THE Magazine for experienced TANDY Colour Computer Users!

$4.50

AUSTRALIAN

COCO MAGAZINE

OPERATION: UTILITIES

Registered by Australia Post – Publication No. QBG 4007 ISSN 0819–9035

WHAT'S ON THE BEST OF CoCoOz

Best of CoCoOz #1. EDUCATION

ROADQUIZ ..... ROB WEBB
SHARE MARKET ..... ALEPH DELTA
HANGMAN ..... ALEPH DELTA
AUSTQUIZ ..... P. THOMAS
ALPHABET ..... ROB WEBB
SPELLING TUTOR..... IAN LOBLEY
TANK ADDITION ..... DEAN HODGESON
FRACTION TUTOR ..... ROBBIE DALZELL
TABLES ..... BARRIE GERRAND
ICOSA ..... BOB WALTERS
KIDSTUFF ..... JOHANNA VAGG
TAXMAN ..... TONY PARFITT
FLAGQUIZ ..... ROB WEBB

Best of CoCoOz #2 part 1 16K GAMES

PYTHON ..... W.ARMSTRONG
COCONUT ..... STEVE COLEMAN
POKERMCH ..... GRAHAM & MATTHEWS
OILSLICK ..... JEREMY GANS
SPEEDMATHS ..... DEAN HODGESON
CCMETEOR ..... BOB THOMSON
BATTACK ..... JEREMY GANS
SKIING ..... JOSHUA GANS
PROBDICE ..... BOB DELBOURGO
RALLY ..... TONY PARFITT
CHECKERS ..... J & J GANS
FOURDRAW ..... JOHANNA VAGG

Best of CoCoOz #2 part 2 32K GAMES

TREASURE ..... DAVIDSON & GANS
SHOOTING GALLERY ..... TOM DYKEMA
MASTERMIND ..... GRAHAM JORDAN
GARDEN OF EDEN ..... DAVE BLUHDORN
ANESTHESIA ..... MIKE MARTYN
YAHTZEE ..... KEVIN GOWAN
OREGON TRAIL ..... DEAN HODGSON
BATTLESHIP ..... CHRIS SIMPSON
ADVENTURE + ... STUART RAYNER
ANDROMEDIA ..... MAX BETTRIDGE
LANDATTACK ..... ALDO DEBERNADIS

Best of CoCoOz #3 UTILITIES

SCREEN PRINT ..... TOM DYKEMA
RAMTEST ..... DYKEMA
PRINT SORT ..... RIES
BEAUTY ..... SON
DATAGEN ..... ROWN
PCOPY ..... DOUGAN
FASTEXT ..... OZ-WIZ
MONITOR + ... FERGUSON
COPYDIR ..... MAS SZULCHA
LABELLER ..... RED BISSELING
SPEED CO ..... PAUL HUMPRIES
2BC ..... WARREN VARNE
CREAT ..... BRIAN FERGUSON
DIS ..... BRIAN DOUGAN
B ..... BOB THOMSON
L ..... GORDON BENTZEN
DIR ..... MORRIE SINGER
HI ..... ALEX.HARTMANN

DELETED

Best of CoCoOz #4 Business

HI ..... ALEX. HARTMANN
(disk; Disk Directory Manager)
PERSMAN ..... PAUL HUMPHREYS
(Personal Finance Management)
BANKSTAT ..... BARRY HATTAN
(Annual & Store Statement)
CC5 ..... GRAHAM MORPHETT
(tape; Sales invoicing)
INSURE ..... ROY VANDERSTEEN
(Analyse Home Contents)
COCOFILE ..... BRIAN DOUGAN
(tape; database)
DPMS ..... PAUL HUMPREYS
(disk; Disk Program Management Sys)
DATABASE ..... PAUL HUMPREYS
(tape; THE tape database)
RESTACC ..... DUNG LY
(tape; Restaurant Accounts)
SPDSHEET ..... GRAHAM MORPHETT
(disk; 22 column spreadsheet)
PRSPDSHT ..... GRAHAM MORPHETT
(disk; prints out "SPDSHEET")
ACS3 ..... GREG WILSON
(disk; Multi disk database)

Best of CoCoOz #5 ADVENTURES

ADV 32K ..... S. RAYNER
QUEST ..... TONY PARFITT
LABYRINTH ..... JAMES REDMOND
ADV + ... SEAN LOVE
CRYSTAL ..... C & K SPRINGETT
PRISON ..... TIM ALTON
OPALTON ..... IAN CLARKE
WIZARD ..... DARRELL BERRY
TREASURE ..... C. DAVIDSON
LOST ..... ALEX. HARTMANN

Best of CoCoOz #6 PRESCHOOL

ALPHABET ..... STUART DAWSON
HATDANCE ..... JOHANNA VAGG
AUSTSONG ..... McDERMOTT FAMILY
ADVANCE ..... McDERMOTT FAMILY
WALTZING ..... McDERMOTT FAMILY
TIMEKANG ..... McDERMOTT FAMILY
BAND ..... McDERMOTT FAMILY
KIDSTUFF ..... JOHANNA VAGG
MATCHER ..... ?
LETTERS ..... JACK FINNEN
BABYSIT ..... JOHANNA VAGG
SPELLING ..... JOHANNA VAGG
SPEEDTAB ..... DEAN HODGSON
10 RACES ..... JOHANNA VAGG

Please Note : Some of the programs on Best of Cocooz # 3 and #4 will not work on the Coco 3.

Best of CoCoOz #7 GRAPHICS

LIL'COCO ..... ANDREW WHITE
THE ROOM ..... HERMANN FREDRIKSON
BACK STREET ..... JOY WALLACE
LOCO ..... MIKE D'ESTERRE
COCO ART ..... SANDY McGREGOR
KANGA ..... JOHANNA VAGG
THE BOAT ..... SANDY McGREGOR
SAD COCO ..... P. BOLLS
TOWER ..... C.A.SYMS
WINDY DAY ..... SARAH LAW
SAILING ..... STEVE YOUNGBERRY
OUTHOUSE ..... STEVE YOUNGBERRY
SMURF ..... JOHANNA VAGG
SUNSTATE ..... STEVE YOUNGBERRY
HELICOPTER ..... ANDREW WHITE
MARTHA ..... ANDREW WHITE
BAD MOON ..... STEVE YOUNGBERRY
MCC ..... JOY WALLACE
EAGLE ..... ?
BLASTER ..... PAUL YOULD
FOGHORN ..... PAUL STEVENSON

Best of CoCoOz #8 16K GAMES

ALIEN ..... STUART SANDERS
QWERL ..... DARRELL BERRY
SHOOTOUT ..... CRAIG STEVART
SHUTTLE ..... CRAIG STEVART
FROG ..... DARREN OTTERY
FROGRACE ..... TOM LEHANE
KINNAT ..... TOM LEHANE
GRANDPRI ..... DOUG GREY
WATER WARS ..... JUSTIN LIPTON
CATERPILLER ..... JUSTIN LIPTON
DETECTIVE ..... VAL STEPHEN
BREAKOUT ..... WHY/BILT

Best of CoCoOz #9 32K GAMES

TRIONINO ..... BOB DELBOURGO
MATCHEM ..... CHARLES BARTLETT
GO ..... BOB DELBOURGO
NARZOD ..... MAX BETTRIDGE
CHOMPER ..... MAX BETTRIDGE
POPBALL ..... MAX BETTRIDGE
LUDO ..... WHY/BILT
SABRE ..... ANDREW SIMPSON
MOVEABOUT ..... KEVIN GOWAN
JIGSAW ..... JAMES REDMOND
LABYRINTH ..... JAMES REDMOND
TANK ..... CRAIG STEWART

Best of CoCoOz #10 Education II

METEOR ..... DEAN HODGSON
DRIVERS TEST ..... ANDREW SIMPSON
SALE ..... JUSTIN LIPTON
TABLES ..... PAT KERMODE
OPALTON ..... IAN CLARKE
CAPITAL LETTERS ..... BOB HORNE
TEST MATCH ..... JEFF SIBEN
SENTENCE ENDINGS ..... BOB HORNE
ESCAPE ..... DEAN HODGSON
RAILMATH ..... BOB HORNE
COUNTDOWN ..... DEAN HODGSON
WHATZIT ..... BOB HORNE
HOMOPHONES ..... BOB HORNE
COMPOUND WORDS ..... BOB HORNE

Best of CoCoOz #11 Education III
This is a DISK only issue!!

CHATVIN MANOR ..... BOB HORNE

TAPE $16 each

DISK $16 each

Telecom

VIATEL®

AUSTRALIA'S NATIONAL VIDEOTEX SERVICE

APPLICATION FORM

DATE OF APPLICATION / /

(BEFORE COMPLETING THIS APPLICATION, PLEASE READ REVERSE SIDE CAREFULLY)

section 1

PLEASE TICK APPROPRIATE BOX TO INDICATE SERVICE REQUIRED

BUSINESS SERVICE ☐ NON-BUSINESS SERVICE ☐

(CHARGES INCURRED ON BUSINESS SERVICES ARE USUALLY TAX DEDUCTIBLE)

SURNAME (OR BUSINESS NAME IF BUSINESS SERVICE) GIVEN NAMES

POSTAL ADDRESS NUMBER/STREET

SUBURB/CITY STATE POSTCODE

TELEPHONE NUMBER ON WHICH SERVICE IS REQUIRED (INCLUDING STD CODE)

section 2

CONTACT NAME (IF BUSINESS SERVICE) GIVEN NAMES

POSTAL ADDRESS FOR BILLING IF DIFFERENT FROM SECTION 1 ABOVE
NUMBER/STREET

SUBURB/CITY STATE POSTCODE

CONTACT TELEPHONE NUMBER (INCLUDING STD CODE)

section 3

PLEASE DESCRIBE NATURE OF BUSINESS (OR OCCUPATION IF NOT A BUSINESS SERVICE)

PLEASE INDICATE TYPE OF EQUIPMENT USED TO ACCESS VIATEL

special instructions

THIS FORM SHOULD BE HANDED IN AT ANY TELECOM BUSINESS OFFICE OR MAY BE MAILED WITHOUT A STAMP TO FREEPOST 20, VIATEL BOX 188C, GPO MELBOURNE, VICTORIA 3001

PLEASE ALLOW TEN WORKING DAYS FOR PROCESSING OF APPLICATION AND RETURN MAIL ADVICE.

telecom use only

DTE PP VN

BG SC CI

GOLDLINK: The place to be on VIATEL

REF

SPAN OVERLAY NO. 140

REGISTRATION AND SUBSCRIPTIONS

Customers must register as a Business Service if the telephone number nominated for the use of the VIATEL Service is a Business Service and/or VIATEL is to be used wholly or mainly for Business, Commercial, Industrial, Professional or Government purposes. (Charges incurred on Business Services are usually tax deductible.)

Where a Business Telephone Service is nominated for the use of VIATEL, but the use of VIATEL is wholly or mainly for Non-Business purposes, the Customer may be registered as a Non-Business VIATEL subscriber, providing the registration is taken out in the Customer's personal name and address and not a Business name.

Telecom Australia will register the Business or Individual named under Section 1 as a Customer of its VIATEL Service and will provide the Customer with a confidential Customer Identity Number and Personal Password by mail.

Where billing address is indicated, bills and bill related correspondence ONLY will be forwarded to that address. All other correspondence will be forwarded to address under Section 1.

Customers should advise VIATEL of any change of address as soon as possible.

If you lose your Customer Identity Number and/or Personal Password, you must advise VIATEL in writing before new numbers are issued. Our postal address is: Freepost 20, Box 188C, GPO Melbourne, Vic. 3001. FOR SECURITY REASONS REPLACEMENT NUMBERS AND PASSWORDS CANNOT BE PROVIDED OVER THE TELEPHONE.

Customers of VIATEL acknowledge that their name and registered VIATEL Number will appear on the VIATEL Mailbox Directory and that Service Providers and/or other registered VIATEL users may send messages to their VIATEL number.

Telecom Australia undertakes no responsibility in relation to the accuracy of the information or service provided by Service Providers on VIATEL. Telecom Australia will not be responsible for any loss or damage arising out of or in any way connected with the use of this information or service.

Attention is also drawn to the terms and conditions governing the provision of information and services by some Service Providers. These terms and conditions may, in some cases, include a disclaimer absolving the Service Provider from liability regarding information and services supplied on VIATEL. The means of accessing these terms and conditions is set out on the Service Provider's Index Page on VIATEL.

Should you require any changes to your existing telephone equipment (e.g. new exchange line, additional socket), please contact your local District Telecom Office.

In a small number of cases VIATEL reception may be unsatisfactory. Correction may incur an additional charge.

BLAXLAND COMPUTER SERVICES PTY. LTD.

SERVICING MODERN TECHNOLOGY

Tandy ELECTRONICS DEALER NO 9251

SOFTWARE-SOFTWARE-SOFTWARE-SOFTWARE-SOFTWARE-SOFTWARE-SOFTWARE-SOFTWARE

OS-9 LEVEL 2 - Includes Basic 09 (Direct from U.S.)	CC3 - $180.00
DESKMATE 3 - For your Coco 3	CC3 - P.O.A.
T.M.S.- 4th Generation Language Database (Single User)	CC2/3 - $299.00
SCULPTOR - 4GL Database Multiuser (OS-9 level 2 req.)	CC3 - $1200.00
THE WIZ - Communications with Windows (L2 & RS232 req.)	CC3 - $160.00
SCREEN STAR - OS-9 Screen Editor with Smart Speller	CC3 - $99.95
SCREEN STAR TEXT FORMATTER - use with Screen Star	CC3 - $69.95
RAMDISK & 512K DIAGNOSTIC - plus utilities	CC3 - $39.95
COLOR SCRIBE - RS/DOS Wordprocessor	CC3 - $99.95
THE WILD WEST - by Tom Mix (Disk)	CC3 - $51.95
NUKE THE LOVE BOAT - Adventure using 512k as ramdisk	CC3 - $69.95
MAGIC OF ZANTH - Adventure with top graphics	CC3 - $69.95
RETURN OF JUNIOR'S REVENGE - All time favourite	CC3 - $69.95
PRO GOLF - 36 Holes, 32K required (Disk)	CC2 - $59.95
MR. DIG - Tape or Disk	CC2 - P.O.A.
ESCAPE: 2012 - 64K Adventure (137 rooms)	CC2 - P.O.A.
TREASURE OF THE AZTECS - 4 voice music, 50 hires screens	CC2 - P.O.A.
ROBOT ODYSSEY - 64K Adventure (Disk)	CC2 - P.O.A.
COMPLETE RAINBOW GUIDE TO OS-9 - 2 Disks, no book	CC2 - $59.95
FIRST RAINBOW BOOK OF ADVENTURES - Book and Tape	CC2 - $32.00
SECOND RAINBOW BOOK OF ADVENTURES - Book and Tape	CC2 - $56.00
THIRD RAINBOW BOOK OF ADVENTURES - Book and Tape	CC2 - $44.00
" " " " " - Book and Disk	CC2 - $54.00
RAINBOW BOOK OF SIMULATIONS - Book and Tape	CC2 - $40.00
SECOND RAINBOW BOOK OF SIMULATIONS - Book and Tape	CC2 - $40.00
INTRODUCTORY GUIDE TO STATISTICS - Book and Tape or Disk	CC2 - $26.00
DISK TUTORIAL - for serious disk basic/ml prog. (2 disk)	CC2/3 - $74.00
COCO GRAPHICS DESIGNER - print your cards, signs,banners	CC2/3 - $59.95
" " " - 100 pre-drawn pictures	CC2/3 - $29.95
3D-GRAPHIMATOR - 3 Dimensional Drawing Design	CC2 - P.O.A.
PROGRAMMERS UTILITY - Commands & Utilities by keystroke	CC2 - P.O.A.
PRINTMASTER	IBM - $96.35
ART GALLERY 1	IBM - $56.65
ART GALLERY 2	IBM - $56.65
NEWSROOM	IBM - $101.95
CLIP ART 1	IBM - $50.95
CLIP ART 2	IBM - $62.35
CLIP ART 3	IBM - $62.35
NEWSMASTER (DESKTOP PUBLISHING)	IBM - $170.00
PRODESIGN (CAD SYSTEM WITH HI-RES PRINTOUTS)	IBM - $630.00
CLIPPER (DBASE III COMPILER)	IBM - $599.00
DAC-EASY ACCOUNTING AUSTRALIAN VERSION	IBM - $250.00
DAC-EASY WORD PROCESSOR	IBM - $129.00
FBS ACCOUNTING SUITE (Inventory,Debtors,Creditors,G/L)	IBM - $830.00
FBS PAYROLL	IBM - $630.00
FBS BILL OF MATERIALS	IBM - $630.00
FBS JOB COSTING	IBM - $630.00
A1 TYPIST WORD PROCESSOR	IBM - $159.00
PEEKS N' POKES FOR IBM	IBM - $66.00
INSIDE TRACK FOR IBM	IBM - $85.00
SPEEDIT PROGRAM EDITOR	IBM - $79.00
PROKEY	IBM - $195.00
MUSICA II DISK OR TAPE	CC2 - $49.00
MUSICA LIBRARY 100/800 EACH SET	CC2 - $49.00
GEOGRAPHY (Africa,South America,Asia,Europe,U.S.A.)	CC2 - $99.95
BOUNCING BOULDERS	CC2/3 - $49.95
GAUNTLET	CC2/3 - $49.95
KARATE	CC2/3 - $49.95
KNOCKOUT	CC2/3 - $49.95
F-16 MISSION ASSAULT	CC2/3 - $49.95
SHOCK TROOPER	CC2 - $33.45
PAPER CHASE	CC2/3 - $49.95
MARBLE MAZE	CC2/3 - $49.95
DECATHLON	CC2 - $33.45
MONEYPOLY	CC2 - $33.45
SAILOR MAN	CC2 - $33.45
COCOTEX VIATEL SOFTWARE	CC2/3 - $99.95
BUSINESS ACCOUNTING SYSTEM	CC2 - $99.95
FAMILY TREE GENEOLOGY	CC2 - $33.45
SUPER BACKUP UTILITY	CC2 - $59.95
TELEWRITER 64 DISK + TAPE	CC2 - $84.45
VIP-WRITER DISK + TAPE	CC2 - $98.95
VIP-DATABASE DISK ONLY	CC2 - $84.45
VIP-CALC DISK + TAPE	CC2 - $84.45
VIP-SPELLER DISK ONLY	CC2 - $77.45
VIP-TERMINAL DISK + TAPE	CC2 - $84.45

BOOKS-BOOKS-BOOKS-BOOKS-BOOKS-BOOKS-BOOKS-BOOKS-BOOKS-BOOKS-BOOKS-BOOKS

AMERICAN RAINBOW - Direct from USA, month of publication	P.O.A.
BACK ISSUE ORDERS - place your order in July and August	P.O.A.
P.C.M. - Portable Computer Monthly, month of publication	P.O.A.
INSIDE OS-9 LEVEL II - A must for Level II	$79.00
500 PEEKS and POKES - for your Coco	$33.95
200 ADDITIONAL PEEKS and POKES - supplement to 500	$19.95
ASSEMBLY LANGUAGE PROGRAMMING - by Barden	$11.95
COMPLETE RAINBOW GUIDE TO OS-9	$39.95
NANOS REFERENCE CARD FOR COCO	$5.95
6809 ASSEMBLY LANGUAGE PROGRAMMING	$41.95
VISICALC APPLICATIONS	$19.95
MULTIPLAN APPLICATIONS	$19.95
UNDERSTANDING COMPUTER SCIENCE	$5.49
DBASE III TIPS AND TRAPS	$37.95
GRAPHICS PRIMER FOR YOUR IBM PC	$45.95
YOUR IBM MADE EASY	$31.99
PC SECRETS	$35.95
PC DOS TIPS & TRAPS	$35.95
YOUR IBM PC	$39.95
GUIDE TO USING LOTUS 1-2-3	$36.95
C LIBRARY	$36.95
USING DBASE III	$37.95
THE 8086 BOOK	$39.95
THE SHAREWARE BOOK	$31.95
MBASIC HANDBOOK	$39.95

HARDWARE-HARDWARE-HARDWARE-HARDWARE-HARDWARE-HARDWARE-HARDWARE-HARDWARE

10MEGABYTE HARD DISK DRIVE - Speed and Mass Storage	CC2/3 - $1299.00
20MEGABYTE HARD DISK DRIVE - Large Business Storage	CC2/3 - $1599.00
512k COCO 3 UPGRADE Includes P.D Software	CC3 - $220.00
TEAC DSDD DRIVE 40T PLUS CONTROLLER - Double Height	CC2/3 - $680.00
TEACx2 DSDD DRIVES 40T PLUS CONTR. - Equal to 4 Tandy.	CC2/3 - $950.00
BARE TEAC DSDD 40T DRIVES - Hi-tech reliability	CC2/3 - $300.00
BARE TEAC DSDD 80T DRIVES - 1.4MB Storage per drive	CC2/3 - $350.00
BARE TANDY SSDD DRIVES - not $519 (subject to stock)	CC2/3 - $250.00
T1000 20MB INTERNAL HD - 1 year warranty, formatted	T1000 - $1399.00
T1000 COMBI RS232/640K/CLOCK/CALENDAR	T1000 - $750.00
THOMSON AMBER/GREEN COMPOSITE MONITORS WITH SOUND	ALL - $350.00
TANDY 1000 SX - 384K, 2 DRIVE	T1000 - $1999.00
TANDY 1000 EX	T1000 - $1299.00
TANDY COLOUR COMPUTER 3 - 128K	CC3 - $449.00
TANDY 102 PORTABLE	T102 - $999.00
TANDY 200 PORTABLE	T200 - $1399.00
MULTIPAK INTERFACE - SWITCHABLE COCO 2/3, LED ON/OFF	CC2/3 - $199.95
VIDEO DRIVER WITH SOUND	CC2 - $38.00
VIDEO DRIVER WITHOUT SOUND (if your monitor has sound)	CC2 - $32.00
MODIFIED ARCHER JOYSTICKS - SUIT COCO 2/3	CC2/3 - $27.00
CITIZEN 120D DOT MATRIX PRINTER - EXTRA TYPEFONTS	ALL - $599.00
DMP-106 DOT MATRIX PRINTER	ALL - $399.00
DMP-130 DOT MATRIX PRINTER	ALL - $599.00
AVTEK MINI MODEM - 300 & 1200/75 baud, economy plus	ALL - $250.00
TANDY INTERNAL MODEM - 300 & 1200/75 + VTEX2 software	T1000 - $699.00
SUPER AUTO MODEM - 300,1200/75,1200 auto ans./dial/con.	T1000 - $820.00
WYSE 30 TERMINALS	CC3 - $999.00

-MISCELLANEOUS-MISCELLANEOUS-MISCELLANEOUS-MISCELLANEOUS-MISCELLANEOUS-

PRINTER PAPER 9-1/16" x 11" PLAIN per 1000 sheets	$17.00
PRINTER PAPER " x " PLAIN per box 4000 sheets	$60.00
PRINTER PAPER 9-1/2" x 11" PLAIN per 1000 sheets clean edge	$25.00
PRINTER PAPER 9-1/2" x 11-2/3" PLAIN A4 per 1000 sheets	$29.00
PRINTER PAPER 11" x 15" PLAIN per 1000	$30.00
PRINTER PAPER 11" x 15" BLUE RULE per 1000 (also SCREEN)	$27.00
NCR 2 PART 9-1/2" x 11" (no carbon required) per 1000	$74.00
MAILING LABELS 11mm x 50mm (4 up) per 1000	$6.00
MAILING LABELS 23mm x 89mm (1 up) per 1000	$16.00
MAILING LABELS 36mm x 102mm (1 up) per 1000	$18.00
CASSETTE LABELS (orange) per100 (10 per sheet)	$5.00 (white) $4.00
BLANK CASSETTES C-10's (10 min.) $1.10 each.	C-30's $1.30
STORAGE CASE, HINGED, HOLDS 5-1/4" holds 100 $25: holds 60	$22.00
STORAGE CASE, HINGED 3-1/2" holds 60	$15.00
3-1/2" DSDD DISKETTES box 10	$55.00
5-1/4" DSDD DISKETTES box 10	$25.00
DISK HEAD CLEANING KIT 3-1/2"	$11.95
DISK HEAD CLEANING KIT 5-1/4"	$11.95
LARGE RANGE PRINTER RIBBON/CARTRIDGES (Tandy/non Tandy)	P.O.A.
LARGE RANGE PRE-PRINTED INVOICE/STATEMENT FORMS AVAILABLE	P.O.A.

bankcard

VISA

AGC CREDIT LINE

BLAXLAND COMPUTER SERVICES PTY. LTD.

134 GREAT WESTERN HIGHWAY,
BLAXLAND 2774 N.S.W.
P.O. BOX 125
PHONE: (047) 39 3903

PRICES SUBJECT TO CHANGE WITHOUT NOTICE.

POSTAGE AND PACKAGING EXTRA

SUBMITTING YOUR WORK

Ah! So you've finally finished that program? And you say to yourself, "What a great program that would make for CoCo Magazine/Softgold Magazine!"

And so you wonder to yourself, "How am I going to send this program in to the magazine?". Some time goes by and you suddenly realise, "Hey, there's an article in this month's magazine about submitting your work. I'll read through that and maybe that'll help me."

So you rip the magazine out of your stack of other CoCo/Softgold magazines and read the article on how to submit your program.

It reads ...

" ... we accept programs stored on both tape and disk ONLY along with a hard copy of the program(s) (optional only; we use it here as a reference to see what the program is/does) and suitable instructions.

Saving to Tape

Each program would be best saved three times with the last save being in ASCII. The tapes we recommend you use are either a C30 or less (the reason for that is that tapes longer than C30 have a tendancy to tear).

It'd be even better if you could include some instructions along with the program, either as a seperate program or in the wordprocessors listed below.

Saving to Disk

With disk, you'd be best to save it three times with the last save being in ASCII. Also, the extension name for the second and third copy should be different, so to distinguish the three copies. A simulation is given below.

' ... I have just saved 3 copies of a program called "HORSE". The directory listing would be:

```
HORSE  BAS  0 B 3
HORSE   1   0 B 3
HORSE   2   0 A 3'
```

Any instructions could be saved in the same system using either a program or in the wordprocessors listed below.

Wordprocessors we use.

Here is a list from our most preferable worprocessors to the drastic measure one could take to tell us how your program works.

1. Telewriter/Telepatch
2. Scripsit
3. PenPal
4. VIP Writer
5. Any form of data file.
6. Instructions written in a seperate program."

"Oh wow!", you think to yourself as you read it with awe and astonishment. So you go about your busy little way saving your program and instructions to tape or disk. Then you say to yourself, "Where do I send it?"

