop to demonstrate the technique:

```
10 LET I = 1
20 PRINT I
30 LET I = I + 1
40 IF I =< 10 THEN 20
50 PRINT "DONE"
60 END
```

If you run this program it will print the digits from 1 to 10, followed with DONE. But what is actually taking place?

line 10 initialises a variable labelled I to equal 1.

line 20 immediately prints its value.

line 30 increments I by one.

line 40 evaluates whether I is equal to, or less than, ten (the symbol =< means equal to or less than), and if it is will send the computer back to line 20 (THEN 20 means goto 20). The computer will loop between lines 20 and 40 until the value of I is greater than ten where it will not goto line 20, but drop through to line 50, which simply prints DONE, and the program terminates.

Whether we use the FOR... NEXT loop, or a decision making branch, IF... THEN will very much depend upon the program we are working on. This is where experience and knowledge combine, knowing which method best to use, and this you will gain as we progress through our subject lessons.

FINALLY, LET'S GET RID OF LET

I have been using the word LET each time I assigned a value, or a string to a label. Although LET is a keyword, it is optional whether you wish to use it. Sometimes it can make the understanding of a program more clear, but mostly it just uses up a little more memory and execution time. It is up to you whether you wish to use it, but we can state:

```
10 X = 10
20 A$ = "LET GO"
30 A = A+B/C*D
40 FIRST = 1: SEC = 2
```

SUMMARY FOR PART TWO

1. In this session we discovered that by using the keyword INPUT the computer can request the user to enter numbers which can then be used to either initialise variables or perform mathematical manipulations upon them.

2. We also discovered that the computer can judge the difference between numbers and strings of alphanumeric characters.

3. We saw that we can use the facilities of the screen editor to make changes to our code, either deleting and/or inserting the changes which may be required.

4. We looked into the INPUT of character strings, how we can program the computer to request information such as names and addresses, telephone numbers, &etc, by using the prefix $ character. We also noted the limitation of using the comma within a string; where the text following the comma will be ignored.

5. We also noted that strings can be up to 256 characters in length, but, because of line restrictions, we have temporary limited the length to 40 characters until later in the series where we will discover the method to overcome this restriction.

6. Part of this sessions discussion showed that it is also possible to initialise number variables as well as strings, by using the LET keyword.

7. By using a colon (:) we can separate multiple statements on a line, but there are certain restrictions when using this option.

8. We looked at two methods which can be used for controlling program loops. The first is the use of the FOR...NEXT statement, and the other is by using a variable as a counter in conjunction with the decision statement IF...THEN.

9. Finally we saw that the LET statement is optional and can be omitted.

That's just about it. We apologise for missing out all of the above, we hope that it did not confuse you too much. Before getting on with this months tutorial proper, we have added a little footnote to finish off MARCH's discussion on ARRAYS.

As a footnote to the dimensioning of arrays we are allowed to dimension several arrays on one line separated by commas, for example:

10 DIM NAME$(10),SCORES(10),AA(100),BB(10,6),NN$(2,2) will separately dimension NAME$ - SCORES - AA - BB - NN$

Finally, we are now up todate with all the tutorial text. We can now continue where we left off last month. (ie; PART FOUR).

NB. Some of the programs and program segments in the tutorials are also listed on this months disk. It is suggested that you do write the smaller segments yourself, but if you should find things a little tricky then you can load in from the disk. The programs which are on the disk will be preceded by their save name just before the tutorial listing. These are BB.EG#<n>, where n is the example number.

COLOUR, FORMAT, AND GRAPHICS

We shall now look at methods to control character colour, cursor positioning, and formatting. The first two, character colour and cursor positioning use a method of embedding within text strings the necessary 'shorthand' commands to perform such functions.

Formatting has the use of two possible keywords, namely, TAB() and SPC(), we shall deal with these in a moment. However, there are other methods for positioning the cursor. One of which which uses a machine code routine within the operating system know as 'PLOT', and yet another method which will allow us to save the current cursor position before we move it to another screen location, then retrieve it to move back again. We can also use the embedded cursor controls held in strings which can then be manipulated by 'cutting up' the strings. We shall deal with these methods a little later on in the series.

By using the special graphic keys on the 64 will allow us to produce very useful graphic features, such as boxes, or windows within which we can place our text, or borders, either plain or fancy, to surround our text, and even quite interesting pictures to enhance our screens.

EMBEDDED COMMANDS

The thing about using embedded commands within text strings is that I will have to use abbreviations to let you know what it is I want you to do. Remember last month I told you about performing the screen clear by using the SHIFT and CLR/HOME keys in conjunction, i.e.

10 PRINT "[SHIFT CLR/HOME]CLEARING THE SCREEN"

using this method printed a reversed heart symbol. Here follows a list of the abbreviations for the embedded commands which I shall be using throughout the series.

ABBREV	KEYPRESS REQUIRED
[CR]	CRSR LEFT/RIGHT KEY
[CL]	SHIFT AND CRSR LEFT/RIGHT KEYS
[CD]	CRSR UP/DOWN KEY
[CU]	SHIFT AND CRSR UP/DOWN KEYS

A number following the cursor controls means press the key that particular number of times. E.G. [CR5] = CRSR LEFT/RIGHT key five times.

[HOME]	CLR/HOME KEY
[CLR]	SHIFT AND CLR/HOME KEYS
[RVSON]	CTRL AND 9 KEYS
[RVSOFF]	CTRL AND 0 KEYS
[INST]	INST/DEL KEY
[DEL]	SHIFT AND INST/DEL KEYS
[PI]	SHIFT AND UP ARROW KEYS
[LARROW]	LEFT ARROW KEY
[UPARROW]	UP ARROW KEY
[SPAC]	SPACE-BAR

A number following the space bar means press the space-bar that number of times. E.G. [SPAC10] = Space-bar ten times, creating ten blank spaces.

COLOURS

[BLACK]	CTRL AND 1 KEYS
[WHITE]	CTRL AND 2 KEYS
[RED]	CTRL AND 3 KEYS
[CYAN]	CTRL AND 4 KEYS
[PURPLE]	CTRL AND 5 KEYS
[GREEN]	CTRL AND 6 KEYS
[BLUE]	CTRL AND 7 KEYS
[YELLOW]	CTRL AND 8 KEYS
[ORANGE]	CBM AND 1 KEYS
[BROWN]	CBM AND 2 KEYS
[LRED]	CBM AND 3 KEYS
[GREY1]	CBM AND 4 KEYS
[GREY2]	CBM AND 5 KEYS
[LGREEN]	CBM AND 6 KEYS
[LBLUE]	CBM AND 7 KEYS
[GREY3]	CBM AND 8 KEYS

GRAPHIC CHARACTERS

[C/LETTER]	CBM AND A LETTER KEY - E.G. [C/A] OR [C/Z]
[S/LETTER]	SHIFT AND A LETTER KEY - E.G. [S/S] OR [S/G]

A number following the C/LETTER OR S/LETTER means press the letter key that particular number of times. E.G. [S/F8] = hold SHIFT down and press the letter F eight times.

Using these abbreviations, or mnemonics (as some like to call them), we can now choose to add colour to our printed output, or move the cursor up, down, left and right, or by using the special graphic symbols create some neat graphic effects. Let's have a look at a few examples of what I mean.

BB.EG#1

```
5 REM *** COLOUR AND CURSOR CHANGES ***
10 PRINT"[CLR][LGREEN]CLEAR THE SCREEN AND COLOURS LIGHT GREEN"
20 PRINT"[CD4][LRED]THIS TEXT HAS MOVED DOWN FOUR LINES"
30 PRINT"[CU3][GREY3]TEXT HAS CHANGED COLOUR AND MOVED UP"
40 PRINT"[CD4]A MULTICOLORED [LRED]W[WHITE]O[GREEN]R[YELLOW]D"
50 PRINT"[LBLUE][CD2]COLOUR BACK TO NORMAL"
```

BB.EG#2

```
5 REM *** A LOOK AT REVERSE VIDEO ***
10 PRINT"[CLR][CD4][LRED][RVSON][SPAC2]A TITLE/MENU BAR[SPAC2][RVSOFF]
20 PRINT"[WHITE][CD]1. MENU ITEM 1"
30 PRINT"2. MENU ITEM 2"
40 PRINT"[CD12][CYAN][RVSON][SPAC]PRESS ANY KEY TO QUIT[SPAC][RVSOFF]"
50 GET A$: IF A$ = "" THEN 50
60 PRINT"[CLR][LBLUE]"
```

In this example I have introduced a new command word in line 50, namely, GET. Let us divert our attention for a moment and examine this more closely; programmers tend to use this key word quite a lot.

A method whereby the computer is able to detect when a user makes a keypress is to employ a system which will continually check the keyboard to see if a key has been pressed. Obviously the computer is doing this constantly from the moment it is switched on. However, we can use the key word GET to find out what was the very last keypress a user might have made and then put this information to our own use. Because the keyboard is all characters (i.e. letters, numbers, and controls), then we need to use a string variable to store the value of last keypress.

In the program above I chose to use the string variable A$. GET will place the last key press into the variable A$. Now if no key has been pressed then A$ is filled with the string equivalent of zero, which is referred to as a 'null' string. As you know by using quotes allows the machine to identify the beginning and the ending of a string of characters. This, " ", for example, means a string of a single space, and this, "Z", means a string which contains only the letter Z. A null string is represented thus, "".

So, line 50 uses GET and places the result in the variable A$. The next command on line 50 is an IF...THEN decision. If A$ = "" THEN 50, in other words if A$ is null then no key has been pressed so go to line 50 and use GET again. The program will loop forever on line 50 until a key has been pressed. Once a key is pressed then A$ will hold a value other than null, therefore, the decision argument becomes false and so the ...THEN part of the argument will not be executed which means the program will 'drop through' line 50 and continue to execute from line 60.

If you consider the foregoing you will realise that A$ will hold the value of the character which was pressed. It readily becomes apparent that you can have a much greater degree of control over what the user receives from what the keypress might have been. For example a few IF...THEN decisions could determine if the key was a "Y" for Yes, or an "N" for No, or whatever. We will be returning to this quite soon.

Back to our examples. Let us now look at what can be achieved using the graphic symbols as well as the embedded commands for colour and cursor.

BB.EG#3

```
5 REM *** GRAPHIC SYMBOLS EXAMPLE ***
10 PRINT"[CLR][CD8][CR7][BLACK][S/O][C/Y24][S/P]"
20 PRINT"[CR7][C/H][YELLOW]THIS COULD BE A DIALOGUE[BLACK][C/N]"
30 PRINT"[CR7][C/H][SPAC24][C/N"
40 PRINT"[CR7][C/H][GREEN][SPAC9]BOX...[SPAC9][
```

```
BLACK][C/N]"
50 PRINT"[CR7][C/H][SPAC24][C/N"
60 PRINT"[CR7][S/L][C/P24][S/@][LBLUE]"
```

At first sight this may seem a little cryptic, but by following the abbreviations, as outlined earlier, it shouldn't prove too much of a hassle. Note also that line 50 is identical to line 30, therefore, instead of typing in line 50 simply cursor up to line 30 and replace the 3 with a 5 (remember to press return, and then move the cursor back down the screen to a blank space ready to type in line 60). When you have finished typing in this example and have RUN it, then the screen should be cleared and a box with the words, THIS COULD BE A DIALOGUE BOX... within it should appear onto the screen.

If you practice using the embedded controls and the graphic symbols by simply 'playing around' with them using the print command you will soon commit all the necessary controls and commands to memory. It really is quite simple.

FORMAT WITH TAB AND SPC

The action of the SPC function is to control the formatting of data as either an output to the screen or to a logical file. However, at this stage we shall only look at the screen function. We can tell the computer how many spaces we require the cursor to move by enclosing the number of spaces required in parenthesis. The number of the argument (spaces) must be in the range from 0 to 255.

```
10 PRINT"WE CAN PRINT HERE";
20 PRINT SPC(10)"AND THERE"
RUN
WE CAN PRINT HERE      AND THERE

READY.
```

From this example you will see that the spaces (10) are counted from the end of HERE to the beginning of AND. Also in this example you will have noted the use of a semi-colon at the end of line 10. This tells the computer not to perform a carriage return after it has finished printing the text of line 10, so, therefore, the next print statement (line 20) will continue printing it's text upon the same line. This will indeed prove to be a very useful facility.

```
10 PRINT"ONE"SPC(10)"TWO"SPC(8)"THREE"
RUN
ONE       TWO       THREE

READY.
```

The action of the TAB function, although it may at first seem the same, moves the cursor to a relative SPC move position on the screen given by a numeric argument in parenthesis but this time starting at the left-most position of the current screen line. Once again the numeric argument is in the range from o to 255.

```
10 PRINT"NAME" TAB(30) "AMOUNT":PRINT
20 PRINT"JOHN" TAB(32) "50"
RUN
NAME                    AMOUNT

JOHN                    50

READY.
```

From this example you can see that the tabulation was from the start of the line. It counted over both NAME, and JOHN. Whereas the SPC command counted from the point where the cursor was currently situated. You can also see that at the end of line 10 I added a second PRINT statement on its own. What this achieves is to print a carriage return, which gives a row space between the line, NAME AMOUNT and the line, JOHN 50.

```
10 PRINT"HELLO"TAB(85)"THERE"
RUN
HELLO

    THERE

READY.
```

That is all for now concerning Tabulation, Embedded Controls, and Graphic keys. You can create your own homework(!), and simply play around with the things we have covered so far. You cannot harm the computer, and if anything should go wrong with a program you may create and the system 'bombs' out, simply switch off, then start again.

POKING AND PEEKING

That subtitle almost sounds as though it could be some exotic chinese dish!

To POKE means to place a value into a memory location. To PEEK means to have a look at a memory location and find out what value it may contain. Of course to either POKE or PEEK a memory location will require you to know the address of the memory location you wish to deal with.

Earlier I explained that the computer has 65,536 bytes of memory, each byte representing a memory location, and, each byte can only hold a number from 0 to 255. The computer's memory is 'mapped' into various areas and segments for better memory management. There are areas in this map which will prove to be of great importance to you as you establish you programming skills. Table 1. sets out a very generalised memory map.

If you have the Commodore 64 User Manual which came with the Commodore when it was purchased, then by turning to page 143 you will find Appendix E, which is a list of the SCREEN DISPLAY CODES. For the time being we shall ignore SET 2 and concentrate on SET 1. You will see that a character is displayed under the SET 1 heading and a number under the POKE heading. What all of this means, in actual fact, is that if we POKE the

value of the character we require, say 1 for the letter A, into an address of the screen area (see Table 1), then that letter will be reproduced onto the screen.

Now if you turn to page 149 of the manual you will discover a pictorial representation of the screen which lists all the memory addresses. You can see that the screen is divided into 1000 character squares in a matrix of 40 squares horizontally by 25 squares vertically. We usually refer to this as X (columns - 40) by Y (rows - 25). The start address, commonly referred to as the Screen Base Address is memory location 1024. Study the screen memory map on page 149 for a moment and all of this becomes quite clear.

So now we shall actually place the letter A onto the screen. Try writing this short program:

```
10 PRINT"[CLR]"
20 POKE 1024,1
```

When you run this you will see that the letter A has been printed into the first character square. So, the POKE statement is saying to the computer take the value of 1, which happens to be the screen code for the letter A, and place it into the 1,024th byte in the computer's memory. Nothing could be more simple, eh?

We can, using that which we have so far learned, do much more than this, let's print a whole string of the letter A. Here we will need a loop to iterate around our POKE statement. Try this:

```
10 FOR X = 0 TO 999
20 POKE 1024 + X,1
30 NEXT X
```

When you run this program you will see that it will quite rapidly fill the screen with letter A's. You can see that in line 20 we simply add the value of X to 1024, thereby incrementing it by one character square each iteration of the loop.

With slight modification we can use the same principle to actually list the entire set of the characters of SET 1. Clear the screen and type in the following short program.

```
10 PRINT"[CLR][CD8]"
20 FOR X = 0 TO 255
30 POKE 1024 + X,X
40 NEXT X
```

Can you work out why the cursor down command in line 10 is there?

After you have RUN this program you will see all the possible characters (256) of the set printed on the screen. The first 128 characters are in normal video and the next 128 characters are identical except they are the reversed video characters. This tells you that when you actually use the RVS/ON command the computer simply adds the value of 128 to the character values in the character string of the PRINT statement. I.E. the letter A = 1 and the reversed letter A = 128+1 = 129.

POKING COLOUR

The screen we have just been looking at is the character screen. To use colour changes we must now access another screen, this is located in a different part of the Memory Map. The Colour Memory Map (or screen) Base Address is located at 55296. Once again this is a matrix of 40 x 25 squares (1000), and you will find a pictorial representation of this located on page 150 of the User Manual. Table 2. lists the values needed to POKE a colour into the Colour Map. So:

```
10 POKE 1024,1
20 POKE 55296,3
```

will print the letter A into the upper left corner square of the screen, and it will be coloured Cyan.

POKING SCREEN AND BORDER COLOURS

Naturally we do not wish to confine ourselves to the Start-up screen and border colours all the time. We might like to change the screen and border colour quite often in a program to add to the aesthetics of things. We might even wish to use the border to indicate something by allowing it to cycle through colours.

There are two memory locations which will alow us to do such things as I have just described. They are, address 53280 for the border colour and address 53281 for the screen colour. Using the colour values as defined in Table 2. we can now POKE a colour to one, or both of these locations.

```
10 POKE 53280,0 : POKE 53281,0
```

Try this and both the border and the screen will turn to black. Add the following line

```
20 FOR X = 0 TO 64 : POKE 53280,X : NEXT X
```

and the border will cycle through all 16 colours four times. Useful if you need to catch the attention of the user of your program.

Using the screen colour in conjunction with character colours can bring your screens alive and make even dull text input programs into something more adventurous.

LOOKING AT BYTES - PEEKING

So far we have only POKEd values into bytes, but now we shall look at PEEKing bytes and how this can assist

us.

When we wish to look into a memory location to find out what value is being held their we use the PEEK statement, or keyword. We must enclose the address inside brackets. For example, PRINT PEEK(53280) will print the current colour of the border. Amending the example border cycle above we could interrupt the cycle on any given colour and print a message:

```
20 FOR X = 0 TO 64 : POKE 53280,X
30 IF PEEK(53280)=247 THEN PRINT"YELLOW"
40 NEXT X
```

You will quickly spot that I have used the value of 147 instead of 7. Why?
Remember, a byte is eight bits, xxxx xxxx which is four low bits and four high bits looking from right to left. The four low bits can be in the range from 0 to 15, and the four high bits can be in the range from 16 to 240. Without going into the reason just why, the high four bits are almost always set to 240, no matter what number you may poke into the byte. For example you could POKE 53280,129 - the byte would look like this, 1000 0001, but faster than you can say "BLA" the byte will be changed to 1111 0001, which means the colour of the border will be white, because the computer only uses the low order bits - the right-most four bits. After saying all that what this means is if you want to PEEK the border or screen colours within a decisional IF...THEN situation, remember to add 240 to the number of the colour you are looking for. Phew!

Actually this is not good programming! Because we only seek a number in the range 0 to 15 we should perform what is called a logical AND (the code would look like this - PEEK(53280)AND15), which isolates and reads only the four low bits of the byte. However, this is racing ahead of ourselves on our learning curve. But we will return to this subject.

There is a very useful memory location at 197 (see Table 1. ZERO PAGE). This location has been set aside by the operating system to use as a storage byte for the last keypress a user may have made. In the February 1991 edition of CDU an article appeared entitled THE 64s KEYBOARD, if you refer to this article there is a complete description of this function, but, more importantly to us at the moment, there is a table within the article which outlines the numerical value held in location 197 and its key equivalent (they are not the same as screen codes). If you plan to use this location within your programs (and I'm sure you will eventually), you would be well advised to keep the table for handy reference.

So, how do we use location 197? Let us write a small program which will test for key input and then execute a print statement when it receives the correct key:

BB.EG#4

```
5 REM *** A PEEK EXAMPLE ***
10 PRINT"[CLR]PLEASE USE NO PUNCTUATION"
20 INPUT"NAME PLEASE";N$
30 INPUT"ADDRESS PLEASE";A$
40 PRINT"[CLR]F1 = NAME"
50 PRINT"F3 = ADDRESS"
60 PRINT"F5 = QUIT"
70 K = PEEK(197)
80 IF K = 64 THEN 70
90 IF K = 4 THEN PRINT N$
100 IF K = 5 THEN PRINT A$
120 IF K = 6 THEN END
130 GOTO 70
```

When you RUN this program you will be prompted to enter first your name and then your address. After which by pressing a function key you will either print your Name, or Address, or Quit. The thing to note about this type of user input statement as opposed to the GET statement we used earlier is that it repeats. This can prove a useful facility in many cases, but we shall be discussing user interface systems in much more detail later. The important point was to enable us too look at another form of PEEK.

If you can't, for the moment, lay your hands on the copy of CDU just mentioned then here is a little program which will allow you to look at every keypress and make up your own table.

```
10 K = PEEK(197)
20 PRINT K
30 GOTO 10
```

If you hold any key down it will keep on repeating, just make a note of the number and the corresponding key. To stop the program simply press the RUN/STOP key.

SPRITES AND SOUNDS

We all know of the amazing capability of the Commodore 64 in it's use of Sprites (also know as Movable Object Blocks, MOBs), and Sounds (whether special sound effects or music). To enable us to use these facilities to their full potential we will need to constantly POKE and PEEK values to and from the various areas of memory where the Sprite and Sound information is stored. We shall be covering these capabilities later in the series.

BEFORE WE CONCLUDE

I would like to draw your attention to page 146, Appendix F, of the User Manual. Here are listed the characters from the character set once more, but a quick observation will soon tell you that many of the numerical values are much different to those from the Screen Display Codes on page 143. The Screen Display Codes are Commodore's own, and are very useful when you want to POKE characters to the screen area - POKEing does have a lot of advantages

over PRINTing as we shall soon come to see. However, there are other aspects where PRINTing has its own particular advantages.

From the heading of page 146 you will see that these character codes are referred to as ASCII and CHR$ codes, you will, on examination also see that some codes appear to be empty, truth is they are not, and they do have functions which we will eventually come to know. You will also see that there are many other 'things' next to the number which do not seem to be characters, for example 14 = switch to lower case, 8 = disable the shift key and the CBM key.

The abbreviation ASCII means American Standard Code for Information Interchange. It is a standard character encoding system which is now used by the great majority of computers.

We can use CHR$ to convert an ASCII code into its character equivalent. The numbers must range between 0 and 255 and be enclosed in parenthesis.

```
10 PRINT CHR$(65) : REM ASCII FOR A = 65, THEREFORE THIS COMMAND WILL PRINT THE CHARACTER A

10 A$ = CHR$(13) : REM 13 = RETURN KEY

10 PRINT CHR$(147) : REM CLEAR THE SCREEN

10 PRINT CHR$(18) : REM TURN ON REVERSE CHARACTER MODE

10 GET A$: IF A$ = ""THEN 10
20 IF A$ = CHR$(160) THEN PRINT" SPACE BAR WAS PRESSED"
30 IF A$ = CHR$(133) THEN PRINT" KEY F1 WAS PRESSED"
40 IF A$ = "L" THEN PRINT CHR$(14): REM IF 'L' PRESSED SWITCH TO LOWER CASE
50 IF A$ = "U" THEN PRINT CHR$(142): REM IF 'U' PRESSED SWITCH TO UPPER CASE
60 IF A$ = CHR$(134) THEN PRINT CHR$(147):REM KEY F3 CLEARS SCREEN
70 GOTO 10
```

We shall be dealing with these codes in much more detail a little later on in the series when we start to construct more substantial programs.

NEXT MONTH

Next issue we shall be exploring the world of random numbers and how we can generate them, and use them in simulations and event timing environments. We shall also return to strings and have a look at just how easily we can cut them up, and manipulate the pieces. Further to this we shall construct a real time digital clock just for fun.

So until next month, happy finger tapping....

TABLE 1.

A GENERALISED MEMORY MAP OF THE C64.

LOCATION OR ADDRESS	USAGE	NOTES
0-255	ZERO PAGE	VITAL ADDRESSES
256-511	THE STACK	DISCUSSED LATER
512-799	CONTROL AREA	NOT MUCH FOR US
780-819	STORAGE & VECTORS	A WAY OF USING OP SYS
819-1023	TAPE BUFFER	DISCUSS LATER
1024-2023	SCREEN	SCREEN IS SITUATED HERE
2040-2047	SPRITE DATA POINTERS	DEAL WITH THIS LATER
2048-40959	BASIC PROG SPACE	OUR PROGRAM SPACE
40960-49151	BASIC INTERPRETER	BASIC INTERPRETER
49152-53247	FREE RAM	FOR M/C PROGRAMS
53248-57343	INPUT/OUTPUT DEVICES	WE WILL USE THIS AREA LOTS
57344-65535	OPERATING SYSTEM	THE BRAINS OF THE COMPUTER

TABLE 2

COLOURS AND THEIR NUMERICAL VALUE

0 - BLACK	8 - ORANGE
1 - WHITE	9 - BROWN
2 - RED	10 - LIGHT RED
3 - CYAN	11 - GREY 1
4 - PURPLE	12 - GREY 2
5 - GREEN	13 - LIGHT GREEN
6 - BLUE	14 - LIGHT BLUE
7 - YELLOW	15 - GREY 3

RASTER MASTER

Yet another Raster utility hits the streets. (Doesn't anyone program anything else these days ?)

DAVID BRYSON

Many utilities now are usually repeats of the same utility some time ago, but better (or perish the thought, worse!) This is why I decided to re-invent the wheel, and its name is "RASTER MASTER" With this utility, you can make up colour data for raster bars, and presentation of your own demos and games would be both easier and better. RASTER MASTER works by changing the colour on every raster line of the screen. Before starting, if you have not come across the term "RASTER", it is a VERY fast beam that draws the picture in a television, and travels across the screen, starting from the top and finishing at the bottom.

GETTING STARTED

When you first load RASTER MASTER and press a key you are presented with a virtually blank screen - but don't despair! Pressing "F7" will take you into the help screen, and you can immediately absorb information there if you don't like reading instructions. The reason why the screen is virtually blank is because no colour data is in memory, therefore to see example colour bars, press "L" on the HELP SCREEN to load a file. Type in the filename "DEMO.COL" and it will load the file from disk. There is also more example files called "DEMO2.COL" and "DEMO3.COL" which you can load in a similar manner. If you do not wish to load or save a file, press any other key to exit, although it is advisable that you press "F7" again, as pressing another key might activate a function accidentally.

USING THE EDITOR

When you have done this, the screen should display lots of pretty colour bars. You can edit the colours by moving up and down with the cursor keys and pressing "A" and "S" to move through the colours of the line that the cursor is on. If you want an even prettier effect, try pressing RETURN a few times. This should scroll the colours up, down or not at all, depending on how many times you pressed it. If you have one of those tellies that display the whole screen range, you may notice a slight glitch at the left-hand edge. You can disable the borders if you don't like it or think it looks better by pressing the "B" key.

BLOCK COMMANDS

One of the powerful commands in the editor is the block commands. To memorise a block, press "1" at the start of the block and "2" at the end. You do not see anything happening when doing this and, obviously, you must press "2" further down than "1". Once you have grabbed a block, you can duplicate it at another position by pressing "3", or if you want a mirror-image of the block press BACKARROW. This is very time-saving when making colour bars that are symmetrical.

THE COLOUR TABLE

The colour table is a stretch of memory which holds all the colours for each raster line. It is in the usual format, i.e. one byte per colour, and can be 192 bytes maximum (which means 192 raster lines, which is the most you can display on the editor screen) The program actually manipulates 256 lines, as you can see when you try inserting colours off the screen, and deleting them back on again. The colour table can be saved out when on the help screen and then be loaded into a demo or a game to be used - in fact, you could use it with Andy Partridges famous raster bars routine, featured in an earlier issue of CDU, but more about this later. The end of the colour table is signified by a thin white line, which you can move about with the keys "P" and "L". When you load a colour table, there is no need to set this, as the program does it automatically. This is why, when saving a colour table, you should always set the white line to the correct place.

USING IT IN YOUR OWN PROGRAMS

On the disk is a small example program ("RASTERBARS.BAS) which shows how you can display raster bars in your own programs, even if you don't use machine code. First, you have to load the machine code file which displays the colour bars, called "RASTERBARS.MC" and also the colour data which you saved previously (you did save it, didn't you?) Remember there is absolutely no need to use my display routine, as the colour data is in standard form, therefore it is possible to use your own if you have knowledge of machine code. In your program call this routine at the start using "SYS 52224" but before it is ready to run, you must tell the program how you want the colours displayed. The length of the colour data is important (which is displayed IN HEXADECIMAL on the help screen of the RASTER MASTER editor) as well as what vertical line you want your colour bars to start. You can choose any line ranging from $32 (50 decimal, top of

display) to $F9 (249 decimal, bottom of display) in the visible area. If you look at the example program on the disk, you will find the "POKES" that have to be made once you have decided on the starting line, and the colour data length.

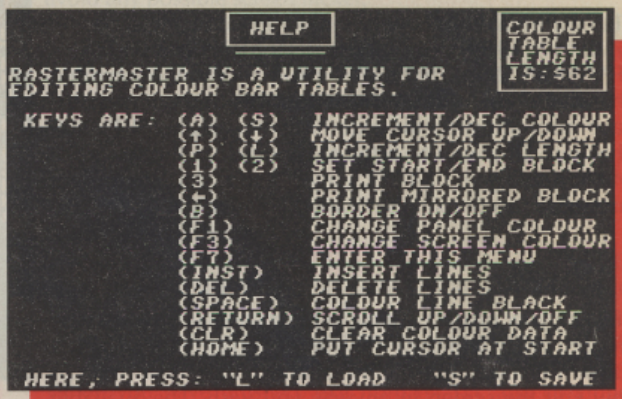
TECHNICAL DETAILS

The editor starts at the address 40888 and is approximately 2K long. The area of memory that the program is based is in the range from 40888-53247 ($9FB8-$CFFF) I have opted for the rarely-used extended colour mode of the VIC, so I could display the cursor without the colour lines interfering with it. Note that this program can be used with basic programs, if that is any help, as the only zero-page locations it uses are $FB-$FF. The "RASTERBARS.MC" program occupies locations $CC00-$CDFF (52224-52735) and the colour data for it and RASTER MASTER is addressed at $C700 (50944)

FINAL MESSAGE

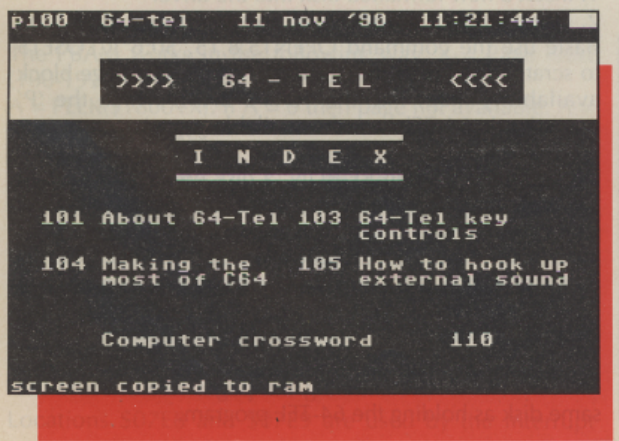
I think all the other commands I have not covered are self-explanatory. I am sorry that I have not included a routine to print the directory of a disk or command the disk drive, but if you are that desperate, you can press "RUN/STOP" and "RESTORE" and load the directory into basic and issue commands as normal.

When you have finished type SYS 40888 to restart the program without losing any data. I hope that this utility comes into great use, or if you don't really program at all, have fun fiddling about with it!



64-TEL

Use your C64 as a TELETEXT terminal. by S.OLATUNBOSUN



This is an information storage and retrieval program. It allows users to view it's pages in an attractive and simple way. New pages can be created to add to those already available. This is done using the on-line Editor. The user can use **64-Tel** in whatever way he or she wants. For example, the pages could be used to hold vital GCSE study notes. Alternatively the pages may be used for addresses as well as hints and tips for users that send disks to each other. The **64-Tel** service is like Teletext TV. What's more, it has the added touch of being more interactive and personal. Its simplicity in use will convenience both the novice and expert programmer.

HOW TO USE 64-TEL

Either select from the CDU menu, or load '64-TEL',8,1 followed by SYS36864.

You will be greeted with a simple 64-TEL logo. At this stage the program prompts you to enter the present time. First enter the hours followed by the minutes then seconds. The inputs must be two digits. (Eg: 8 o'clock would be 08). Enter the current date in the same way. When you have finished there will be some disk accessing as the index page is listed. You are now in a 'Teletext' service. Pages from 100-199 can be entered by the user. The requested page will be loaded from disk. Note that the index page resides in memory for prompt access. This page appears if page 100 or an unavailable page is requested. All pages available on the disk are found by pressing 'P'. That is basically how to use the program. Some other operational keys are given below.

F1 is go into Editor mode.
F7 is leave 64-Tel. (Do not use RUN STOP/RESTORE).
SPACEBAR Reveal stop. It's use is to show up hidden characters. Useful for answers to jokes and questions!! I have also activated the external voice of the SID chip. This will allow you to listen to an external sound source whilst reading the 64-TEL pages. I find that some light jazz music accompanies my viewing quite well!

THE EDITOR

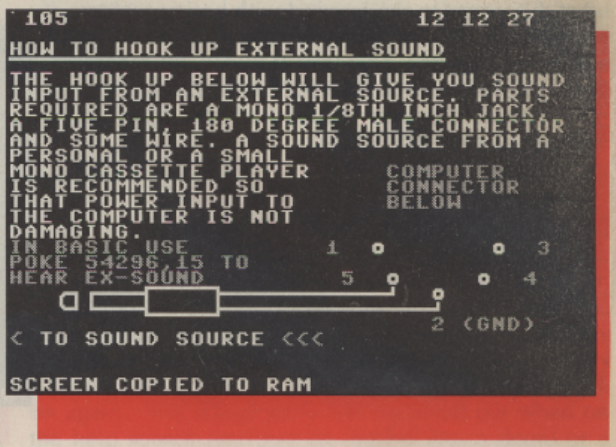
Creating your own, or editing existing pages is a very simple process. When you use F1 to enter the Editor from 64-TEL, a cursor will appear in the top left hand of the screen. You can now Edit the screen in anyway you wish. The available editing area is 40*23 characters. (The top screen line is used by 64-TEL and the bottom screen line displays function messages). All the control stops are available. Therefore you can use the cursor keys to position the cursor. The SHIFT plus CLEAR/HOME keys to clear the screen, maybe CTRL and 3 to change the text to RED. The following functions are also available.

F1 - Return to 64-TEL.
F2 - Sets page (Background) colour. Continuous pressing of F2 will scroll through all 16 colours.
F3 - Toggle UPPER/LOWER case. Use F3 instead of CBM/SHIFT to properly register the type case of the page.
F4 - Remove a page from disk. The page removed is the one currently specified by the page number. Use F7 before using F4.
F5 - Toggle FLASH/REVEAL mode. Flash mode is indicated by the cursor flashing. Entering the Editor from 64-TEL automatically sets it to reveal mode.
F6 - Save a page to disk. The saved page is labelled with the current page number. Use F7 beforehand.
F7 - Enter the new page number. A '1' will appear in the top left of the screen, just enter two more digits.
F8 - Copy SCREEN to RAM. (More on this later).

CBM - This is the Editors reveal key. The spacebar is already being used for Editing purposes. CBM will let you see how your reveal CLR-UP is working.
When you have finished designing your page and are ready to save it to disk, press F8. (This is essential otherwise the page you later on load back from disk may look quite different). Next, press F6 to save your page. Continue editing or creating new pages, or return to 64-TEL through F1.

FLASHING CHARACTERS

Flashing characters can emphasise parts of the screen, making them immediately noticeable. Introduce FLASH CHARS in the following way. Design all of your page first and copy it to RAM (F8). Now put the Editor into flash mode (F5) (You will notice this as the cursor slowly flashes). Now move the cursor to the character you want to flash. Put a SPACE in it's place (In other words, hit the space bar). Set the text colour of the flashing characters beforehand using the standard CTRL/CBM 1-8, with all the flashing characters introduced. Use F6 to save the page to disk. That's it!!



REVEAL MODE

Reveal mode works in the same way as above, except that the characters are hidden and do not appear until you press CBM in the Editor, or SPACE in 64-TEL. Follow the same procedure as above for the FLASH characters, but remember to have the Editor in REVEAL mode.

A FRESH START

The user may want to create his/her own pages on a fresh disk. They can either:

1. Insert a new formatted disk into the drive.
2. Use the Editor to remove all existing pages. Then from Basic use the command OPEN15,8,15,"S0:B":CLOSE15 to scratch the PBAM file. (The PBAM file is a page block availability map, like the 1541's BAM, which the 'P' operation in 64-TEL uses the available pages).