You read the article on ...

" ... any articles and programs should be sent to this address:

Submissions Editor,
Freepost 5
PO Box 1742,
Southport, Qld, 4215

All mail to this address need not be paid for.

All tapes and disks received will be returned after three months in case we need to refer to something or re-print something."

So place your tape/disk along with your hardcopy of the listing in a postpack (or suitable wrapping) and pop it in the mail.

All done!!

Utilities

inside COCO

Articles

Applications

Tutorials

Plus: In a Nutshell, Com Station 642, Dr CoCo, Hints.

Founder: Greg Wilson.
Managing Editor: Graham Morphett.
Editor: Alex Hartman.
Accounts: Karen Court
Production: Paul Wynne
Advertising: Ken Allen

Subeditors: John Redmond, Fred Bisselling, Jack Fricker.

GOLDSOFT

Special thanks to: Geoff Fiala, Martha Gritwhistle, Mike Turk.

Deadlines: 7th of preceeding month.
Phone: (075) 39-6177 or contact Viatel *64213#
Registered Publication QBG 4009.

IN A NUT SHELL

G'day! We here at Goldsoft are introducing quite a new number of things!

The 'Flippie' disk concept

If you subscribe to the disk version of CoCoOz, and you own a version of OS-9 and Basic09, then you can take advantage of our new service.

As the programs become available, we will be putting the normal CoCoOz programs on one side of the disk and put all the OS-9 programs and such on the other side.

So all you need to do is turn the disk over for all the OS-9 programs that appear in the magazine.

The disk itself will be in standard OS-9 format, ie a 35 track, single sided disk, so that everyone can take advantage of this new service.

As well as that, half this month's magazine has been devoted to OS-9 articles and programs.

Contributor's Database

Another new service we are introducing (although this mightn't affect some users) is a contributors' database.

This database will store the following:

* Your name and address,
* The programs you have submitted,
* What magazine that program was published,
* The date your submission was received,
* Whether or not you have received the letter of confirmation (letter stating, "Ah - we've received your program and we are going to put it in the magazine in the near future.", etc ...)
* If it is a disk submission or a tape submission,

... and many more features.

In this way, your program is guaranteed to go into the magazine.

For those interested, all the above was previously written into a big book. Finding this or that, collating information about this program or that program was nearly impossible. This way the database makes everything and anything so much easier.

For example, if I wanted to search the entire database for people who hadn't received their confirmation letter, all I would do is press a few buttons and CoCo would tell me right away.

Contributing a Program

Yes, that is going to change too. The section titled, "Submitting your Work", page 60 in last month's CoCo Magazine has to have a few changes made to it.

For example, we now accept your submission in OS-9 as well. If you have Stylo, you can write your information up on Stylo and send it in to us.

The format of your disk? Well, we cater for the following disk formats:

* 35 track single sided,
* 40 track single sided,
* 40 track double sided,
* 80 track double sided.

... practically anything you can throw at us.

CoCoOz/Softgold Monthly Tapes

I understand that some of you are having problems with the above.

In the magazine, you'll see a label of some sort. It will read either '16K ECB' or '32K ECB' or whatever, and will be located in the top left corner of each program.

Now, these programs have been graded to 'what CoCo they will best fit'. That means that if a label says '16K ECB', it will fit into a 16K CoCo with ECB.

Now you say, "but it won't fit/I get an ?OM ERROR" ... etc - but it will, provided you try one of the following:

* If you have a 16K CoCo with no disk drives, type 'PCLEAR3'. If it still doesn't fit, type 'PCLEAR2', etc. When you reach 'PCLEAR1' and it still doesn't fit, then do this:

```
POKE25,6:POKE3584,0:NEW
```

All the graphics pages that you could once use can now be used for program storage.

* If you have a 16K CoCo with disk drives, do the above - but when it comes to the point of having to POKE CoCo to death, do the following instead:

```
POKE 25,14:POKE3584,0:NEW
```

Generally, if you have any problems with programs, you can do anyone of the following:

* Call Alex on (075) 39-6177,
* Write a letter to Dr.CoCo, c/o Goldsoft, PO Box 1742, Q., 4215, stating your problem and the 'symptoms'.

The above goes for ANY other hassle users might be having, so don't despair - write away or call me!

January's Reference Issue

Every January we bring out a reference issue. It's probably the next best thing to get besides a CoCo and a subscription to Aussie CoCo/ Softgold magazine, mainly because it contains everything that's not in the manual, nor will it be found in any other reference book.

There's only one way to make it great, and that is for you to send in any hints and tips, etc you might have.

Remember, what you think is "common knowledge" is someone else's "answer to everything".

So send it in - it might only be very small, but it will help another user somewhere else!

Also, we are considering doing a printer control code conversion table, eg a ...

```
PRINT#-2,CHR$(31)
```

... tells a DMP 100 to turn the bold print on, while a ...

```
PRINT#-2,CHR$(27)CHR$(31)
```

... does the same thing to a DMP-200.

All we ask of you is to send in a copy of your printer codes from your printer manual and what each specific printer code does.

So far we have the following printers covered:

* DMP100
* DMP110
* DMP120
* DMP130
* DMP200
* Sakata
* Epson
* Gemini
* Amust
* Riteman II

If you have a printer, and it doesn't appear above, then send in a copy of the control codes and what they do.

The idea of all this is this: Let's say you have a program that works for a DMP110. Now you have a DMP130. How do YOU know what any of the printer control codes do?

You probably could find out, after lengthy searching; but wouldn't it be easier to just grab a copy of the January issue of CoCo?

So that's it for this month - have fun until next month!

```
GOLDLINK                    ViaTw 64290136A
     Clubroom                     Member
  726286690   TUE 06 OCT 1987 21:46
     Tonight is Computer night!!

> dBest 2 whats this program you are af
ter do?

   You haven't by any chance come accros
s the control codes for the Meagre, x/y
positioning and so on.

U F O

G'Day ... how are ya?
I've got a dumb question: what is the
correct syntax in order to have
the xcopy and xdir commands to work
properly on OS-9 level 1 (I know it
don't work on 12!)
  Alex

                  *

GOLDLINK                    ViaOn 64290137A
     Clubroom                     Member
  648230650   TUE 06 OCT 1987 21:44
     Tonight is Computer night!!

> UNO the trip to the schol in sydney w
as great. 'C' is great all I need now is
thetime to practice it. Although when I
get back to work ( I'm on holidays for
1 month) I will have to write a couple o
f small programmes to enterface with the
 bata base. I am painting the outside of
 the house at the moment, a wek should p
ull that up and then I'll get into the C
.

                T.N.F.T.S.

When you've finished with your "C"
program, why not send it in? Encourage
some more people to write the same
stuff ...  alex

                  *

GOLDLINK                    ViaTw 64290142A
     Clubroom                     Member
  705471270   TUE 06 OCT 1987 22:16
     Tonight is Computer night!!

> Alex, you need to use xcopy rs%1:name
/ext /d0/name. The part with rs%1 indica
tes a Radio shack disk in drive 1 and
name is a filename. The /ext is also
checked, and any strange combinations
are not accepted. Now, how about telling
me how to make cocomax work on a 3?


   Jeff
  (Xpert)
Syntax to copy rs(1) to os9(0)
 xcopy rs%1:filename/ext /d0/filename
da? cocomax on coco 3? cant, sorry.

                  *

GOLDLINK                    ViaOn 64290144A
     Clubroom                     Member
  648230650   TUE 06 OCT 1987 22:17
     Tonight is Computer night!!

> Alex I had plained to do that, send i
n the C programme that is. The xcopy com
mand is something like xcopy RS%1:fname/
txt /d0/text/fname I have forgotten but
that is close.

                T.N.F.T.S.

Thank you. Next question. iN basic09,
where you have a loop and in one line
in this loop is a question. If it
proves true, it will exit this loop
without finishing the loop off.
 How is that done?
   alex                              _

                  *

GOLDLINK                    ViaTw  6420613A
                       TUE 06 OCT 1987 20:29
                       Member       726286690



>>>>> Does anyone know the cursor cont
rol codes? or where to find them?

U F O
```

```
GOLDLINK                    ViaOn 64290145A
     Clubroom                     Member
  648230650   TUE 06 OCT 1987 22:44
     Tonight is Computer night!!

> Alex I think you mean the loop endloo
p command if so use exitif.

  eg.
  loop
    exitif a=0 then
       (do something)
       (goto a line number)
    endexit
    more lines
  endloop

          T.N.F.T.S.
Great, but I don't want it to go to a
line number ... pseudocode shoulda't
have line numbers ... thank you anyway!

                  *

GOLDLINK                    ViaTw 64290146A
     Clubroom                     Member
  705471270   TUE 06 OCT 1987 22:54
     Tonight is Computer night!!

> WHAT? But the Gnome said you knew how
to fix! In basic09 inside a loop:

EXITIF x<0 THEN PRINT "x<0"
ENDEXIT


          Jeff
         (Gone)
loop: input"name: ",name; if name$="xx"
then(what??); endif; input"address: ",a
adr$; etc ... etc
  Alex

Com Station 642      *      ViaTw 64206112A
                       SUN 27 SEP 1987 17:27
                                  709681090
                                     Member



>>>>> When are we going to see Softwar
e for the Amiga on viatel now that the
new version of SUPERTEX 2.0 is out.

By the way Supertex 2.0 is GREAT...



Its coming soon! G

                  *

GOLDLINK                    ViaOn  6429031A

   The  OS9  Users Board
   TUE 06 OCT 1987 21:11
   Member       648230650
  Sponsored by Paris Radio

> Noel. the hard drive works great on b
oth my coco 2 and now 3 512k. I bought i
t from blaxland computers there adds are
 in coco and softgold. my 5meg HD. cost
$1000 about 9 months ago. I'm a member o
f the national OS9 user group and there
latest Mag. had a list of all the member
s. If you wish to contact them try Ream
e Nichols 9 milham ct. Forestville 2087
or the group BBS 02-451 2954. the modem
there is a multi speed unit.
                        Cheers Ross

                  *

Com Station 642          ViaOn 64291406A
     Tandy 1000 Board     Member
  753517750       TUE 22 SEP 1987 22:58


>  and the information will be revealed
you may use anything but as long as it
is in the format:

SET something=something

To clear it just type:

SET something=
              eg SET Jim`phone=

       and it is cleared..
              Deon       _
```

```
Com Station 642          Viate  6429032A
     OS-9 Users' Board     Member
  234491110       FRI 02 OCT 1987 16:34
    Paris Radio serves OS9 users!!

> Jeff, no I am not... are you sure it
improved with C66 removed? Might try it
on mine...no, second thoughts, might
just leave well alone!! Ron Wright come
to the rescue please! (see previous MB)

Ross, Deskmate3 requires the VDGINT
module up and running. I haven't had a
play yet to see if you can have both it
and GRFINT running, if not just have a
separate boot disc for using Deskmate3
and others which need VDGINT (Koronis
for one, I think).
                        --Rosko!-_

                  *

Com Station 642          ViaTw 64206111A
                       SUN 27 SEP 1987 14:45
                                  934186720
                                     Member



>>>>> Hello all. I have a problem with
regards to certain software bugs, in a
couple of games that I've got. My questi
on is; are there any repair or backup
programs avalible for the Amiga??? Some
of these disks that I have, cannot be
backed-up, owing to protective devices,
so if they come up with a "guru", I've
lost them. Any help or advice would be
greatly appreciated.  P.S. My mb number
is 934186720, thanks.
                              MUN.
                  *

Com Station 642          ViaOn 64291405A
     Tandy 1000 Board     Member
  753517750       TUE 22 SEP 1987 22:52

>  Remember when your busy working on
your computer and the phone rings...
you can't find a pen... and it takes to
long to find the SideKick disk, well,
you don't need it!! Just use the SET
command on your computer (while in the
MS-DOS prompt is there!) For example..
Jim rings and tells you his new phone
number well type:
SET JIM`PHONE=075-510015 and it is stor
ed in the computers enviroment space.
To retrieve it just type SET at the MS-
DOS command ..<CONT_

                  *

Com Station 642          ViaOn 64290210A

 The Tandy Users Board
 SUN 04 OCT 1987 20:52
 Member       755100150   Electronics

> .CoCo Users:
      To eliminate all output to screen
            Poke 359,0
      To restore
            Poke 359,126

                        G

                  *

Com Station 642          ViaOn 64290212A

 The Tandy Users Board
 SUN 04 OCT 1987 22:12
 Member       705471270   Electronics

> Has anyone fixed Cocomax 2 to work on
a Coco3. It seems easy enough (to change
the Joystick input pack to a different
address) and the address of the pointer
in the disk version of the program is
$562C, but there is something more that
needs modifying.
                        Jeff



Yes Jef, they have. I'll have Alex
give the fix tomorrow night. G  _
```

Dear Dr CoCo,

With regard to K.K. Martin's quiry in the September issue I have another solution to the problem which may help. I had the same trouble with this line and I found with a lot of help from our Woodridge user contact that we had, somewhere in the line, held the shift key down when pressing a letter. The shift key is used a lot in this line and if it is held down when a letter is pressed the CoCo accepts the letter as lower case although printing it as a capital. As all commands must be in the uppercase mode, it will give an FM error.

After carefully retyping the line it worked perfectly until the same problem arose in line 570, with the same solution.

My question is why don't the dice change in the 2nd and 3rd throw?

Also, if L.C.W. Bartell would like to get a copy of Australian Rainbow - February 1987 he will find what he is looking for on page 48.

And regardless of what Graham might think, I'll bet it took more than 5 minutes to think up.

Keith McWhirter
Loganlea, Qld

Keith,

The matter regarding the upper/lower case problem can be quite disasterous in a program, especially in say data statements. In some cases (extreme) it can 'warm start' the computer, thus losing your program and all information.

The simple solution is that if there is a line that requires you to press the shift key a lot of times, you type that line VERY slowly - that way you can't go wrong!

As for your problem regarding "Yahtzee" and "why won't the dice change the second and third times?", please re-check lines 730 to 770 and 430 to 450 ... the answer COULD lie within these lines.

*

Dear Dr CoCo,

My daughter is having trouble getting the "Hangman" game in the February issue of CoCo, in line 2000. I went to the Tandy shop in Lismore and they suggested I write to you. My daughter has a 16K machine, and the game is for a Nico.

He said it should still work, but if it doesn't, would you have a have a hangman game that she could play on her computer?

R. Nommenson
Coraki, NSW

Mrs Nommenson,

When one gets an error in a data statement, it is usually when you are telling the computer to READ a number and in actual fact is trying to read a string. This is what is happening here.

In lines 1100 to 1350 the data statements are all numbers while lines 2000 to 2240 are string data statements. Somewhere in the program, the computer isn't reading enough number data statements.

The solution? Check line 200, and see that it reads ...

```
FOR P=1 TO 26:FOR N=1TO 4:
FOR C=1TO 3
```

If I multiply 24*4*3, I get 312. This means that the computer will have to read 312 number data statements before it can read the string data statements.

Also check line 280 ... it should read

```
READ R$
```

... making sure that the string (or the dollar sign) is after the 'R'.

That should solve your problem.

Any more problems? Write me a letter - remember, help is only a letterbox away!

Christmas Party

Don't forget!!

The Christmas Bash on the 12th and 13th December on the beach at Southport is definitely still on!!

There will be Viatel users from all over the east coast - as well as a number of Tandy Computer users! And we will be using a Cellular Phone Viatel terminal to update Viatel LIVE from the Beach all night!

Yes - the party is an all night party - don't expect to leave before 10am!!

FREE-SOFT INTERNATIONAL LIBRARY

OUTSTANDING SOFTWARE

For IBM PC's and Compatibles

At last FREE-SOFT ® International is here! We have more than 900 programmes in our international library, you can get any disk for $12, and if you join FREE-SOFT NETWORK ® it will be $10 only per disk! plus getting our SOFT-LINE ® Newsletter.. full of tips to help you get the most out of your PC! plus receiving regularly our SUPER-SOFT ® sheet listing in details the best 10 software programmes available in our library! plus SPECIAL PRIVILEGED STATUS ® .. you will have advance access to the latest Public Domain and User-Supported software programmes before its release to the public! plus having 24 hours, 7 days hot line to order from! Membership in the NETWORK is $39 annually! So, may we say.. WELCOME to the world of FREE-SOFT.

 # 0125 PC-PROFESSOR — Your computer will teach you all about BASIC programming in colour!

 # 0180 PC-TUTOR — Tutor will teach you all what you need to know about Your PC and its DOS!

 # 1000 PC-WRITE+ — Super word processor, comes in 2 diskettes, this is part 1, full-featured package with 55000 word dictionary in colour, even support a Laser printer.

 # 1001 PC-WRITE+ — Part 2 as above.

 # 0054 SIDE-WRITER — It will allow your printer to print SIDEWAYS on paper! a must for lotus users!

 # 0051 EZ-FORMS — allows you to generate master forms tailored to your need. Super for business.

 # 0028 PC-MUSICIAN — Great programme, you can create and play songs on your PC!

 # 1003 PC-FILE+ — Just when you thought PC-FILE couldn't get any better File+ create new standard in Database managers, comes in 2 diskettes, this is part 1, it is easier, faster and more. . . more powerful.

 # 1004 PC-FILE+ — Part 2 as above.

 # 0130 PERSONAL FINANCE MANAGER — Good personal accounting system. You can keep track of all household money matters from Cheque account to Investments.

 # 0148 PC-TOUCH — Your Computer will be your typing tutor, let you go at your own pace and keep track of how well you are doing.

 # 0147 SLIDE — Images can be created, edited, saved, displayed and printed using the programme. Handy for Disktop Publisher.

SPECIAL ANY 5 DISKS PLUS 1 YEAR MEMBERSHIP ONLY $39

 # 0172 THE LIBRARY for lotus — 20 Super worksheets for lotus 123, from Cheque Book balancer. Cash Flow Manager to New Venture Budget!

 # 0197 HARD DISK UTILITIES — Super collection of Hard disk Utilities from a utility tells you which files have not been backed up to the one helps you create sub-directory no one knows about but you!

 # 0174 KID'S WORD PROCESSOR — Excellent word processor written for Children (and adult too!) in super colour and sound, features graphic menus and the lot!

 # 0175 PC-DRAW # 1 — A must as a part of your Desktop Publishing Library, it is a combination of programmes, providing keyboard, screen drawing, graphics printing and slide show capability.

 # 0176 PC-DRAW # 2 — A selection of drawings and pictures made by PC-Draw #1, plus a super slide show, you must have PC-DRAW #1 to be able to use it.

 # 0201 PROCOMM — The professional communications programme, if you have a Modem then you need Procomm.

 # 0046 PTROOPER — A game, in Super Colour, keep the invading paratroopers from landing in your country!

 # 0049 PC-CHESS — Very good Chess game, you can play against the computer or a friend!

 # 0065 AFGHAN-WAR — Good WAR GAME, in colour based on Afghanistan War.

 # 0157 LANDER — In excellent graphics and colour, can you land a space ship on a pad without crashing?

 # 0165 SPACEWAR — Arcade game in colour and graphics, combines the best features of Asteiods and Startrek with a few tricks of its own!

YES! I want the best! Send me my MEMBERSHIP KIT in FREE-SOFT NETWORK ®, plus the following diskettes. (write catalogue # of any FIVE of the above list)

At your SPECIAL OFFER for $39
Plus, postage & handling ..... $ 3
(if you want more than five diskettes, just add for each extra diskette $10)

As per ☐ My Cheque ☐ Bankcard ☐ Visa ☐ MasterCard ☐ Am. Express

Card No.....Exp. Date.....

Signature .....Name .....

Address.....Sub.....

State.....Postcode.....Phone.....

Post to . . . (No stamps Needed) . . .
Or by using our 24 hrs,
7 days a week,
HOTLINE (03) 859 4697

FREE POST No. (1),
FREE-SOFT International,
P.O. Box 398,
NORTH BALWYN, VIC. 3104.

FREE-SOFT INTERNATIONAL LIBRARY

ASSEMBLER

32K DECB UTILITY

by
Charles Bartlett

I FELT PROMPTED to write this program after receiving my July subscription disk.

On it I found Assembler output files. As I don't have an assembler or the inclination to manually poke in the hex values, I found these files as usful as two left feet (or a Commodore).

So this program will read in such a file and create the binary file from it. The program has a 24k buffer and, except for listings with multiple ORGs, should cope with most listings.

To use it, simply type in the input file name and an output file name and a couple of minutes later the binary file will be stored on disk.

The Listing:

```
0 GOTO10
3 SAVE"30B:3":END'7
10 ' ASSEMBLER
    (C) CHARLES BARTLETT 30/6/87

20 CLEAR1000:FORX=&H03B6 TO &H03
BD:READ A$:POKEX,VAL("&H"+A$):NE
XT:EXEC&H03B6
30 DATA CC,6E,01,1F,02,7E,96,A5
40 BS=&H0E01:BE=&H6E00
50 CLS:INPUT"ASSEMBLER FILE NAME
 ";AN$
60 INPUT"BINARY FILE NAME ";BN$
70 GOSUB530
80 OPEN"I",#1,AN$
90 IF EOF(1)=-1THEN220
100 LINEINPUT#1,A$
110 IFMID$(A$,40,3)="ORG" THEN S
A=VAL("&H"+MID$(A$,7,4)):SS=SA:E
A=BS:GOTO130
120 GOTO90
130 PRINT"ORG ";HEX$(SA)
140 IF EOF(1)=-1THEN280
150 LINEINPUT#1,A$
160 FS=VAL("&H"+MID$(A$,7,4)):IF
 FS<>SA THEN140
170 PRINT:HX$="0000"+HEX$(FS):HX
$=RIGHT$(HX$,4):PRINTHX$;" ";
180 FORZ=12TO25:B$=MID$(A$,Z,1):
IFB$=" "THEN210
190 B$=MID$(A$,Z,2):B=VAL("&H"+B
$):POKE EA,B:EA=EA+1:SA=SA+1:Z=Z
+1:IFEA>BE THEN270
200 PRINTB$;
210 NEXT:GOTO140
220 PRINT"NO ORG FOUND. ASSUME O
RG 0000H":CLOSE:OPEN"I",#1,AN$:S
A=0:SS=0:EA=BS
230 IFEOF(1)=-1THEN260
240 LINEINPUT#1,A$:IF MID$(A$,7,
4)="0000"THEN160ELSE230
250 PRINT"DISK ERROR":END
260 PRINT"CAN'T FIND ADDRESS":CL
OSE:END
270 PRINT"BUFFER FULL":CLOSE:END
280 CLOSE
290 OS=EX-SS:XX=BS+OS
300 SAVEM BN$,BS,EA,XX
310 OPEN"D",#1,BN$,1:IF LOF(1)=0
 THEN CLOSE:KILL BN$:GOTO250
320 FIELD#1,1 AS F$:LF=LOF(1)
330 FOR Q=1TO5:GET#1,Q:BY(Q)=ASC
(F$):NEXTQ
340 B=0:FORQ=LF-4 TO LF:B=B+1
350 GET#1,Q:BE(B)=ASC(F$):NEXTQ
360 LD$=HEX$(BY(4)*256+BY(5))
370 E$=HEX$(BE(4)*256+BE(5))
380 LN$=HEX$(BY(2)*256+BY(3))
390 PRINT
400 OF=VAL("&H"+E$)-VAL("&H"+LD$
)
410 NE=SS+OF
420 BY(4)=INT(SS/256)
430 BY(5)=SS-(256*(INT(SS/256)))
440 BE(4)=INT(NE/256)
450 BE(5)=NE-(256*(INT(NE/256)))
460 H1$="0000"+HEX$(SS):H1$=RIGH
T$(H1$,4):PRINT"START ADDRESS =
";H1$
470 H2$="0000"+HEX$(SS+VAL("&H"+
LN$)):H2$=RIGHT$(H2$,4):PRINT"EN
D ADDRESS  = ";H2$
480 H3$="0000"+HEX$(BE(4)*256+BE
(5)):H3$=RIGHT$(H3$,4):PRINT"EXE
C ADDRESS  = ";H3$
490 FORQ=1TO5:LSET F$=CHR$(BY(Q)
):PUT#1,Q:NEXTQ
500 B=0:FORQ=LF-4 TO LF:B=B+1
510 LSET F$=CHR$(BE(B)):PUT#1,Q:
NEXTQ
520 CLOSE:PRINT:PRINT"COMPLETED"
:END
530 OPEN"I",#1,AN$
540 PRINT"SEARCHING FOR 'END' LA
BEL"
550 IFEOF(1)=-1THEN640
560 LINEINPUT#1,A$:IFMID$(A$,40,
3)="END" THEN 580
570 GOTO550
580 LB$=MID$(A$,48,6):LB$=LB$+"
     ":LB$=LEFT$(LB$,6):CLOSE:OP
EN"I",#1,AN$
590 PRINT"FOUND":PRINT"SEARCHING
 FOR ";LB$;" LABEL"
600 IFEOF(1)=-1THEN650
610 LINEINPUT#1,A$:LC$=MID$(A$,3
2,6):IFLC$=LB$ THEN630
620 GOTO600
630 PRINT"FOUND":PRINT:EX=VAL("&
H"+MID$(A$,7,4)):CLOSE:RETURN
640 PRINT"CAN'T FIND 'END' LABEL
":CLOSE:END
650 PRINT"CAN'T FIND LABEL":CLOS
E:END
```

COCO 3 COLOUR CHART CORRECTIONS

by Brian Bere–Streeter

COLOUR CHART FOR the CoCo 3' by Rick Adams & Dale Lear in the February 1987 magazine, looked good, and I keyed in the program with great anticipation to see 64 colours on screen at once.

But to my dismay I found it would not work correctly. An examination of line 280 in the machine language listing revealed the problem - it was set-up for the American N.T.S.C. TV systems. Rather than fiddle with the machine language section and make a 'hash' of it, I chose to modify the basic section.

Change the following lines:

```
190 HPRINT(4,20),"COLOR COMPUTE
R 3 - COLOR CHART"
240 HLINE(X*80+10,5)-(X*80+40,1
33),PSET,B
260 HPRINT(X*10+5,Y+1),X*16+Y
270 HLINE(X*80+10,Y*8+13)-(X*80
+40,Y*8+13),PSET
280 HPAINT(X*80+20,Y*8+10),8+X,
1
```

The program will now run correctly on Australian 625 line PAL TV systems.

Enjoy the 64 colours on screen simultaneously, and in passing, they look superb on the CM-8 RGB monitor.

BASIC MACHINE LANGUAGE SORT

By George McLintock

UTILITY
32K ECB

BASIC PROGRAMS which handle a large number of strings frequently have a requirement to sort these strings into some specific sequence

If the number of strings to be sorted is small, then it is reasonable to perform the sort as part of the Basic program itself. However as the size of the sort increases, the time required to complete it, using Basic code, increases at a rapid rate.

The obvious solution to the time problem for large arrays is to use a machine language routine to perform the sort.

Sort algorithms are fairly standard, and the main problem with ML sort routines to be used with a Basic program is with the complexity of the interface with Basic (calling sequences), and the flexibility of the functions to be performed as part of the sort, eg if you use a simple single call, then the sort routine is normally restricted to a single sort function like sort the whole array into ascending sequence and change the sequence of the source array as part of the sort.

In general, the more alternatives you provide for the sort, the more complex the calling sequence becomes.

The routine submitted, called ASORT, is a ML routine which provides most of the options likely to be required for sorting arrays, with a graduated level of complexity required for the calling sequences.

Sort options provided include the ability to ...

* sort both string and numeric arrays,
* sort the array into either ascending or descending sequence,
* sort part of an array only
* use a part of each string as a sort key for sorting string arrays,
* perform a non-destructive (or indirect) sort of the array,
  - ie sort the array without changing the order of the array being sorted,
* perform multi-level sorts
  - ie to sort A$(N) within B$(N) within C$(N) etc
  - or to sort A$(N) first part within A$(N) second part
  - or A(N) (numeric) within A$(N) (string)
* sort arrays with more than one dimension,
  - ie to sort A$(X,Y,Z), any dimension, within any of the criteria above,

If required, it will sort a multi-dimensioned array as if it were a simple array, and leave you to work out what you've got at the end of it.

The sort algorithm used is Shell-Metzner which is described later for those who might be interested.

The sort operates by changing the VARPTR's of the elements. The actual string data in string sorts is not moved.

The numeric sort operates my modifying the 5 byte floating point number to a 6 byte string for each comparasion, and performing a normal string type sort with the result.

CALLING SEQUENCE

The method of achieving a simple calling sequence for the routine is based on the use of a separate parameter array to pass values from Basic to the ML routine.

The ML routine is then entered with a single parameter that points to the start of the parameter array.

For those who haven't come across this concept before, it is a way of passing a number of parameters to a ML routine without the requirement to POKE values into memory from the Basic program before calling the ML routine.

While basic provides a convenient way of passing a single parameter to a ML routine (the USR call), if you require more than one parameter, then you would normally POKE the extra parameters into memory where they can be accessed by the ML routine.

However, this becomes messy if you have a requirement for a number of parameters which exceed 255 in value. You finish up with a lot of statements in your program like ...

```
POKE N,INT(N/256):POKE N+1,N -
INT(N/256)*256
```

Even so, parameter arrays are probably not so common with the CoCo because you cannot define variables to be integer, and hence the ML routine has to convert each floating point value in the parameter array to an integer before it can use them.

For some operations with this routine, this conversion is required anyway, so the advantages of using a separate parameter array are more obvious.

The various options available require up to 11 parameter values to be passed to the routine. The calling sequence requires a parameter array to be defined by a DIM command, eg DIM P(10). The various elements of P(X) are then set equal to the values required, and the routine is entered with a call like ...

```
X=USR(VARPTR(P(0)))
```

PARAMETERS USED

The use for each element in the array is as follows (the purpose of each parameter is described in more detail later):

P(0) = number of elements to sort
P(1) = VARPTR of starting

element for the sort
P(2) = VARPTR of matching array
P(3) = switch for ascending /descending sort
P(4) = starting position for sort key in strings
if P(4)=0 the sort is numeric
P(5) = length of sort key

The next 5 elements are for the move routine, which is used in association with some sorts.

P(6) = VARPTR of start of source array
P(7) = VARPTR of start of destination
P(8) = number of elements to move
P(9) = VARPTR of start of pointer elements
P(10) = direction of move

If a parameter is not required for a particular operation, its value can be left zero.

ENTRY POINTS FOR THE ROUTINE

The routine has two entry points for single simple calls, which require two USR entry points to be set, eg

```
DEFUSR0 = 'address 0'
DEFUSR1 = 'address 1'
```

A further facility is available through an EXEC address ('address A3').

These addresses are defined with respect to the start position of the routine. The examples used assume a start address of 32,000, but this will vary on how the routine is used (discussed later).

Address 0 = 32219 is the entry point to perform a sort (single call),
Address 1 = 32000 is the entry point to move VARPTR's (single call),
Address A3 = 32119 is the EXEC address to null VARPTR's.

ALTERNATIVE ENTRY POINTS

For the more complex sorts, I find it more convenient to separate the calls to the routine into a single USR call to initialise the routine, and then use the simple EXEC 'address' command to perform the functions required.

When used in this way, the entry points are:

Address A0 = 32187 - entry point to initialise the routine only
Address A1 = 32223 - EXEC address to perform a sort
Address A2 = 32006 - EXEC address to move VARPTR's
Address A3 = 32119 - EXEC address to null VARPTR's

The separate initialise only routine sets some switches for the ML routine which tend to remain the same for all subsequent calls to the routine for any particular use of it.

Once these have been set, the routine can be used for any number of similar sorts by using the EXEC 'address' calls.

The parameters set by the initialise only routine are:

* The starting address of the parameter array,
* The ascending/descending switch,
* The start position and length of the sort key for string arrays,
* Numeric or string sort.

Once the routine is initialised, these parameters can also be changed by POKE's to specified memory locations.

This is described in the section on using the routine.

CODING RESTRICTIONS ASSOCIATED WITH THE PARAMETER ARRAY

Most of the variables in the parameter array are the VARPTR's of other arrays, which is the address in memory where these arrays are located.

Whenever Basic defines a new simple variable for the program, the actual location of all arrays in memory are altered. Hence, the Basic program that uses these routines must NOT define any new simple variables between the time that the parameter values are set, and a call is made to the routine.

This restriction would still apply even if the values were POKE'd into memory instead of being passed by a parameter array.

If you use the alternative entry points for the routine, then no new simple variables can be defined between the time that the routine is initialised, and any subsequent call to the routine, without re-initialising.

The easy way to ensure that no new simple variables are defined when they should not be, is to include all simple variables used in that part of the Basic program associated with the sort, in the DIM statement which sets up the arrays at the start of the program, eg

```
DIM A$(100),B(100),P(10),X,Y,
A,B,P,Q
```

... etc.

A further advantage of defining counters and other frequently used tempory variables in this way is that the program will also run just a little bit faster as well.

SORTING STRING OR NUMERIC ARRAYS

With these routines, there is very little difference between the operations performed on any array, irrespective of whether the array is a numeric array or a string array.

In this respect CoCo Basic is different to most other Microsoft Basics because all variables have a 5 byte VARPTR, irrespective of whether it is a number or a string.

Hence the same effective operation is performed on all arrays, and the type of variable makes no difference for most operations.

The only difference is the way in which the elements are compared during the actual sort itself, and this is allowed for according to the value assigned to the variable P(4) in the parameter array. If P(4)=0 then a numeric sort is performed, otherwise a string sort is done.

For the VARPTR move routines, no distinction is made between moving the VARPTR for a number or a string.

SAMPLE CODE

As an example of the calling sequences required to use these routines, I have included an outline of Basic programs which could be used to perform the various sorts available.

In all cases, other code will be required between the DIM command and the sample code to set up the initial values of the arrays to be sorted.

SIMPLE SORT OF SINGLE DIMENSIONED ARRAY

Assume a single array of 100 elements to be sorted, eg A$(1) to A$(100), in ascending sequence and A(1) to A(100) in descending sequence.

This could be done with the following Basic code:

```
DIM A$(100),A(100),P(10),X,Y
DEFUSR0 = 32219
 'address 0 entry
P(0)=100
 'number of elements to be
sorted
P(3)=0
 'switch for ascending sort
P(4)=1
 'start position of sort key
(and string sort)
P(5)=255
 'length of sort key, cover all
strings
P(1)=VARPTR(A$(1))
 'starting element for the sort
X=USR(VARPTR(P(0)))
 'do sort with single call
P(4)=0
 'alter for numeric sort
P(3)=1
 'change for descending sort
P(1)=VARPTR(A(1))
 'start element for sort
X=USR(VARPTR(P(0)))
 'do sort with single call
```

COMMON OPTIONS

Ascending/descending sort

The value of P(3) determines if the sort will be in ascending or descending sequence, and it operates independently for all other sort functions.

If P(3)=0 then the sort will be ascending. If P(3) is not equal to zero, the sort will be descending.

Sort key

This is a method of sorting strings, where a part only of the total string is used as a sort key.

The routine will automatically adjust for the actual string length being less than the length of the specified sort key, so you can use any value which is greater than the longest string to be sorted where necessary.

The values of start position equals one, and length of key equals 255 will cover all strings.

String data can be organised in many different ways, and it can be useful at times to be able to sort on a sub-set of the total string.

For example, suburb and postcode could be held in separate strings if you want to sort on both fields. However, if memory is limited, you can reduce string overheads by 5 bytes per entry by combining these fields into a single string, where the first 4 bytes are the postcode, and the rest of the string is the suburb.

For strings of this nature, you can sort on postcode only by specifing the sort key as starting from position 1, with a length of 4, eg,

```
P(4)=1:P(5)=4
```

To sort the same string on suburb, you specify the start position as 5, with a length of the rest of the string, eg

```
P(4)=5: P(5)=250
```

When sorting a numeric array, the parameter at P(4) is set equal to zero to show a numeric sort. The parameter at P(5) is not used for numeric sorts.

SORTING PART OF AN ARRAY

The routine uses two parameters to specify which elements of an array are sorted:

* The number of elements to be sorted,
  - Which is provided by P(0)
* and the address of the first element in the array to be sorted
  - which is provided by P(1),

... so that to sort the top half of an array only, say A$(51) to A$(100) requires:

```
P(0)=50
P(1)=VARPTR(A$(51))
```

... and then call the sort.

Other partial sorts follow a similar procedure, ie to sort elements from A(21) to A(50) use

```
P(0)=30
P(1)=VARPTR(A(21))
```

When using the alternative entry points, the parameters from P(0), P(1) and P(2) are picked up each time, and the routine does not require reinitialising if these are the only three parameters which change.

INDIRECT SORT

This is a method of sorting which allows you to sort an array without altering the order of the array being sorted.

This is done by using a separate matching numeric array as pointers to the elements of the array to be sorted. The numeric array is then sorted into the sequence required for the original source array, and is then used to reference the elements in sorted sequence.

As implemented here, the original source array is effectively copied to another tempory array (T$(N)) which is then sorted into the sequence required.

As part of the sort, the matching numeric array (N(N)), is sorted into the same sequence as T$(N), so that at the end of the sort N(N) is in the same sequence as T$(N), which is the sequence required for A$(N)

To demonstrate the operations performed, the following outlines how this operation could be performed using Basic, where A$(100) is the array to be sorted, and N(100) is the pointer array:

```
DIM A$(100),T$(100),N(100),X,Y
FOR X=1 TO 100
T$(X)=A$(X) ' copy A$ to T$
N(X)=X      'set pointers to
existing sequence
NEXT X
```

A sort routine would then be used to sort the array T$(100).

Each time two elements of T$(N) are swapped during the sort, the same two elements of N(N) are also swapped, so that at the end of the sort, N(N) is in the same sequence as T$(N) which is the sequence required for A$(N).

The array A$(N) can then be printed in sorted sequence with the following loop:

```
FOR X=1 TO 100
PRINT A$(N(X))
NEXT X
```

Other relationships which also apply after the sort are:

```
FOR X=1 TO 100
PRINT T$(X)
NEXT X
```

... which will print T$(N) in the sequence required for A$(N), while:

```
FOR X=1 TO 100
PRINT T$(N(X))
NEXT X
```

... will print T$(N) in the same sequence as A$(N).

The operation in Basic to copy A$(N) to T$(N) uses up free string space because the data for each string in A$(N) is duplicated in T$(N).

To avoid this, I have included a ML move routine, which effectively performs the same function. (Full details of the move routine are provided later)

An example Basic program which performs an indirect sort on A$(100) is as follows:

```
DIM A$(100),T$(100),N(100),P(10
),X,Y
DEFUSR0=32219
 'entry for sort
DEFUSR1=32000
 'entry for move
A3=32119
 'entry to null strings
P(6)=VARPTR(A$(0))
 'start source elements to move
P(7)=VARPTR(T$(0))
 'start destination for move
P(8)=101
 'number of VARPTR's to move
X=USR1(VARPTR(P(0))
 'move VARPTR's '
FOR X=0 TO 100
 'set pointers in N(X)=X

 'pointer array to
NEXT X
 'natural sequence
P(0)=101
 'number of elements to sort
P(1)=VARPTR(T$(0))
 'starting element for sort
P(2)=VARPTR(N(0))
 'start matching pointer array
P(3)=0
 'ascending sort
P(4)=1
 'start sort key
P(5)=255
 'length sort key
X=USR(VARPTR(P(0)))
 'do sort
EXEC A3
 'set T$(N) to null strings
```

SOME POINTS TO NOTE ON THIS SORT

The ML move of A$(N) to T$(N) does not duplicate the string data for A$(N), all it does is copy the VARPTR's of A$(N) to be the VARPTR's of T$(N). This makes T$(N) reference the same string data in memory.

However, if left this way, it may interfer with Basic's normal garbage collect routines the next time that it occurs. To avoid this, the entry at 'address A3' is executed to reset the VARPTR's of T$(n) to null strings.

If used with a numeric array, the routine at 'address A3' will set the numeric elements to zero.

The parameters used for this operation, P(7) and P(8) are the same values as used to move the VARPTR's in the first place, so these would not normally have to be altered before the EXEC A3.

The VARPTR's actually sorted are the VARPTR's of T$(N), and it is this parameter which is used in P(1) for the sort.

The sort moves the VARPTR's of N(N) in the same way as the VARPTR's of T$(N), which leaves N(N) in the desired sequence.

The value of the parameter at P(2) is the VARPTR of the starting element in the numeric array to be used as the pointer array. The setting of this parameter to a non-zero value before calling the routine sets a switch for the extended sort operation.

The other parameter options at P(3) to P(5) work independently for this and any other sort, and the same general procedure can be used with either string or numeric sorts.

The procedure for an indirect sort can also be used to sort two related string arrays into the same sequence. For example if A$(N) and B$(N) are related and you want to finish up with the same relationship maintained between them, but sorted into sequence for A$(N).

All the indirect sort does is to move the VARPTR's of the pointer array in the same way as the VARPTR's of the array being sorted, so that if P(1) is set to VARPTR(A$(0)) and P(2) set to VARPTR(B$(0)), then the relationship between A$(N) and B$(N) will be retained, but both arrays will be sorted into the sequence for A$(N).

The same logic of course applies to numeric arrays as well, and to any mixture of numeric and strings.

the element at P(6) should normally be set to the zero element...

MULTI-LEVEL SORTS

A multi level sort is one where related strings (or numbers) are sorted in sequence with respect to each other, eg, if you have a list of records (music type) where A$(N) contains the artist and B$(N) contains the song, and you want to sort B$(N) (the song) in sequence for each value of A$(N) (the artist).

To do this, you first sort the higher level array (A$(N)) into sequence using an indirect sort as described above.

To sort B$(N) within A$(N), you move B$(N) to T$(N) in the sorted sequence for A$(N), and then sort each group of B$(N) that corresponds to a common value in A$(N), again with an indirect sort.

This second sort will then leave N(N) in the desired sequence, so that you can print both arrays in sequence with ...

```
FOR X = 1 TO N
PRINT A$(N(X)), B$(N(X))
NEXT X
```

If coded in Basic, the move of B$(N) to T$(N) would use code like:

```
FOR X = 0 TO N
T$(X) = B$(N(X))
NEXT X
```

The ML move routine has been set up to do a similar move of VARPTR's, which eliminates the requirement to duplicate the string data in these circumstances as well.

The parameter array element, P(9), controls this aspect of the move. If P(9) is set to the VARPTR of the starting position in the pointer array before the call to the move routine, it will move the source elements to the destination array in sorted sequence.

For this move, the element at P(6) should normally be set to the zero element of the source array, even if moving only part of the array. The detail of the move operation is described later.

An example Basic program which performs a multi-level sort of this nature is as follows:

B$(N) is to be sorted in sequence within A$(N).

Note that the alternative entry points are used in this example.

```
DIM A$(100),B$(100),T$(100),N(
100),P(10),X,Y
DEFUSR0 = 32187
 'entry to initialise only
A1 = 32223
 'EXEC address for sort
A2 = 32006
 'EXEC address for move
A3 = 32119
 'EXEC address to null strings
P(3)=0:P(4)=1:P(5)=255
 'set as before
P(2)=VARPTR(N(0))
 'address matching pointer
array
X=USR(VARPTR(P(0)))
 'initialise routine
FOR X = 0 TO 100
 'set pointers
N(X)=X
 'to natural sequence
NEXT X
P(6)=VARPTR(A$(0))
```

```
 'start source to move
 P(7)=VARPTR(T$(0))
 'start destination for move
 P(8)=101
 'number of VARPTR's to move
 EXEC A2
 'do move A$ to T$ '
 P(0)=101
 'number of elements to sort
 P(1)=VARPTR(T$(0))
 'start of elements to sort
EXEC A1
 'sort T$ for A$ '
 P(6)=VARPTR(B$(0))
 'new source array to move
 P(9)=VARPTR(N(0))
 'start position in pointer
array for move
 EXEC A2
 'move B$ to T$ in sorted
sequence for A$
 'sort B$(N) within A$(N)
 'define each group in A$(N)
which has the same value
 'and sort those sub-groups '
 P(0)=1:Y=0
 'initial values FOR
 X = 1 TO 100
 'loop starts at one for
comparasions
 IF A$(N(X))=A$(N(X-1))THEN
 P(0) P(0)+1:GOTO 300

 ... Line 100 follows.

 IF P(0)=1 THEN 200
 'no sort required
 P(1)=VARPTR(T$(Y)):
 P(2)=VARPTR(N(Y))
 'pointers this sort
 EXEC A1
 'sort subset of string
 ' for this sort, T$(Y) is
starting element to sort
 'N(Y) is start of matching
pointer
 ' and P(0) is number of
elements to sort

 ... Line 200 follows.

 P(0)=1:Y=X
 'reset pointers for next
sub-set sort

 ... Line 300 follows.

 NEXT X
 IF P(0)>1 THEN
 P(1)=VARPTR(T$(Y)):
 P(2)=VARPTR(N(Y)):EXEC A1
 'this test is required at the
end of the loop
 'to sort the last sub-group of
B$(N)
 EXEC A3
 'reset T$(N) to null
```

Further levels of sorting, ie C$(N) within B$(N) within A$(N) etc, can be obtained by repeating the sequence to sort B$ within A$, and substituting C$ for B$, and B$ for A$ in this code.

This allows any number of multiple levels of sorting to be achieved, and any array in the sequence can be either string or numeric.

SOME NOTES ON THIS SORT

The grouping of elements with the same value in A$(N) is coded in Basic. While this operation could be done with another ML routine, the extra time required to perform this particular function in Basic is not excessive, because no copying of string data is required. Hence the gain in time from a ML routine is not that significant, and it is not a repetitive operation anyway.

The description is based on using an indirect sort for multi level sorts. It is also possible to change the order of each array as stored in memory as part of a multi-level sort if required.

The easiest way to achieve this is to follow the general procedure for an indirect sort, and then after each array is sorted in T$(N), move the sorted VARPTR's from T$(N) back to the original source array again.

If doing this, remember to set P(9)=0 before the move back so that you get a simple direct move.

The other thing which will change is that when grouping the elements in A$(N) which are the same, use A$(X) and A$(X-1) in the IF test instead of A$(N(X)) and A$(N(X-1)).

If the multi-level sort involves only two levels ie A$(N) and B$(N), and you want to change the order in memory, then you can use the procedure outlined under indirect sort to sort both A$(N) and B$(N) at the same time, and then perform the loop to sort B$(N) within A$(N).

However, if the multi-level sort extends beyond two levels then you have to use the indirect sort procedure to retain the relationship between all elements.

For sorts of this nature, you also have to ensure that Basic is not allowed to do a garbage collect operation during the sort and associated moves.

The easiest way to ensure that this does not happen is to not use any string variable on the left hand side of an equals sign at any time during these operations in the Basic program.

SORTING MULTI-DIMENSIONED ARRAYS

Multi-dimensioned arrays are held in memory with the highest order of the dimension in adjoining memory locations, eg with DIM A$(20,2,1) the VARPTR's are stored in memory from the start as

```
A$(0,0,0) .... A$(20,0,0)
A$(0,1,0) .... A$(20,1,0)
A$(0,2,0) .... A$(20,2,0)
A$(0,0,1) .... A$(20,0,1)
A$(0,1,1) .... A$(20,1,1)
A$(0,2,1) .... A$(20,2,1)
```

This sort routine requires the elements to be sorted to be held in adjoining memory locations. Hence only the highest order dimension of an array can be sorted directly without some extra setting up, but any one of then be sorted this way.

Where the sort is to be done on the highest order dimension, then the sort is entered normally, eg to sort the array above for elements A$(0,0,1) to A$(20,0,1) set P(0)=21 and P(1) to VARPTR(A$(0,0,1)) and execute the normal sort routine

Any other sort on the highest order dimension can be done in a similar way.

A sort of the lower order dimensions requires an indirect sort. In effect the elements are moved to the T$(N) array and sorted there.

For example, if the array is dimensioned as DIM A$(20,20), and you want to sort the low order elements for the high order elements of zero. ie A$(0,0) to A$(0,20), then the Basic code to move these elements to the T$(N) array would be:

```
FOR X = 0 TO 20
T$(X) = A$(0,X)
NEXT X
```

The ML move routine can also be used to move the VARPTR's of these elements in the same way, so that it can be used to avoid the requirement to duplicate the string data for these moves as well.

THE MOVE ROUTINE

The move routine has five parameters for its full operation P(6) is the address of the first VARPTR in the source array P(7) is the address in the destination array where the first VARPTR from the source

array will be stored. P(8) is the number of VARPTR's to move P(9) is the address of the starting variable in the pointer array to control the move P(10) is a switch to control the direction of the move with a pointer array.

If P(9) is zero, then the move is done as a straight move of the VARPTR's from the source array to the destination array. The parameter at P(10) has no effect when P(9) is zero.

When P(9) is not zero, then the move is done by using the value of each element in the pointer array as an offset from the first VARPTR in the source array, (ie the address in P(6)), to find the actual element in the source array to be moved.

For example, if moving from B$(N) to T$(N) using N(N) as the pointer, and the first three elements of N(N) are:

```
N(0)=15
N(1)=35
N(2)=0
```

... then following the move

```
T$(0)=B$(15)
T$(1)=B$(35)
T$(2)=B$(0)
```

... the parameters for this move are:

```
P(6)=VARPTR(B$(0))
P(7)=VARPTR(T$(0))
P(9)=VARPTR(N(0))
```

If you code your programs in such a way that you don't use the zero elements in arrays (ie you always start your subscripts from 1), then the address of the starting element for the source array (in P(6)) must still be B$(0) when the lowest value in N(N) is 1.

The value in the pointer array is used simply as an offset from the address in P(6) to find the start of the 5 bytes of the VARPTR to move. There is in fact, no check that that it is even in the same array.

As a result of this, the move routine can also be used to move the low order elements of a multi-dimensioned array, but you have to know how far apart they are.

MOVING LOW ORDER ELEMENTS

From the description of how multi-dimensioned arrays are stored in memory, you can work out how many elements there are between each occurrance of the low order elements of such an array, eg for DIM A$(X,Y,Z) as DIM A$(20,2,1), there are 21 elements between each of the 'Y' elements, and 21*3 elements between each of the 'Z' elements.

So that to move low order elements to the T$(N) array for an indirect sort, you set up a separate pointer array containing the offsets required for such a move.

From our earlier example with A$(20,20), to achieve the same move as

```
FOR X = 0 TO 20
T$(X) = A$(0,X)
NEXT X
```

... you would set up values in a separate array, B(20), with:

```
FOR X = 0 TO 20
B(X) = X*21
NEXT X
```

... and call the move routine with:

```
P(6)=VARPTR(A$(0,0))
P(7)=VARPTR(T$(0))
P(8)=21
P(9)=VARPTR(B(0))
```

In effect, the routine starts by moving the VARPTR of A$(0,0) to T$(0). It then skips, 21 elements, and picks up the next VARPTR which is A$(0,1), and moves that to T$(1).

This same set up will work with any of the 21 low order elements in the array. The only parameter which would require a different value is P(6), which should be set to the first of the low order elements to be moved, eg to do the sort with the fifth element, ie for a move like:

```
FOR X = 0 TO 20
T$(X) = A$(5,X)
NEXT X
```

... P(6) would be set equal to VARPTR(A$(5,0)).

SORTING LOW ORDER ELEMENTS

When you actually do an indirect sort of the T$(N) array, you would still use the pointer array A(N) for tracking the elements, ie the pointer array for the move (B(N)), is not the same pointer array that is used for the sort (A(N)).

Multi-level sorts on the low order elements of a multi-dimensioned array thus become a little more complicated.

For this type of sort you require another step to set up the correct values in the pointer array for the move (B(N)), because it is not the same as the pointer array for the sort.

For example, if you are sorting A$(5,X) within A$(0,X), then the first sort would be a normal indirect sort of A$(0,x), using the move as described above.

The Basic code to move A$(5,X) to T$(N) for the second sort (where A(N) is the pointer for the sort) would be:

```
FOR X=0 TO 20
T$(X) = A$(5,A(X))
NEXT X
```

The code to set B(N) to the correct values for the same move would be:

```
FOR X = 0 TO 20
B(X) = A(X)*21
NEXT X
```

... with P(6) set equal to VARPTR(A$(5,0)), which would then move the elements of A$(5,X) to T$(N) in the sorted sequence for A$(0,X), when using B(N) as the pointer for the move, and A(N) as the pointer for the sort.

While this may seem rather complicated, the logic for what is required can be related directly to how you would do it if you were coding it all in Basic, and this can be useful for the more complex sorts. If you don't use multi-dimensioned arrays, then sorts of this nature are not necessary.

However, sorting is normally an incidental requirement for what the whole program is doing. In a number of cases, the overall design and coding of the total program can be made a lot easier by using multi-dimensioned arrays to structure and access the data.

In these circumstances, being able to sort the lower dimensions can be very useful, for simplifing the coding of the rest of the program.

While it would be possible to use Basic code to set up the T$(N) array each time, rather than use the move routines, this requires the string data to be duplicated. If you have large arrays, this can use a lot of memory, which is not required when using the move routines as well.