When 64-TEL is then RE-RUN, a disk error will occur as the program tries to load the index page. Ignore this and start creating your pages. It is a good idea to write your index page first. This can always be Edited later, but its initial presence is to stop disk error occurring next time 64-TEL is used. If you forget which pages you have created, use the key 'P' operation in 64-TEL.

NOTE - The disk containing the pages need not be the same disk as holding the 64-TEL program.

```
p103  64-tel    15 mar '91  09:10:16
In 64-TEL

F1      - Go to editor
F7      - Leave 64-TEL
P       - Display available pages
SPACE   - Reveal hidden characters

In Editor

F1      - Return to 64-TEL
F2      - Set page colour
F3      - Toggle upper/lowercase
F4      - Remove page from disk
F5      - Toggle flash/reveal mode
F6      - Save page to disk
F7      - Enter new page number
F8      - Copy screen to ram
<cbm>   - Activate reveal

          C O M M A N D S
```

HOW IT WORKS

An assembler listing, M/C 64-TEL, is provided on the disk. The program is full machine code and is interactive with an IRQ interrupt routine. (Which handles the clock display update and the dual background colour screen). The program consists of two main parts, 64-TEL and the EDITOR. 64-TEL handles PAGE and PBAM disk loading whilst the EDITOR deals with the page saving. The 64's memory is used in the following way.

$A000-$FFFF - Kernal ROM, I/O, Basic etc.
$9000-$9A00 - 64-TEL program.
$1900-$8FFF - RAM.
$1800-$18FF - PBAM (Used ares $1864-$18C8).
$1400-$17FF - Index page colour.
$1000-$13FF - Index page text.
$0C00-$09FF - Current page colour.
$0800-$0BFF - Current page text.
$0400-$07FF - Text screen display.
$0000-$03FF - Operating system and Basic ROM use.

The portion loaded and saved to disk each time is memory area $0800-$1000. The contents of a current page colour area reflect somewhat colour ROM at $D800-$DBFF. The current page text is 'identical' to screen RAM, $0400-$0800.

PAGE BLOCK AVAILABILITY MAP

The PBAM works in the following way. X = Page Next (100-199).
A = PEEK($1800+X). If A = 0 then page not available. A = $FF page available.

Screen RAM really runs from $0400-$7BFF. Therefore 4 of the spare addresses of $07C0-$07FF are used.

$07E8/$0BE8 - Screen background colour.
$07E9/$0BE9 - Flash/Reveal mode.
$07EA/$0BEA - Upper/Lower case mode.
$07EB/$0BEB - Page next.
$0BE8-$0BEB have the same contents as $07E8-$07EB.

Locations $07E8 and $07E9 are used by the interrupt routine. If a viewed page has a non-black background colour, you will notice 2 RASTER screen splits. One below the top screen line, the other above the bottom screen line. Between these 2 RASTER splits a different colour background can be used instead of the black line that appears on the top and bottom screen lines. FLASH/REVEAL mode works by altering VICMEM address $D018. This states where the screen display will start. The IRQ routine will set $D018 to display the screen at $0400 or $0800. The IRQ routine operating from a RASTER IRQ runs at 100Hz. The keyboard scan routine in ROM is called on every other IRQ call. Therefore it runs at normal pace. The operating systems normal interrupt mechanism at address $DC0D was disabled to avoid any possible screen flicker.
64-TEL is situated just below Basic ROM. Area $C000-$CFFF is used too frequently, and besides, putting 64-TEL in this area would have made it incompatible with the

TURBO disk loader.
If you are feeling quite ambitious you can try the following;

POKE44,25:POKE6400,0:NEW
LOAD"64-TEL",8,1:NEW

Now you can use 64-TEL in conjunction with any Basic program that you may be developing or running. For example, instead of thumbing through your manual just run 64-TEL (SYS36864) and load up that help page from disk. When you have finished, exit from 64-TEL and continue with whatever you was doing previously. Your Basic code can use up to about 30K before it starts to corrupt 64-TEL. While in Basic you can see that 64-TEL is still active through the continuously updated clock. You can remove this by hitting RUN/STOP and RESTORE.

FINALLY

I have included a simple demo program to illustrate how 64-TEL can be used with a Basic program. If you have followed the commands above (POKE44 etc), then load the demo program and run it. Your own programs should not alter the IRQ vector at $0314-$0315 or your program will hang up. WARNING, 64-TEL handles no disk errors. Therefore it is not failsafe. However, if all the guide lines are followed properly then everything should be Okay.

```
P104  64-TEL    11 NOV '90  11:42:23
THE 64 AND EXTERNAL SOUND
●●●●●●●●●●●●●●●●●●●●●●●●●

THE COMMODORE IS STILL QUITE A WELL
KNOWN HOME COMPUTER. CONTRIBUTING TO
ITS POPULARITY IS THE BRILLIANT SOUND
IT POSSESSES. WHILE THINGS ARE GOING
WELL WITH PEOPLE SYNTHESIZING ALSORTS
OF SOUNDS FROM THE SID CHIP, THE SID'S
EXTERNAL INPUT SOUND HAS BEEN MADE
REDUNDANT.
EXTERNAL INPUT SOUND BRINGS A FURTHER
DIMENSION OF SOUND REALITY. IT COULD
BE USED IN CONJUNCTION WITH THE 3 SID
VOICES, ENHANCING ANY MUSIC COMPOSITION

THEN AGAIN HOW ABOUT
READING THESE 64-TEL
PAGES WITH A LITTLE
CLASSICAL MUSIC ?

MORE ON 105...
SCREEN COPIED TO RAM
                    MUSIC MAN !!!
```

BASIC MACHINE LANGUAGE TECHNIQUES

Our tutorial into Machine Code programming continues, by J Simpson.

"A series of lessons designed for the beginner to enter the world of machine code programming on the commodore 64." That's what the man said, and that's exactly what this series of tutorials hopes to bring you.

PART TWO - EIGHT-BIT ADDITION

Last month we constructed (albeit rather clumsily) an eight-bit addition routine. We shall now construct a modified version of an addition routine.

```
10      * = $C000
20      ;
30      ADDITION  LDA VALUE1
40                CLC
50                ADC VALUE2
60                STA SUM
70                RTS
80      VALUE1    BYT 20
90      VALUE2    BYT 20
100     SUM       BYT 0
110     END
```

LINE 10 this line organises where in memory the routine will locate from. If this was a sub-routine called on a regular basis from within a program then this line would be omitted.

LINE 30 starts with our routine label called, ADDITION and the first mnemonic instruction LDA tells the processor to load the Ac with the value held in the memory byte labelled VALUE1. If this was a subroutine then it is the label, ADDITION, which would be called (more about calling subroutines later in the series).

LINE 40 clears the carry (flag) ready to be set if the addition overflows the byte, i.e. if the value becomes greater than 255.

LINE 50 adds the value of the memory byte, VALUE2, plus the value of the carry bit (in this case 0, because we cleared it earlier), to the contents of the Ac (which held the value of VALUE1).

LINE 60 stores the total now held in the Ac into the memory location SUM.

LINE 70 ends the routine with a return to the caller.

LINES 80 to 100 are the three memory locations set aside for variable data storage.

LINE 110 is the assembler directive to terminate assembly.

Within a program which required the addition of the two bytes, VALUE1 and VALUE2, would be the necessary code which would place the values required into these two data bytes ready for computation (we shall look at this more closely when we start discussing subroutines).

EIGHT-BIT SUBTRACTION

When programming we will also require the use of subtraction just as much as addition, so let us construct another small routine which does just that.

```
10      * = $C000
20      ;
30      SUBTRACT LDA VALUE1
40               SEC
50               SBC VALUE2
60               STA SUM
70               RTS
80      VALUE1   BYT 50
90      VALUE2   BYT 40
100     SUM      BYT 0
110     END
```

As you can easily see, there is not a great deal of difference between the two routines, ADDITION and SUBTRACT. The differences, apart from the values held in the variable data bytes, is confined to Lines 40 and 50.

LINE 40 this instruction is the opposite of CLC, in that SEC means SEt the Carry. In other words the carry bit (flag) is now equal to 1.

LINE 50 instead of adding with carry we now SuBtract with Carry. This allows to subtract from the Ac the data at the specified address without a borrow.

In our example we first loaded the Ac with the contents of VALUE1 (50). We then subtracted the contents of VALUE2 (40) from the Ac. And stored the result in SUM (10). During the subtraction process we used the carry flag to test for an underflow, if the result of the subtraction had been less than zero, then the carry flag would have become clear.

Understanding the full implications of using the Carry will become more clear when we look at 16 bit addition and subtraction.

SIXTEEN-BIT ADDITION

An eight-bit addition will only allow the addition of numbers within the range of 0 to 255. For more practical operations it is necessary to use 'multi-precision' and to add numbers which use sixteen bits or more. I shall demonstrate addition on 16-bits, but they can readily be extended to 24, 32 or more bits. We always use multiples of eight-bits or a byte. Check this example:

```
10      ADDITION    LDA VAL1
20                  CLC
30                  ADC VAL2
40                  STA SUM
50                  LDA VAL1+1
60                  ADC VAL2+1
70                  STA SUM+1
80                  RTS
90      VAL1        BYT 255,160
100     VAL2        BYT 237,46
110     SUM         BYT 0,0
120     END
```

First of all let us examine the numbers we are adding together. The first pair of numbers held in the two bytes of VAL1 and VAL1+1 represent 41215 ($A0FF). The second pair of numbers held in VAL2 and VAL2+1 represent 12013 ($2EED). We store the low eight-bits of the 16-bit number in the first byte, followed by the high eight-bits in the second byte.

You probably know that it is standard to store a large value in the Low/high byte formula but if you are unsure just how I came to arrive at 255 (lo), 160 (hi) VAL1, and 237 (lo), 46 (hi) VAL2 I will explain. To calculate the high and low bytes of a 16-bit number, that is a number in the range of 0 to 65536, we must first find the hi-byte. This is simply done by dividing the 16-bit number by 256, and ignoring any fractional parts. To obtain the lo-byte we then subtract the hi-byte from the original number and multiply the result by 256. A simple line of BASIC code will do this for us, namely:

```
10 NU = <NUMBER TO SPLIT>:HI = INT(NU/256):LO =
NU-INT(HI)*256
```

Most good assemblers will have some form of number conversion command and if you convert the decimal number into its hex equivalent then it is simple to split the number into Hi/Lo bytes (e.g. $A0FF = $A0 and $FF. $2EED = $2E and $ED), this would then be incorporated into your code using a suitable pseudo-op.

```
90 VAL1     BYT $FF,$A0
or if your assembler uses the pseudo-op, hex
100 VAL2    HEX ED21
```

If we now look at the 16-bit addition program you will see that lines 10 to 40 are just the same as in the 8-bit addition program, however when we add the two Hi-bytes of VAL1+1 and VAL2+1, we did not precede the addition with a CLC instruction (we only use CLC at the beginning of an addition). This means we add together the two Hi-byte values plus the value of the carry. In the case of our addition this is what took place:

When VAL2 was added it took the value held in the Ac (taken from VAL1) which took it beyond the range of a byte, 255. This resulted in the carry bit being set (1). Now the carry will represent 256 to be added to the Hi-byte, or adding 1 to bit 0 in the Hi-byte (which has the column value of 256). The balance, 236, which was left in the Ac is then stored in SUM1 at line 40. Then lines 50 to 70 added together the values held in VAL1+1, VAL2+1, plus the carry. We know the carry bit was set, so we carry over 256 to be added to the result of VAL1+1 and VAL2+1. Here is a graphic example of what took place:

(LOW BYTE)

```
 VAL1 = 255          %1111 1111
+VAL2 = 237          %1110 1101

=     492            %1110 1100   CARRY = 1
                     (REPRESENTING 256)
                     - CARRY 256

SUM1   236          %1110 1100
```

(HI BYTE)

```
 VAL1+1 = 160        %1010 0000
+VAL2+1 =  46        %0010 1110
+CARRY  =   1        %0000 0001   (The first column
                     represents 256)

SUM1+1    207       %1100 1111
```

16-BIT VALUE OF RESULT = 53228 $CFEC %11001111 11101100

SIXTEEN-BIT SUBTRACTION

```
10      SUBTRACT    LDA VAL1
20                  SEC
30                  SBC VAL2
40                  STA SUM
50                  LDA VAL1+1
60                  SBC VAL2+1
70                  STA SUM+1
80                  RTS.
90      VAL1        BYT $FF,$A0
100     VAL2        BYT $ED,$2E
110     SUM         BYT $00,$00
120     END
```

Subtraction is the reverse of addition. We SEt the Carry to one at the start of the operation, and if the result of the lo-byte becomes a minus number then the carry is cleared and thus the lo-byte would have borrowed from the hi-byte (the value of 255). The result would be correct when the hi-byte is calculated because we have borrowed the least significant bit (the right-most bit) from the hi-byte during the calculation. If the result of the lo-byte subtraction did not underflow, then the carry would remain set which would indicate a "no-borrow" condition. Study the two programs, addition and subtraction until you are familiar with them. Try using different values, and stepping through each line of code with a paper and pencil noting how the changes occur (tracing, or single stepping, on paper is explained more fully towards the end of this months article).

MULTIPLICATION (GETTING TRICKIER)

The multiplication of binary numbers is somewhat more complex. For reasons that only Commodore will know they omitted instructions to perform either multiplication or division.

It is not necessary for me to explain how we multiply two numbers in decimal notation, we all learned this at school. The method for multiplying two binary numbers is exactly the same. Let us multiply 5 x 3:

```
(5)     101  Multiplicand (MPD)
(3)   x 011  Multiplier   (MPR)

        101  Partial Product (PP)
        101
        000

(15)  01111  Result (RES)
```

The multiplication is performed by multiplying the right-most digit of the Multiplicand by the Multiplier, then one multiplies the next digit and so on. We offset the digits to the left by one position. Or, equivalently we could say that the partial product had been 'shifted' one position to the right before adding.
In order to perform multiplication the computer will operate the same as we did above. It will 'shift' each line one place to the right then add together all of the lines. However, because we are dealing with a full byte, eight bits, then the computer must shift all eight lines, or eight 'shifts'.
Here is a routine to multiply two eight bit numbers (8 x 8 multiply).

```
10 INIT   LDA #0      ; SET Ac TO ZERO
20        STA TMP     ; ZERO THE TEMPORARY DATA
                      ADDRESS
30        STA RES     ; AS WELL AS THE RESULT BYTES
                      (LO)
40        STA RES+1   ; AND (HI)
50        LDX #8      ; WE ARE USING THE X INDEX
                      REGISTER AS A COUNTER
60 MULTIPLY
70        LSR MPR     ; SHIFT MULTIPLIER RIGHT
80        BCC NOADD   ; TEST CARRY, IF CLEAR NO
                      ADDITION SO BRANCH
90        LDA RES     ; LOAD Ac WITH LO-BYTE OF
                      RESULT
100       CLC         ; PREPARATION TO ADD
110       ADC MPD     ; ADD MULTIPLICAND TO
                      RESULT
120       STA RES     ; NOW SAVE LO-BYTE OF RESULT
130       LDA RES+1; ADD REST OF SHIFTED
                      MULTIPLICAND (HI-BYTE)
140                   ADC TMP  ; ADD TEMPORARY DATA
BYTE
150       STA RES+1 ; SAVE HI-BYTE OF RESULT
160 NOADD
170       ASL MPD   ; LEFT SHIFT MULTIPLICAND
180       ROL TMP   ; LEFT SHIFT AND SAVE BIT
190       DEX       ; DECREMENT THE COUNTER
200       BNE MULTIPLY ; IF NOT FINISHED DO IT
                       AGAIN
210       RTS
220       ;
230       TMP    BYT 0
240       RES    BYT 0,0
250       MPD    BYT 5
260       MPR    BYT 3
270       END
```

Because we are doing eight-bit multiplication we will have to test each bit of the multiplier (MPR), but unfortunately there is no instruction with which we can sequentially test the bits. It is only the bits (flags) in the status register (SR) which can conveniently be tested. This is a limitation of most microprocessors, and as a result of this limitation it will be necessary to shift the byte and test the bit. What actually occurs when shifting is that every bit in the byte will move right or left by one position. The bit which is shifted out of the byte, or 'falls' of the end, is placed into the carry bit of the SR. Fig 1 demonstrates the 'shift' (you will note that the figure demonstrates the left shift, or ASL, to study a right shift, or LSR, then simply reverse the direction of the arrows). There are different possibilities depending on the shifted 'bit' but more of this will be discussed later.
However, since we need to test each of the eight bits of the Multiplier, and since we can easily test the carry bit, then we must shift the Multiplier by one position eight separate

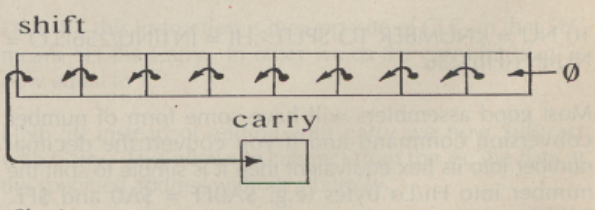


fig1

times - its right-most bit will 'fall' into the carry bit (flag) where we will test it each time in order to take the appropriate action. Next the Partial Product (PP) which is accumulated with each of the successive additions will require the use of sixteen bits, obviously multiplying two eight bit numbers could produce a 16-bit result (255 x 255

= 65025). Unfortunately the 6510 has few internal registers which means that the Partial product, the Multiplier, and the Multiplicand cannot be stored within the 6510 itself. Therefore we must set aside some RAM memory locations, or external registers, within which to store them (Partial product into RES and RES+1, Multiplier into MPR, and Multiplicand into MPD). We will shift the Multiplicand left and into an address labelled TMP prior to adding it to the Partial product (RES, RES+1).
We shall now examine this routine in more detail. There are several important points to discuss, as well as the instructions which we have not covered before. It is very important, when developing good software, be it games, utilities or even demos, to have a thorough understanding of the multiplication issue.

DETAILED EXAMINATION

Lines 10 to 40 are quite simple, we have dealt with load and store instructions, and so they do not require further explanation, save to say that if this is an often called routine from within a larger program, then it is good programming practice to zero the temporary data byte and the two-byte result before execution.

Line 50 LDX #8. This is our first use of the X index register and we are going to use it within this routine merely as a loop counter. So, by using the # symbol, the same as the LDA instruction, we have instructed the CPU that the value following it is the literal value of 8 which we store within the X register and not the contents of memory location 8. We are using the counter in order to stop shifting the MPR at the right time. We need eight shifts, one for each bit, so each time a shift occurs we decrement the X index register. As soon as the value of the X register reaches zero the multiplication has finished.

Line 70 LSR MPR. This instruction means 'Logical Shift Right' the contents of the byte MPR. We must test the least significant bit (that is, the right-most bit) of the MPR, and, as I indicated earlier, this cannot be done with a single instruction. So first we must 'shift' it, then we must test the carry bit to find out if the bit shifted out was a 1 or a 0, and use this to either branch or not. If the bit was a 0 then this indicates a 'no addition' situation.

Line 80 BCC NOADD. Test the the value of the carry flag. BCC NOADD means 'Branch if Carry (bit) is Clear' to the address labelled NOADD. If the carry is clear, then this tells us that the bit shifted out of the byte was a zero and so it does not require to be added.
This is our first encounter with a branch instruction. So far the routines we have dealt with have all been sequential. In other words each instruction being executed after the previous one. Obviously we need to perform logical tests, such as testing the carry bit, which will direct us to another part of the program, similar to the IF...THEN of Basic. Branch instructions perform such functions. In our branch instruction BCC the CPU will branch to a new address if the carry bit = 0, in other words the next instruction at address NOADD (i.e. line 170 - ASL MPD). If the test fails, in other words the carry was set (=1) then no branch would occur thus program execution would continue with the next instruction after BCC NOADD.

Line 90 to 150 The test failed and the carry bit is set. We must add the MPD to the Partial product (here the RES registers). The MPD is held in TMP and MPD. The 16 bits of the Partial product are held in RES and RES+1. We simply add together the two 16-bit numbers as we did in the 16-bit addition of the earlier program (Sixteen-bit Addition).

LDA RES - the Ac is loaded with the lo-byte of RES.
CLC - prior to any addition the carry bit must be cleared
ADC MPD - add the MPD to the Ac which already contains RES lo-byte
STA RES - store the result of the low part of the addition in RES lo-byte (the carry will be either set or clear depending upon the result of the addition, and any carry which may have been generated will carry over to the hi-order part of the result).

LDA RES+1
ADC TMP
STA RES+1

These three instructions complete our 16-bit addition. We have now added the MPD to RES, however we must still shift it by one position to the left in anticipation of the next add.

Line 170 ASL MPD - 'Arithmetic Shift Left' will shift the contents of the register MPD one position to the left, the low part of the Multiplicand. However we cannot lose the bit which falls off the left end of the byte. It 'drops' into the carry but we cannot leave it there because it would quickly be destroyed when we perform our addition. It must, therefore, be placed into a semi-permanent location. We use TMP for this.

Line 180 ROL TMP - 'ROtate Left' the contents of TMP. This is different to SHIFT in that it performs the same function but with the additional feature that the bit in the carry is forced into the right-most bit position and the bit which 'falls' off the end is placed into the carry (see Fig 2. Rotates. The figure demonstrates the left rotate, ROL, to demonstrate a right rotate, or ROR, simply reverse the direction of the arrows). So the bit which fell off the left-most end position, during execution of the previous instruction ASL MPD (line 170), and held in the carry, will now be forced into the right-most position of the register TMP.
This finishes with the arithmetic operations of the routine but we must still test whether the operation has been performed a full eight times.

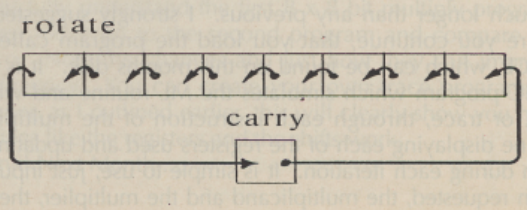


fig 2

Line 190 DEX - 'DEcrement index register X'. If X contains 8 then its contents will contain 7 after this operation.

Line 200 BNE MULTIPLY - 'Branch if the result (in this case, of the X decrement) is Not Equal to zero' to memory address MULTIPLY (Line 60). This is another test-and-branch instruction. While the X index register decrements to a non-zero value the CPU will automatically branch back to the the memory address following the branch instruction, in this program, MULTIPLY. If X has decremented to zero then the result of the test will be false and so the CPU will execute the next sequential instruction.

The flag used for this test is the Z flag (See Status Register - Part 1).

Line 210 RTS - returns to the part of the main program which called this multiplication routine.

Lines 230 to 260 are the memory locations for the various external registers used in this routine.

THE TRACE PROGRAM

It is very important, if you wish to program efficiently in ML, that you fully understand such a typical program as the 16-bit multiply routine in complete detail. I have introduced you to many new instructions, and the routine is much longer than any previous. I strongly suggested, before you continue, that you load the program called TRACE, which can be found on this months disk. It is a BASIC program which simulates the ML routine and will step, or trace, through each instruction of the multiply routine displaying each of the registers used and updating them during each iteration. It is simple to use, just input, when requested, the multiplicand and the multiplier, then use the space-bar to 'step' through each line of the program. All of the registers are displayed in binary digits which will give you a much clearer picture of exactly what is taking place, especially during the 'shift' and 'rotate' instructions.

It is not the best of programs, just the 'bare bones' but it will prove sufficient for our purpose of a clearer understanding of ML. After you have used the trace, maybe after several times, you will fully understand the mechanism by which instructions manipulate the contents of memory, of the CPUs registers 'Ac' and 'X', and just how the carry bit (flag) 'C', is being used.

You can, if you wish, create your own trace program on paper and verify that the program works by hand. Every time that a routine is written it is often checked thoroughly by hand to ascertain that its result will be correct.

DIAGRAM 1. Shows an example of a hand trace operation.

The first instruction is LDA #0. After this is executed the contents of the X register are unknown, so eight dashes would indicate this. The A register would contain zero, so eight zeros would be placed under the A heading. You would place the relevant binary bits under the MPR and MPD. The C-bit and the TMP, RES, and RES+1 registers are undetermined, so these would also contain dashes. The next instruction STA TMP will change the dashes in TMP to zeros, and so on with each instruction. I've started the first of the eight iterations for you but it is up to you to finish it.

Using the hand trace method in conjunction with the BASIC trace program will allow you to verify that which you do by hand. All of this will, essentially, teach you very good programming practice and technique and will ensure that when you start writing your own programs you will create more fluent programs with fewer 'bugs' or logic errors to iron out.

DIAGRAM 1

LABEL	INSTRUCTION	-X-	-A-	MPR	C	TMP	MPD	RES	RES+1
INIT	LDA #0	---- ----	0000 0000	0000 0011	-	---- ----	0000 0101	---- ----	---- ----
	STA TMP					0000 0000			
	STA RES							0000 0000	
	STA RES+1								0000 0000
	LDX #8	0000 1000							
MULTIPLY	LSR MPR			0000 0001	1				
	BCC NOADD								
	LDA RES		0000 0000						
	CLC				0				
	ADC MPD		0000 0101						
	STA RES							0000 0101	
	LDA RES+1		0000 0000						
	ADC TMP		0000 0000						
	STA RES+1								0000 0000
NOADD	ASL MPD						0000 1010		
	ROL TMP				0	0000 0000			
	DEX	0000 0111							
	BNE MULTIPLY								
2nd ITERATION									
MULTIPLY	LSR MPR			0000 0000	1				
	BCC NOADD								
ETCETERA)									

There are many ways in which a program can be written, in fact the 8 x 8 bit multiply that we have just developed is such a case. We can always find ways to modify and sometimes improve a program. For example we could have shifted the result by one position to the right before adding it to the multiplicand instead of shifting the multiplicand to the left before adding. What are the advantages of such modification? Well we would not have needed the temporary register, TMP - a saving of one memory location! This is not such a drawback, unless we are really pushed for memory. However, it might make the program run somewhat faster, and this can be very important.

Effective programming requires us to spend time looking at detail so that we can reduce the length of a program, and improve its execution speed. Using shifts for the result and the multiplier consume instructions and time. The 'trick' which we can use in multiply algorithms is that every time the multiplier is shifted to the right then a bit position on the left is freed up. We can also see that the first result will take up, at the most, nine bits and that after the next shift the result will be increased by one bit again. What this boils down to is that we can reserve one memory location for the result (or partial product) and then use the bit positions which are being freed up by the multiplier as it is shifted. Now we are going to shift the MPR (multiplier) right which will free a bit position to the left. We will enter the left-most bit of result into this position that has become free. We will use the X register as our counter for the number of bits being shifted. Unfortunately the CPU only has one internal register which can be shifted, and that is the Accumulator. Because we are going to shift both the result and the multiplier we need to determine which one we should put into the Ac. Well, since the result must be added to the MPD every time a set bit is shifted out, and since the CPU also adds something only to the Ac, then the result is best served using the Ac. The other numbers will reside in memory locations.

Ac (result high-byte) and RES (result low-byte). MPR and MPD will be as before. So, then, here is our alternative 8 x 8 bit multiplier:

```
10      MULT   LDA #0
20             STA RES
30             LDX #8
40      LOOP   LSR MPR
50             BCC NOADD
60             CLC
70             ADC MPD
80      NOADD  ROR A
90             ROR RES
100            DEX
110            BNE LOOP
120            RTS
130     RES    BYT 0
140     MPR    BYT 5
150     MPD    BYT 3
160     END
```

Line 10 and 20. LDA #0
STA RES
here we are using the Ac for the high byte of the result, and RES for the low byte of the result. So we initialise them to zero.

Line 30 LDX #8 once again we shall use the X register for our shift counter, so this is initialised to the value of 8.

Line 40 LSR MPR the same as the earlier routine, we shift the multiplier to right.

Line 50 BCC NOADD the result of the shift has two possibilities, either the carry is set or clear. If clear, then no addition is required so branch, otherwise continue with the next instruction which is to perform the addition.

Line 60 and 70 CLC
ADC MPD

We must always clear the carry before addition (and we know it must have been set simply to have reached this stage of the program). So we add the multiplicand to the accumulator (the Ac holds the high-byte of the result).

Line 80 and 90 ROR A
ROR RES

this is the Partial Product held in Ac and RES. By shifting (rotating) the Ac right one bit, the left-most bit will 'drop off' into the carry. The carry bit is then rotated into the RES (result low) register which holds the low-byte of the result.

line 100 and 110 DEX
BNE LOOP

simply test the X index and if it has not yet reached zero, then branch back to LOOP for the next iteration.

Line 120 RTS return from subroutine to caller.

Line 130 to 150 are the memory registers we are using for our data.

If you examine this modified routine you will see that it uses around half the number of instructions, and, as a result of this, and the selection of the correct registers, it will execute a lot faster.

A straightforward program design will work, but with a little thought we can make it work more efficiently.

IN CONCLUSION

Well, we have covered a lot of tricky ground this month, but never-the-less it is ground that needed to be covered. Examine the addition and subtraction programs, and create your own. See just how you might improve on things. I recommend that you use the trace program until you fully understand the first 8 x 8 bit multiply program, then move on to the second program and compare the two. Note the changes and just how they can improve efficiency. You can create your own trace program on paper, as I outlined earlier, this will clearly show you how things like the registers and the shifts work.

Until Next Month.....

...it's dynamite!

POWER CARTRIDGE

FOR YOUR COMMODORE 64/128

"...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

SO MUCH FOR SO LITTLE

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	HARDCAT	RENUMBER
AUDIO	HARDCOPY	REPEAT
COLOR	HEX$	SAFE
DEEK	INFO	TRACE
DELETE	KEY	UNNEW
DOKE	PAUSE	QUIT
DUMP	PLIST	MONITOR
FIND	ILOAD	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	DVERIFY	DIR
DSAVE	MERGE	DEVICE
DISK		

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	SAVE	VERIFY
MERGE	AUDIO	

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

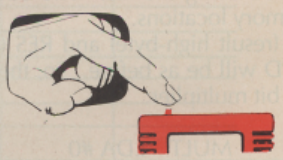
The POWER CARTRIDGE contains a very effective Printer-Interface, that self detects if a printer is connected to the Serial Bus or User-Port.
It will print all Commodore characters on Epson and compatible printers.
The printer-interface has a variety of set-up possibilities. It can produce HARDCOPY of 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

WORDSEARCH

Create your very own customised WORDSEARCH puzzles with ease . by STEVEN BURGESS

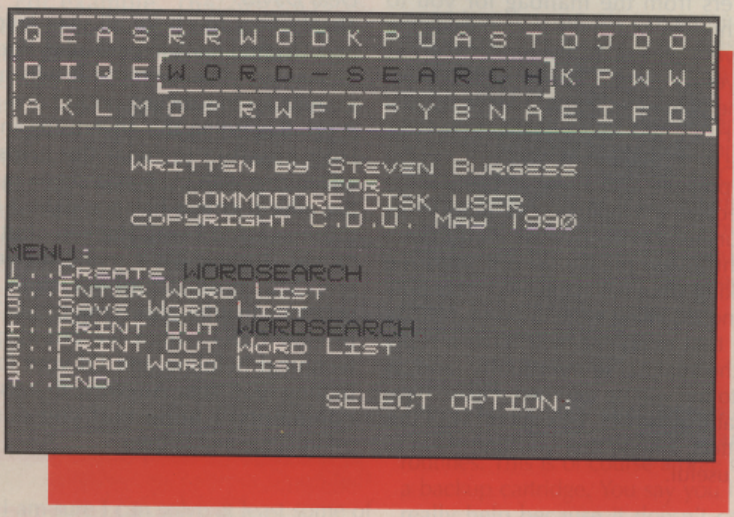
Are you mad about those wordsearch puzzles? Wouldn't you just love to make your own customised wordsearches to baffle your friends and colleagues (and, if you lose the grid showing all the answers, even yourself.)

Well, my dear friend, read on and all your dreams will be realised. All your hopes fulfiled and all your friends and colleagues will hate your guts.

When the program has loaded via the C.D.U. menu, you will be presented with another menu. The options are as follows:

CREATE WORDSEARCH

This option cannot not be selected, and you shouldn't try, until you have either loaded a list of words or have typed in a new set. When you have, however, it sets up the wordsearch grid. When it has placed all the words in your list it prints out the grid without all the extra letters so that you can easily see where your words are. Have the printer ready.



ENTER WORD LIST
This option allows you to enter the word list. You can choose how many words you want to enter but I wouldn't go too high. Once you have typed in a word there is no going back, except to start afresh, so make sure you type very carefully indeed.

SAVE WORD LIST
Allows you to save your word list to diskette. Asks for filename and then saves as a sequential file.

PRINT OUT WORDSEARCH
Prints the Wordsearch grid out to the printer. The grid is 30x30 and uses thirty printer lines. Between each letter across there is one space.

PRINT OUT WORD LIST
Prints out the word list currently in memory.

LOAD WORD LIST
Allows you to load a previously created word list from diskette.

END
This simply returns you to the C64 startup screen. If you pressed it by mistake then switch on the drive, insert the C.D.U disk and type: SYS 2152

HOW IT WORKS

The wordsearch grid is created on a perfectly random basis. First the computer decides which way the word will run, and there are eight different ways. Then it will decide upon the x and y coordinates in the grid where the will start. It checks first if the word will fit and, if it will, if there are any other letters in its path. If there aren't then it places the word. If there are, and the word can still be placed because it can safely cross through the other word then the word is placed. If neither of these two are true then the computer chooses another direction, another x and another y coordinate. This data can be seen flashing up on the screen when you select CREATE WORDSEARCH.

FOOTNOTE: This program was.written with the aid of LASER BASIC, from the OCEAN IQ range of utilities. It was compiled with LASER COMPILER.

For owners of LASER BASIC, there is, on the disk, a copy of the source program which you can mess about with.

TECHNO-INFO

Can you solve my problem Dad? No! But I know a man that can... "JASON FINCH" circa 1990

When I closed last month I told you that I would bring you news of a program that gave FULL DISK JACKET it's compatibility with the STAR LC10C. Well, here it is. A couple of people, namely MR CLIFF KENDALL OF CUMBRIA and MR DENIS REDINGTON OF ESSEX, very kindly sent me copies of the program that they had modified for use with the above printer. I had to choose one and, because they were both of a high quality, I had to go on a first come first served sort of basis which meant that MR.KENDALL was the lucky man. On the disk this month you will find his program, filed as "STAR FDJ". I hope that everyone will appreciate his work. This month I have selected ten letters from the mailbag for you to peruse at your leisure, three of which are from outside the UK which makes me very happy. To everyone that has sent letters and hasn't yet received a formal reply, please bear with me. Due to the volume of letters that we receive at the TECHNO INFO HQ it just isn't possible to answer them all personally. The great majority of you will receive replies as soon as it is possible and a few of you will be privileged enough to receive a telephone call from yours truly if you supply your number!! Anyway, if you don't receive a reply straight away, please do not send me copies of your letter asking what has happened. This only confuses both myself and my filing system! From this day on I am guaranteeing that everyone will have a reply either by post or in the magazine eventually, and as always, any disks sent to me will be returned. Please have patience. Well, now that that is over, let's get on with this month's selection of letters. I hope that you find them to be both interesting and useful.

THANKS FOR CDU

Dear CDU,
I doubt that I will be the first this year to praise your magazine for its content, but I feel that I should make my feelings known. I have only recently found out about CDU and was, to say the least, pleased that someone had the sense to bring out a magazine for those who realise that computers are for more than just vegetating their brain on games. Especially do I find "Techno-Info" interesting as it helps everyone to get more from their machine - even us lowly ones that know very little machine code. However, if I was to be critical, I would say that it lacked only two things. Firstly, advertisements. In the years BC (Before CDU) I used to buy the Commodore games 'comics' for the software and hardware adverts. What I would like to see, therefore, in CDU is perhaps more adverts? Especially from companies such as Datel Electronics, etc. Secondly, size - not so much a criticism, rather a point of view. I would like to see more features in the magazine and for this I would be willing to pay extra and still consider it good value for money. What this really points to is that the magazine is great. Keep up the good work!
David Brown, Newport.

Dear David,
Thanks very much for all your kind remarks about CDU and "Techno-Info" in particular. It really is great to receive letters like yours because it tells me that I have got the formula for this section correct and I'll do my best to keep it up. Regarding the advertising in the magazine: it just isn't possible to twist the arms of companies into advertising with us. Everything in our power is done to get the advertisement balance right. I have no objections to the magazine being thicker - but it's not as simple as "Right, I think we'll make it twelve pages extra this month". Those sort of decisions have to be taken by echelons that lie above my head I'm afraid. I'm all for a thicker CDU - then we could have an even longer technical helpline and I'd vote for that any day. More money, you see. Anyway, thanks once again for your comments. I hope that you continue to find CDU informative and helpful.