CHANGING VARPTR's IN MEMORY

The set up as described so far allows for an indirect sort of the low order dimensions of a multi-dimensioned array. Under some circumstances you may in fact want to change the order in which these elements are actually stored in memory, ie to do the same function as a simple sort which actually changes the order in which the elements are stored.

To allow this, I have provided a switch (at parameter P(10)) which makes the move routine do the indirect move in the opposite direction, ie it will move the sorted VARPTR's from T$(N) back to the array A$(20,20) in the same manner as they were extracted before the sort, but they will now be in sorted sequence.

To use this facility, simply set P(10)=1 after the sort, and execute the move routine again.

The same parameter values as set for the original move will then apply for the move back again.

If doing this, set P(10) back to zero again immediately after the move to avoid problems later on if you forget to do it.

If you intend to change the VARPTR's in memory, and you don't want to retain any existing relationships between the high order elements, then you don't need to do an indirect sort on T$(N), you can do a simple sort.

An example program to sort all the high order elements of an array A$(20,20), by changing the VARPTR's in memory follows, ie to do the equivalent of ...

```
FOR X = 0 TO 20
 'do all high order elements
FOR Y = 0 To 20
 'for each low order one
T$(Y) = A$(X,Y)
 'move for sort
NEXT Y
 'sort T$(N) into required
sequence
FOR Y = 0 TO 20
 'move sorted elements
A$(X,Y) = T$(Y)
 'back to A$(20,20)
NEXT Y
NEXT X
```

This in turn is equivalent to writing a special ,sort routine in Basic to do the comparasions on the low order elements of a two dimensioned array sort.

Example program follows.

```
DIM A$(20,20),T$(20),P(10),X,
Y,Q
DEFUSR = 32187
 'Entry to initialise
A1 = 32223
 'EXEC address for sort
A2 = 32006
 'EXEC address for move
A3 = 32119
 'entry to null '
P(3)=0
 'ascending sort
P(4)=1
 'use full string
P(5)=255
 'as sort key
P(2)=0
'don't use indirect sort
X=USR(VARPTR(P(0)))
 'initialise routine '
FOR X=0 TO 20
 'set pointer array
N(X) = X*21
 'for each move
NEXT X
P(7)=VARPTR(T$(0))
 'set move parameters
P(8)=21
  'which don't change
P(9)=VARPTR(N(0))
P(0)=21
 'will always sort 21 elements
P(1)=VARPTR(T$(0))
 'in T$(N)
 'sort all low order elements
FOR     X     =     0     TO     20
P(5)=VARPTR(A$(X,0))
 'set to move
EXEC A2
 'move elements
EXEC A1
 'sort T$(N) array
P(10)=1
 'set for reverse move
EXEC A2
 'move back P(10)=0
 'reset switch
NEXT X
EXEC A3
 'null VARPTR's in T$(N)
```

If you are in fact doing a multi-level sort of the low order elements, using an indirect sort, and changing the order in memory, then the pointer array for the move from the source array to T$(N) in sorted sequence will be different to the pointer array used to move them back again.

The pointer array for the move to T$(N) in sorted sequence will be:

```
FOR X = 0 TO 20
B(X) = A(X)*21
NEXT X
```

The pointer array for the move back after the sort will be:

```
FOR X = 0 TO 20
B(X) = X*21
NEXT X
```

OTHER RESTRICTIONS WHEN MOVING STRING VARPTR's

When ever you use these routines to move string VARPTR's around in memory for sorting etc, you have to ensure that Basic is not forced to do a garbage collect operation in normal string space. If it does then the references to the string data may become unpredictable.

The easiest way to avoid this is to not use any string variable on the left hand side of an equal sign at any time during the sort and related operations.

For this reason, the calling sequences for these routines have been designed specifically to not require any string variable on the left hand side of an equasion.

So play safe, and avoid doing any other operations in the same part of any program which does a sort or move associated with a sort.

The easy way to do this is to use a GOSUB to a routine that does the actual sort, and do nothing else before the RETURN.

THE USE OF VARPTR's

All VARPTR's in CoCo Basic are 5 bytes long, and this feature is used by these routines so that it does not have to distinguish between the VARPTR of a numeric or a string array.

In most cases they can be treated in the same way.

For a string variable the contents of the 5 byte VARPTR are:

* Byte 0 is the length of the string,
* Bytes 1 & 4 are not used,
* Bytes 2 & 3 contain the address of the first byte of the data for the string.

For a numeric variable the contents of the 5 byte VARPTR is the numeric value of the variable itself.

The value is held as a normalised floating point number where byte zero is the exponent (plus 128), and bytes 1 to 4 contain the normalised mantissa.

A common aspect for all VARPTR's is that if byte zero is

zero (ie contains the value zero), then a numeric variable will have the value zero, while a string variable will be a null string.

CONVERTING A FLOATING POINT NUMBER TO AN INTEGER

This utility contains a small routine (28 bytes long) which converts a normalised floating point number to a positive integer value between zero and Hex FFFF.

It does its conversion differently to Basic's INTCNV routine which converts to a 16 bit two's complement integer between -32768 and +32767.

The conversion routine used here assumes a valid floating point value for the conversion, and does not test if it is valid or not. If an invalid value is used, then the result produced will normally be zero.

Hence if your machine 'locks up' when using this routine, check that the parameters in the parameter array are in a correct range. In some circumstances a zero value may overwrite a critical part of basic's communication area.

The high order bit of byte one is used to show the sign of the floating point number. For a positive number, this bit is zero. The conversion routine simply turns this bit on. So a negative value will convert to its absolute value.

Valid values for the exponent of a positive integer between 1 and Hex FFFF are from 129 to 144.

The conversion routine substracts the actual exponent from 144 and uses the result as a counter for the number of 16 bit shifts to the right for the first two bytes of the mantissa, eg if the exponent is 129 (eg for 1) then the bits in the first two bytes of the mantssa are moved 15 places to the right, which produces the correct integer value.

If the actual exponent is 144 then no shift to the right is required, the first 2 bytes of the mantissa is the correct integer value.

If the exponent is invalid, then the routine will perform more than 15 shifts to the right, which will produce an integer value of zero.

To avoid doing 144 shifts to the right for a floating point number which is zero, the routine first tests for a zero exponent, and if it is then the routine returns a zero value directly.

SORTING NUMERIC VARPTR's AS STRINGS

Positive numeric values (including zero) will sort in the correct natural sequence if the 5 byte VARPTR is treated as normal string data and compared byte for byte from byte zero to 4.

However, negative numbers will not. This is because the exponent for -10 is the same as the exponent for +10. The only difference between these two variables as floating point numbers is that the high order bit of byte 1 is zero for +10 and one for -10. This also means that -10 will sort above +9 if compared in this way.

For much the same reason -10 will also sort as being greater than -9 if compared in this way.

The conversion of floating point numbers to a string format for sorting is done in two parts. Before the comparasion is made, the 5 byte VARPTR is moved to working storage and converted to 6 bytes, which are then compared as normal string data.

The extra byte added is a sign byte to ensure that the natural sequence for sign applies. The sign byte is set to 2 for positive numbers, 1 for a zero, and to zero for negative numbers.

For positive numbers, the five byte numeric value is moved directly to working storage.

However for negative numbers, the complement of the five byte value is stored as the string data for comparasions. This ensures that negative numbers will sort in the correct natural sequence.

With these transformations, the normal string data comparasion routines will produce the correct natural sequence for numeric variables in a sort.

The pointers to the string data elements (also held in working storage) are set to point to the working storage area, and if a swap is required, the numeric VARPTR's are swaped in the same way as if they were string VARPTR's.

HINT:

Loading ML programs

The following will load in a machine language program at ANY address! Just set the variable 'x' to the desired loading address.

```
POKE 465,0:EXEC &HA64C:X=1024:A=
INT(X/256):B=X-(A*256):POKE &H1B
7,A:POKE &H1B8,B:EXEC &HA505
```

USING THE ROUTINES

This utility is relocatable and can be loaded into any convenient area of memory for execution. eg for incidental use it can be loaded into protected memory above 32000 and executed from there, or in the graphic screen area.

However for normal use by any particular Basic program, I consider that the most appropriate place to put it is at the end of the Basic program itself.

The main advantage of this approach is that the ML routine then forms part of the Basic program itself and will always load with it, and hence is always available when required. It also does not require any special loading procedures to be used, and no high memory has to be protected from Basic.

There are no real disadvantages associated with using ML routines in this area. The Basic program can still be handled normally for EDIT, RENUM, (C)SAVE etc. The only normal operation which will damage ML routines in this area is to (C)SAVE the program in ASCII format. This effectively removes the ML routine from the program.

Other utilities which will effectively remove the ML routines are those which alter the physical storage of the Basic program in memory. eg that remove blanks & comments etc.

The utility is submitted as a Basic program containing DATA statements which are POKE'd into memory to set up the ML code.

This program is set up to incorporate the ML routine at the end of another Basic program, and then to delete itself from that program.

The line numbers used for the program start from 55000 to

allow it to be merged at the end of some other program. After merging this program, RUN 55000 to set up the ML routines.

If you wish to remove blanks etc from the original program, this can be done after merging these lines, but before the Run command.

Within the Basic program that uses this utility, the starting point for the routine is obtained by:

```
M=PEEK(27)*256+PEEK(28)-583
```

Entry points are then calculated with reference to the value of 'M' eg,

```
* single entry sort at M+219
* single entry move at M+0
* initialise only at M+187
* EXEC sort at M+223
* EXEC move at M+6
* EXEC null VARPTRS at M+119
```

If you want to use the routine at some other fixed area of memory, say 32000, then delete lines 55110-55150 and replace with:

```
CLEAR 500,32000
A=32000
```

... then continue as is.

The ML routines will then be POKE'd into memory from A to A+580.

The small ML routine used to add the ML code to the Basic program (in Line 55110) contains the length of the routine to be added (in Hex) as bytes 9 to 12 in the string A$.

If this value is altered to suit, the same routine can be used to add any ML routine to the end of a Basic program, by following the same general procedure as used here.

Note that the value substracted from the address in Loc 27 & 28 is 3 more than the value used in A$.

WORKING STORAGE

The ML utility requires 46 bytes of memory for working storage, and to hold the parameter values passed to it. The direct page register is used to point to the working storage area.

As set up the routine uses the cassette buffer for working storage. The DP register is loaded with 2 for this purpose. This can be altered if desired by POKE'ing a different value into location M+175 (ie 32175) within the ML code.

For example, if you want to display some action on the screen while the sort or move is going on, you can POKE 4 (or 5) into this location, and this will display the variables on the text screen. However if you do this then the routine will not be able to retain parameters between calls and you may not be able to use the alternative calling sequence. You can if you don't clear the screen or print anything on it that makes it scroll between operations.

Alternatively you may prefer to use the graphic screen area instead. Use 6 for Extended Basic or 14 for disk.

If you use the alternative calling sequence, then a number of parameters which are initialised by the initialise only call, can be altered directly by POKE's to the appropriate memory locations.

If DP is set equal to the start of working storage, ie to the value obtained by PEEK(M+175)*256, then the POKE addresses for the parameters are:

```
DP+0 start position for sort key in string
DP+1 length of sort key in string
DP+2 address of parameter array
DP+4 ascending/descending switch
DP+35 switch to do numeric sort (=0) or string sort (<>0)
```

If changing parameters with pokes, note that it is the switch at DP+35 which actually determines if a numeric sort is done or not.

The move routine does not retain any switch values for its options. These are set directly from the parameter array for each call, and hence cannot be altered by POKE's.

The null varptr routine also picks up its parameters directly from the parameter array when entered, and hence the routine must have already set its pointer to the parameter array before this routine is called ie, the initialise only routine, or a sort or a move must have been executed before executing the null varptr routine.

THE SHELL-METZNER SORT

I don't know the difference between a Shell sort and the variation called Shell-Metzner.

The algorithm used here is the one described in 'Programming Techniques For Level II Basic' by William Barden jr. This is a publication by Radio Shack for the Model 1, and is a good one.

Programming techniques don't change much, and some of the older ones are easier to understand.

The description of the sort presented here is structured around the program logic used to implement it.

The sort itself is based on breaking down the total sorting task into several smaller sorts of sub-groups of elements within the total.

Each smaller sort is performed by a form of bubble sort, where each sub-group of elements to be sorted is selected in such a way that the total number of comparasions (and swaps) between individual elements for the sort, is substantially reduced, compared with a normal bubble sort.

Because of the conceptual similarities between the bubble sort and each individual sort of lists in the Shell-Metzner sort, a brief description of the bubble sort is included, using a similar approach as adopted for the more complex sort.

BUBBLE SORT

Assume we have 16 elements to be sorted into ascending sequence. The bubble sort would loop through the list and compare each element with the one next to it, and swap them over if the first was greater then the second. It then repeats this operation until it does a complete pass of the list with no swaps.

For example, the bubble sort would start with a switch, SW=0.

It compares element 1 with element 2 ...

- if element 2 is less than element 1, it exchanges (or swaps) elements 1 and 2
- as part of the swap it also sets SW=1. It then compares element 2 with element 3
- and does the same comparasion and swaps the elements if necessary.

It then continues with a comparasion (and swap) of elements 3 & 4, 4 & 5, 5 & 6, etc to 15 & 16.

When it has compared all elements, it tests SW. IF SW=0 then it knows that no elements were swaped, and hence the list is now in the desired sequence.

If SW=1 then at least one swap

occurred so it goes back and repeats the operation again.

It keeps doing this until SW is still zero at the end of a pass, when the list is sorted.

In the worst case, the maximum number of swaps that may be required for the sort is 15+14+13 ... +3+2+1 = 120.

SHELL-METZNER SORT.

The Shell-Metzner sort uses a similar logic for sorting each individual list, in that it sets a switch to zero at the start of a pass. It then loops through the list, comparing elements in the list, and swapping them if necessary to get the required sequence. Each time it does a swap it sets the switch non-zero. At the end of the loop it tests the switch, and if it is not zero, it goes back and does the loop again.

The main difference with the Shell-Metzner sort is that it does not compare adjoining elements in the original list.

It sets up a 'gap' between elements in the original list, and does the comparasion between elements separated by that gap.

In this way it avoids the requirement to 'float' an element all the way from the bottom of the list to the top, which happens with the bubble sort. With Shell-Metzner, a low element near the bottom of the list would be moved to near the top of the list as a single swap early in the sort.

The added complexity of the Shell-Metzner sort comes from the logic required to control the gap between the elements within each list to be compared, and the number of lists to be sorted. The logic to test when each list is sorted is the same as for the bubble sort.

NUMBER OF LISTS AND GAP BETWEEN EACH ELEMENT IN THE LIST

While these are separate parameters for the sort, they both have the same value (called M1 in this description).

For the first pass of the sort, M1 (gap and number of lists) is set equal to INT(N/2), where 'N' is the number of elements to be sorted.

For each subsequent pass of the sort, M1 is reduced to M1=INT(M1/2). When M1 equals zero, the sort is finished.

So that with an example of 16 elements in the list:

The first pass has 8 lists to be sorted, with a gap of 8 between each element in the list.

So we do a normal bubble type sort on 8 separate lists, with the following elements in each list:

1 & 9: 2 & 10: 3 & 11: 4 & 12: 5 & 13: 6 & 14: 7 & 15: 8 & 16

In this case the maximum number of swaps that can occur is 8, ie a swap in each list sorted.

The second pass has 4 lists to be sorted, each with a gap of 4 between each element in the list.

So we do another normal bubble type sort on 4 separate lists, with the following elements in each:

1, 5, 9 & 13: 2, 6, 10 & 14:
3, 7, 11 & 15: 4, 8, 12 & 16

Because of the way that these elements were sorted during the first pass, the maximum number of swaps that can occur within each list is 3, giving a maximum of 3*4=12 swaps for the pass.

The third pass has 2 lists to be sorted, each with a gap of 2 between each element in the list.

So we repeat the sort on 2 separate lists, with the following elements in each list.

1, 3, 5, 7, 9, 11, 13 & 15:
2, 4, 6, 8, 10, 12, 14 & 16

Again, because of the way that these elements were sorted in previous passes, the maximum number of swaps that might be required within each list is 7, giving a maximum of 2*7=14 swaps for the pass.

The fourth pass has a single list with a gap of 1, and while the logic for this sort is effectively the same as the logic used for a bubble sort, because of the way that the elements have been sorted by the previous passes, the maximum number of swaps that might now be required is only 16.

So that the worst case, maximum number of swaps required for the sort has been reduced to a total of 50 for the Shell-Metzner, compared with the 120 required for the bubble sort.

For larger lists, the reduction in the maximum number of swaps required becomes even more dramatic.

SORTS WITH AN ODD NUMBER OF ELEMENTS

The logic used to sort each list is effectively:

1. get the first element,
2. add the gap to find the next element
   - if the next element is greater than the number of elements to be sorted, have finished this loop for this list
   - go and test switch for swaps
3. compare the elements and swap if necessary
4. replace the first element with the second element, and repeat the operations from 2.

Hence, if the number of elements to be sorted is 15, then the lists sorted are:

First pass. M1=7, elements sorted are 1, 8 & 15: 2 & 9: 3 & 10: 4 & 11: 5 & 12: 6 & 13: 7 & 14.

Second pass. M1=3, elements sorted are 1, 4, 7, 10 & 13: 2, 5, 8, 11 & 14: 3, 6, 9, 12 & 15.

Third pass is a single list with a gap of 1.

I haven't tried to work through the maximum number of swaps for this one, but it is still substantially less than the number that might be required for a bubble sort.

NUMBER OF COMPARASIONS & NUMBER OF SWAPS

For a string sort in Basic it is normally the number of swaps required that determines how long the sort might take, rather than the number of comparasions.

This is mainly because of the requirement for regular and frequent garbage collections by Basic to provide sufficient free string space for its operations.

For example if you swap A$(I) with A$(J) using normal Basic commands like:

```
T$=A$(I): A$(I)=A$(J): A$(J)=T$
```

... then you use up free string space at a rapid rate, and this forces frequent garbage collection.

For this reason, any large Basic sort on strings can be speeded up significantly by swapping the VARPTR's with PEEK's and POKE's instead, eg to do the swap with code like:

```
IX=VARPTR(A$(I))
JX=VARPTR(A$(J))
FOR X=0 TO 4
T=PEEK(IX+X)
POKE
IX+X,PEEK(JX+X)
POKEJX+X,T
NEXTX
```

While this arithmetic takes longer to do a single swap, the total sort time will be shorter because it avoids the need for garbage collection.

However, with a ML sort, swaping VARPTR's takes very little time at all, with most of the time used is spent finding the string data and doing the comparasions between the string data.

As well as reducing the maximum number of swaps required for a sort, the Shell- Metzner algorithm also provides a substantial reduction in the number of comparasions required for a sort.

For example, with 16 elements to be sorted, the bubble sort required a worst case maximum of 15 comparasions per pass, by 16 passes (240 max). The last pass is required to confirm that the list is sorted.

The first pass of the Shell-Metzner requires a maximum of 2 comparasions per list for 8 lists (ie 16). It also requires the extra pass after a swap to confirm that the list is sorted.

The second pass requires three comparasions per list for a max of 3 loops for 4 lists (ie 36).

The third pass requires 7 comparasions per list for a max of 4(?) loops for 2 lists (ie 56(?)).

I must admit that I am not too clear on what happens after the second pass. I would expect that if you start with a number of elements which is a power of 2, then the maximum number of loops required for any list would be 3, but I can't prove it and my reference book doesn't say.

However, even if the maximum number of loops per list increases by one for each pass, the total number of comparasions required is still substantially less than for a bubble sort.

Listing One

```
1 '**  ASORT   (ML SORT)
       BY GEORGE MCLINTOCK
       SEPT 87
2 GOTO 55000
3 SAVE"54:3":END'7
4 'A GENERAL PURPOSE ML SORT FOR
 BASIC ARRAYS
5 'SET UP TO ADD ML AT END OF AN
OTHER BASIC PROGRAM - TO USE  ME
RGE AT END ANOTHER PROGRAM AND
RUN 55000
6 ' ENTRY POINTS - M=PEEK(27)*25
6+PEEK(28)-583 -SORT M+219    MO
VE M+0  NULL M+119
7 'INITIAISE ONLY M+219  EXEC MO
VE M+6  EXEC SORT M+223
8 'TO PUT IN FIXED AREA OF MEMOR
Y - DELETE LINES 55110-55150  -
REPACE WITH CLEAR 200,32000:A=32
000
9 '
55000 LN=56000:FOR X=0 TO 580 ST
EP 25:IF X<574 THEN N=25 ELSE N=
5
55010 GOSUB 55030:NEXT X
55020 RESTORE:GOTO 55110
55030 PRINT LN;:A=0:FOR Y=0 TO N
-1
55040 READ C$:B=VAL("&H"+C$):A=A
+B
55050 NEXT Y:READ C$:IF A<> VAL(
"&H"+C$) THEN PRINT "ERROR IN LI
NE NO";LN:STOP
55060 LN=LN+10:RETURN
55070 '
55080 FOR Y= 0 TO N-1:READ C$:PO
KE A,VAL("&H"+C$)
55090 A=A+1:NEXT Y:READ C$:RETUR
N
55100 '
55110 M$="9E1B308902446F806F806F
809F1B39":Y=&H01DA
55120 B=0:FOR X=1 TO 30 STEP 2:N
=VAL("&H"+MID$(M$,X,2)):B=B+N:PO
KE Y,N:Y=Y+1:NEXT X
55130 IF B <> &H578 THEN PRINT "
ERROR IN LINE NO 55110":STOP
55140 EXEC &H1DA:CLEAR
55150 A=PEEK(27)*256+PEEK(28)-58
3:LN=56000
55160 FOR X=0 TO 580 STEP 25:IF
X<574 THEN N=25 ELSE N=5
55170 GOSUB 55080:NEXT X
55180 '
55190 PRINT "ASORT ML NOW ADDED
TO END OF    BASIC PROGRAM": PRI
NT "AND EXTRA BASIC CODE DELETED
"
55200 '
56000 DATA 17,0,AB,17,0,BB,17,0,
A5,9E,2,30,88,1E,17,0,81,1F,3,30
,5,8D,7B,1F,2,5DE
56010 DATA 30,5,8D,75,DD,12,30,5
,A6,5,97,8,8D,6B,DD,E,1F,1,DA,E,
26,11,9E,12,C6,83D
56020 DATA 5,A6,C0,A7,A0,5A,26,F
9,30,1F,26,F3,20,77,DC,12,58,49,
58,49,D3,12,D3,E,DD,AFD
56030 DATA 10,DF,1A,8D,42,DD,1C,
58,49,58,49,D3,1C,DE,1A,33,CB,C6
,5,D,8,26,F,A6,C0,973
56040 DATA A7,A0,5A,26,F5,30,5,9
C,10,25,DF,20,46,A6,A0,A7,C0,20,
EF,8D,35,9E,2,30,88,ADD
56050 DATA 23,8D,12,1F,3,30,5,8D
,C,1F,1,6F,C4,33,45,30,1F,26,F8,
20,25,6D,84,27,15,65C
56060 DATA 86,90,A0,84,97,B,EC,1
,8A,80,D,B,27,6,44,56,A,B,26,FA,
39,4F,5F,39,86,88D
56070 DATA 2,1F,8B,C6,28,DD,25,3
9,4F,1F,8B,39,8D,F1,8D,2,20,F6,8
D,CF,DD,2,1F,1,30,9B5
56080 DATA 88,14,A6,1B,97,4,8D,C
2,D7,0,D7,23,30,5,8D,BA,D7,1,39,
8D,D1,8D,E2,8D,CD,BCC
56090 DATA 9E,2,30,5,8D,AB,DD,18
,30,5,8D,A5,DD,6,DA,6,D7,5,DC,6,
93,18,DD,16,9E,A26
56100 DATA 2,8D,95,DD,E,58,49,58
,49,D3,E,D3,18,DD,10,DC,E,44,56,
DD,E,DA,E,27,62,9EA
56110 DATA DC,E,DD,12,58,49,58,4
9,D3,E,DD,C,9E,18,9F,14,9E,14,9F
,1A,F,8,D,23,10,810
56120 DATA 27,0,C6,8D,30,97,9,4F
,D6,0,5A,E3,2,DD,1E,DC,C,30,8B,9
F,1C,9C,10,24,34,90B
56130 DATA 8D,1A,97,A,4F,D6,0,5A
,E3,2,DD,20,D,4,27,4,8D,55,20,2,
8D,30,9E,1C,9F,7FF
56140 DATA 1A,20,C8,A6,84,27,D,9
1,0,25,A,90,0,4C,91,1,25,2,96,1,
39,4F,39,4F,1F,67B
56150 DATA 8B,39,D,8,26,A6,9E,12
,30,1F,9F,12,27,84,9E,14,30,5,20
,96,9E,1E,DE,20,D,764
56160 DATA A,26,7,D,9,27,2,20,31
,39,D,9,27,FB,A,A,A,9,A6,80,A1,C
0,27,E7,25,619
56170 DATA EF,20,1E,9E,1E,DE,20,
D,9,26,6,D,A,27,E1,20,10,D,A,27,
DB,A,A,A,9,5B8
56180 DATA A6,80,A1,C0,27,E8,22,
CF,9E,1A,DE,1C,96,5,97,22,4C,97,
8,5F,A6,85,97,B,A6,B4A
56190 DATA C5,A7,85,96,B,A7,C5,5
C,C1,5,26,EF,D,22,27,AE,F,22,DC,
16,30,8B,33,CB,20,A35
56200 DATA E0,DE,25,DF,1E,8D,19,
DC,C,30,8B,9F,1C,9C,10,10,24,FF,
72,DF,20,8D,9,86,6,A56
56210 DATA 97,9,97,A,16,FF,3D,6D
,84,26,B,CC,1,5,A7,C0,6F,C0,5A,2
6,FB,39,5F,A6,1,9D7
56220 DATA 2D,E,86,2,A7,C0,A6,85
,A7,C0,5C,C1,5,26,F7,39,4F,A7,C0
,A6,85,43,A7,C0,5C,C1B
56230 DATA C1,5,26,F6,39,21B
56240 DEL 55000-56240
```

For assembly listing please turn over....

HINT:

'Replacing' the text screen

This little one replaces the video display area to the address defined by the variable 'RE'. Change 'RE' to the desired address, and make sure 'RE is evenly divisible by 512.

```
RE=3584:B=RE/512:A=1:FORI=65478T
O
65490       STEP2:POKE        I,(B
ANDA)=A),0: A=A*2:NEXT
```

Listing Two

```
                        00100 * CALLED  ASORT  SORTS BASIC ARRAYS
                        00110 * WITH VARIOUS OPTIONS
                        00120 *
7D00                    00130          ORG     32000
          0000          00140 SSK      EQU     0       START POS SORT KEY
          0001          00150 LSK      EQU     1       LEN SORT KEY
          0002          00160 PARAMS   EQU     2       ADDR PARAM ARRAY
          0004          00170 AD       EQU     4       ASC/DESC SW
          0005          00180 ASW      EQU     5       SW FOR INDIRECT SORT
          0006          00190 ARRAY    EQU     6       ADDR POINTER ARRAY
          0008          00200 SW       EQU     8
          0009          00210 IL       EQU     9
          000A          00220 JL       EQU     10
          000B          00230 T1       EQU     11
          000C          00240 M1       EQU     12
          000E          00250 M        EQU     14
          0010          00260 END      EQU     16
          0012          00270 CNT1     EQU     18
          0014          00280 ST       EQU     20
          0016          00290 OFFSET   EQU     22
          0018          00300 SAV      EQU     24
          001A          00310 I        EQU     26
          001C          00320 J        EQU     28
          001E          00330 IA       EQU     30
          0020          00340 JA       EQU     32
          0022          00350 GX1      EQU     34
          0023          00360 NUMSW    EQU     35
          0025          00370 STRNUM   EQU     37      COMPARE NUMBERS
          0028          00380 NOFSET   EQU     40      FROM START
                        00390 *
                        00400 *MOVE ARRAY POINTERS - CALLED BY USR
                        00410 *OR BY EXEC AFTER EITHER ROUTINE INITIALISED
                        00420 *PARAMETERS ARE
                        00430 *P(6) START SOURCE STRING - VARPTR(A$(0))
                        00440 *P(7) START DESTINATION VARPTR(T$(X))
                        00450 *P(8) NUMBER OF VARPTRS TO MOVE - N(X)
                        00460 *P(9) START POSITION IN POINTER ARRAY - VARPTR(A(X))
                        00470 *     IF P(9) = 0 THEN DO A DIRECT MOVE
                        00480 *P(10) SW FOR DIRECTION OF MOVE
                        00490 *
7D00 17   00AB          00500 MOVE     LBSR    SETDP   ENTRY TO
7D03 17   00BB          00510          LBSR    INITAL  INITIALISE
7D06 17   00A5          00520 MEXEC    LBSR    SETDP   EXEC ENTRY
7D09 9E   02            00530          LDX     <PARAMS START POINTER ARRAY
7D0B 30   88 1E         00540          LEAX    30,X    TO 6TH ELEMENT
7D0E 17   0081          00550          LBSR    CONVT
7D11 1F   03            00560          TFR     D,U     SOURCE ARRAY
7D13 30   05            00570          LEAX    5,X
7D15 8D   7B            00580          BSR     CONVT
7D17 1F   02            00590          TFR     D,Y     DESTINATION ARRAY
7D19 30   05            00600          LEAX    5,X
7D1B 8D   75            00610          BSR     CONVT
7D1D DD   12            00620          STD     <CNT1   NUM TO MOVE
7D1F 30   05            00630          LEAX    5,X
```

```
7D21 A6   05      00640          LDA     5,X     DIRECTION
7D23 97   08      00650          STA     <SW     SWITCH
7D25 8D   6B      00660          BSR     CONVT
7D27 DD   0E      00670          STD     <M      POINTER TO NUMERIC
7D29 1F   01      00680          TFR     D,X
7D2B DA   0E      00690          ORB     <M
7D2D 26   11      00700          BNE     INDIR
                  00710 *SIMPLE MOVE OF VARPTRS
7D2F 9E   12      00720          LDX     <CNT1
7D31 C6   05      00730 SM0      LDB     #5
7D33 A6   C0      00740 SM1      LDA     ,U+     MOVE EACH
7D35 A7   A0      00750          STA     ,Y+     VARPTR
7D37 5A           00760          DECB
7D38 26   F9      00770          BNE     SM1
7D3A 30   1F      00780          LEAX    -1,X
7D3C 26   F3      00790          BNE     SM0     DO ALL
7D3E 20   77      00800          BRA     RBASIC
                  00810 *DO INDIRECT MOVE FROM POINTER ARRAY
7D40 DC   12      00820 INDIR    LDD     <CNT1
7D42 58           00830          LSLB            MUL * 5
7D43 49           00840          ROLA
7D44 58           00850          LSLB
7D45 49           00860          ROLA
7D46 D3   12      00870          ADDD    <CNT1
7D48 D3   0E      00880          ADDD    <M      ADDRESS FOR
7D4A DD   10      00890          STD     <END    END MOVE
7D4C DF   1A      00900          STU     <I      SAVE START
                  00910 *
7D4E 8D   42      00920 INDIR0   BSR     CONVT   VAL TO MOVE
7D50 DD   1C      00930          STD     <J      CONVERT TO BYTES
7D52 58           00940          LSLB
7D53 49           00950          ROLA
7D54 58           00960          LSLB
7D55 49           00970          ROLA
7D56 D3   1C      00980          ADDD    <J
7D58 DE   1A      00990          LDU     <I      START SOURCE/DESTINATION
7D5A 33   CB      01000          LEAU    D,U     OFFSET
7D5C C6   05      01010          LDB     #5      MOVE VARPTR
7D5E 0D   08      01020 INDIR1   TST     <SW
7D60 26   0F      01030          BNE     INDIRX
7D62 A6   C0      01040          LDA     ,U+
7D64 A7   A0      01050          STA     ,Y+
7D66 5A           01060 INDIR2   DECB
7D67 26   F5      01070          BNE     INDIR1
7D69 30   05      01080          LEAX    5,X
7D6B 9C   10      01090          CMPX    <END
7D6D 25   DF      01100          BLO     INDIR0
7D6F 20   46      01110          BRA     RBASIC
                  01120 *
                  01130 *DO REVERSE MOVE
                  01140 *
7D71 A6   A0      01150 INDIRX   LDA     ,Y+
7D73 A7   C0      01160          STA     ,U+
7D75 20   EF      01170          BRA     INDIR2
                  01180 *
                  01190 *SET VARPTRS TO NULL AFTER SORT
                  01200 *P(7) = START OF DESTINATION ARRAY
```

```
                    01210 *P(8) = NUM TO NULL
                    01220 *
7D77 8D   35        01230 NULL    BSR     SETDP
7D79 9E   02        01240         LDX     <PARAMS
7D7B 30   88 23     01250         LEAX    35,X    TO 7TH ELEMENT
7D7E 8D   12        01260         BSR     CONVT
7D80 1F   03        01270         TFR     D,U     START ELEMENTS
7D82 30   05        01280         LEAX    5,X
7D84 8D   0C        01290         BSR     CONVT
7D86 1F   01        01300         TFR     D,X     NUMBER
7D88 6F   C4        01310 NULL1   CLR     ,U
7D8A 33   45        01320         LEAU    5,U
7D8C 30   1F        01330         LEAX    -1,X
7D8E 26   F8        01340         BNE     NULL1
7D90 20   25        01350         BRA     RBASIC
                    01360 *
                    01370 *CONVERT NORMALISED FLOATING POINT NUMBER
                    01380 *POINTED TO BY X
                    01390 *INTO 2 BYTE INTEGER IN D
                    01400 *RESULTS VALID ONLY FOR POSITIVE VALUES TO HEX FFFF
                    01410 *INVALID NUMBERS FORCED TO ZERO
                    01420 *
7D92 6D   84        01430 CONVT   TST     ,X
7D94 27   15        01440         BEQ     ZERO
7D96 86   90        01450         LDA     #144    MAX VALID VALUE
7D98 A0   84        01460         SUBA    ,X      EXPONENT
7D9A 97   0B        01470         STA     <T1
7D9C EC   01        01480         LDD     1,X     MANTISSA
7D9E 8A   80        01490         ORA     #$80    ASSUME POSITIVE
7DA0 0D   0B        01500         TST     <T1
7DA2 27   06        01510         BEQ     EXCONV  NO MOVE REQUIRED
7DA4 44             01520 CONVT1  LSRA            MOVE TO INTEGER
7DA5 56             01530         RORB            POSITION IN D
7DA6 0A   0B        01540         DEC     <T1
7DA8 26   FA        01550         BNE     CONVT1
7DAA 39             01560 EXCONV  RTS
7DAB 4F             01570 ZERO    CLRA
7DAC 5F             01580         CLRB
7DAD 39             01590         RTS
                    01600 *
                    01610 *SET DIRECT PAGE
                    01620 *
7DAE 86   02        01630 SETDP   LDA     #2
7DB0 1F   8B        01640         TFR     A,DP
7DB2 C6   28        01650         LDB     #NOFSET AND ADDRESS FOR
7DB4 DD   25        01660         STD     <STRNUM COMPARE NUMBERS
7DB6 39             01670         RTS
                    01680 *
                    01690 *RETURN TO BASIC
                    01700 *
7DB7 4F             01710 RBASIC  CLRA
7DB8 1F   8B        01720         TFR     A,DP
7DBA 39             01730         RTS
                    01740 *
                    01750 *SORT ROUTINE
                    01760 *CALLED BY X=USR(VARPTR(P(0)))
                    01770 *PARAMETERS ARE
```

```
                    01780 *P(0)=NUMBER STRINGS TO SORT
                    01790 *P(1)=FIRST ELEMENT TO SORT
                    01800 *P(2)=MATCHING POINTER OR ZERO
                    01810 *P(3)=SW ASCENDING/DESCENDING SORT
                    01820 *P(4)=START POSITION SORT KEY
                    01830 *P(5)=LENGTH OF SORT KEY
                    01840 *FOR FULL STRING P(4)=1 & P(5)=255
                    01850 *
7DBB 8D   F1        01860 INONLY  BSR     SETDP   TO INITIALISE ONLY
7DBD 8D   02        01870         BSR     INITAL
7DBF 20   F6        01880         BRA     RBASIC
                    01890 *
7DC1 8D   CF        01900 INITAL  BSR     CONVT   GET PARAMETER VALUE
7DC3 DD   02        01910         STD     <PARAMS SAVE IT
7DC5 1F   01        01920         TFR     D,X
7DC7 30   88 14     01930         LEAX    20,X    TO ELEMENT 4
7DCA A6   1B        01940         LDA     -5,X    ASC/DESC SW
7DCC 97   04        01950         STA     <AD
7DCE 8D   C2        01960         BSR     CONVT
7DD0 D7   00        01970         STB     <SSK    START KEY
7DD2 D7   23        01980         STB     <NUMSW  SW NUMERIC OR STRING
7DD4 30   05        01990         LEAX    5,X
7DD6 8D   BA        02000         BSR     CONVT
7DD8 D7   01        02010         STB     <LSK    LENGTH SORT KEY
7DDA 39             02020         RTS
                    02030 *
                    )2040 *ACTUAL ENTRY FOR SORT
                    02050 *
7DDB 8D   D1        02060 START   BSR     SETDP   ENTER HERE FOR INITIALISE
7DDD 8D   E2        02070         BSR     INITAL  AND SORT
7DDF 8D   CD        02080 EXECS   BSR     SETDP   ENTER HERE FOR EXEC
7DE1 9E   02        02090         LDX     <PARAMS
7DE3 30   05        02100         LEAX    5,X     FIRST ELEMENT FOR SORT
7DE5 8D   AB        02110         BSR     CONVT
7DE7 DD   18        02120         STD     <SAV
7DE9 30   05        02130         LEAX    5,X     MATCHING
7DEB 8D   A5        02140         BSR     CONVT   POINTER
7DED DD   06        02150         STD     <ARRAY  ARRAY
7DEF DA   06        02160         ORB     <ARRAY  GET SWITCH
7DF1 D7   05        02170         STB     <ASW    INDIRECT SORT
7DF3 DC   06        02180         LDD     <ARRAY  CALC OFFSET
7DF5 93   18        02190         SUBD    <SAV    TO POINTER
7DF7 DD   16        02200         STD     <OFFSET ARRAY
7DF9 9E   02        02210         LDX     <PARAMS SET BY INIT
7DFB 8D   95        02220         BSR     CONVT
7DFD DD   0E        02230         STD     <M      NUM TO SORT
7DFF 58             02240         LSLB            MUL * 5
7E00 49             02250         ROLA
7E01 58             02260         LSLB
7E02 49             02270         ROLA
7E03 D3   0E        02280         ADDD    <M
7E05 D3   18        02290         ADDD    <SAV    ADDRESS FOR END
7E07 DD   10        02300         STD     <END    OF SORT STRINGS
                    02310 *
                    02320 *
                    02330 * SET FOR OUTTER LOOP IE EACH LIST
                    02340 *
```

```
7E09 DC   0E      02350 RP0    LDD    <M      GAP FOR COMPARE
7E0B 44           02360        LSRA           INT(M/2)
7E0C 56           02370        RORB
7E0D DD   0E      02380        STD    <M
7E0F DA   0E      02390        ORB    <M      TEST ZERO
7E11 27   62      02400        BEQ    RTSB    END OF SORT
                  02410 *
7E13 DC   0E      02420        LDD    <M      SET INITIAL VALUE
7E15 DD   12      02430        STD    <CNT1
7E17 58           02440        LSLB           MUL * 5
7E18 49           02450        ROLA           TO GET
7E19 58           02460        LSLB           GAP IN BYTES
7E1A 49           02470        ROLA
7E1B D3   0E      02480        ADDD   <M
7E1D DD   0C      02490        STD    <M1     BYTES IN GAP
                  02500 *
                  02510 * SET FOR INNER LOOP
                  02520 *
7E1F 9E   18      02530        LDX    <SAV    FIRST GROUP IN LIST
7E21 9F   14      02540 RP1A   STX    <ST
7E23 9E   14      02550 RP1    LDX    <ST     START EACH LIST
7E25 9F   1A      02560        STX    <I
7E27 0F   08      02570        CLR    <SW     SWITCH
                  02580 *
                  02590 * SORT THE LIST
                  02600 *
7E29 0D   23      02610 RP2    TST    <NUMSW
7E2B 1027 00C6    02620        LBEQ   DONUM   DO NUMERIC SORT
7E2F 8D   30      02630        BSR    VALID1  SET LEN TO SEARCH
7E31 97   09      02640        STA    <IL
7E33 4F           02650        CLRA
7E34 D6   00      02660        LDB    <SSK
7E36 5A           02670        DECB           ADJUST FOR ADD
7E37 E3   02      02680        ADDD   2,X     START POS IN STRING
```

```
7E39 DD   1E      02690          STD     <IA     ADDRESS TO START SEARCH
7E3B DC   0C      02700          LDD     <M1     BYTES IN GAP
7E3D 30   8B      02710          LEAX    D,X
7E3F 9F   1C      02720          STX     <J      SECOND STRING TO COMPARE
7E41 9C   10      02730          CMPX    <END    END STRINGS
7E43 24   34      02740          BHS     ENDLP1  AT END THIS PASS
                  02750 *
7E45 8D   1A      02760          BSR     VALID1  LEN UPPER STRING
7E47 97   0A      02770          STA     <JL
7E49 4F           02780          CLRA
7E4A D6   00      02790          LDB     <SSK
7E4C 5A           02800          DECB
7E4D E3   02      02810          ADDD    2,X
7E4F DD   20      02820          STD     <JA     ADDRESS TO START SEARCH
                  02830 *
7E51 0D   04      02840 CONT1    TST     <AD     ASCEND/DESCEND
7E53 27   04      02850          BEQ     AS1
7E55 8D   55      02860          BSR     DES     DESCENDING SORT
7E57 20   02      02870          BRA     NX1
7E59 8D   30      02880 AS1      BSR     ASC     ASCENDING SEARCH
                  02890 *
7E5B 9E   1C      02900 NX1      LDX     <J      INCREMENT POINTERS
7E5D 9F   1A      02910          STX     <I
7E5F 20   C8      02920          BRA     RP2
                  02930 *
                  02940 * SET LENGTH OF STRING TO SEARCH
                  02950 *
7E61 A6   84      02960 VALID1   LDA     ,X
7E63 27   0D      02970          BEQ     RTS     NULL STRING
7E65 91   00      02980          CMPA    <SSK
7E67 25   0A      02990          BLO     NX2     START > STRING
7E69 90   00      03000          SUBA    <SSK    START SEARCH
7E6B 4C           03010          INCA            ADJUST FOR COUNT
7E6C 91   01      03020          CMPA    <LSK    LEN TO SEARCH
7E6E 25   02      03030          BLO     RTS     STRING < SEARCH
7E70 96   01      03040          LDA     <LSK    GET LEN TO SEARCH
7E72 39           03050 RTS      RTS
7E73 4F           03060 NX2      CLRA            MAKE ZERO LEN
7E74 39           03070          RTS
7E75 4F           03080 RTSB     CLRA
7E76 1F   8B      03090          TFR     A,DP
7E78 39           03100          RTS
                  03110 *
                  03120 * END OF LOOP 1
                  03130 *
7E79 0D   08      03140 ENDLP1   TST     <SW
7E7B 26   A6      03150          BNE     RP1
7E7D 9E   12      03160          LDX     <CNT1
7E7F 30   1F      03170          LEAX    -1,X
7E81 9F   12      03180          STX     <CNT1
7E83 27   84      03190          BEQ     RP0     DO NEW LIST
7E85 9E   14      03200          LDX     <ST     DO NEXT SET OF LISTS
7E87 30   05      03210          LEAX    5,X
7E89 20   96      03220          BRA     RP1A
                  03230 *
                  03240 * DO ASCENDING SORT
                  03250 *
7E8B 9E   1E      03260 ASC      LDX     <IA     ADDRESS LOWER
7E8D DE   20      03270          LDU     <JA     ADDR UPPER
7E8F 0D   0A      03280 ASC1     TST     <JL     LEN UPPER
7E91 26   07      03290          BNE     NZ1
7E93 0D   09      03300          TST     <IL     LN LOWER
7E95 27   02      03310          BEQ     NOSWAP  IF BOTH ZERO THEN NO SWAP
7E97 20   31      03320          BRA     SWAP    IF UPPER ZERO & LOWER NOT THEN SWAP
7E99 39           03330 NOSWAP   RTS             RETURN
                  03340 *
7E9A 0D   09      03350 NZ1      TST     <IL
```

```
7E9C 27   FB      03360          BEQ     NOSWAP  IF LOWER IS ZERO & UPPER IS NOT THEN NO SWAP
                  03370 * BOTH NON-ZERO COMPARE NEXT BYTE
7E9E 0A   0A      03380          DEC     <JL     DECREASE COUNTERS FOR
7EA0 0A   09      03390          DEC     <IL     COMPARASION
7EA2 A6   80      03400          LDA     ,X+     LOWER BYTE
7EA4 A1   C0      03410          CMPA    ,U+     UPPER BYTE
7EA6 27   E7      03420          BEQ     ASC1    STILL EQUAL
7EA8 25   EF      03430          BLO     NOSWAP  LOWER IS LOWER THAN UPPER
7EAA 20   1E      03440          BRA     SWAP    IS GREATER THAN SWAP THEM
                  03450 *
                  03460 * DESCENDING SORT
                  03470 *
7EAC 9E   1E      03480 DES      LDX     <IA     LOWER
7EAE DE   20      03490          LDU     <JA     UPPER
7EB0 0D   09      03500 DES1     TST     <IL     LEN LOWER
7EB2 26   06      03510          BNE     NZ2
7EB4 0D   0A      03520          TST     <JL     LEN UPPER
7EB6 27   E1      03530          BEQ     NOSWAP  BOTH ZERO NO SWAP
7EB8 20   10      03540          BRA     SWAP    IF LOWER ZERO & UPPER NOT THEN SWAP
7EBA 0D   0A      03550 NZ2      TST     <JL
7EBC 27   DB      03560          BEQ     NOSWAP  IF UPPER NON ZERO & LOWER ZERO THEN SWAP
7EBE 0A   0A      03570          DEC     <JL
7EC0 0A   09      03580          DEC     <IL
7EC2 A6   80      03590          LDA     ,X+     LOWER BYTE
7EC4 A1   C0      03600          CMPA    ,U+     UPPER BYTE
7EC6 27   E8      03610          BEQ     DES1    STILL EQUAL
7EC8 22   CF      03620          BHI     NOSWAP  LOW BYTE IS > UPPER NO SWAP
                  03630 *IS LESS THAN SWAP THEM - FALL THROUGH
                  03640 *
                  03650 * SWAP VARPTR OF STRINGS
                  03660 *
7ECA 9E   1A      03670 SWAP     LDX     <I
7ECC DE   1C      03680          LDU     <J
7ECE 96   05      03690          LDA     <ASV
7ED0 97   22      03700          STA     <GX1    SWITCH
7ED2 4C           03710          INCA
7ED3 97   08      03720          STA     <SW     SHOW SWAP
7ED5 5F           03730 S0       CLRB
7ED6 A6   85      03740 S1       LDA     B,X
7ED8 97   0B      03750          STA     <T1
7EDA A6   C5      03760          LDA     B,U
7EDC A7   85      03770          STA     B,X
7EDE 96   0B      03780          LDA     <T1
7EE0 A7   C5      03790          STA     B,U
7EE2 5C           03800          INCB
7EE3 C1   05      03810          CMPB    #5
7EE5 26   EF      03820          BNE     S1
7EE7 0D   22      03830          TST     GX1
7EE9 27   AE      03840          BEQ     NOSWAP  RETURN DONT DO ARRAY
7EEB 0F   22      03850          CLR     <GX1    ZERO FOR NEXT TIME
7EED DC   16      03860          LDD     <OFFSET SWAP ARRAY AS WELL
7EEF 30   8B      03870          LEAX    D,X
7EF1 33   CB      03880          LEAU    D,U
7EF3 20   E0      03890          BRA     S0      SWAP AS WELL
                  03900 *
                  03910 *
                  03920 *DO NUMERIC SORT
                  03930 *BY CONVERTING NUM VARPTR TO STRING
                  03940 *
7EF5 DE   25      03950 DONUM    LDU     <STRNUM
7EF7 DF   1E      03960          STU     <IA     ADDR TO START SEARCH
7EF9 8D   19      03970          BSR     SETNUM  SET NUM TO STRING
7EFB DC   0C      03980          LDD     <M1     BYTES IN GAP
7EFD 30   8B      03990          LEAX    D,X     NEXT VARPTR TO COMPARE
7EFF 9F   1C      04000          STX     <J
7F01 9C   10      04010          CMPX    <END
7F03 1024 FF72    04020          LBHS    ENDLP1  END THIS PASS
```

```
7F07 DF   20      04030          STU     <JA     SECOND STRING
7F09 8D   09      04040          BSR     SETNUM  SET IT UP
7F0B 86   06      04050          LDA     #6
7F0D 97   09      04060          STA     <IL     LENGTH TO
7F0F 97   0A      04070          STA     <JL     SEARCH
7F11 16   FF3D    04080          LBRA    CONT1   CONTINUE AS FOR STRING
                  04090 *
                  04100 *SET NUMERIC VARPTR TO SORT AS A STRING
                  04110 *
7F14 6D   84      04120 SETNUM   TST     ,X
7F16 26   0B      04130          BNE     NOTZRO
7F18 CC   0105    04140          LDD     #$0105  SET FOR
7F1B A7   C0      04150          STA     ,U+     ZERO
7F1D 6F   C0      04160 ZR1      CLR     ,U+
7F1F 5A           04170          DECB
7F20 26   FB      04180          BNE     ZR1
7F22 39           04190          RTS
                  04200 *
7F23 5F           04210 NOTZRO   CLRB            FOR COUNTING
7F24 A6   01      04220          LDA     1,X     BYTE WITH SIGN
7F26 2D   0E      04230          BLT     XNEG    IS NEGATIVE
7F28 86   02      04240          LDA     #2      MOVE FOR
7F2A A7   C0      04250          STA     ,U+     POSITIVE NUM
7F2C A6   85      04260 POS1     LDA     B,X
7F2E A7   C0      04270          STA     ,U+
7F30 5C           04280          INCB
7F31 C1   05      04290          CMPB    #5
7F33 26   F7      04300          BNE     POS1
7F35 39           04310          RTS
7F36 4F           04320 XNEG     CLRA            SET FOR
7F37 A7   C0      04330          STA     ,U+     NEGATIVE NUM
7F39 A6   85      04340 XNEG1    LDA     B,X     COMPLEMENT
7F3B 43           04350          COMA            DURING MOVE
7F3C A7   C0      04360          STA     ,U+
7F3E 5C           04370          INCB
7F3F C1   05      04380          CMPB    #5
7F41 26   F6      04390          BNE     XNEG1
7F43 39           04400          RTS
                  04410 *
          7F44    04420 ZZEND    EQU     *
          7DDB    04430          END     START
00000 TOTAL ERRORS
```

HEADINGS

by
Michael Shoobridge

THIS PROGRAM IS rather light-weight, ie it only makes pretty patterns. But the underlying idea is to explore some of the potential uses of STRING$, used for making BORDERS. The following points are brought out:-

1) Define the STRING$ early in the program, then call them up when needed by a simple command, such as:-

PRINT A$

(... B$, C$ or A$+B$+C$+D$).

2) Define some STRING$ a bit later in the program, with a variable included in it, which may be used to vary the appearance of the graphic called by the STRING$ (such as:-

A$=STRING$(N, X+16*Y)

... where N = number of times that the colour block will be repeated; X = the shape of the graphic character; and Y = the colour of the character - where 1 = yellow, 2 = blue, 3 = red, and so on).

X may be modified by defining it as being equal to 127 plus any number from 1 to 16: By this means RND(16) may be used to give any graphic character.

3) Use another number (C5 in the program) to change the background colour by a CLS(c5) instruction. (See line 460).

4) Make a fixed block of, say, 24 characters in length, into which any number of words (not more than 24 characters in total length) may be inserted.

This device (LINES 300-340) enables one to wrap a coloured border around this defined. block, and to change both the border, and the message in the centre, with only one 'instruction' line (any of lines 810-880), and a universal print statement (line 470).

5) A simple device for making follow-on menus. If these are to be fixed menus (not likely to be changed later) many bytes may be saved by omitting the PRINT@ statements, and wrapping them up in one or several long strings, when they will take only one or two lines instead of the 9 or 10 for all the PRINT@ statements. The beauty of this type of menu is that you can come back to the 2nd or the 3rd menu, or return to the 1st, whenever you want to.

This is controlled by MK=1, MK=2, MK=3, MK=7 in lines 180, 360, 500-505, 695, 740, 995, 1030.

I am not claiming any special originality for these ideas, merely that they may be useful (especially for beginners, like myself) to have as a working example to study.

This is one of the reasons why there are two LISTINGS, so that comparisons may be made between a simpler and a more complex version (line 470 for instance).