OCP AND THE 120D

Dear CDU,
Help! Could you please send me information on how to configure a Citizen 120D printer for use with the OCP Advanced Art Studio. Thanking you in anticipation.
Norman Merritt, Cambridge.

Dear Norman,
Certainly I can. Here are the answers that you should enter to the questions as they crop up: s, 4, 8, t, 27 65 8, 480, 27 75 480, 960, 27 76 960, 0, 13, 0. That should take you to the summary screen. If not then check everything again. Entries like "27 65 8" should be entered exactly as shown in the on-screen examples. That should

do the trick and give you the option of double density printing as well.

DIVISION SUMS

Dear CDU,
I need a bit of advice (and a devil of a lot of instruction!). The attached algorithm for calculating the date of Easter Sunday was taken from the "Scientific American" magazine many years ago. When I came across it the other day I decided to try to convert it into a C128 program but found it beyond my capabilities. And so I come to you. A few of the steps ask for the remainder of a division sum to be isolated and others say the same thing about the quotient. Can this be done on this toy machine of mine? All the best and my thanks for a good, readable magazine even if you can't help me on this occasion!
Eric Frost, West Sussex.

Dear Eric,
First of all, why do you say that the C128 is a "toy" machine? The C128 can be turned, through a few pieces of relatively inexpensive hardware, into a very powerful machine. Secondly and more importantly, I can help you. If you are performing the sum N/D (variable 'N' divided by variable 'D') then you can calculate the quotient by using the formula Q=INT(N/D). This takes just the whole number part of the answer. For example, 164/29=5.655 (5, remainder 19). The INT function isolates the number before the decimal point so the quotient can be found quite easily. The formula for the remainder is a little more complicated but can be found with a little mathematical trickery: R= ((N/D)- INT(N/D)) *D. To use the previous example again; 'N/D'=5.655, 'INT(N/D)'=5 and therefore the expression in the brackets is equal to 5.655 minus 5 which is 0.655. By multiplying this by the original number that you were dividing by, you will get 0.655 times 29 which you should find is near enough 19, which is indeed the remainder. You therefore have those two equations for calculating the quotient, Q, and the remainder, R. Hopefully that will have been of some assistance.

SNAIL TRAIL

Dear CDU,
If I may be permitted I am sure I can solve at least some of the problems posed by your correspondent PAULETTE YVES (Volume 4, Number 4, Feb 1991). Firstly she should tell her advertising manager that his salary can stay at five figures but in future these will be £123.45 per month. The employee who took an unscheduled holiday in Germany should be told, "We congratulate you on your personal initiative and we don't know how we could manage without you but from Monday we are going to try!" Auf Wiedersehen, Pet (or C64)! Also by allowing your stud snails to feed off your neighbour's lettuces you can cut feeding costs drastically. Software to control the whole operation must be in machine code. This will increase the speed of various snail activities, especially breeding propensity, and will stimulate production and make a lot of snails very happy. A suitable piece of software is marketed by GASTROPOD (Gets All Snails Tabulated Releasing Overwhelming Piles Of Data). I confess I am just a little surprised that PAULETTE YVES, whose name bears a striking resemblance to that of our revered Editor, did not approach him with the business problems as he has shown initiative and competence in keeping CDU magazine alive with the minimum of annoyance to readers.
NEDWOB (whose name bears a striking resemblance to A.H.Bowden), Bishop's Stortford.

Dear Mr.Bowden,
I must thank you for writing in with that help for Paulette. I presume that your last remark is a favourable one for Sir Eves but I must admit that I had not spotted the similarity between Paulette's name and that of our great Editor. The world is full of coincidences, isn't it! I'd like also to thank you for the interesting expenditure pie charts that you included with your original letter which we unfortunately can't print here. Thanks once again for your correspondence, such a great reply to the spoof letter.

TAPE TO DISK

Dear CDU,
I'm a C64 owner and ever since I saw your mag I've been one of your many fans. Well, I think I should get straight to the point so: 1) I've recently bought a 1541-II disk drive and I am transferring all my games from cassette to disk. However, I'm facing problems with the multistage or multiload (or whatever they're called) games - the ones that when you finish one level, then loads the next one. Could you give any help? 2) Where can I find a book telling me what the machine language commands are equivalent to in BASIC. Could you publish it? Quite difficult, right? 3) Is there a possibility of transforming the ILS program for the C128 into one for the C64 (I own the ACTION REPLAY MkVI - will the SLIST command help?). Thanking you in advance. Keep up with the good work and try to put in more competitions.
John Kopsidas, Greece.

Dear John,
The secret for converting tape multiload games to disk lies in the conversion of tape load routines to disk load routines. This is of course a drag to do and so is best left to a backup cartridge. You say you have AR6 but that doesn't really help because it doesn't cater for multiload things. Trilogic's EXPERT Cartridge will be able to help you. It isn't really possible to relate machine code commands to BASIC ones because machine code is so sort of "raw" and down to the bare minimum. One machine code command may require several BASIC ones (and vice versa). However, a number of them can be directly transposed and hopefully if the King of all Editors, Paul Eves, has done his job, you should find a table elsewhere in the Techno-Info section illustrating some of the more common relationships. Regarding ILS, it is not possible to convert the program for use on the C64 directly. Another version would need to be written. The SLIST command idea might work but it would be very tedious and take an eternity, and besides, the BASIC program is very long and you may get an OUT OF

MEMORY error with the large number of variables used by the program. Also, some commands of the C128 are not implemented on the C64. It would be best to wait and see whether the author writes a C64 version.

GEOS COMPATIBILITY

Dear CDU,
I hope that you can be of some help to me. I have a 64C and disk drive and a BROTHER HR-5 printer. The problem is when I load my GEOS disk and I set everything up to start to write, when I get to print what I have typed in my printer does not work. I have set the printer up as Commodore Compatible. Is this that my printer is not compatible with GEOS? I have tried setting my printer with various DIP switches but to no good. Have I bought a pig in a poke or can you set me in the right direction? Also I wonder if you can tell me where I can obtain the following books. I have asked the publishers but they don't sell them now. Any help would be appreciated as I reckon that with all the 64Cs that are still being sold, other users would like to get some books to help them. The books are: The Complete Commodore 64 ROM Disassembly by P.Gerrard and K.Bergin; Impossible Routines for the Commodore 64 by K.Bergin; Further Adventures 64, Will you still love me when I'm 64?, Advanced BASIC and M/C for the 64, Sprites and Sound on the 64, all by P.Gerrard; and Sound Effects and Music on the 64 by W.Turner and A.Velta. Hoping that you can be of help.
H.Lane, Clwyd.

Dear Mr.Lane,
Unless your printer is the Commodore version of the printer, ie the HR-5C, you will not get any results using the Comm.Compat printer driver. This is for printers connected on the serial bus which I presume yours isn't although you didn't say. You will need to purchase an interface for the computer and use a different printer driver if you want all to work perfectly. The printer should work with the computer with a suitable interface. For more information on the necessary interface I would recommend that you telephone a company called Meedmore in Merseyside on 051-521-2202. They are usually very helpful and know what they are going on about. With relation to the correct printer driver, I would ring Tim Harris at FSSL on 0386-553153. He'll be able to take you further on that side of the operation. Regarding the books, the only place I can think of is a good town library. If they are no longer sold by the publishers then it is unlikely that they will still be available in shops. An alternative, however, to one I can recommend. It is "The Anatomy of the Commodore 64", sold by FSSL, which has a complete ROM disassembly of the 64.

128 QUERIES

Dear CDU,
Last year I purchased a Commodore 128D computer which has a 1571 disk drive, MPS-801 printer and a colour monitor. This is the very first computer I have ever owned. This system was chosen above the IBM range - the reason being because it is so user-friendly and I felt it was the ideal computer for the whole family to use. I now buy your magazine each month. There are a couple of problems at the moment that have me confused. Some of your programs load in the 80 column mode. What does this mean exactly? The other problem is that on some of them they have a less than sign after the file type in the directory. You say that if one uses a disk editor to remove this protection one can then load the program. Please could you explain how one goes about doing this and also where would I get a disk editor. Also, how does one move the file entry to the top of the directory? Could you possibly also give me some information on whom I may contact in the UK for the purchase of software by mail. Hoping you can be of assistance to me and also, keep up the good work in your magazine.
B.G.Carroll, South Africa.

Dear Mr.Carroll,
The C128 has a built-in eighty column mode which is only available if you have the necessary monitor or a composite monitor with a certain special lead. Programs for eighty column mode will not produce any effect on the screen of a standard monitor for the forty column mode. These programs have been specially designed with an 80 column screen format in mind. The removal of the less than signs is complicated to explain in text so I have provided a program on the disk, filed as PROB, that will do it for you. Just load and RUN the program, follow all on-screen prompts and then sit-back whilst the protection is removed, after the program has located the file in the directory of course. To help it do this you are asked if you know what sector of the directory track the entry can be found in. It is not possible to know this without a disk editor really so just press 'N' for no and the program will do its stuff. I can only recommend that you possibly invest in a back issue if you can and look back to my article "Directories Explained" in the February 1990 issue if you really want to know detailed things about directory manipulation. That article should provide you with all the information you need for rearranging your directories - unfortunately it is too complicated to go into here but I hope I have been of some help already. There are many companies in the UK that sell software and hardware by mail. There follows just three (not necessarily mail order companies but they do mail things if you get what I mean!) that come to mind straight away: DATEL Electronics Ltd., Fenton Industrial Estate, Govan Road, Fenton, Stoke-on-Trent, England; Evesham Micros, High Street, Evesham, England; (and now a company that is definitely mail order) FSSL, Masons Ryde, Defford Road, Pershore, Worcestershire, WR10 1AZ, England. If you are looking particularly for games software then perhaps you should join the Home Computer Club. I've only got an Amiga version of their advert, but the address on that is The Home Computer **Club, Swindon, SN3 4BR, England.**

CHEATS NEVER PROSPER

Dear CDU,
I have been struggling with machine code for some time. I

use 6510+ Assembler, Trilogic Expert Cartridge, and Codemaster - all indispensable in my quest for enlightenment. CDU is also indispensable and is probably as near perfection as possible. Of course CDU should be published weekly and the low price is too low. My only interest in games is what they can teach me about coding. I have a BASIC loader for a cheat for Spindizzy. It does not appear to work. GOOD! I don't want to cheat or even play Spindizzy. I am more interested in what it does, how it does it, and why it doesn't. The loader starts as follows: 10 REM I LIED, 20 F=679 and then it goes on to read in a load of DATA and store it at that 679 onwards. Conerting it all to machine code I get: SEI, LDA #$34, STA $01, LDX #$06, LDA $0708,X, STA $00DF,X, DEX, BPL somewhere, LDA #$37, STA $01, CLI, RTS. Now for the questions. This routine is supposed to be loaded beginning at address 679. Monitering reset memory I get: 02A5 00 BRK, 02A6 01 00 ORA ($00,X), 02A8 00 BRK. I guess that 02A6 01 00 ORA ($00,X) is there for the benefit of the C64 or perhaps is a product of the cartridge used to reset memory. Question: What is 02A6 01 00 ORA ($00,X) for? Loading the short machine code routine to the specified address causes problems for address 02A6 because the next address is 02A8 - it is supposed to load at 02A7 which is obviously wrong because the instruction at 02A6 takes two bytes. So should I move it to 02A8 and SYS it at 680 decimal? The machine code routine sets the interrupt flag, alters the memory layout, copies some locations across then resets the memory map to default. More questions: I think it is a spoof. Hence the REM statement 'I LIED' in the loader because it just appears to copy blank memory to blank memory. Am I wrong? If so, what does it do? The only thing I can say is that bit 2 of location 1 is zero. Apologies for the garbled nature of this letter. I normally use GeoWrite but my printer's not well. Thanks for any help.
Jim Wyatt, Derby.

Dear Jim,
Glad you think that CDU is near perfection. Sorry it can't be published weekly though! To answer your questions, you have firstly misinterpreted the machine code I'm afraid. You have not taken into account the lo-hi byte storage system. The commands that you identify as LDA $0708,X and STA $00DF,X are in fact LDA $0807,X and STA $DF00,X. This then perhaps makes more sense because the codes for the "I LIED" statement will be poked into memory high up. Spindizzy would I presume check this location high up in memory to see whether the text was there. If so, cheat mode enabled! I don't think that it is a spoof. If it does not in fact work then remove the space before the REM and the "I". That may then help the proceedings. The reason why you get spurious results upon resetting and reading memory is that the area in question is filled with null and important bytes when the machine is reset. Nothing can be stored there permanently so to speak, like it can at places elsewhere in memory. By permanently I mean the code will still be there after a reset, not after switching off and then on again. The code that you disassembled means absolutely nothing and is just the null byte information. To summarise then, the program is probably all right and seems to do something and it is just that that section of memory is reset itself upon resetting of the machine.

110 VOLT PRINTER PROB

Dear CDU,
My problem is with a STAR NEX1000 printer that was purchased in the USA along with a CD Grappler. This system worked perfectly on my brother's Commodore 64 system IN the USA, but when hooked up similarly (with the appropriate 110 volt power supply) and using the same PRINTSHOP package, as again used in the USA, printing occurred but in a haphazard manner. I have enclosed a copy of a letter heading using this system for your perusal and any comment you would care to make to rectify this problem would be appreciated. I do recall, but as this time cannot find, a problem being stated with a STAR printer in a previous issue of CDU where it was stated, as I remember, that the 'internal clock' was causing the fault. Could this be? If so, what is the remedy as it seems to me that all talk of compatibility with Commodore 64s by STAR is hogwash. Not that it isn't a good printer, it's superb, but my MPS-801 is far more compatible. I will close with a heartfelt thanks to all your staff. Pick at this letter as you wish.
A.J.Blackwell, Northants.

Dear Mr.Blackwell,
First of all I must thank you for sending in a sample of what was going on with your printer. It certainly would seem that it is the internal clock that is playing up because each time you produced the letter head, it was doing the same thing which points to a fault that is not just random. It is due to the USA supply being sixty hertz frequency and not 50Hz like it is in the UK. The voltage is only one part of the problem. The printer will be expecting a frequency higher than that given to it which will, in simple terms, confuse it. I think that Tandy shops sell converters although most large, good electrical companies will be able to tell you the best thing to buy. You need a supply frequency of 50Hz. Failing that I can't see why the printer is playing up. Regarding STAR compatibility - I can say with total confidence that talk of compatibility with Commodore is NOT hogwash. I have a STAR LC10 printer myself and am most satisfied with it. It works with every printing program that I have and as of yet I have come across no problems with it. With a bit of luck I will have been of some assistance.

DISK FORMATTED?

Dear CDU,
First of all, from February - happy birthday to Techno-Info, and long live CDU! Here is my problem: in a BASIC program of my own (C64 and 1541), I need to write a data file to disk. Past experience told me to be slightly cautious, so I told the program to perform some tests before writing the file. I'm able to test the following within the program: If the drive is on, if there is a disk in the drive, if the file already exists and needs to be scratched. But I'm not able to test if the disk in the drive is already formatted. Could you help?
Roland Dept, Belgium.

Dear Roland,
The only way to check whether or not a disk has been

formatted that is also easy to understand is by attempting to load the directory. If you get a read error then it is probable that the disk hasn't got a directory and is therefore not formatted. Other circumstances could bring about the same results but I should think that this method will suffice. Execute the line: OPEN 15,8,15: OPEN 2,8,0,"$": INPUT#15,E: CLOSE 2: CLOSE 15. If it ends up that the variable E is not zero - check whether it is the error value for one of the read errors that can be found in your drive's manual - then an error has occurred and you may assume that the disk is worth formatting because it can't find a directory.

UPDATE

Now to this month's Techno-Info Update. There is just one thing to report back on. MR SIMON COLLINS OF NORTH HUMBERSIDE has informed us of the POKEs that MR SANDERSON OF SOUTH AFRICA was after in the December 1990 issue. They relate to the game 'GAME OVER' and provide infinite lives for it. You should load up the game, reset the computer, and then enter: POKE 15244,234: POKE 15245,234: SYS2304. Thanks very much for that little bit of help, Simon.

TIP OF THE MONTH

This month you are going to have to put up with a tip from me I'm afraid. Please keep those tips rolling in and I shall publish any that will be of general interest to the readers. But for now, I'm going to give a few hints out to users of BASIC. In all the reference books that I've read there is always something to the effect "The programmer should ensure that loops are complete before he breaks out of them" but it never tells you how to go about that. Following are just a few little simple hints for BASIC programmers. Imagine you have the following program:

```
10 FOR T=1 TO 10
20 IF A$(T)="END" THEN 50
30 NEXT
40 PRINT "NOT FOUND": STOP
50 ...rest of program
```

This would exit with the BASIC stack confused. It is the same as performing a GOSUB and then not RETURNing from it properly (Try RUNning the following: 10 GOSUB 10 - the computer gets confused because you never have a RETURN command executed correctly). Always ensure that for every GOSUB there is a RETURN. Errors occur if there is a RETURN without a GOSUB but not the other way around. Be careful. But back to the loops. Just use a flag:

```
10 F=0: FOR T=1 TO 10
20 IF A$(T)="END" THEN F=1: T=10
30 NEXT: IF F=1 THEN 50
40 PRINT "NOT FOUND": STOP
50 ...rest of program
```

Many of you who already write complicated BASIC programs will have thought that that was below you. Just go back and check your loops and I bet you'll be surprised how many times you jump out of loops without finishing them.

Finally, if your program pauses for a while now and then for no apparent reason, it is because all unwanted variables are being disposed of. To rectify it, execute the command G=FRE(0) or similar (by that I mean the variable 'G' can be changed) at regular intervals. If your program has a menu, give the command just before the menu is displayed. In that way you will get a number of very short pauses instead of one or two very long ones.

I hope that a lot of you will find those quick tips useful. Remember that if you do have any tips then send them to me at the normal Techno-Info address for publication. What's more, if you have any hardware or programming problems, no matter how complex they may seem, please do not hesitate to write to us and we'll do our best to sort them out for you. The address is, as always: CDU TECHNO-INFO, 11 COOK CLOSE, BROWNSOVER, RUGBY, WARWICKSHIRE, CV21 1NG. See you all next time when it will be June, the month of my birthday - so I expect lots of cards to arrive by the first!! See yer then.