The Listing:

```
   ****************************
   *                          *
   *         HEADINGS         *
   *            by            *
   *    MICHAEL SHOOBRIDGE    *
   *                          *
   ****************************
1 GOTO5
2 SAVE"337:3":END'10
3 '
4 '
5 ' HEADINGS/BAS  MULTIPLE MENUS
 WITH ONE OR SEVERAL COLOURED SU
RROUNDS  MPK SHOOBRIDGE  14 SEPT
 1987 23 NAUGHTON GROVE BLACKBUR
N 3130 VICTORIA
10 CLEAR 2000
20 ON BRK GOTO 30: 'FOR COCO 3 O
NLY. DELETE FOR COCO 1 AND 2
25 Z1$=STRING$(15,140+48):
   Z2$=STRING$(15,131+16):
   Z3$=STRING$(1,131+48):
   Z4$=STRING$(1,131+16):
   Z5$=STRING$(15,131+48):
   Z6$=STRING$(15,140+16)
30 CLS:PRINT@33,Z1$;Z2$:FOR X=65
TO 353 STEP32:PRINT@X,Z3$;Z4$: N
EXT X
40 PRINT@76,"MENU"
50 PRINT@100,"1) HORIZONTAL"
60 PRINT@132,"2) UPRIGHT LINES"
70 PRINT@164,"3) RED WHITE & BLU
E"
80 PRINT@196,"4) MOSAIC"
90 PRINT@228,"5) BLACK & WHITE"
100 PRINT@260,"6) STREET SCENE"
110 PRINT@292,"7) INDIAN BLANKET
S"
120 PRINT@324,"8) BLACK SURROUND
"
130 PRINT@356,"9) further select
ions"
140 FORX=93TO 381 STEP32:PRINT@X
,Z3$;Z4$: NEXT X
150 PRINT@385,Z5$;Z6$
160 AN$=INKEY$:IF AN$="" THEN160
170 IF AN$<"1" OR AN$>"9" THEN30
180 MK=1: ON VAL(AN$) GOSUB210,2
20,230,240,250,260,270,280,290
190 IF AN$="9"THEN 600
200 GOTO300
210 S1$="RAILWAY":S2$="TRACKS":C
1=3:C2=5:C3=1:C4=7:C5=2:C6=0:E=1
40:F=140:G=140:H=140:CNT=9:RETUR
N
220 S1$=" PICKET":S2$=" FENCE ":
C1=1:C2=7:C3=6:C4=2:C5=3:C6=0:E=
138:F=138:G=138:H=138:CNT=9:RETU
RN
230 S1$="UNION ":S2$="JACK":C1=4
:C2=4:C3=2:C4=7:C5=5:C6=0:E=143:
F=143:G=143:H=143:CNT=8:RETURN
240 S1$="****":S2$="****": C1=6:
C2=4:C3=3:C4=1:C5=7:C6=0:E=136:F
=134:G=130:H=134:CNT=2:RETURN
250 S1$="ADVERTISING": S2$="CIGA
RETTES ":C1=2:C2=5:C3=4:C5=4:C6=
0:  E=128:F=128:G=128:H=138: CNT
=6: RETURN
260 S1$="TOWN": S2$="HOUSES": C1
=7:C2=5:C3=1:C4=1:C5=2:C6=0:E=13
7:F=128:G=139:H=135:CNT=2:RETURN
270 S1$="": S2$="SOMBRE": C1=3:
C2=1: C3=4: C4=6: C5=4: C6=0: E=
130: F=133: G=143: H=128: CNT=3:
 RETURN
280 S1$=CHR$(143+48)+CHR$(143+32
): S2$=STRING$(8,CHR$(143+16)):
C1=7: C2=3: C3=2: C4=2: C5=5: C6
=0: E=128: F=128: G=128: H=128:
CNT=8: RETURN
290 RETURN
300 P1=LEN(S1$): P2=(24-P1)/2: P
3=INT((24-P1)/2): IF P2>P3 THEN
```

Tandy ELECTRONICS

CoCo3 Home Educational Package

SPECIAL CHRISTMAS PACKAGE

Enjoy reliable computer technology from a user-friendly Tandy computer package that is easy on the budget. Be entertained and educated at the same time. With this great computer package you can get down to work or fun from day one. Enjoy super quality computer equipment in your own home. This is an exciting offer that should not be missed.

Package Includes:

Your own computer system for fun, entertainment, professional application and an excellent saving! Make a fantastic investment this Christmas and treat yourself and your family to computing fun with Tandy. This package includes a computer cassette recorder, a set of joysticks, a handy nylon marker pen, 2 education and 2 game programs and the CoCo3 keyboard. This package is too good to miss, so rush into Tandy now before it is too late. 26-1208, 26-3008, 26-9649, 26-9407, 26-9550, 26-7967, 26-7968

35% Off!

Reg 690.69

$449

BUY NOW AND SAVE!

Save $80

Color Computer 3

Reg 449.00 **$369**

128K Extended BASIC Color Computer 3. Be entertained, educated and get professional applications such as graphics, programming, budgets & more. 128K memory expandable to 512K. Connect to your TV for instant use. 26-3334

1000 EX Great Home Education Package

Tandy Puts It All Together For You

$1499 Reg 2198.00

Save $699

BONUS
Software Pack Includes:

- **Typing Tudor III, Improve Speed & Accuracy**
- **Smat Money, to help organize finances**
- **P-16 Falcon—a game**
- **Crosscheck-B/W—a crossword puzzle**

Suggested Retail Value: $300

Tandy 1000 EX. A wonderful computer package to give, great to get! Best of all, buy a PC-Compatible 1000 EX before Christmas and get the software bonus pack to the value of $300.00. Take advantage of Personal Desk-Mate at no extra cost. It's the best computer package put together for Christmas. Buy today and save! 25-1050, 25-1023, 25-9650

Same package as above with the VM-2 Monochrome Monitor. 26-3211
Reg 1948.95..**Save 649.95..$1299**

Trademark OZISOFT
Monitor stand not included

Save $100

Triple-Mode Personal Printer

Reg 599.95 **499.95**

DMP 130. Word-processing, data-processing and dot-addressable graphics modes. 26-1280

WE SERVICE WHAT WE SELL!

Tandy ELECTRONICS

A DIVISION OF TANDY AUSTRALIA LIMITED INC. IN N.S.W.

Nearly 350 Stores Australia-Wide

Available From 350 Stores Australiawide Including Tandy Computer Centres

or Order On VIATEL *642614#

Independent Tandy Dealers may not be participating in this ad or have every item advertised. Prices may also vary at individual Dealer Stores

the GOLDSOFT II WISHBOOK

The Goldsoft Wishbook

The following products are available on order from us.

To order, contact us by phone, Viatel or letter, giving your name, address, phone number and credit card number, as well as the Item # shown beside the product as listed below.

All items include post and packing.

CoCo Hardware

Item #	Description	Price
G 001	The CoCoConnection - Use your CoCo to control models, alarms - anything electrical	$206.00
G 002	Video Amplifier with sound - attach your CoCo 1 or 2 to a Video monitor	$35.00
G 003	The Probe - A temperature sensing unit you plugin to the joy stick port.	$49.95

CoCo Software

Item #	Description	Price
G 1001	Say the Wordz - two Curriculum based speller programs for your Tandy Speech / Sound Pack (32K ECB)	$29.95

The CoCo 3 Tape/Disk

Item #			Price
G 1002		# 1	$16.00
G 1003		# 2	$16.00
G 1004		# 3	$16.00
G 1015	#10	Education	$16.00
G 1016	#11	Education (Disk only)	$16.00

The Best of CoCoOz

Item #	Description		Price
G 1005	# 1	Education	$16.00
G 1006	# 2 Part 1	16K Games	$16.00
G 1007	# 2 Part 2	32K Games	$16.00
G 1008	# 3	Utilities	$16.00
G 1009	# 4	Business	$16.00
G 1010	# 5	Adventure Games	$16.00
G 1011	# 6	Preschool Edn	$16.00
G 1012	# 7	Graphics	$16.00
G 1013	# 8	16K Games	$16.00
G 1014	# 9	32K Games	$16.00

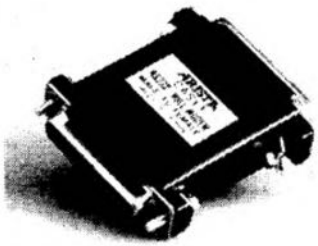
Fun

Item #	Description	Price
Q 1020	Ancient Art of War	$96.00
Q 1021	Print Shop	$119.00
Q 1022	Gato	$68.00
Q 1023	Sargon III	$96.00
Q 1024	Zork I	$79.00
Q 1025	Zork II	$79.00
Q 1026	Zork III	$79.00
Q 1027	Trinity	$79.00
Q 1028	Ballyhoo	$79.00
Q 1029	Hitch Hicker's Guide to the Galaxy	$79.00
Q 1030	Crossword Magic	$68.00
Q 1031	The American Challenge	$68.00
Q 1032	Balance of Power	$89.00
Q 1033	Racter	$79.00
Q 1034	Jet	$114.00
Q 1035	Moonmist	$79.00
Q 1036	Shanghai	$68.00
Q 1037	Championship Golf	$89.00
Q 1038	Borrowed Time	$68.00
Z 2018	The Great International Paper Airplane Construction kit	$49.95
Z 2019	Star Trek	$49.95
Z 2020	Championship Boxing	$69.95
Z 2021	Ultima II	$69.95
Z 2022	Decision in the Desert	$69.95
Z 2023	F-15 Strike Eagle	$69.95
Z 2024	Kings Quest	$69.95
Z 2025	Mean 18	$69.95
Z 2026	Boulderdash	$49.95
Z 2027	Boulderdash II	$49.95
Z 2028	Conflict in Vietnam	$69.95
Z 2029	Dambusters	$69.95
Z 2030	Kings Quest II	$69.95
Z 2031	PSI-5 Trading Company	$69.95
Z 2032	Silent Service	$69.95
Z 2033	Solo Flight	$69.95
Z 2035	Star Fleet	$59.95

Education

Item #	Description	Price
Z 2036	Chem Lab	$69.95
Z 2037	Creature Creator	$59.95
Z 2038	Crypto Cube	$59.95
Z 2039	Decimal Dungeon	$49.95
Z 2040	Donald Duck's Playground	$59.95
Z 2041	European Nations and Locations	$59.95
Z 2042	Fraction Action	$49.95
Z 2043	Math Maze	$59.95
Z 2044	Mickey's Space Adventure	$69.95
Z 2045	Mission Algebra	$59.95
Z 2046	Race Car 'Rithmetic	$49.95
Z 2047	Remember!	$89.95
Z 2048	Ships Ahoy	$59.95
Z 2049	Spellagraph	$59.95
Z 2050	Spellakazam	$59.95
Z 2051	Spellicopter	$59.95
Z 2052	Ten Little Robots	$49.95
Z 2053	Winnie The Pooh in the 100 Acre Wood	$69.95

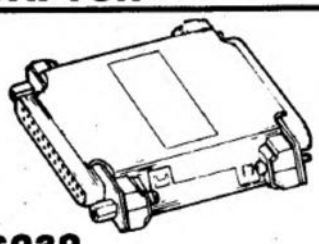
CHRISTMAS GIFTS FOR YOUR COMPUTER



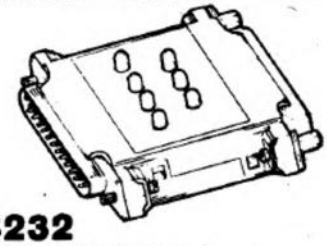
RS232C NULL MODEM ADAPTOR **$13.35**

RS232 DATA SWITCH WITH TESTER **$99.95**

RS232 BREAK OUT BOX **$94.95**



RS232 SURGE PROTECTOR **$28.80**



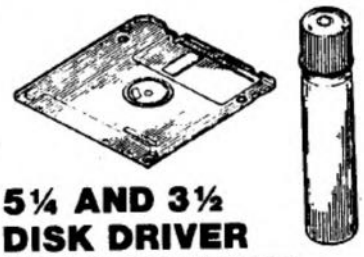
RS232 MINI TESTER **$24.70**

RS323 GENDER CHANGER **$12.45**

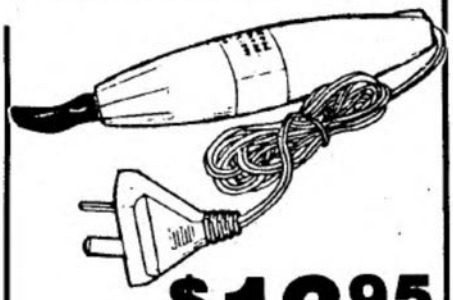
Accessories to keep your system running

MAGNETIC BULK ERASER

$39.95



5¼ AND 3½ DISK DRIVER HEAD CLEANER **$6.55 ea**

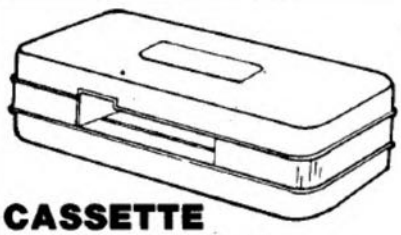


HEAD DEMAGNETISER **$13.95**

GOLDSOFT DISKETTES 5¼ DS DD

BOX OF 10 **$17.99**

TAPE HEAD CARE KIT **$4.35**



CASSETTE TAPE ERASER **$13.15**

GOLDSOFT

Available from:

GOLDSOFT

P.O. BOX 1742, S'PORT, Q, 4215

TEL: (075) 39 6177 or order on VIATEL *64213 #

MasterCard bankcard welcome here VISA

```
P2=P3+1
310 P2$=STRING$(P2,143+16*4): P3
$=STRING$(P3,143+16*4)
320 Q1=LEN(S2$): Q2=(24-Q1)/2: Q
3=INT((24-Q1)/2): IF Q2>Q3 THEN
Q2=Q3+1
330 Q2$=STRING$(Q2,143+16*4): Q3
$=STRING$(Q3,143+16*4)
340 X1$=STRING$(24,143+16*4)
350 X=127
360 IF MK=7 THEN370
365 GOTO430
370 C1=RND(7):     C2=RND(7):
    C3=RND(7):     C4=RND(7):
    C5=RND(8):'         1=YELLOW
 2=BLUE     3=RED        4=BUFF
 5=CYAN     6=MAGENTA  7=ORANGE
 OR SUBSTITUTE FIXED NUMBER (1-7
)    IN C1/C2/C3/C4/C5
380 E=X+RND(16): F=X+RND(16): G=
X+RND(16): H=X+RND(16): '128-143
 -> VERTICAL    = 133/138
    HORIZONTAL = 131/140
    PLAIN       = 128/143
    CHECKER     = 134/137
    L SHAPES= 129/130/132/136
    REVERSE"= 135/139/141
390 CNT=CNT+1: IF CNT>3 THEN E=F
400 IF CNT>5 THEN G=E
410 IF CNT>7 THEN H=E
420 IF CNT>9 THEN CNT=0
430 A$=STRING$(2,E+16*C1):
    B$=STRING$(2,F+16*C2):
    C$=STRING$(2,G+16*C3):
    D$=STRING$(2,H+16*C4)
440 K$=A$+B$:       L$=C$+D$:
    O$=K$+L$
450 Q$=O$+O$:       R$=Q$+Q$:
    U$=R$+R$:       V$=U$+B$+C$+X1
$+D$+K$+C$:         W$=D$+A$+C$+D$
:   Y$=O$+X1$+K$+D$+R$+Q$+O$+K$+
C$
460 CLS(C5)
470 PRINT V$;P2$S1$P3$;W$;Q2$S2$
Q3$;Y$
480 PRINT@320,"colours     C1="C1
" C2="C2" C3="C3" LINE 370   C4=
"C4" C5="C5" C6="C6:PRINT@384,"c
haracters E="E"F="F"G="G"LINE 38
0   H="H;"      count=";CNT
490 AN$=INKEY$:: IF AN$=""THEN490
ELSE IF AN$="P"THEN520
500 IF MK=2 THEN600 ELSE IF MK=3
 THEN900
505 MK=7:S1$="COLOURFUL MARGINS
DEMO":S2$="<P>  TO PRINT OUT DAT
A":  ' MK=7 SENDS PROGRAM TO RND
(7) OR RANDOM COLOURS (LINES 300
-550)
510 GOTO300
520 CLS:INPUT"IS THE PRINTER ONL
INE ?          <ENTER> WHEN READY
";R$
530 PRINT:PRINT"PRINTING THE SET
TINGS REQUIRED":PRINT#-2,"THE BO
RDER from your HEADINGS/BAS prog
ram was made up as follows:-

540 PRINT#-2,"COLOURS:-      C1="
C1"   C2="C2"   C3="C3"   C4="C4
"   C5="C5"   C6="C6
550 PRINT#-2,"CHARACTERS:-  E="E
"  F="F"  G="G"  H="H"  and COUN
```

'...these ideas may be useful, especially to beginners...'

```
T="CNT:PRINT#-2,"COUNT=1-3 (rand
om)  CNT=4-5 (2 Chars same 2 dif
ferent)  CNT=6-7 (3 chars same)
    CNT=8-9 (all chars same).":P
RINT#-2:PRINT#-2:CLS:GOTO500
600 CLS:PRINT@33,Z1$;Z2$:IF MK=2
 THEN605
603 FOR X=65TO 353 STEP32:PRINT@
X,Z3$;Z4$:NEXT X:GOTO610
605 FOR X=65TO 353 STEP32:PRINT@
X,B$: NEXT X
610 PRINT@100," #1     VARIOUS"
620 PRINT@132," #2     PATTERNS"
630 PRINT@164," #3     FOR YOU"
640 PRINT@196," #4     TO PLAY"
650 PRINT@228," #5     WITH, OR"
660 PRINT@260," #6     TO WHICH"
670 PRINT@292," #7     YOU MAY AD
D"
680 PRINT@324," #8     YOUR OWN."
690 PRINT@356," 9) further selec
tions"
695 IF MK=2 THEN705
700 FOR X=93TO 381 STEP32:PRINT@
X,Z3$;Z4$: NEXT X:GOTO710
705 FOR X=93 TO381 STEP32:PRINT@
X,D$:NEXT X
710 PRINT@385,Z5$;Z6$
720 AN$=INKEY$: IF AN$="" THEN72
0
730 IF AN$<"1" OR AN$>"9" THEN60
0
740 MK=2:ON VAL(AN$) GOSUB810,82
0,830,840,850,860,870,880,890
750 IF AN$="9"THEN900
760 GOTO300
795 '-------------------------
-------------------------------
---------------
798 '
799 '
800 ' ONLY TYPE 800 ON IF YOU WA
NT MORE EXAMPLES, OR TO SEE HOW
MULTIPLE MENUS WORK
805 ' -------------------------
-------------------------------
---------------
810 S1$="#1":S2$="":C1=2:C2=7:C3
=1:C4=4:C5=4:C6=0:  E=143:F=143:G
=143:H=134:CNT=7:RETURN
820 S1$="#2":S2$="":C1=7:C2=1:C3
=5:C4=4:C5=6:C6=0:  E=138:F=134:G
=143:H=131:CNT=1:RETURN
830 S1$="#3":S2$="":C1=1:C2=6:C3
=1:C4=2:C5=2:C6=0:  E=133:F=134:G
=143:H=131:CNT=1:RETURN
840 S1$="#4":S2$="":C1=5:C2=5:C3
=7:C4=2:C5=5:C6=0:  E=140:F=140:G
=140:H=140:CNT=8:RETURN
850 S1$="#5":S2$="":C1=5:C2=5:C3
=1:C4=7:C5=2:C6=0:  E=140:F=143:G
=130:H=137:CNT=3:RETURN
860 S1$="#6":S2$="":C1=7:C2=6:C3
=6:C4=2:C5=8:C6=0:  E=143:F=143:G
=129:H=143:CNT=4:RETURN
870 S1$="#7":S2$="":C1=7:C2=7:C3
=7:C4=3:C5=7:C6=0:  E=143:F=143:G
=137:H=143:CNT=4:RETURN
880 S1$="JUST IN CASE YOU THOUGH
T":S2$="NOTHING WAS GOING HERE":
C1=4:C2=3:C3=4:C4=7:C5=5:C6=0:  E
=143:F=143:G=143:H=140:CNT=7:RET
URN
890 RETURN
900 CLS:IF MK=3 THEN904
902 PRINT@33,Z1$;Z2$:GOTO906
904 PRINT@33,Q$+O$+K$+C$
906 FOR X=65 TO353 STEP32: PRINT
@X,Z3$;Z4$: NEXT X
910 PRINT@68," #1     FURTHER ONE
S"
920 PRINT@100," #2     FOR YOUR"
930 PRINT@132," #3     AMUSEMENT.
"
940 PRINT@164," #4     TRY SOME"
950 PRINT@196," #5     OF YOUR OW
N"
960 PRINT@228," #6     BY SUBSTIT
UTION"
970 PRINT@260," #7     IN LINES"
980 PRINT@292," #8     1110 TO 11
90"
985 PRINT@357,"(9 =  MAIN MENU)"
990 FORX=93TO 381 STEP32:PRINT@X
,Z3$;Z4$: NEXT X
995 IF MK=3 THEN1005
996 PRINT@385,Q$+O$+K$+C$
1000 PRINT@385,Z5$;Z6$:GOTO1010
1005 PRINT@385,Q$+O$+K$+C$
1010 AN$=INKEY$:IF AN$="" THEN10
10
1020 IF AN$<"1" OR AN$>"9"THEN30
1030 MK=3: ON VAL(AN$) GOSUB 111
0,1120,1130,1140,1150,1160,1170,
1180,1190
1040 IF AN$="9" THEN30
1050 GOTO300
1110 S1$="#1":S2$="":C1=1:C2=3:C
3=5:C4=2:C5=4:C6=0:  E=143:F=143:
G=143:H=143:CNT=0:RETURN
1120 S1$="#2":S2$="":C1=4:C2=1:C
3=4:C4=4:C5=6:C6=0:  E=143:F=143:
G=135:H=142:CNT=5:RETURN
1130 S1$="#3":S2$="":C1=2:C2=4:C
3=6:C4=3:C5=3:C6=0:  E=133:F=131:
G=143:H=143:CNT=3:RETURN
1140 S1$="#4":S2$="":C1=6:C2=5:C
3=5:C4=3:C5=4:C6=0:  E=138:F=143:
G=142:H=140:CNT=1:RETURN
1150 S1$="#5":S2$="":C1=6:C2=6:C
3=6:C4=1:C5=2:C6=0:  E=141:F=141:
G=141:H=134:CNT=6:RETURN
1160 S1$="#6":S2$="":C1=3:C2=6:C
3=1:C4=1:C5=2:C6=0:  E=131:F=138:
G=131:H=134:CNT=2:RETURN
1170 S1$="#7":S2$="":C1=5:C2=5:C
3=5:C4=5:C5=7:C6=0:  E=136:F=139:
G=140:H=133:CNT=2:RETURN
1180 S1$="#8":S2$="":C1=4:C2=4:C
3=4:C4=3:C5=5:C6=0:  E=140:F=140:
G=140:H=134:CNT=7:RETURN
1190 RETURN
```

CLOCK CHIPS 'n THINGS

By Ken Wagnitz
OS9

I FINALLY HAVE the clock chip which I have been mumbling about, connected to my CoCo. Admittedly at the time of writing, it is still on a prototyping board. But it works fine. I can boot OS-9 and get the right time onto it, without entering the time from the keyboard.

The RTC (Real Time Clock) chip and an 8 bit I/O (Input/Output) chip, are connected via the CoCo cassette socket, and powered from two 'AA' batteries. They should last for over a year.

Additional I/O chips, or battery-powered memory chips can be added without further hardware. The cassette connections are used as a serial input and output ports.

The relay is unused, and could still be operated independantly. I chose the cassette port because it is not normally used under RS-DOS and never used under OS-9.

The big delay has been software! My goal was to produce a cheap circuit which could be added externally to the CoCo, with software anyone could use to control it.

The initial stimulus was the pain of entering the date and time to OS-9, so naturally my first driving software was for that DOS.

However the hardware will work under RS-DOS as well, and would be ideal for those people using B-DOS who enter the time and date, or for anyone who wants to control things for their CoCo with digital input and output.

This is a simple add-on which is easy to use.

The Circuit

This approach to the addtion of a real time clock was a result of stumbling across the family of chips designed to hang off the Phillips 'I2C' bus (that is 'I' squared 'C'). They only require a ground, a bi-directional data lead (SDA) and an output-only (from the computer) clock lead (SCL).

Lots of chips can be connected to those three wires, each having its own address.

The only problem with the cassette port is that it has only one input and one output. I tried using the relay as a clock output, but it nearly chattered to death! So the other two connections together make a bi-directional (each able to send data) data lead.

The clock is synthesized by the circuitry you see in the circuit diagram below.

How It Works

The battery provides a 3V rail to the chips. If the I/O chip is not loaded, it draws virtually nil current. The clock chip will draw about 15 microamps with the computer off or on. Current is only drawn in the milliamp range when the clock chip is being accessed.

I only do that when booting OS9. The I/O chip only draw current if any of its ports which are programmed high, are supplying current to a load.

It resets to all ports high, and should be use such that it switches on any external loads by its ports going low. The same goes for the output leads on the RTC. That way, no extra current will be drawn from the battery.

The external circuit should be powered from the mains or elsewhere, as shown.

The cassette output is normally low, hence the DOut lead is heigh. Software takes the cassette out high, driving DOut low and triggering the monostable, whose output goes high.

At the end of the pulse it triggers the second mono, which provides a clock pulse of about 50 microseconds. Before the clock pulse starts, software has driven the DOut line high or low, as required.

The receiving chips read the data line only when the clock is high.

The first byte sent is the address of the chip which is being written to or read from. The least significant bit is a '0' if the computer is sending to the chip, or a '1' if it wants the chip to send. At the end of each by transmitted, the receiving device (computer or chip) sends an acknowledge bit, a '0'.

Notice that the data line is pulled up with resistors only, not driven high. When a chip has to send data, the cassette port drives the data line low briefly, to start a clock pulse sequence, then returns it high, so the chip can leave the lin high or drive it low.

Thus the line is bi-directional. It is read only by the cassette input. The capacitor and diodes on that line to the computer are to provide a +ve and -ve swinging signals, which the cassette input needs.

The data line to the chip is decoupled from it by a diode, so that when the chips drive the line low, they don't drive out DOut low and start a clock pulse.

The I/O chip is novel in that each lead is simultaneously an output and an input. Each port is normally high after a reset (zero supply volts). It can be programmed to output a low or high, it can be treated as an input and driven low by an external signal.

Its internal pullup resistor is large, so it can't source much current (about 100 microamps), unless it has an external pullup provided. When the chip is read, the state of its leads is read, then the state depending on the voltage appearing on them.

The INT output from the chip

will go low if any of its inputs change. This would be used in other applications to send an interrupt to the MPU, which would then poll the chips to see which lead changed.

The RTC chip has an alarm in it, which if programmed as active, will drive the COMP output low when the time matches the alarm time. It is reset by software.

Also the power-fail-input sets a status flag which can be read. If the supply volts to the clock drops below 1.2 volts, the computer can tell if the battery has failed, making the time reading incorrect.

Now look at the pulse timing diagrams. The pulse trains begin with a 'start' sequence, and end with a 'stop' sequence, as shown.

After the start sequence, the first byte sent is the chip's address. Each different type of chip in the family has a different high nibble address, and further address bits are strappable to be able to attach a few devices of the same type, eg up to eight I/O chips.

The least significant bit of the address byte determines whether following bytes wil be sent to the chip, or by it.

The I/O chip is easy. Its address byte is $40 to send to it, with its address leads strapped low. The next byte sets its outputs (set what are inputs as high). To read from it, send a $41 address byte, and receive the next byte from it. Note that after all reads from chips, the computer acknoledges each data byte received (a low bit at the end of the received byte) except the last one. This tells the chip not to send any more data on further clocks.

The clock chip (RTC) is more complex. It has 9 read/write locations for control, status, months, days, hours, minutes, and alarm months, days, hours and minutes. It does not have read or write capacity for years or seconds except that the seconds count can be zeroed.

So on the January 1 of each year, one has to change the year in the driving software! That would be a great time to change the batteries as well.

A fix on Feb 29 would be needed also. I can live with these great burdens. They beat the hell out of entering the time/date on each boot.

The general working of the bus used, and the clock chip is too complex to present here. The data on this family of ICs is in the Phillips Telecommunications Data book. You may be able to get data from the suppliers. In SA, you can buy the chips from Burtons at Unley, or Soanar at Forestville. They cost around $10 each.

The Software

I wanted to, as a first go, write driving software for these chips which could be easily used by beginners. This meant a Basic09 program (for the OS-9'ers) since a few people have 'C' and fewer use assembly language.

Although my first choice and need was for an OS9 program, these same principles and code could also be used from RS-DOS.

The monostable timings could be increased to allow operation from (slow) Basic or Basic09, in fact my early tests were in Extended Colour Basic. The only penalty is speed. It takes the order of a second to send or receive data under Basic. Basic09 is a bit faster, but the problem is that the task scheduler in OS9 is continually interrupting any running programs, in case any other tasks are running.

This upsets the timing needed by the monostables. Turining off interrupts for such a long time to get around this would be bad practice. So the driver had to be in machine code. That would then be hard to use in a general way. So what I have done (as an interim measure) is write a machine code subroutine which is called from Basic09.

To find out how to do that, I studied the supplied example of 'inkey'. It is listed in Appendix B of the level 2 Basic09 handbook. These subroutines should apply equally well to all versions of OS9.

The Basic09 Parts

The machine code subroutine is called with 'run i2c (string), (number)'.

The 'string' has as its first character, the address byte. If the least significant bit (lsb) of that byte is zero then the following characters in the string are those to be written to the chip. If the lsb is a one, then the following locations in the string are a

continued next page

3 BUFF

By Colin North
UTILITY
CoCo3

WHEN CONVERTING YOUR CoCo 2 graphics programs to run on your CoCo 3, don't you find those DIM statements a little hard to convert?

I mean, if you have something that requires a GET statement on the CoCo 2, you need a DIM(x,y), right?

Well, on the CoCo 3 you need a HBUFF<screen number>,<size>. But just how do you convert the DIM values to a HBUFF value?

Simple - use this program!

The Listing:

```
0 GOTO10
1 '***** "3HBUFF"
2 '***** COLIN NORTH
3 SAVE"20C:3":END'7
10 POKE&HFFBD,63
20 CLS5
30 GOTO300
40 PRINT:PRINT
50 INPUT"  HSCREEN NUMBER";H
60 IF H=1 THEN H=4 ELSE IF H=2 T
HEN H=2 ELSE IF H=3 THEN H=8 ELS
E IF H=4 THEN H=4
70 RETURN
80 PRINT:INPUT" ENTER FIRST X CO
ORDINATE";X
90 INPUT" ENTER FIRST Y COORDINA
TE";Y
100 INPUT"ENTER SECOND X COORDIN
ATE";X1
110 INPUT"ENTER SECOND Y COORDIN
ATE";Y1
120 P=X:P1=X1
130 IF P>P1 THEN X1=P:X=P1
140 P=Y:P1=Y1
150 IF P>P1 THEN Y1=P:Y=P1
160 A=INT(X/H):A=A*H
170 A1=INT(X1/H):A1=A1*H
180 XCOORD=(A1-A+1)/H
190 XD=INT(XC)
200 IF XC>XD THEN XD=XD+1
210 Y1=Y1-Y+1
220 PRINT:PRINT:PRINT
230 BUFFER=XD*Y1-1
240 PRINT"         HBUFF 1,"BUFFE
R
250 PRINT:PRINT:PRINT"         AN
OTHER < Y/N >
260 A$=INKEY$:IFA$=""THEN260
270 IFA$="Y" THEN RUN20
280 IF A$="N" THEN CLS:END
290 GOTO260
```

continued on p 47

continued from previous page

buffer in which is returned the bytes read from the chip.

The 'number' is an integer which is the number of bytes after the address byte, to be written to or read from, the chip.

Notice that I read or fill the string by finding its address, then peeking or poking into it. I couldn't find any easier way to read the bytes of the string, though you could assign it a value with:

```
'string=chr$(xx)+chr$(yy) ...'.
```

"SETCLOCK" simply sets the time/date registers of the RTC. It doesn't check for a power failure or set the alarm registers.

The registers are loaded with 2 BCD nibbles, hence the a*16+b construction.

Listing One:

```
PROCEDURE setclock
 0000       REM setclock -sets real time clock
 0021       DIM instr:STRING[10]
 002D       DIM outstr:STRING[6]
 0039       DIM ad:INTEGER
 0040       ad=ADDR(outstr)
 004A       INPUT " Input date/time as mmddhhmm (no years or secs):  ",
             instr
 0084       POKE ad,$D0 \REM address of clock -write mode
 00AC       POKE ad+1,0 \REM mode byte- execute address
 00D4       POKE ad+2,VAL(MID$(instr,5,1))*16+VAL(MID$(instr,6,1))
 00F5       POKE ad+3,VAL(MID$(instr,7,1))*16+VAL(MID$(instr,8,1))
 0116       POKE ad+4,VAL(MID$(instr,3,1))*16+VAL(MID$(instr,4,1))
 0137       POKE ad+5,VAL(MID$(instr,1,1))*16+VAL(MID$(instr,2,1))
 0158       RUN i2c(outstr,5)
 0165       POKE ad+1,$20
 0171       RUN i2c(outstr,1) \REM reset the seconds count
 0198       END
```

WRCLOCK reads the date/time from the clock chip and writes it in the form yymmddhh as you would type it in when running SETIME. So the following command line will set the time.

```
wrclock ! setime
```

What it does is pipe the output from WRCLOCK to SETIME, which treats it as if it came from the keyboard. You have to have PIPE, PIPER and PIPEMAN in memory for it to work.

Listing Two

```
PROCEDURE wrclock
 0000       REM wrclock -write date/time from clock to stdout
 0030       DIM st:STRING[5]
 003C       DIM ad:INTEGER
 0043       DIM n:BYTE
 004A       ad=ADDR(st)
 0054       POKE ad,$D0 \ POKE ad+1,0
 0068       RUN i2c(st,1) \REM set read address to 1st locn (hours)
 009C       POKE ad,$D1
 00A5       RUN i2c(st,4) \REM read date/time from clock chip
 00D3       n=PEEK(ad+4) \REM months
 00E8       PRINT "87"; STR$(n/16); STR$(LAND(n,$0F));
 0100       n=PEEK(ad+3) \REM days
 0113       PRINT STR$(n/16); STR$(LAND(n,$0F)); " ";
 012A       n=PEEK(ad+1) \REM hours
 013E       PRINT STR$(n/16); STR$(LAND(n,$0F));
 0151       n=PEEK(ad+2) \REM minutes
 0167       PRINT STR$(n/16); STR$(LAND(n,$0F))
 0179       END
```

The Machine Code

'I2C' is a self-contained subroutine. It can be called from any other proram, but I wrote in the form required by Basic09. Its useage is as in the pograms above. The level 1 asembler works fine under level 2 OS-9. And the same subroutine would work under RS-DOS, except that the timings are set for a 1.8Mhz clock. If anyone is interested in the code, I can supply them with a copy, or put it on the AMUS Bulletin Board for downloading. I won't waste space here by explaining here how it works, since the majority of readers aren't assembly language programmers. The listing is that produced by the assembler.

Listing Three

```
00001 *** SUBROUTINE I2C for Basic09 by Ken Wagnitz.
00002
00003 0001             Vers    equ     1
00004 0000 B7CD00FA             mod     ModEnd,ModNam,$21,
$82, Entry,0
00005 000D 6932E3       ModNam  fcs     "i2c"
00006
00007 * Entry: stack offsets as follows:
00008 * (after S is increased by 4 for storage)
00009 0004              Return  equ     4       return
address of caller
00010 0006              PCount  equ     6       # of
parameters following
00011 0008              String  equ     8       addr of
str. 1st byte = addr
00012 000A              StrLen  equ     10      length of
string
00013 000C              BCount  equ     12      # of bytes
to send/receive
00014
00015 0000              hi      set     $00     hi value
for PIAD
00016 00FC              lo      set     $fc     lo value
for PIAD
00017 FC00              lohi    set $fc00
00018 FF20              PIAD    equ $ff20
00019 *-----------------------------------
00020 * Storage offsets
00021 0001              bits    equ     1       bit counter
00022 0002              word    equ     2       space to
assemble tx/rx byte
00023 0003              bytes   equ     3       # of bytes
to tx/rx
00024 *-----------------------------------
00025
00026 0010 327C         Entry   leas    -4,s    create
storage for variables
00027 0012 1F43         tfr     s,u     point to that
storage
00028 0014 3401         pshs    cc
00029 0016 1A50         orcc    #$50    mask interrupts
00030 0018 EC46         ldd     PCount,u get the # of
parameters
00031 001A 10830002     cmpd    #2      right number?
00032 001E 10260081     lbne    ParamErr abort if no
00033 0022 ECD80C       ldd     [BCount,u] get # bytes to
tx/rx
00034 0025 E743         stb     bytes,u save it
00035 0027 CCFC00       ldd     #lohi
00036 W 002A F7FF20     stb     PIAD    make sure SDA is hi
00037
00038 *** Write a Start sequence ***
00039 W 002D B7FF20     sta     PIAD    trigger & hi
00040 W 0030 F7FF20     stb     PIAD
00041 0033 1700B8       lbsr    Delay
00042 W 0036 B7FF20     sta     PIAD    go lo
00043 0039 1700B2       lbsr    Delay
00044 W 003C F7FF20     stb     IAD     restore hi
00045
00046 *** Send the device address ***
00047 003F AE48         ldx     String,u point to address
byte
00048 0041 A684         lda     ,x      get the addr byte
00049 W 0043 17006C     lbsr    WByte   send it
00050 W 0046 1025005D   lbcs    NoAck   exit to Ack error
00051 004A A680         lda     ,x+     get address again
00052 004C 46           rora            put r/w bit into
carry
00053 004D 2435         bcc     Wrt
00054
00055 *** Read from device into buffer ***
00056 004F C608         Rd      ldb     #8 bit count
00057 0051 E741         stb     bits,u
00058
00059 0053 CCFC00       bitlp   ldd     #lohi
00060 W 0056 F7FF20     stb     PIAD    make sure SDA hi
00061 W 0059 B7FF20     sta     PIAD    trigger & hi
00062 W 005C F7FF20     stb     PIAD
00063 005F 17008C       lbsr    Delay
00064 W 0062 B6FF20     lda     PIAD    get i/p into
lsbit
00065 0065 44           lsra            put bit into
carry
00066 0066 6942         rol     word,u  put it in byte
receiver
00067 0068 170083       lbsr    Delay
```

```
00068 006B 6A41          dec     bits,u    dec bit count
00069 006D 26E4          bne     bitlp
00070
00071 006FA642           lda     word,u    Get the
assembled byte
00072 0071 A780          sta     ,x+       put it in buffer
00073 0073 6A43          dec     bytes,u   dec byte count to
read
00074 0075 2602          bne     Ack
00075 0077 2015          bra     Stop n    no ack for last
byte
00076
00077 *** Send an Acknowledge ***
00078 0079 CCFC00        Ack     ldd       #lohi
00079 W 007C B7FF20      sta     PIAD      trigger & lo
00080 W 007F 170066      lbsr    Delay2
00081 0082 20CB          bra     Rd
00082
00083 *** Write buffer to device ***
00084 0084 A680          Wrt     lda ,x+   get byte from
buffer
00085 0086 8D2A          bsr     WByte     send it
00086 0088 251D          bcs     NoAck
00087 008A 6A43          dec     bytes,u   dec byte count
00088 008C 26F6          bne     Wrt
00089
00090 *** Send a stop sequence ***
00091 008E CCFC00        Stop    ldd #     lohi
00092 W 0091 B7FF20      sta     PIAD      trigger & lo
00093 W 0094 170057      lbsr    Delay
00094 W 0097 F7FF20      stb     PIAD      take SDA hi
00095 W 009A 17004B      lbsr    Delay2
00096
00097 *** Done and OK -Exit ***
00098 009D 3501          Exit    puls      cc
00099 009F 5F            clrb               no errors
00100 00A0 3264          leas    4,s       restore stack
00101 00A2 39            Dummy   rts       exit subroutine
00102 *----------------------------
00103 * Wrong number of parameters
00104 00A3 C638 ParamErr         ldb       #$38 parameter erro
00105 00A5 2004          bra     EExit
00106 *----------------------------
00107 * No Acknowledge from device
00108 00A7 C6F6 NoAck    ldb     #$f6      device_not_ready
error
00109 00A9 2000          bra     EExit
00110 *----------------------------
00111 00AB 3501 EExit    puls    cc
00112 00AD 1A01          orcc    #1        set the carry
00113 00AF 3264          leas    4,s       restore stack
00114 00B1 39            rts
00115
00116 ************************************************
00117 * Subroutine WByte. Write the A reg to the port.
00118 00B2 A742 WByte    sta w   ord,u     save send byte
00119 00B4 C608          ldb     #8        bit count
00120 00B6 E741          stb     bits,u
00121 00B8 CCFC00        ldd     #lohi
00122 00BB 6942          wblp    ol        word,u put bit to
write into carry
00123 W 00BD F7FF20      stb     PIAD      make sure SDA is hi
00124 00C0 2408          bcc     wr0
00125 W 00C2 B7FF20      wr1     sta       PIAD trigger & hi
00126 W 00C5 F7FF20              stb       PIAD
00127 00CB 2003                  bra       wrnxt
00128 W 00CA B7FF20              wr0       sta PIAD trigger &
leave lo
00129 W 00CD 17001B      wrnxt   lbsr      Delay2
00130 00D0 6A41          dec     bits,u    dec bit count
00131 00D2 26E7          bne     wblp
00132
00133 *** read the acknowledge ***
00134 W 00D4 F7FF20      stb     PIAD      make sure SDA is hi
00135 W 00D7 B7FF20      sta     PIAD      trigger & hi
00136 W 00DA F7FF20      stb     PIAD
00137 W 00DD 17000E      lbsr    Delay
00138 W 00E0 B6FF20      lda     PIAD      get i/p
00139 W 00E3 170008      lbsr    Delay
00140 00E6 46            rora               put bit into carry
-expect 0.
00141 00E7 39            rts
00142 *----------------------------
00143 00E8 108E0019      Delay2  ldy       #25
00144 00EC 2004                  bra       Dloop
00145
00146 00EE 108E0005      Delay   ldy       #5
00147 00F2 313F          Dloop   leay      -1,y
00148 00F4 26FC                  bne       Dloop
00149 00F6 39                    rts
00150
00151 00F7 7991F7        eeod
00152 00FA               ModEnd  equ       *
00153 end

00000 error(s)
00028 warning(s)
$00FA 00250 progran bytes generated
$0000 00000 data bytes allocated
$0232 00562 bytes used for symbols
```

⊕

Notes

The circuit of the cassette interface and the code listed here are all my own work.

I have no objection to their use by private or commercial interests, as long as the source of the material is quoted. The circuitry shown would work on the cassette port of virtually any home computer, and I welcome someone producing a PCB for the project.

The software timing was tuned using my prototype circuit, and depends on the monostable time constants. Do NOT use an inferior monostable chip.

OS-9ers are welcome to contact me about the circuit or software on (08) 277-1404.

Ed's note: programs "clockset" and "wrclock" will appear on the other side of this months CoCoOz, in 40 track single sided format.

⊕

35mm Animation

By Colin Gawn
UTILITY
32K DECB

35MM IS A SMALL program that simulates a 35mm machine, ie animation. The idea is that you draw a little man (or whatever takes your fancy) using the arrow keys. Press the spacebar when you've completed that one picture.

You can then see the animation on one part of the screen, or in "walking motion", or moving across the screen.

After that, you can save it, and re-load it later.

The Listing:

```
1 '***   35MM    ***
2 '  **BY C GAWN**
3 '*10 SIMOUNDS ST ALICE SPRINGS
5 'USE ARROW KEYS TO MOVE AND
   PRESS "0" TO ERASE AND USE
   "5" TO DRAW LINE.
6 PCLEAR8
15 DIM MA(30,30)
25 X=100:Y=50
35 H=115:V=65:ST=1:E=10:F=90:E1=
E+30:F1=F+30
45 PMODE4,5:PCLS
55 PMODE4,1:PCLS:SCREEN1,1
65 'put graphix on pages 5 to 8
75 FOR W=0 TO 210 STEP 30
85 FOR W1=0 TO 150 STEP 30
95 GOSUB 175
105 DF=DF+1
115 GET(100,50)-(130,80),MA,G
125 PMODE4,5
135 PUT(W,W1)-(W+30,W1+30),MA,PS
ET
145 NEXT W1,W
155 GOTO 315
165 'go draw the little man
175 PMODE 4,1:PCLS:SCREEN1,1
185 LINE(90,40)-(140,90),PSET,B
205 A$=INKEY$:IF A$<>""THEN 255
ELSE 215
215 IF PEEK(341)=247 THEN V=V-ST
225 IF PEEK(342)=247 THEN V=V+ST
235 IF PEEK(343)=247 THEN H=H-ST
245 IF PEEK(344)=247 THEN H=H+ST
255 IFA$="0" OR PS=2 THENPS=2:PS
ET(H,V,0):PSET(H,V,5):PSET(H,V,0
)
265 IFA$="5" OR PS=1 THENPS=1:PS
ET(H,V,5)
275 IFPEEK(345)=247 THENRETURN
285 IF A$="D" THEN315
295 GOTO 205
305 'load,save,or see display
315 '
325 CLS:PRINT"WANT TO SEE [D]ISP
LAY OR [S]AVE TO DISKETTE OR [L
]OAD FROM DISKETTE OR [R]ET
URN FOR ANOTHER DRAW"
335 INPUT W$
345 IF W$="R" THEN RUN
355 IF W$="D"THEN425
365 IF W$="S"THEN395
375 IF W$="L"THEN415
385 IF W$<>"D" OR W$<>"S" OR W$<
>"L" OR W$<>"R" THEN325
395 LINEINPUT"FILENAME :";G$
405 SAVEM G$,9728,15871,9728
415 IFW$<>"L"THEN425ELSELINEINPU
T"FILENAME :";G$:LOADM G$
425 INPUT"IS FIGURE TO BE <S>TAT
IC OR <R>UNAWAY ";U$:IFU$="S
"THEN435ELSEIFU$="R"THEN545ELSEI
FU$<>"S" OR U$<>"R"THEN425
430 '** see display **
435 PMODE4,1:SCREEN1,1:PCLS
445 FOR W=0 TO 210 STEP 30
455 FOR W1=0 TO 150 STEP 30
465 GH=GH+1:IFGH=DF THENGH=0:GOT
O325
475 PMODE4,5
485 GET(W,W1)-(W+30,W1+30),MA,G
495 PMODE4,1
505 PUT(105,80)-(135,110),MA,PSE
T
515 NEXT W1,W
525 ' ** go for another run **
535 GOTO 325
545 PMODE4,1:SCREEN1,1:PCLS
555 FOR W=0 TO 210 STEP 30
565 FOR W1=0 TO 150 STEP 30
570 IFDI>195 THENDI=0
575 PMODE4,5
585 GET(W,W1)-(W+30,W1+30),MA,G
595 PMODE4,1
605 PUT(E+DI,F)-(E1+DI,F1),MA,PS
ET
610 DI=DI+2
615 NEXTW1,W
625 GOTO 325
700 END
9999 GOTO 9999
10000 PMODE4,5:SCREEN1,1
10001 GOTO 10001
10002 SAVE"27B:3":END'7
```

PRIVATE PROGRAM

by Keith Holzapfel
UTILITY
32K ECB

THIS LITTLE UTILITY will come in handy with people who keep their bank account and other data files on tape or disk as it stops others from looking at things that you don't wan't them to.

When loaded and RUN, it displays a warning that if an incorrect password is entered the program will be deleted (NEW'ed).

Once the program is RUN, it can't be stopped by pressing <reset> and can't be listed.

However, you can press break only in one place - but you still won't see the passwords.

The program has two passwords:

* one for the listing in case you want to change the password or alter the program and ...

* one for the running of the program.

To use this program you will need to renumber your program so that this utility will fit on top (renum28,?,?) and change the passwords in lines 18 & 19 to whatever you want.

There are two other programs for cassette based systems that might be handy for you. The first is by Bill Snow and it is called "AutoLoad" & "Protect" (CoCo June 87 P31)

The other is by John L Nicolettos and is called "Merge" (CoCo May 87 P28).

I think that about covers it, so I'll say cheers and good health to all.

The Listing:

```
0 GOTO 4
1 '********* PRIVPROG **********
   ******** REVISED BY *********
   **** KEITH V HOLZAPFEL ******
   ********* 28/08/87 **********
2 '*** LINES 4,5,6 FROM COCO ***
  '* PEEKS POKES & EXEC JAN 86 *
3 SAVE"58:3":END'7
4 POKE383,158
```

continued on p 47

ROTATE & EDITOR

By Colin Gawn
UTILITY
32K DECB

WELCOME TO THE editor. The editor is a program to help you create and edit points on an x, y, and z axis, so you can watch a 3-dimensional object rotate, using the "Rotate" program (listing 2).

The idea is that you set the x-coordinate, the y-coordinate and the z-coordinate on your screen, and then doing the same to all the other points.

When all sets of points have been given, write them down and place them into a section of the program called, "New Display".

By selecting the "Edit" functions, you can edit the x, y and z points, to check if you have not made any mistakes.

If there aren't any (or there aren't any more), select the "Compute Points" section and wait until filing is complete.

When program has finished, RUN"ROTATE" and watch the display!

When the computer asks "Spin around X axis", this means (if you say 'Y') that the outside point of your object will spin around 0, 0, 0, which is exactly in the middle.

This also applies to the Y and Z axis.

If an object has over 500 points, adjust the DIM statements for a figure over 500 points.

The points of your object must be within -100 and +100 and the X, Y, and Z axis, with point 0 being the dead center.

Remember, the more points you input into the computer, the slower it will be to calculate, although "Rotate" won't be affected by the number of points inputted.

It is also a great idea to plot your co-ordinates on a piece of paper first. When you put the co-ordinates in, remember also to put them into the computer in sequence.

On the disk version of the CoCoOz monthly, there will be an example of picture analysis.