Some machine code commands and the equivalent BASIC statements for JOHN KOPSIDAS OF GREECE:

```
ADC #xx;  A=A+xx+C: IF A>255 THEN A=A-256: C=1
ADC xxxx;  A=A+PEEK(xxxx)+C: IF...THEN as above
AND #xx;  A=(A AND xx)
ASL A;  effectively A=A*2
ASL xxxx;  effectively POKE xxxx, PEEK(xxxx)*2
BCC xxxx and BCS xxxx;  IF C=0 THEN xxxx and IF C=1...
CLC and SEC;  C=0 and C=1
CMP #xx f/b BEQ xxxx;  IF A=xx THEN xxxx
CMP xxxx f/b BNE yyyy;  IF PEEK(xxxx)<>A THEN yyyy
CPX #xx f/b BEQ xxxx;  IF X=xx THEN xxxx
DEC xxxx;  POKE xxxx, PEEK(xxxx)-1: to 255 if =0
INC xxxx;  POKE xxxx, PEEK(xxxx)+1: to 0 if =255
INX and INY;  X=X+1 and Y=Y+1: check as for ADC #xx
JMP xxxx;  GOTO xxxx
JSR xxxx;  GOSUB xxxx
LDA #xx;  A=xx
LDA xxxx;  A=PEEK(xxxx)
LDA xxxx,X;  A=PEEK(xxxx+X)
LDA (xx),Y;  A=PEEK((PEEK(xx)+PEEK(xx+1)*256)+Y)
LDX/LDY as appropriate for LDA
LSR as for ASL but divide not multiply
NOP very short pause
ORA #xx;  A=(A OR xx)
ORA xxxx,Y;  A=(A OR PEEK(xxxx+Y))
RTS;  RETURN
SBC #xx;  A=A-xx
STA xxxx;  POKE xxxx,A
STA xxxx,Y;  POKE xxxx+Y,A
STA (xx),Y;  POKE (PEEK(xx)+PEEK(xx+1)*256)+Y,A
STX/STY as appropriate for STA
TAX and TXA;  X=A and A=X
TAY and TYA as for TAX/TXA
```

NUMBERS AND BYTES

Another excursion into the arithmetics of our computers by JOHN SIMPSON

The third in a series of single part discussions dealing with most aspects of programming computers. This discussion deals with the fundamentals of Binary Coded Decimal (BCD)

The first principle of representing BCD is to encode each decimal digit separately. To do this will require the use of four binary bits, or a half byte (commonly referred to as a nyble). As you may remember from an earlier discussion, a nyble can hold the decimal representation of 0 through to 15, but as we now only require the digits 0 through to 9 for decimal, the representation of 10 through to 15 can be disregarded, see table 1.

BITS	DEC	BITS	DEC
0000	0	1000	8
0001	1	1001	9
0011	2	1010	unused
0101	3	1011	unused
0100	4	1100	unused
0101	5	1101	unused
0110	6	1110	unused
0111	7	1111	unused

TABLE 1

By using a full byte size we can, in BCD, represent the numbers from 0 through to 99, that is "1001 1001". For larger numbers we simply add more nybles to meet with the required size of the number. For example, let us use the number 49,382. In this example we will require the use of 5 nybles, or 3 bytes (the final nyble being left unused), to represent our 5 digit decimal number.

Byte 2 Byte 1 Byte 0

0000 0100 1001 0011 1000 0010

— 4 9 3 8 2

When we wish to operate the computer in BCD mode, we must inform the processor of our intention. This is easily done by setting the decimal flag to true (1), which is bit No 3, in the Status Register. There is an Assembler coded instruction for this purpose, namely "SED" (SEt Decimal mode). The effect of this will ensure that during arithmetic operations on bytes, an "internal" carry, or borrow, will be generated so that each nyble will not represent numbers beyond 9.

For example, if a byte contained the BCD value of, say, 27 (0010 0111) and we were to add 6 to this number, then as soon as the Lo nyble reached 9 a carry to the Hi nyble would be generated making the byte value 33 (0011 0011).
If our original example number was 97 and we added the 6, then when the carry from the Lo nyble is added to the Hi nyble, this would cause a further carry to the eighth bit, or, effectively setting the carry flag. (0000 0011 - C=1). At this point we would need to take account of the carry flag, and if we wanted numbers greater than 99 we would need to incorporate another nyble and add the carry to this, and so on (0001 0000 0011 - 103).

As you can see, by using multiple bytes we can easily create numbers as large as will be reasonably needed.

This leads us on to signed BCD numbers. Deciding upon the size of the numbers we require we can "put aside" a section of bytes to handle the maximum size, and by adding an extra nyble for use as either the positive or negative sign of the number. To indicate how many digits are being used it is also possible to incorporate an extra nyble, or byte (depending on quantity), to specify this. Here is an example of a multibyte BCD signed integer:

0 3 0 7 4 5 (3 Bytes)
number
"745"
Sign of
number
0000 = positive
0001 = negative

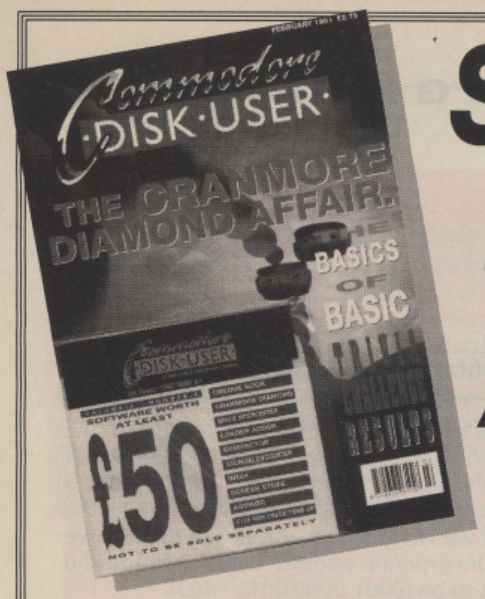
Quantity of
digits in
the number

Fractional numbers can also be represented using BCD. For example the number +7.45, may be represented by:

0 3 0 2 0 7 4 5 (4 Bytes)
———— "745"
number of —
digits +

decimal point
is on the left
of second digit

The great advantage of BCD is that it will yield absolutely correct results. However it does have disadvantages in that it will use large amounts of memory which result in slow arithmetic operations. This is a very useful system for the accounting environment where absolute accuracy is of prime importance. There are also many other areas where unsigned BCD is used to great advantage, more usually within the games environment for keeping tabs on scores, etc.
Okay, then, this wraps up the final discussion. I hope that our short excursion into the realms of Numbers and Bytes has helped clarify some of the aspects of the numbering systems and their effects on bytes for you in some way.



Subscribe now . . . And Save £8

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

We've gone mad and are offering you a once only opportunity of receiving Commodore Disk User's next twelve issues for the staggeringly low price of *£25 if you live in the U.K. We will even post it to you free as well.**

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.

Don't Delay Respond Today

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:

Subscriptions Manager, Alphavite Publications Ltd., 20 Potters Lane, Kiln Farm, Milton Keynes, MK11 3HF.

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

BUSIBASIC

Now you can develop your own Business programs with ease by FERGAL MOANE

Basic is ideally suited to writing 'serious' programs like databases or utilities, where speed is not necessary. It is easy to learn and debug, and a great deal of memory management is done for you. Arrays and strings are easy to store and manipulate. Unfortunately, Basic 2 is not very helpful in easing the workload. This is where Busibasic comes in. It's fifty-odd commands give you a full WIMP environment and assist with I/O and variable manipulation. It has an interpreter as large as the C64's (8K) but still leaves you with 34K for yourself!

POINTS TO NOTE

There are a few things to note about the Busibasic operating system.

1) All ordinary programs will work in the Busibasic environment, but obviously Busibasic programs will only work with Busibasic present.
2) There are a number of commands which use the system interrupt. These will not operate at the same time, and the last executed command will have priority.
3) Watch out for Basic strings corrupting Busibasic! Locations 52 and 56 should always be below 144. A crash will nearly always be the result of corruption.

MEMORY ALLOCATION

Although Busibasic takes up a huge 8K of RAM, you are left with 34K free for Basic. The area from $C000-$D000 and $9000-$A000 are strictly reserved and any attempt to overwrite these areas will crash your program. Zero Page is used extensively, especially locations $FB-$FE. There is no room in the machine for any other machine code, except by lowering Basic.
Busibasic has no preference for screen and graphics data. It does not force any particular location for graphics, and most commands will detect the new screen location and adjust accordingly.

MEMORY MAP

HEX LOCATION	DESCRIPTION
0000-00FA	BASIC WORKSPACE
00FB-00FF	BUSIBASIC PARAMETER BLOCK
0100-02A6	STACK AND ZERO PAGE
02A7-02CC	TEMP. DATA AREA
0300-03FF	COMMAND VECTORS AND BUFFER
0400-07E7	SCREEN
07FB-07FF	ICON POINTERS
0800-8FFF	MAIN BASIC AREA (34K)
9000-9FFF	INTERPRETER AND COMMAND CODE 1
A000-BFFF	BASIC ROM/WINDOW STORAGE
C000-CFFF	COMMAND CODE 2
D000-DFFF	VIC SID CIA IO 1+2
E000-FFFF	KERNEL ROM/FREE STORAGE AREA

BUSIBASIC USERS GUIDE

An alphabetical summary of commands follows. Error massages are discussed and a command summary is given.

$
Syntax: $hexnumber
See also: %

The dollar sign allows hexadecimal numbers to be used in variables, expressions or commands. Just precede the hex number by the dollar sign and it's decimal equivalent will be calculated and processed.
EG. SYS $9400
POKE $D000,$FF
PRINT $FF

%
Syntax: %binarynumber
See also: $

The percent sign followed by a binary number will allow the use of binary constants. Binary numbers are especially useful in setting individual bits in a byte.
EG. POKE 54296,%10001011

@
Syntax: @"disk command"
OR @
or @$
See also: DOS

The ampersand takes the place of the DOS command for sending commands to the disk drive. It has been

included for compatibility with other DOS Support programs.See DOS for a full description of syntax.

.DEFPROC
Syntax: .DEFPROC procname (optional parameters)
See also: .PROC .ENDPROC

DEFPROC defines a procedure. Anyone who has used BBC Basic will know how useful a procedure is. It is in effect a mini-program, separate from the rest of the program, which can be called at any time and parameters passed to it. .DEFPROC defines a procedure with the name procname. This name can be up to 128 characters long and is used to reference the procedure with the .PROC command. The optional parameters allow variables to be specified inside brackets separated by commas. These variables can be passed from the main program to the procedure and used there.

```
EG. 10      .DEFPROC WAIT(N)
    20      TI$="000000"
    30      IF INT(TI/60) < N THEN 30
    40      .ENDPROC
    50      .PROC WAIT(5)
    60      SEC=10
    70      .PROC WAIT(SEC)
```

This procedure allows a program to be paused for a certain number of seconds. The procedure is completely independent of the main program. It is impossible to give a full explanation of procedures in this small space, but see the demo program for examples.

.ENDPROC
Syntax: .ENDPROC
See also: .DEFPROC .PROC

.ENDPROC specifies the end of a procedure. The .ENDPROCs are paired with the last .DEFPROC. Note how the area of program marked by a .DEFPROC/.ENDPROC is completely ignored until it is called by a .PROC command. Procedures have a full-stop in front of the command for easy identification of a procedure command. It is possible to exit before an .ENDPROC

.PROC
Syntax: .PROC procname(optional parameters)
See also: .DEFPROC .ENDPROC

.PROC references a previously defined procedure with the name procname. Control is passed to the procedure and any variables transferred. Execution resumes at the next command after the .PROC when a .ENDPROC is encountered. It is similar to the GOSUB/RETURN construction. Note that the correct number of values must be supplied, but these may be different variables from the ones specified in the .DEFPROC command. This gives increased flexibility. See the example above. The procedure stack can handle around 25 procedures at once. See the error message dictionary.

ARROW
Syntax: ARROW speed,colour
See also: POINTER

ARROW allows parameters to be specified for the POINTER command. The smaller the speed value, the faster the pointer moves. The colour parameter follows normal colour codes.

BAR
Syntax: BAR screen location,colour,height
See also: LOCATE BOX WSTORE WDISPLAY

BAR draws a vertical solid bar with resolution of one pixel. It uses character graphics, and is therefore compatible with ordinary text. The bar is always eight dots wide. Screen location is the number from 1024 to 2024 where the bar is to start in the screen matrix. Height is in pixels. This command allows quite complex histograms to be drawn very quickly. It is invaluable when representing data, and is very effective when combined with boxes and windows.

BEEP
Syntax: BEEP
See also: BOFF

BEEP sets up a 'blip' once a key is pressed. This facility is found on expensive computers, and gives programs that professional touch. It is interrupt driven and will continue until RUN STOP/RESTORE is pressed, BOFF is executed, or it is cancelled by another interrupt command.

BINPUT
Syntax: BINPUT,X,Y,"prompt string",length,variable
See also:

Busibasic INPUT is an extended INPUT command. The Commodore form can be crashed easily and has a number of weak points. This allows the input of a value, string or numeric, to the specified variable. It will occur at the specified XY position and will not exceed the length given. Editing is supported, but the cursor cannot move out of the given area. Once RETURN is pressed, control is returned to your program, with the variable specified containing the input. Note that the prompt string is optional.

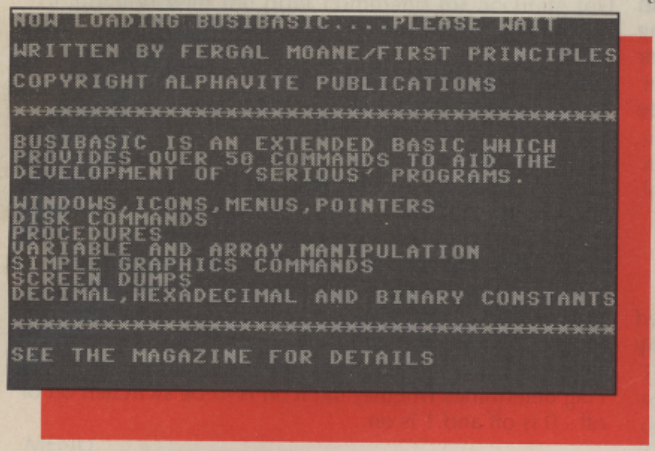
BOFF
Syntax: BOFF
See also: BEEP

This command switches the keybeep off. It will also kill any other undesired interrupt commands and return the HIV to normal.

BOX
Syntax: BOX startX,finishX,startY,finishY,top line,bottom line,colour,reverse flag

See also: WSTORE WDISPLAY

BOX draws a character graphic box of the specified dimensions. It can be used to outline options or menus, and is especially useful with the Window commands. It can outline a specified window to give a professional



effect. The top and bottom line allow two lines to be drawn inside the main box. This allows the highliting of an option, or the underlining of a title. They are Y values only, and should fall between startY and finishY. The reverse flag will fill in the box with the foreground colour if it is 1, or any other value will draw an empty box. See the demo program for effective use of the BOX command.
EG. BOX 10,30,5,14,6,13,1,0

CLINE
Syntax: CLINE linenumber
See also: CLS LOCATE

CLINE clears a horizontal line of the screen. Linenumber is a Y value between 0 and 24. It allows clearing of certain parts of the display without disturbing other screen data. It can also be used for special clear screen effects.

CLS
Syntax: CLS
See also: CLINE

CLS simply clears the screen without any silly hearts or CHR$(147) . God knows why Commodore left this command out in the first place!

COLOUR
Syntax: COLOUR cursor,background,screen
See also:

COLOUR simply sets up the three colours of the screen without having to use any POKEs in difficult to remember locations.

DAPPEND
Syntax: DAPPEND"filename"
See also: DLOAD

DAPPEND joins a program from disk onto the end of the one currently in memory. This allows the construction of tried and tested subroutines on disk to be appended to your latest program. Try and make sure that the line numbers are different or problems could occur. Use the DLIST command to make this check

DIR
Syntax: DIR
See also: DOS @

DIR displays the directory of the disk in the drive without disturbing any memory. Scrolling of the directory can be slowed by holding down the CTRL key.

DLIST
Syntax: DLIST"filename"
See also: TYPE

DLIST will list the Basic program specified directly from disk. This means that the program is not loaded, and other programs can be examined without disturbing the current Basic program. Lines can be grabbed by DLISTing to the correct place, pressing STOP and pressing RETURN over the correct lines. This has proved to be a very useful command. See TYPE for listing other filetypes. Note that DLIST is not very reliable with control characters or BUSIBASIC commands.

DLOAD
Synatx: DLOAD"filename"
See also: DSAVE DAPPEND

DLOAD merely loads the specified Basic program from disk. No ,8 is needed after the filename. DLOAD"*" is much neater than LOAD"*",8

DOKE
Syntax: DOKE location,16 bit value
See also:

DOKE performs a double-byte POKE. It will split the 16 bit value into high and low bytes and store them at location and location+1 in the correct order. This is useful for setting vectors, start of Basic etc. and it is improved when you realise that hex or binary values can be used.

DOS
Syntax: DOS"command"
OR DOS
OR DOS$
See also: @ DIR

DOS is a multi-function command that addresses the disk

operating system. DOS"command" will send any command that would normally be sent by OPEN15,8,15,"command":CLOSE 15. It allows the scratching and copying of files etc. See the drive handbook for more information. These commands are essential when working with business programs.
DOS will display the contents of the disk error channel in reverse video. It will explain the cause of the flashing LED on the drive. DOS should be used after every major disk access.
DOS$ will display the disk directory to the screen. See DIR for more information.

DSAVE
Syntax: DSAVE"filename"
See also: DLOAD

Saves the current Basic program to disk with the specified filename. A save with replace can be executed by placing @0: in front of the filename.

DUMP
Synatx: DUMP
See also: FKEY

Performs a dump of the text screen to a device 4 printer. This is a relatively fast method, and takes care of ASCII conversions. If a printer is not connected, the screen will be dumped to the screen! You may like to assign this command to one of the function keys for a one-key screen dump.

FAST
Synatx: FAST
See also: SLOW

FAST increases the processor clock rate by switching off the screen and sprites and setting interrupts to a minimum. A speed increase of around 10% is usual. This is useful in speeding up operations where the screen is not necessary.

FKEY
Syntax: FKEY"F1,F3,F5,F7,F2,F4,F6,F8"
See also: BOFF

FKEY will assign your own definitions to the eight function keys. The function keys are already defined by Busibasic as below:

F1 - DOS
F3 - DIR
F5 - LIST
F7 - RUN
F2 - DLOAD
F4 - DSAVE
F6 - OLD
F8 - QUIT

SYS $9400 will restore these definitions after RUN STOP/RESTORE. BOFF will disable the function keys.
When defining your own function keys, there is a limit of ten characters on each key. A carriage return is signified by a back arrow. Separate each definition by a comma.

ICON
Synatx: ICON icon number,flag
See also: SETICON POSICON PTRICON

ICON enables or disables one of eight icons. Icons run from 0-7 and are sprites. They follow all the rules for sprites, and the four ICON commands take the place of the dozens of pokes necessary for sprites. Icons can be designed with any sprite editor. They are useful in giving a graphic representation of a program or operation and are essential in a WIMP environment. See the Programmer's Reference Guide for an easy introduction to icons/sprites and the techniques associated with them. Flag determines whether the icon is to be switched on or off - 0 is off and 1 is on.

INSTRING
Syntax: INSTRING target string,search string,result
See also: VARPTR

INSTRING finds the occurrence of one string within another string. It is useful in anything requiring a search facility. The target string is compared with the search string, and the first occurrence of the search string within the target string is noted. This value is passed back in the result variable

EG. INSTRING A$,B$,RESULT
Result will now contain the position of the first occurrence of B$ within A$.

JOY
Syntax: JOY
See also: BOFF MENU

JOY provides a useful editing facility. It allows you to position the cursor by using a joystick in port 2. It simulates the appropriate cursor key press, and can therefore be used within programs requiring cursor movement. It is interrupt driven so BOFF will disable JOY. The fire-button simulates the RETURN key.

KEYWAIT
Syntax: KEYWAIT
See also:

KEYWAIT will wait indefinitely for a keypress. This saves the use of GET and is useful in many applications. This command is not ideal for receiving a value from the keyboard, but location 2 will contain the value that would have been in location 197. See a reference guide for details of these values.

LOCATE
Syntax: LOCATE x,y
See also:CLINE

LOCATE positions the cursor at the x,y position where x is less than 40 and y is less than 25. Use this in conjunction with the print command for accurate text positioning. It is faster and neater than cursor commands in quotes.

LOWER
Syntax: LOWER
See also: UPPER

LOWER switches to the C64's lower-case business mode character set. This reduces the graphics characters available to you, but is essential for wordprocessing. Note that graphics characters used in BOX and BAR may be affected. This command should be avoided if you are moving your character sets.

MENU
Syntax: MENU return variable%,choice array(0),zone
See also: WSTORE JOY

MENU creates drop down menus similar to those found on the Amiga. It stores the screen before the menu is displayed, and replaces it after the choice has been made. Return variable should be an integer; this will contain the number of the chosen option. An ON..GOSUB structure could be used to handle this choice.
The choice array should be a one dimensional string array. The element 0 is displayed at the top of the menu as a title and cannot be chosen. The other elements (up to 23) should be strings of eight characters or less. These choices can be scrolled up and down with the cursor keys, or joystick by using the JOY command. The current option is highlighted in black. Control is returned to your program once a choice has been confirmed by pressing the Return key.
Zone ranges between 0 and 4 and determines where the menu will appear on screen. 0 is flush to the left border while 4 is flush to the right border.

eg. MENU RT%,A$(0),2

MENU stores it's screen information in the same area as the window commands. To avoid corrupting windows, create a dummy window which fills the screen. This will take up the correct amount of space for the MENU screen store, and avoid touching other windows. You should of course not use that particular window.

OLD
Syntax: OLD
See also: QUIT

OLD will resurrect a Basic program after a reset or a NEW command. It will restore pointers and give a listing to the screen. OLD is a very useful and reliable command which will save some angry words.

POINTER
Syntax: POINTER
See also: ARROW JOY

POINTER will display a joystick driven pointer on-screen. It is controlled by the port 2 joystick and cannot move off the screen. The routine will take control of your program until the fire button is pressed, indicating that a screen item has been selected. You can then get three items of information about the pointers location.

251 - X co-ordinate 0-39
252 - Y co-ordinate 0-24
253 - screen code of character under cursor

PEEK the above locations to find the correct information. The X and Y co-ordinates are most useful as they relate to screen positions rather than sprite positions. Use this to decide which action to take.
The cursor speed and colour can be changed with the ARROW command. Note that the pointer occupies one of the eight icons available.
The cursor keys and space can be used to control the pointer.

POP
Syntax: POP
See also:

POP is used to make a premature escape from a subroutine. Subroutines can be branched out of, but the return address is left on the stack and clogs it up. This results in an eventual OUT OF MEMORY error. POP allows the program to ignore the existence of a subroutine. This means that POP is used just before a jump out of a subroutine. This might just be a saviour to your program structure!

POSICON
Syntax: POSICON icon number,X,Y
See also: PTRICON ICON SETICON

This command POSitions an ICON. The icon number is the standard identifier from 0-7. X and Y are in the normal sprite ranges. Experiment to find the border positions on your monitor. There is no need to fiddle about with the MSB of X - X numbers over 255 will be automatically calculated. Note that the other three commands must be used before the icon becomes visible.

PTRICON
Syntax: PTRICON icon number,design block
See also: POSICON ICON SETICON

PTRICON sets the PoinTeR to the appropriate ICON. This

is essential to tell Busibasic where your icon definition starts in memory. The design block is calculated by dividing the address from the start of the bank by 64. For most applications where bank 0 is in use, just divide the address by 64.

eg. 12288 / 64 = 192
start address/ 64 = design block

QUIT
Syntax: QUIT
See also: OLD

QUIT will return to the normal Commodore operating system. It will make Busibasic invisible to the system, but still in memory. Basic programs are not affected by the QUIT command. You can return to Busibasic by typing SYS 37888. This means that you can exit and return to Busibasic at will.

REVERSE
Syntax: REVERSE screen location,number of characters
See also: CLINE

REVERSE toggles specified characters between reverse and ordinary mode. This is useful for highlighting a title, or for drawing a bar. It is thousands of times faster than it's Basic equivalent. The screen location should be the number between 1024 and 2024 in the screen matrix where the reverse should start. The number of characters is obviously the number of screen positions that should be reversed. This could be useful in conjunction with the POINTER to indicate what has been selected.

SETICON
Syntax: SETICON icon number,type,colour,Xexp,Yexp,priority,colour1,colour2
See also: POSICON PTRICON ICON

SETICON sets up parameters for icons. The type should be 0 for hi-res and 1 for multicolour. Colour is obvious. Xexp and Yexp determine if expansion is switched on or off on the icon, 0 for off and 1 for on. Priority dictates the priority of the icon over the background, 0 background and 1 icon. The two following colours are only specified if the icon is in multicolour mode. All colours are from 0 to 15. SETICON should be used along with the other commands to make an icon visible.

SLEEP
Syntax: SLEEP time
See also:

SLEEP makes your program pause for a specified amount of time. This allows the user time to read displays etc. and avoids the use of clumsy FOR..NEXT loops. The time should be between 1 and 255. For a rough guide, multiply the time figure by 4 for the time in seconds. This of course varies with the interrupt rate.

SLOW
Syntax: SLOW
See also: FAST

SLOW restores the screen and reverts to normal speed after a FAST call. It should ONLY be used after FAST has been used. This is because FAST stores essential information in the parameter block at 251. If this has been corrupted by other Busibasic commands, the SLOW command will crash the C64.

SORT
Syntax: SORT array name
See also: VARPTR

SORT performs a speedy sort of a string array into alphabetically ascending order. This is a very useful command in business programs. The array name should only be the first one or two letters of the array name (i.e. no $ string identifier). The array should be one dimensional and have more than one element. See the error messages for more information.

TYPE
Syntax: TYPE "filename,filetype"
See also: DLIST

TYPE displays the contents of a file directly to the screen. This often results in rubbish, but it is very successful with sequential files. Basic programs should be displayed with the DLIST command. The filetypes are shown below:

S - sequential
P - program
L - relative
U - user

UPPER
Syntax: UPPER
See also: LOWER

UPPER returns the C64 to the default capitals/graphics character set.

VARPTR
Syntax: VARPTR variable,return variable
See also: SORT

VARPTR will return the start address of the variable. It will store the address in the return variable specified. This command is limited in it's usefulness.

WCLR
Syntax: WCLR
See also: WDELETE WLOAD

WCLR clears the area under the Basic ROM for the storage of windows. It MUST be used before any window commands, and it is wise to include it at the start of your program. The windows will not work without first being

CLeaRed. This does not apply if you are loading in previously created windows.

WDELETE
Syntax: WDELETE window number
See also: WCLR

WDELETE will delete the specified window from the buffer. It will free up memory for other windows. Make sure the window already exists or you will get a WINDOW ERROR

WDISPLAY
Syntax: WDISPLAY window number,X,Y,flag
See also: WSTORE

WDISPLAY displays a previously stored window to the screen. A copy of the stored window is left in the buffer. The window contents will be displayed at the X and Y cursor position specified. This is flexible, as the X and Y positions for displaying need not be the same as those when storing the window.
Flag has an important bearing on how the window will be displayed:

flag= 0 the window will overwrite the screen
flag= 128 the window will have reversed colours
flag= 64 the window is swapped with the screen
flag= 192 the window will have reversed colours

The last two values are potentially most useful, as repeating the command will restore the original display after the window has been used.

WLOAD
Syntax: WLOAD"filename",8
See also: WSAVE

WLOAD will load a previously stored set of windows into the buffer area under the Basic ROM. This means that windows can be created and stored outside your program, and your program can access them as they are needed. The inclusion of this command means that the number of windows is only limited by disk space. Note that WCLR is not necessary if WLOAD is used.

WSAVE
Syntax: WSAVE"filename",8
See also: WLOAD

WSAVE saves the 8K window buffer area under the Basic ROM to disk. Windows can be created and stored outside of your program, and WLOADed when needed. This shortens your program, and provides scope for a huge amount of information to be stored on disk and accessed when required. WSAVE will only save the number of windows that have been used, therefore using as little disk space as possible.

WSTORE

Syntax: WSTORE window number,X,Y,width,height
See also: WDISPLAY BOX

This command is the heart of the window operating system. It stores a specified section of the screen to an 8K buffer, assigning it a unique window number. Note that a border is not drawn around the windows, the BOX command has been designed for this purpose. You should make sure that all parameters are correct or you will get a WINDOW ERROR. See the Error Dictionary for an explanation of the numbers. Be sparing with your width and height settings or you will run out of memory.

ERROR DICTIONARY

Busibasic issues it's own errors, as well as modifying some existing ones. The error checking in most routines is secure, but you should calculate correct parameters beforehand to avoid mistakes. An explanation of errors is given below.

?ARRAY NOT FOUND

A string array has been referenced in a SORT command which does not exist. Make sure that your array has been dimensioned properly.

?.END WITHOUT PROC

An .ENDPROC has been found without a corresponding .DEFPROC. This is similar to ?RETURN WITHOUT GOSUB. Mkae sure that all procedures have associated .DEFPROCs and .ENDPROCs and you will have no bother. Note that the line number specified is the line in which the procedure has been called by .PROC

?ILLEGAL VARIABLE

An array variable has been used as a parameter in a procedure. This is not allowed.

?NO .END FOUND

A .DEFPROC has been found but with no .ENDPROC associated with it. Always close each procedure with a .ENDPROC

?NOT ENOUGH ELEMENTS

A string array has been specified in a SORT statement and it has only one element. An array of one string cannot be sorted!

?OUT OF MEMORY

The 256 byte procedure buffer has been filled and there is no more room for procedure definitions. There is room for approx. 25 definitions in one program, though this amount is rarely exceeded. You could also have too many nested FOR...NEXT or GOSUB...RETURN structures. Try using POP to escape from subroutines.

?SYNTAX ERROR

The Busibasic interpreter cannot understand your Basic program line, and the error cannot be included in the other categories. Check the spelling of the command, that parameters are in range, that commas are in the right places and that you have supplied the correct amount of parameters.

?TOO MANY DIMENSIONS

The SORT command can only deal with arrays of one dimension.

?TYPE MISMATCH

Busibasic expected a certain variable type which was not provided. Maybe you put a string in place of a number or vice-versa. This can also occur if the variable type expressed in a .PROC is not the same as in the associated .DEFPROC

?UNDEF. PROCEDURE

You have called a procedure by a name which cannot be found in a .DEFPROC command. Check the spelling of both procedure names.

?WINDOW ERROR

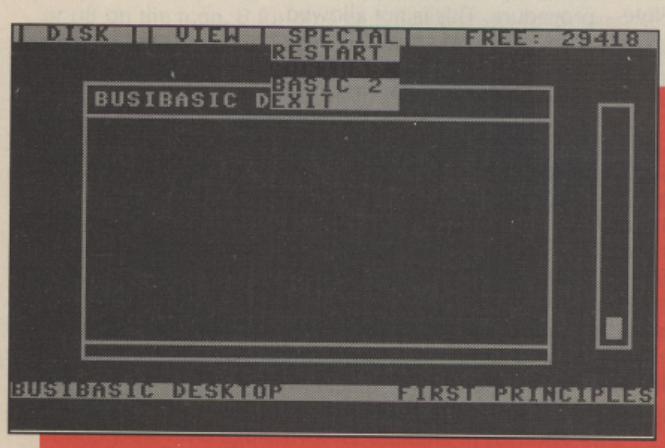
You have made some error in a window related command. Their numbers and meanings are listed below:

ERROR	MEANING
1	SYSTEM FAILURE-RELOAD BUSIBASIC
2	NO WINDOW EXISTS
3	WINDOW ALREADY EXISTS
4	BAD X
5	BAD Y
6	BAD WIDTH
7	BAD HEIGHT
8	BAD X + HEIGHT
9	BAD Y + HEIGHT
10	BAD WINDOW NUMBER
11	NO MEMORY AVAILABLE

DEMO PROGRAM

The Busibasic demo program is loaded from Busibasic by typing DLOAD"BUSIDEMO". It is an attempt to demonstrate how a file handling desktop environment could be created. It is not an elegant piece of programming, but it works and it shows off the system and it's commands. It could be improved further by the utilisation of windows, icons and properly structured subroutines.
When writing this I noticed something odd. I have set the pointers for strings and Basic storage at 52-53 and 55-56 to $8FFF, just below Busibasic. This is to ensure that it does not get corrupted. But during development, I noticed that the Basic 2 system kept resetting the string pointer to $A000. This is in spite of the fact that the highest legal address in 55 and 56 was still $8FFF! This results in the frequent corruption of the Busibasic operating system by variables. Is this a bug in the ROMs or my programming ? Either way, you will have to be very careful about variables; save your program regularly.

When the desktop is up and running you have a joystick or cursor-keys controlled pointer. The bar at the bottom is disk status. The bar on the right is the amount of space on the current disk. The bigger the bar, the more space available. The bar along the top gives free memory and the three pull down menus. The centre window is where all dialog takes place. Click on any of these components and they will react. Below is a description of the menu functions:

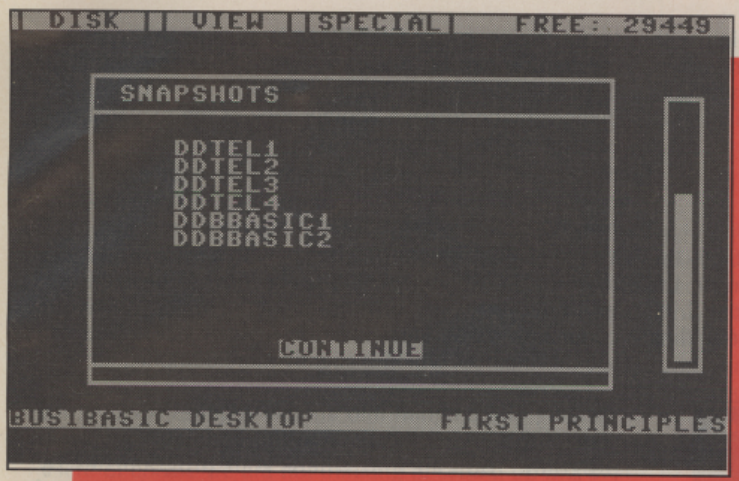


DISK: Normal disk commands. Type the required J6 , and click on cancel or confirm when prompted. Exit will leave this menu.

VIEW: The most important menu. DIR will slowly read the directory in from disk into an array. It will display the filenames in blocks of ten. Click on continue at the bottom of the window to see more filenames. DIR should only be used when a new disk has been inserted. LIST will display the filenames again without any disk access. Click on any of the filenames to load that particular file. This is active at any time when a filename is in the dialog window. The file is presumed to be Basic. SORT will

sort the directory into alphabetical order. DUMP will print the directory to a device 4 printer. When used with sort,, this is a very powerful asset. Use DIR to return to the normal format directory.

SPECIAL: Gives a number of options to exit the program. Choose BUSI to end and list the desktop and BASIC 2 to reset and exit Busibasic.
The desktop program is really self-explanatory. Untidy



but functional. Why no try customising the desktop for your own needs to really get the hang of Busibasic?

CLOSING POINTS

It is difficult to summarise the working of a new Basic language within the constraints of a magazine. There is bound to be something that I have forgotten to mention. My advice: experiment.
The demo program provided should point the way to the using of the commands in a program.
There may even be some programming oversights on my part!
If you have any bother using Busibasic, you can contact me through the magazine.However, exam pressure may postpone a reply. I certainly use Busibasic to make my programming life easier - I hope that you find the same use for it. The Editor may even consider publishing programs written with Busibasic. Why not try your luck and submit your masterpiece - it may earn you some money!

BUSIBASIC COMMAND SUMMARY

$ hexnumber
% binarynumber
@
@"disk command"
@$
.DEFPROC procname (optional parameters)
.ENDPROC
.PROC procname (optional parameters)
ARROW speed,colour
BAR screen location,colour,height
BEEP
BINPUT,x,y,"prompt string",length,variable
BOFF
BOX x1,x2,y1,y1,top line,bottom line,colour,reverse flag
CLINE screen line
CLS
COLOUR cursor,background,screen
DAPPEND "filename"
DIR
DLIST "filename"
DLOAD "filename"
DOKE location,16-bit value
DOS
DOS"disk command"
DOS$
DSAVE "filename"
DUMP
FAST
SLOW
FKEY"f1,f3,f5,f7,f2,f4,f6,f8"
ICON icon number,flag
INSTRING target string,,search string,result
JOY
KEYWAIT
LOCATE x,y
LOWER
MENU return variable%,choice array(0),zone
OLD
POINTER
POP
POSICON icon number,x,y
PTRICON icon number,design block
QUIT
REVERSE screen location,number of characters
SETICON
icon number,type,colour,Xexp,Yexp,priority, col1,col2
SLEEP time
SLOW
SORT array name
TYPE "filename,filetype"
UPPER
VARPTR variable,return variable
WCLR
WDELETE window number
WDISPLAY window number,x,y,flag
WLOAD "filename",8
WSAVE "filename",8
WSTORE window number,x,y,width,height

BASIC SLIDER

Another routine for producing pleasing text. by DAVID READ

I was at my computer one night, experimenting with strings and the commands that deal with them, when, quite by accident, I discovered a small routine that could improve the presentation of a program with a large number of text screens.

HOW IT WORKS

The routine works by taking one character from the string, then two, and so on, and printing them on the beginning of the line. (the word THIS would print S, IS, HIS and THIS.) This gives the effect of the line sliding across the screen.

```
This very simple text-sliding technique
uses two key BASIC commands:

 RIGHT$ and LEN(a$).

How they are used in the subroutine will
be explained later, as will how to put
the subroutine into your own programs.

            Press any key
```

The routine, filed on the disk as SLIDER.SUB, when given a line of text 40 characters long in A$, will slide it across the line on the screen character by character, which would make the screen more interesting to look at. To use the routine is easy. Just make these two commands part of your program:
e.g.

```
10    A$="TEXT 40 CHARACTERS LONG"
20    GOSUB 10000
30    .... rest of program ....
```

The routine supports user-defined characters, and will not interrupt the variables in a program, as long as A$ is not used. You can also put a delay between each letter, by adding the following line to the routine:

10007 FOR <variable>=0 to <delay>:NEXT <variable>

DEMONSTRATION

Also on the disk is a demonstration program called Basic Slider, which shows how the routine works with another program, and uses user-defined characters (from the program CHARS.MC) as a example of the effects that can be made. You then have the option of loading the Basic Slider routine, restarting the demonstration to have another look at the explanations, or resetting the computer.

MERGING IT IN

You can put the routine into your own program by loading the SLIDER.SUB program from the disk and building your program around it, or using a MERGE command from a version of extended basic, or you can use this method, which is slightly more complicated.

Merging one program into another requires resetting the start of the BASIC locations to the end of your current program. You can find where your program lies by PEEKing locations 51 and 52. You then set the start of BASIC locations to these values, and you can now load the routine in. Once it has loaded, POKE 43,1 and POKE 44,8 to get your original program back.
If you wish to use the character set in your own programs, you must put these two lines at the front of your program:

```
1A=A+1:IF A=1 THEN LOAD "CHARS.MC",8,1
2 IF A=2 THEN POKE 53272,24
```

To switch off the characters, type POKE 53272,21

BASIC SLIDER

Basic Slider is only a simple way of sliding text. A machine code version would slide the text much more smoothly, and would allow colours and sound to be used as well, but as this is beyond me (at the moment), perhaps a talented machine code programmer will come up with a better version.

```
Here is the program that slided that
last piece of text....
 10 a$="cdu is a great magazine for Util
ity prog":gosub 10000
 10000 FOR a=LEN(a$) TO 0 STEP -1
 10005 PRINT RIGHT$(a$,LEN(a$)-a);"□"
 10010 NEXT
 10015 RETURN
 Now you should now everything there is
to know about the Basic Slider, and can
use the subroutine in your own programs.
```

ADVENTURE WRITING

Character Animation comes to a close JASON FINCH

This is only a short Adventure Writing just to finish off the section on character animation. In the next article we will start on the programming aspects of the adventure. Yippee! I will have finished with the theory and ideas behind writing an adventure so we will be able to embark on how to actually get the thing up and running.

COMMUNICATIONS

Here of course we are not thinking of setting up our own telephone network or anything throughout the adventure; we are talking about the player communicating on a one-to-one basis with the other characters. You must also bear in mind how the computer will tell the player what actions the other character is performing, if any, and also what is happening elsewhere in the adventure if it is of importance.

TALK TO THE ANIMALS

The most obvious way in an adventure that the player talks to the animals and other characters is by way of the SAY and ASK commands. These usually take the form SAY HELLO TO THE ELF. However, you may want to make your adventure so that quote marks are required around the speech. This is probably best because it clearly defines what the player wishes to say. And how are you going to make your adventure cope with inputs that are "around the wrong way" such as SAY TO THE ELF HELLO. Quote marks make it easier to instantly pick out the text: SAY TO GANDALF "PICK UP THE AXE". I won't go to deeply into the way that your program will analyse that particular command - that will be talked about next month. Suffice it to say that the program will take the word(s) immediately following 'TO' to be the character and the word(s) in the quotes to be what you want to say. It really is as simple as that. You must make a clear distinction between SAY and ASK if you are to include the ASK option. You should ensure that SAY "HELLO" TO THE ELF creates a suitable response whereas something like ASK THE ELF "HELLO" sounds obviously strange. You usually ASK a question, not a statement.

RESPONSES

When the program has sussed out who it is you want to talk to and what it is you wish to say, the computer must decide what a suitable reply for that character would be. This is a simple form of AI (Artificial Intelligence to most people). You must choose how complex you want the communications side of your adventure to be. If you have no characters then all well and good, you don't need to worry about communicating, but the more characters you have, and the more you want the player to rely on speaking to others (which shouldn't be too much!) then the more complicated your speech analysis routine will become. If it isn't all that important then you can have a set of fixed responses to certain statements or questions. Otherwise you can have the routine "think" out what sort of question is being answered and then put together a reply. For instance, the former may result in the computer displaying "The little man says 'no'" if you say to him PUT THE CUP ON THE TABLE, whereas the more advanced reply routine may say "The little man says he wants to hold the cup". This response would be changed if you said PUT THE SWORD ON THE TABLE or something. I hope that will give you some food for thought on the complexity of your routines and whether you actually need a super-advanced system.

WHAT'S HAPPENING?

The last word for now on communications is telling the player what other characters are actually doing. Most of this will be discussed in the programming aspects section next time, but for now just give a bit of thought to the personalities that your characters, if any!, will have. In The Hobbit there was one character that constantly talked to himself. Fine, so you display messages like "The elf talks quietly to himself". But movement is harder to do. The ideas are simple enough, "The guard leaves the room". But how will it be possible to tell whether he can, and how will the computer know where he should then be placed? All will be revealed next month!!

With a bit of luck this won't have extended too far onto a second side, if at all. So without further ado, I shall SAY "CHEERIO FOR NOW" TO THE READERS! See you all again next time, I hope.

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.

SOFTWARE

-THE DEDICATED PD LIBRARY-

IN OUR SECOND YEAR OF SUPPLYING 64 PD, WE STILL CONTINUE TO PROVIDE THE MOST USER FRIENDLY SERVICE AROUND! WE HAVE A WIDE SELECTION OF ALL THE LATEST MIND BLOWING DEMOS THAT REALLY SHOW OFF JUST WHAT THE CBM 64 IS CAPABLE OF DOING. EG. GIANT SPRITE MULTI-PLEXORS, BOBS, VECTORS, PLOTTERS, SINUSES, DYSP'S, DYPP'S & MANY MORE COMPLEX ROUTINES!

MANY DEMOS ARE SO IMPRESSIVE, THEY WOULDN'T EVEN LOOK OUT OF PLACE ON THE AMIGA! EACH ONE IS CREATED BY THE WORLDS BEST DEMO GROUPS & IS PROGRAMMED TO A HIGH STANDARD WITH GREAT GRAPHICS & ORIGINAL MUSIC ALSO INCLUDED. WE ALSO HAVE AN EXCELLENT COLLECTION OF THE BEST PD UTILITIES, MANY OF WHICH ARE BETTER THAN COMMERCIALLY RELEASED ONES! THEY INCLUDE: GRAPHICS, SFX & MUSIC EDITORS, DISK COPIERS, CRUNCHERS, LETTER WRITERS & MANY, MANY MORE. WE EVEN HAVE A FEW PROGRAMMES THAT WILL LET A COMPLETE NOVICE CREATE A DEMO OF THEIR OWN! ALL UTILITIES ARE OF HIGH QUALITY & REALLY HAVE TO BE SEEN TO BELIEVED. IF YOU ENJOY A QUICK GAME OR TWO THEN LOOK NO FURTHER. WE HAVE A RANGE OF THE BEST PD GAMES AROUND, ARCADE, PUZZLE & ADVENTURE. SOME OF WHICH ARE COMMERCIAL QUALITY! WE'RE NOT ONLY THE LEADING SUPPLIER OF 64PD, BUT WE ALSO OFFER FREE EXPERT ASSISTANCE SHOULD ANY PROGRAMMING OR PD PROBLEMS ARISE.

FOR A COPY OF OUR LATEST LIST SEND AN SAE TO THE FOLLOWING ADDRESS, ALTERNATIVELY SEND £2.00 FOR A MORE DETAILED COMPUTERISED LIST & A FEW EXTRA DEMOS.

SILVER WING SOFTWARE
185 CALLOWBROOK LANE,
RUBERY,
BIRMINGHAM . B45 9TG

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

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

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

DEALERS

WORTHING COMPUTERS

Tel: 0903 210861

**7 WARWICK ST
WORTHING
WEST SUSSEX**

Stockists of a large range of 8 Bit and 16 Bit Software and Peripherals.

REPAIRS

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

HOME WORKERS WANTED

(Mailing Envelopes)
£3. per Hour.
STAMPED S.A.E. TO
MATKIZA
27 WOODSIDE PLACE
GLASGO G37QL.

CITIZEN PRINTERS

Silica presents some great offers on the award winning range of high quality dot matrix printers from Citizen. Each Citizen printer is built in the UK to exacting standards, ensuring superb reliability and a very high quality of output. Our confidence in the quality of Citizen printers is such that we are pleased to offer a unique two year guarantee with every printer. Plus, if you purchase your Citizen printer from us, we will give you a Silica Printer Starter Kit (worth £29.95), *FREE OF CHARGE!*

FREE DELIVERY
Next Day - Anywhere in the UK mainland

FREE STARTER KIT
Worth £29.95 - With every Citizen printer from Silica.

FREE COLOUR KIT
Worth £39.95 - With Swift 9 and Swift 24 printers.

2 YEAR WARRANTY
Silica offer a 2 year warranty (including the printer head) with every Citizen printer purchased from Silica.

WINDOWS 3.0
Free Windows 3.0 driver - In the Silica Starter Kit.

FREE HELPLINE
Technical support helpline open during office hours.

MADE IN THE UK
Citizen printers are manufactured to high standards.

144 CPS DRAFT 9 PIN

CITIZEN 120D+

The Citizen 120D+ is one of the UK's best selling printers. It has a stylish appearance and excellent features and performance for such an inexpensive printer. The 120D+ is available with either a serial or parallel interface and is an ideal first printer.

- *9-pin Printhead*
- *Print Speed 144cps Draft*
- *30cps NLQ*
- *Epson & IBM Graphics Emulation*
- *Pull Tractor & Bottom Feed*
- *Superior Graphics - 240×216dpi*
- *FREE Starter Kit*

RRP £228.85
STARTER KIT £29.95
TOTAL RRP: £258.80
SAVING: £110.45
SILICA PRICE: £148.35

£129
+VAT = £148.35

144 CPS DRAFT 24 PIN

CITIZEN 124D

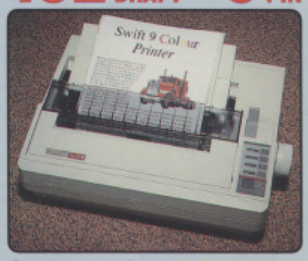
The award winning Citizen 124D brings high quality 24-pin dot matrix printing within every computer users reach. It is the ideal choice where high quality printing is required at a budget price.

- *24-pin Impact Printer*
- *Print Speed 144cps Draft*
- *2 LQ Fonts (48cps)*
- *8K Buffer*
- *Epson, IBM & NEC P6+ Emulation*
- *Advanced Paper Parking*
- *Superior Graphics - 360×360dpi*
- *FREE Starter Kit*

RRP £320.85
STARTER KIT £29.95
TOTAL RRP: £350.80
SAVING: £144.95
SILICA PRICE: £205.85

£179
+VAT = £205.85

192 CPS DRAFT 9 PIN



SWIFT 9 - COLOUR!

The Citizen Swift 9 is perfect for those who require high quality dot matrix black or colour printing at a budget price. The print quality of Swift 9 rivals that of other manufacturer's 24-pin models.

- *9-pin Impact Printer*
- *Print Speed 192cps Draft*
- *3 NLQ Fonts (48cps)*
- *8K Buffer*
- *Epson & IBM Graphics Emulation*
- *Advanced Paper Parking*
- *FREE Starter Kit*
- *FREE Colour Kit*

FREE! COLOUR KIT

RRP £309.35
STARTER KIT £29.95
COLOUR KIT £39.95
TOTAL RRP: £379.25
SAVING: £161.90
SILICA PRICE: £217.35

£189
+VAT = £217.35

192 CPS DRAFT 24 PIN

SWIFT 24 - COLOUR!

The Citizen Swift 24 is one of Europe's best selling printers and has won awards including Printer Of The Year 1990. Its rapid print speed, quality and black or colour options, make it a natural choice.

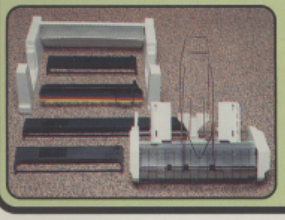
- *24-pin Impact Printer*
- *Print Speed 192cps Draft*
- *4 NLQ Fonts (64cps)*
- *8K Buffer*
- *Epson, IBM & NEC P6+ Emulation*
- *Advanced Paper Parking*
- *FREE Starter Kit*
- *FREE Colour Kit*

FREE! COLOUR KIT

RRP £419.75
STARTER KIT £29.95
COLOUR KIT £39.95
TOTAL RRP: £489.65
SAVING: £191.80
SILICA PRICE: £297.85

£259
+VAT=£297.85

PRINTER ACCESSORIES



SHEET FEEDERS
PRA 1200 **120D** £74.75
PRA 1215 **124D/Swift 9/24** £96.60
PRA 1228 **124D/Swift 9/24** £46.00

SERIAL INTERFACES
PRA 1189 **120D+** £62.10
PRA 1209 **Swift 9/124D** £34.50
PRA 1709 **Swift 24** £28.75

PRINTER STAND
PRA 1242 **124D/Swift 9/24** £26.95

ORIGINAL RIBBONS
RIB 3502 **120D/Swift 9 Black** £4.03
RIB 3924 **124D/Swift 24 Black** £5.18
RIB 3936 **Swift 9/24 Colour** £17.25

COLOUR KITS
PRA 1236 **Swift 9/24** £39.95

All prices include VAT and Free delivery

FREE! STARTER KIT

Every Citizen printer from Silica, comes complete with the Silica Printer Starter Kit, including everything you need to get up and running with your new printer immediately, FREE OF CHARGE!

- *3½" Dual Format Disk with Amiga & ST Printer Drivers*
- *3½" Disk with Drivers for Microsoft Windows 3*
- *2 Metre Parallel Printer Cable*
- *200 Sheets of High Quality Continuous Paper*
- *200 Continuous Address Labels on Tractor Feed*
- *5 Continuous Envelopes on Tractor Feed*

If you already own a printer, and would like a Silica Printer Starter Kit, you may order one (ref: KIT 5000) for the special Silica price of £24.95 - £5 off RRP!

NORMAL RRP **£29.95**

SILICA SYSTEMS OFFERS YOU

- ***FREE OVERNIGHT COURIER DELIVERY:*** On all hardware orders shipped in the UK.
- ***TECHNICAL SUPPORT HELPLINE:*** Team of PC technical experts at your service.
- ***PRICE MATCH:*** We normally match competitors on a "Same product - Same price" basis.
- ***ESTABLISHED 12 YEARS:*** Proven track record in professional computer sales.
- ***£13m TURNOVER (with 60 staff):*** Solid and reliable with maintained growth.
- ***BUSINESS/EDUCATION/GOVERNMENT:*** Volume discounts available for large orders.
- ***SHOWROOMS:*** Demonstration and training facilities at our London & Sidcup branches.
- ***THE FULL STOCK RANGE:*** All of your PC requirements from one supplier.
- ***FREE CATALOGUES:*** Will be mailed to you with offers and software/peripheral details.
- ***PAYMENT:*** By cash, cheque and all major credit cards.

Before you decide when to buy your new printer, we suggest you think very carefully about WHERE you buy it. Consider what it will be like a few months after you have made your purchase, when you may require additional peripherals or software, or some technical help and advice. And, will the company you buy from contact you with details of new products? At Silica Systems, we ensure that you will have nothing to worry about. Silica Systems is one of the UKs leading independent computer dealers and provides a quality service to users at home, in education and in business throughout the nation. Silica have been established for over 12 years, and have an annual turnover of £13 million. With our unrivalled experience and expertise, we can now claim to meet our customers requirements with an understanding which is second to none. But don't just take our word for it. Complete and return the coupon now, for our latest Free literature on the Citizen printer range and begin to experience the "Silica Systems Service".

SILICA SYSTEMS

MAIL ORDER:	1-4 The Mews, Hatherley Rd, Sidcup, Kent, DA14 4DX	**Tel: 081-309 1111**	
Order Lines Open: Mon-Sat 9.00am-6.00pm	No Late Night Opening	Fax No: 081-308 0608	
LONDON SHOP:	52 Tottenham Court Road, London, W1P 0BA	**Tel: 071-580 4000**	
Opening Hours: Mon-Sat 9.30am-6.00pm	No Late Night Opening	Fax No: 071-323 4737	
LONDON SHOP:	Selfridges (1st Floor), Oxford Street, London, W1A 1AB	**Tel: 071-629 1234**	
Opening Hours: Mon-Sat 9.30am-6.00pm	Late Night: Thursday until 8pm	Extension: 3914	
SIDCUP SHOP:	1-4 The Mews, Hatherley Rd, Sidcup, Kent, DA14 4DX	**Tel: 081-302 8811**	
Opening Hours: Mon-Sat 9.00am-5.30pm	Late Night: Friday until 7pm	Fax No: 081-309 0017	

To: Silica Shop, Dept CDU-0591-54, 1-4 The Mews, Hatherley Rd, Sidcup, Kent, DA14 4DX

PLEASE SEND CITIZEN PRINTER INFORMATION

Mr/Mrs/Ms: ..... Initials: ..... Surname: .....

Address: .....

.....

..... Postcode: .....

Tel (Home): ..... Tel (Work): .....

Company Name (if applicable): .....

Which computer(s), if any, do you own? .....

E&OE - Advertised prices and specifications may change - Please return the coupon for the latest information.

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.

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

ROMBO
Limited