Listing One

```
0 GOTO5
1 '***** "ROTATE"
2 '***** COLIN GAWN
3 SAVE"27:3":END'7
5 C1=65495:C2=65494:C3=C1:C4=C2:
C5=65497:C6=C1
10 DIMX(500),Y(500),Z(500),XZ(50
0),YZ(500)
15 A=1:M=330:I=30:AZ=1
20 CLS3:PRINT@6,"WELCOME TO THE
EDITOR";
21 PRINT@128,"WHICH MODEL OF COM
PUTER":PRINT@196," (1) COCO 1";:
PRINT@260," (2) COCO 2";:PRINT@3
24," (3) COCO 3";:KN$=INKEY$:IFK
N$="1"THENFP=C1:SP=C2:ELSEIFKN$=
"2"THENFP=C1:SP=C2:ELSEIFKN$="3"
THENFP=C5:SP=C1:ELSE21
22 CLS3:PRINT@8,"m    e    n
u";:PRINT@65,"iNSTRUCTIONS";:PRI
NT@99,"nEW DISPLAY";:PRINT@133,"
eDIT RECORDS";:PRINT@167,"rUN RO
TATE";:PRINT@201,"pRINTER";:PRIN
T@235,"dIRECTORY";:PRINT@269;"Ex
AMPLE";:PRINT@303,"cOMPUTE POINT
S";:PRINT@361,"EXIt";
23 PRINT@418,"TYPE LOWERCASE";:P
RINT@454,"CHARACTER TO BEGIN";
24 A$=INKEY$:IFA$="I"THENGOTO400
ELSEIFA$="N"THENGOTO30ELSEIFA$="
E"THENGOTO40ELSEIFA$="R"THENGOTO
1000ELSEIFA$="P"THENGOTO900ELSEI
FA$="D"THENGOTO800
25 IFA$="X"THENGOTO700ELSEIFA$="
C"THENGOTO49ELSEIFA$="T"THENDRIV
E0:ENDELSE24
30 CLS:PRINT"NEW DISPLAY":PRINT:
INPUT"DRIVE NO.";DR:DRIVEDR:INPU
T"NAME OF FIRST FILE";F1$
31 INPUT"NO. OF POINTS";PO:PRINT
:PRINT"ENTER DATA AS X,Y,Z":RN=1
32 OPEN"D",#1,F1$,30:FORRE=1 TO
PO:PRINT"NO.";RE;:INPUT"X,Y,Z";X
(RE),Y(RE),Z(RE):WRITE#1,X(RE),Y
(RE),Z(RE):PUT#1,RN:RN=RN+1:NEXT
RE
33 PRINT:PRINT"ALL DONE":SOUND1,
5:CLOSE#1:GOTO22
40 CLS:PRINT" (1) EDIT LINE"
41 PRINT" (2) LIST RECORDS"
42 PRINT" (3) EXIT"
43 EXEC45045 A$:IFA$="1"THEN44EL
SEIFA$="2"THEN47ELSEIFA$="3"THEN
22ELSE43
44 INPUT"DRIVE NO.";DR:DRIVEDR:I
NPUT"FILE NAME";F1$:INPUT"WHICH
RECORD NO.";RE:OPEN"D",#1,F1$,30
:NU=LOF(1)-1:IFRE>NU THEN22ELSEG
ET#1,RE:INPUT#1,X(RE),Y(RE),Z(RE
):CLOSE#1:PRINT"NO.";RE,X(RE);Y(
RE);Z(RE):EXEC44539
45 INPUT"EDIT THIS LINE (Y OR N)
";J$:IFJ$="Y"THEN46ELSEIFJ$="N"T
HEN40ELSE45
46 PRINT:INPUT"NEW POINTS";X(RE)
,Y(RE),Z(RE):OPEN"D",#1,F1$,30:W
RITE#1,X(RE),Y(RE),Z(RE):PUT#1,R
E:CLOSE#1:GOTO40
47 INPUT"DRIVE NO.";DR:DRIVEDR:I
NPUT"FILE NAME";F1$:OPEN"D",#1,F
1$,30:NU=LOF(1):FORRE=1 TO NU:GE
T#1,RE:INPUT#1,X(RE),Y(RE),Z(RE)
:PRINT"NO.";RE,X(RE);Y(RE);Z(RE)
:NEXT:CLOSE#1:EXEC44539:GOTO40
49 CLS
50 INPUT"SPIN AROUND X AXIS (Y O
R N)";C$
60 IFC$="N"THENINPUT"LOCK ON POI
NT(IN DEGREES)";C
70 INPUT"SPIN AROUND Y AXIS (Y O
R N)";D$
80 IFD$="N"THENINPUT"LOCK ON POI
NT(IN DEGREES)";D
90 INPUT"SPIN AROUND Z AXIS (Y O
R N)";B$
100 IFB$="N"THENINPUT"LOCK ON PO
INT(IN DEGREES)";B
110 '**INPUT COORD**
112 PRINT:PRINT:INPUT"NAME OF FI
RST FILE";F1$
113 PRINT:INPUT"DRIVE NO.";DR:DR
IVEDR:INPUT"NAME OF SECOND FILE"
;F2$
120 FORS=0 TO M STEP1:R=S/57.295
77951
130 PRINT@290,"STEP POINT";S
140 OPEN"D",#1,F1$,30:NU=LOF(1):
FORRE=1 TO NU:GET#1,RE:INPUT#1,X
(RE),Y(RE),Z(RE):NEXT:CLOSE#1
145 POKEFP,0
150 IFB$="Y"THEN170ELSEIFB=0 THE
N190ELSEE=R
160 IF B>0 THEN R=B/57.29577951
170 FORA=1TOPO:X=X(A)*COS(R)+Y(A
)*SIN(R):Y=-X(A)*SIN(R)+Y(A)*COS
(R):X(A)=X:Y(A)=Y:NEXT
180 IFB$="Y"THEN190ELSER=E
190 IFC$="Y"THEN210ELSEIFC=0 THE
N230ELSEE=R
200 IFC>0 THENR=C/57.29577951
210 FORA=1TOPO:Z=Z(A)*COS(R)+Y(A
)*SIN(R):Y=-Z(A)*SIN(R)+Y(A)*COS
(R):Y(A)=Y:Z(A)=Z:NEXT
```

```
220 IFC$="Y"THEN230ELSER=E
230 IFD$="Y"THEN250ELSEIFD=0 THE
N270ELSEE=R
240 IFD>0 THENR=D/57.29577951
250 FORA=1TOPO:Z=Z(A)*COS(R)+X(A
)*SIN(R):X=-Z(A)*SIN(R)+X(A)*COS
(R):Z(A)=Z:X(A)=X:NEXT
260 POKESP,0:IFD$="Y"THEN270ELSE
R=E
270 OPEN"D",#1,F2$,10
280 FORA=1TOPO:X(A)=X(A)+128:Y(A
)=96-Y(A):XZ(A)=FIX(X(A)):YZ(A)=
FIX(Y(A)):WRITE#1,XZ(A),YZ(A):PU
T#1,AZ:AZ=AZ+1:NEXT
290 CLOSE#1
300 NEXTS
305 GOTO 22
400 CLS
405 PRINT"WELCOME TO the editor"
410 PRINT"TO START OFF, EVERY FE
ATURE HAS  BEEN THOUGHT OF TO MA
KE THIS A  VERY VERSITILE PROGRA
M."
415 PRINT"(1) WITH THE ExAMPLE P
ICTURE     GIVEN PLACE 1ST POINT
 IN WHICH  OBJECT IS TO START BY
 GETTING   THE X POINT PLUS THE
Y POINT    AND THE Z POINT THEN
GOING TO   THE NEXT SET AND DOIN
G THE SAME"
420 PRINT"(2) WHEN ALL SETS OFF
POINTS     HAVE BEEN GIVEN WRITE
 THEM DOWN AND PLACE THEM INTO n
EW DISPLAY"
425 PRINT@482,"press a key to co
ntinue";:EXEC44539
430 CLS:PRINT"(3) CHECK TO SEE I
F YOU HAVEN'T  MADE ANY MISTAKES
 WITH eDIT     IF NOT PULL cOMPU
TE POINTS      AND WAIT UNTIL FI
LING IS        COMPLETE."
435 PRINT"(4) WHEN COMPLETED RUN
 *ROTATE*  AND WATCH FOR DISPLAY
"
440 PRINT"(5) file1=THE POINTS Y
OU WRITE   DOWN AND FEED TO THE
COMPUTER'S FILING SYSTEM."
445 PRINT"(5) file2=POINTS CALCU
LATED BY   COMPUTING SETS OF POI
NTS ABOUT  360 DEGREES THEN FILI
NG THEM    INTO A SECOND FILE(fi
le2)."
450 PRINT@482,"press a key to co
ntinue";:EXEC44539
455 CLS:PRINT"(6) WHEN COMP ASKS
 'SPIN AROUND X AXIS' THIS MEANS
 THE OUTSIDE  POINT OF YOUR OBJE
CT WILL BE   SPINNING AROUND PO
INT 0,0,0    WHICH IS EXACTLY I
N THE MIDDLE  WHERE ALL THE PLAN
ES MEET."
460 PRINT"THIS ALSO APPLIES TO Y
 AND Z     AXIS."
465 PRINT"(7) THERE IS ALSO A OP
TION FOR   SENDING YOUR COORDINAT
ES TO      PRINTER FOR CLOSER INS
PECTION"
470 PRINT"(8) YOU CAN CALL DIREC
TORY TO    CHECK YOUR DISK WHAT T
HE COMP    IS WRITTEN OR JUST ABO
UT ANTHING"
475 PRINT@482,"press a key to co
ntinue";:EXEC44539
478 CLS:PRINT"IF YOU ARE ATTEMPT
ING TO MAKE ANOBJECT WITH OVER 5
00 POINTS      REDESIGN THE DIM S
TATEMENT TO    SUIT YOUR NEEDS.":
PRINT"ALSO POINTS MUST RANGE WIT
HIN    -100 TO +100 ON X PLANE Y
PLANE AND Z PLANE WITH POINT 0 A
T DEADCENTER."
479 PRINT" a trial data file has
 been added to this disk for an
example of the points and how th
ey are      displayed on screen."
:EXEC44539
480 CLS:PRINT"(9) THE MORE POINT
S YOU INPUT    INTO THE COMP THE
SLOWER IT WILLBECOME TO CALCULAT
E, BUT THE     =ROTATE= PROGRAM W
ILL DISPLAY    THE OBJECT IN A VE
RY SHORT TIME."
485 PRINT"ONE LAST POINT...MAKE
SURE YOU   WORK OUT YOUR OBJECT F
IRST ON    PAPER THEN FIND YOUR S
ETS OF     POINTS."
490 PRINT"ALSO REMENBER
           each point must be dra
wn to the next one with no break
s in       between ie i n  s e q
u e n c e."
699 EXEC44539:GOTO22
700 PMODE4,1:SCREEN1,1:PCLS:LOAD
M"EXAMPLE":EXEC44539:GOTO22
800 CLS7:PRINT:PRINT:INPUT"DRIVE
 NO.";DR:DIR DR:PRINT:PRINTFREE(
0)*2305;" BYTES LEFT":PRINT:PRIN
T"PRESS ANY KEY TO RETURN":EXEC4
4539:GOTO22
900 CLS2:PRINT:INPUT"ACTIVATE PR
INTER [Y OR N]";P$:IFP$="N"THEN2
2ELSEINPUT"BAUD RATE 1=600   2=2
400";BA:IFBA=1 THENPOKE149,0:POK
E150,87ELSEPOKE149,0:POKE150,18
905 INPUT"NAME OF FILE";F1$:OPEN
"D",#1,F1$,30:NU=LOF(1)-1:FORRE=
1 TO NU:GET#1,RE:INPUT#1,X(RE),Y
(RE),Z(RE):PRINT#-2,"NO.";RE,X(R
E);Y(RE);Z(RE):NEXT:CLOSE#1:PRIN
T:PRINT"PRINTING COMPLETED":EXEC
44539:GOTO22
1000 CLS8:PRINT:PRINT"RUNNING -R
OTATE- WILL TERMINATE THIS PROGR
AM":INPUT"ACTIVATE [Y OR N]";G$:
IFG$="Y"THENRUN"ROTATE"ELSEIFG$=
"N"THENGOTO22
```

Listing Two

```
1 'THE ROTATE PROGRAM
2 'ADAPTED FOR 32K/64K DECB
3 'BY COLIN GAVN
4 '10 SIMOUNDS ST
5 'ALICE SPRINGS 5750
6 '22:6:1987
7 CLS:PRINT"ACTIVATE SPEED POKE"
:PRINT"   (1)COCO 1 & 2
          (2)COCO 3
          (3)NO POKE 1 & 2
          (4)NO POKE 3
8 EXEC44539:P1=65495:P2=65497:Q=
0:P3=65494
9 A$=INKEY$:IF A$="1"THEN POKE P
1,Q ELSE IF A$="2"THEN POKE P2,Q
 ELSE IF A$="3"THEN POKE P3,Q EL
SE IF A$="4"THEN POKE P1,Q
10 DIMXZ(500),YZ(500):CLS4:PRINT
@106,"ROTATE";:PRINT@169,"BY C.G
AWN";:PRINT@259,"3 ROTATIONS IN
3 PLANES";:PRINT@419,"PRESS ANY
KEY TO BEGIN";:EXEC44539
11 CLS:PRINT"   (1) RUN FOR NEW D
ISPLAY
                (2) RERUN SAME DI
SPLAY
                (3) RUN EDITOR

                (4) DIRECTORY"
12 :PRINT"
            (5) EXIT ROTATE"
13 EXEC44539
14 A$=INKEY$:IFA$="1"THEN30ELSEI
FA$="2"THEN40ELSEIFA$="3"THENRUN
"EDITOR"ELSEIFA$="4"THEN500ELSEI
FA$="5"THENPOKEP3,Q:ENDELSE14
30 CLS:PRINT:PRINT:INPUT"DRIVE N
O.";DR:DRIVEDR:INPUT"FILE NAME:"
;F2$
31 PRINT:PRINT:PRINT"PRESS A KEY
 TO SEE DISPLAY"
32 PRINT:PRINT"AS EACH PICTURE I
S DISPLAYED    PRESS A KEY TO SE
E NEXT ONE":PRINT:PRINT"PRESS [m
] KEY TO RETURN TO MENU":EXEC445
39
40 DRIVE DR:AZ=1:M=330:I=30
50 PMODE4
60 SCREEN1,1
70 PCLS5
80 COLOR0,5
90 OPEN"D",#1,F2$,10:PO=LOF(1)/1
2
100 FORZ=0TOM STEPI
110 EXEC44539:PCLS
120 FOR A=1 TO PO
130 A$=INKEY$:IFA$<>"M"THEN140EL
SE240
140 GET #1,AZ
150 INPUT #1,XZ(A),YZ(A)
160 IFA=1 THENGOSUB230
170 LINE-(XZ(A),YZ(A)),PSET
180 IF AZ=LOF(1) THEN 200
190 AZ=AZ+1
200 NEXTA,Z
210 CLOSE #1
220 GOTO 40
230 LINE(XZ(A),YZ(A))-(XZ(A),YZ(
A)),PSET:RETURN
240 CLOSE#1:GOTO11
500 CLS:PRINT:INPUT"DRIVE NO.";D
R:DRIVEDR
505 DIRDR:PRINT:PRINTFREE(0)*230
5;" BYTES LEFT":EXEC44539
510 GOTO 11
512 SAVE"27A:3":END'7
```

HINT....

Donkey King

STILL can't past that first level? Well, mnaybe you need a helping hand ... try this:

1. type: (C)LOAD"DONKEY
2. type: POKE18888,23
3. type: EXEC

DATATRAN

By Gunnar Adamzewski

DATA TRANSFER UTILITY

DATATRAN ENABLES you to transfer (copy) ASCII data files between disk drive(s) and tape recorder in any sequence. That is, you load the file once from either tape or disk and then copy it as many times as you like to either tape or disk with the option of saving it with a different filename for each save.

It also allows you to view the file on the screen or send the file to a printer.

When printing to the screen, the data is printed as a line up to 249 characters long (per screenful) or up to the next carriage return in the file, whichever comes first. Note that this means some word processor files may not be transferred completely if they contain sentences longer than 249 characters as such lines are truncated and the excess 'lost'.

This is a restriction imposed by using LINE INPUT to read the source data file. However, ASCII data files generated by BASIC programs will transfer OK.

If anyone knows an easy way around this restriction please let me know!

WARNING!! The program has the option to exit to a COLD START in line 1180. It is essential that you (C)SAVE the program BEFORE RUN-ning it in case you make a 'miss-steak' and WIPE all your typing effort!

Note also that the program has a high-speed POKE in line 90 and that the printer BAUD RATE is set to 4800 in line 50.

Also, if your printer does not print in the 'wide' mode, change line 960 to:

```
960 PRINT#-2,IN$
```

All other instructions and prompts are included in the program.

The Listing:

```
0 GOTO10
3 SAVE"56:3":END'7
10 '********datatran*********
    *  GUNNAR ADAMCZEWSKI   *
    *     9 GANT ST.,       *
    *    LENAH VALLEY,      *
    ****TASMANIA. 7008*******
20 GOTO 990
30 SAVE"DATATRAN.BAS":END
40 AUDIO ON
60 POKE 150,7 '*BAUD RATE=4800*
70 DIM A$(150)
80 CLS
90 S$=CHR$(128):V$=CHR$(34)
100 POKE 65495,0
102 GOSUB 989
103 FOR I=1 TO 150
104 A$(I)=""
105 NEXT I
106 POKE 65494,0
110 PRINT@129,"LOAD DATA FROM:"
120 PRINT@203,"1. DISK"
130 PRINT@235,"2. TAPE"
140 PRINT@263,"OR:"
150 PRINT@299,"3. DISK DIRECTORY
"
160 PRINT@331,"4. END (COLD STAR
T)"
170 PRINT@457,"<SELECT 1-4>"
180 LC=469:GOSUB 930
190 ON CH GOTO 290,300,280,980
200 SOUND 100,4:GOTO 180
280 GOSUB 981:GOTO 100
290 TD$="DISK":GOTO 310
300 TD$="TAPE"
310 GOSUB 890
320 PRINT@225,"ENTER NAME OF "+T
D$+" FILE TO BE   LOADED:"
330 IF CH=1 THEN PRINT@325,"<FIL
ENAME.EXT:DRIVE>":GOTO 360
340 PRINT@325,"<FILENAME>"
350 PRINT@449,"PRESS <ENTER> IF
NAME UNKNOWN"
360 PRINT@358,"";:LINEINPUT IN$
400 IF CH=1 THEN A=1 ELSE A=-1
405 '**LOAD TAPE OR DISK FILE**
410 GOSUB 890
420 SOUND 200,2
430 PRINT@229,"loading "+IN$+"..
."
440 OPEN"I",#A,IN$
450 L=0
460 IF EOF(A) THEN 500
470 LINEINPUT#A,A$(L)
480 L=L+1
490 GOTO 460
500 CLOSE#A
510 GOSUB 890
530 PRINT@101,IN$+" loaded"
535 SOUND 200,1
540 PRINT@164,"OUTPUT DATA TO:"
550 PRINT@205,"1. DISK"
560 PRINT@237,"2. TAPE"
570 PRINT@269,"3. SCREEN"
580 PRINT@301,"4. PRINTER"
590 PRINT@330,"OR 5. DIRECTORY"
595 PRINT@365,"6. LOAD NEW FILE"
600 PRINT@455,"< SELECT 1-6 >"
610 LC=470:GOSUB 930
620 ON CH GOTO 640,650,749,816,8
82,880
630 SOUND 100,4:GOTO 610
635 '**OUTPUT TO TAPE OR DISK**
640 TD$="DISK":GOTO 660
650 TD$="TAPE"
660 GOSUB 890
670 PRINT@133,"*OUTPUT DATA TO "
+TD$+"*"
695 PRINT@453,"PRESS <ENTER> TO
ABORT"
700 PRINT@261,"enter"+S$+"output
"+S$+"filename"
710 IF CH=1 THEN PRINT@325,"<FIL
ENAME.EXT:DRIVE>":B=1:GOTO 730:E
LSE B=-1
720 PRINT@325,"<FILENAME>"
730 PRINT@358,"";:LINEINPUT F$
735 IF F$="" THEN SOUND 100,4:GO
SUB 890:GOTO 530
738 GOSUB 890:SOUND 100,4
739 PRINT@262,"saving "+F$
740 OPEN"O",#B,F$
741 FOR I=0 TO L-1
742 PRINT#B,A$(I)
743 NEXT I
744 CLOSE #B
745 GOSUB 890
746 PRINT@101,F$+" saved"
747 GOTO 535
748 '**OUTPUT DATA TO SCREEN**
749 GOSUB 890
750 FOR I=0 TO L
760 IF A$(I)="" THEN PRINT@266,"
<BLANK LINE>" ELSE PRINT@160,A$(
I)
770 PRINT@451,"PRESS ANY KEY- <X
> TO EXIT";
780 LC=478:GOSUB 930
785 IF I$="X" THEN GOSUB 890:GOT
O 810
787 GOSUB 890
790 NEXT I
810 GOTO 510
```

continued on p 52

ZOOMER

By Dennis Mellican
UTILITY
32ECB

WARNING - THIS PROGRAM requires a lot of patience!!
Zoomer lets you:

* Draw very detailed graphics pictures,
* Zoom in on pictures, 6 times magnification,
* Reverses the video
* Copies graphics pages,
* has save and load a picture facilities.

Using Zoomer

If your CoCo can't take the speed up POKE (POKE65496,0) then take it out. It appears on lines 70 and 850.

After typing RUN, you will see the graphics screen with a flashing cursor, in which you can move using a joystick or a mouse.

Once the cursor is positioned over a desired area, press the fire button.

After a few seconds you will be in the "Draw mode". In this mode you can see your "zoomed-in" area and a range of colours, which you can use by positioning the cursor over and press the fire button.

After picking a colour, position the cursor over the zoomed in area and press the fire button. This will draw on the graphics screen using the zoomed in area as a help or an aid.

To see what you did on the graphics screen, press <d>.

If you want to clear the screen, press <f>, to fill the zoomed in area with a colour you previously selected.

To quit the drawing mode, press <q>. You will then return to the graphics screen with the flashing cursor. You can repeat the above process or go to the main menu by pressing <m>.

The menu will display eight functions:

1. Display Picture
This lets you see your graphic screen.
2. Draw or change picture
Returns to the graphics screen with the "flashing cursor" scenerio.
3. Zoom in on picture
Like it says, you can zoom in on your picture. The size of the picture depends upon the number of times you zoom in. The speed of drawing the "zoomed-in" area depends on this also.
Note that the graphics pages 1 to 4 are copied. To exit from this function, press <e>.
Use the arrow keys to move the square (which when you press <enter> becomes the enlarged, or "zoomed in" area), and wait for the result.
4. Reverse Video Picture
This lets you reverse the whole graphics screen you last viewed.
5. Display Graphics Page
This lets you choose which graphics pages you want to draw on or display.
6. Copy Graphics Pages
Copies graphics pages - what else??
7. Save results on tape
Saves your creations on tape for you.
8. Load from tape
Loads in your creations, from tape.

Have fun using "Zoomer"!

The Listing:

```
0 GOTO10
1 '***** "ZOOMER"
2 '***** DENNIS MELLICAN
3 SAVE"23:3":END'7
10 '''''''''''''''''''''''
20 '       ZOOMER       '
30 ' BY DENNIS MELLICAN '
40 '''''''''''''''''''''''
50 PCLEAR8:CLEAR1000
60 CLS
70 POKE65495,0 'SPEED POKE *THER
E ARE MORE SPEED POKES IN THE LI
STING.DELETE SUCH STATEMENTS IF
YOUR SYSTEM IS SLOW
80 SR=1:M=4:S=1
90 DIM W(17,12),P(8,12)
100 PMODE4,SR:SCREEN1,S:IF M=1 T
HEN M=3
110 C=JOYSTK(0):D=JOYSTK(1)
120 A$=INKEY$
130 IF A$="M" THEN SCREEN0,0:GOT
O 750
140 A=C*3.80952381:B=D*2.8571428
6
150 GET(A,B)-(A+17,B+12),W,G
160 PUT(A,B)-(A+17,B+12),W,PRESE
T
170 IF PEEK(65280)=126 OR PEEK(6
5280)=254 THEN PK=1
180 PUT(A,B)-(A+17,B+12),W,PSET
190 IF PK=1 THEN PK=0:GOTO 210
200 GOTO 110
210 CLS:SCREEN0,0:C=A:D=B
220 FOR E=32 TO 416 STEP 32:FOR
T=8 TO 25
230 P=PPOINT(A+(T-8),B+(E/32)-1)
240 IF P=1 AND M=3 THEN IF S=0 T
HEN CC=-1 ELSE CC=207
250 IF P=2 AND M=3 THEN IF S=0 T
HEN CC=159 ELSE CC=223
260 IF P=3 AND M=3 THEN IF S=0 T
HEN CC=175 ELSE CC=239
270 IF P=4 AND M=3 THEN IF S=0 T
HEN CC=191 ELSE CC=255
280 IF M=3 THEN 320
290 IF P=5 THEN CC=207
300 IF P=1 THEN CC=207
310 IF P=0 THEN CC=128
320 IF CC=-1 THEN POKE1024+T+E,3
2:GOTO 340
330 PRINT@T+E,CHR$(CC);
340 CC=128
350 NEXTT:F=F+1:NEXT:F=0
360 E=-1:FOR T=96 TO 320 STEP 32
:E=E+1:PRINT@T+5,CHR$(143+(E*16)
);:NEXT:POKE1024+5+96,32:PRINT@5
+352,CHR$(128);
370 A=JOYSTK(0):B=JOYSTK(1)
380 A$=INKEY$
390 E=INT(A/2):F=INT((B/4))*32
400 PK=PEEK(1024+E+F)
410 POKE1024+E+F,42:POKE1024+E+F
,PK
420 IF PEEK(65280)=126 OR PEEK(6
5280)=254 THEN 470
430 IF A$="D" THEN SCREEN1,S:EXE
C44539:SCREEN0,0
440 IF A$="Q" THEN 100
450 IF A$="F" THEN PRINT@480,"AR
E YOU SURE (Y/N) ?";:GOTO 550
460 GOTO 370
```

```
470 IF INT(E)=5 AND F>=96 AND F<
=352 THEN SOUND1,1:CC=PK:GOTO 46
0
480 IF INT(E)>7 AND INT(E)<26 AN
D F>16 AND F<448 THEN PK=CC:SOUN
D198,1:POKE1024+E+F,PK:GOTO 500
490 GOTO 460
500 Z=(PK-143)/16+1
510 IF Z=-5.9375 THEN Z=1
520 COLOR Z
530 PSET(C+(E-8),D+(F/32)-1)
540 GOTO 460
550 A$=INKEY$:IF A$="" THEN 550
ELSE IF A$="Y" THEN 560 ELSE PRI
NT@480,"
   ";:GOTO 460
560 PRINT@480,"
          ";
570 FOR E=32 TO 416 STEP 32:FOR
T=8 TO 25
580 P=CC
590 PK=(1024+E+T)
600 POKE PK,CC
610 Z=(CC-143)/16+1
620 COLOR Z
630 PSET(C+(T-8),D+(E/32)-1)
640 NEXTT,E
650 GOTO 460
660 SCREEN 1,S
670 FOR T=0 TO 192 STEP 13
680 FOR E=0 TO 255 STEP 16
690 GET(E,T)-(E+15,T+12),W,G
700 IF A$="S" THEN PUT(E,T)-(E+1
5,T+12),W,PSET:GOTO 720
710 PUT(E,T)-(E+15,T+12),W,PRESE
T
720 NEXT E,T
730 SCREEN1,S
740 EXEC44539:SCREEN0,0:RETURN
750 GOTO 1140
760 SCREEN1,S:EXEC44539:SCREEN0,
0:GOTO 750
770 GOSUB660:GOTO 750
780 CLS:PRINT"SELECT PAGES TO BE
 COPIED:          a  PAGES 1 TO 4
                  b  PAGES 5 TO 8
                  c  EXIT"
790 A$=INKEY$:IF A$="" THEN 790
ELSE IF A$="A" THEN 800 ELSE IF
A$="B" THEN 810 ELSE IF A$="C" T
HEN 750 ELSE 790
800 PCOPY1 TO 5:PCOPY 2 TO 6:PCO
PY 3 TO 7:PCOPY 4 TO 8:GOTO750
810 PCOPY5 TO 1:PCOPY6 TO 2:PCOP
Y7 TO 3:PCOPY8 TO 4:GOTO 750
820 CLS:INPUT"WHICH PAGE (1-5)";
A$:GOSUB1110:GOTO750
830 IF SR<>1 AND SR<>8 THEN PMOD
E4,SR:SCREEN1,S:EXEC44539
840 PMODE4,1:SCREEN1,S:EXEC44539
:PMODE4,5:SCREEN1,S:EXEC44539:SC
REEN0,0:GOTO 750
850 POKE65494,0:PMODE4,1:SCREEN
1,S:CLOADM:EXEC44539:SCREEN0,0:P
OKE65495,0:GOTO750
860 CLS:INPUT"HOW MANY TIMES";MG
870 IF MG>6 OR MG<2 THEN 750
880 M1=INT(256/MG):M2=INT(192/MG
)
890 GOTO 1000
900 EXEC44539:GOTO 750
910 PCLS1:K=0:L=0:COLOR2
920 FOR B=E TO E+M2-MG:FOR A=T T
O T+M1
930 PMODE4,1:IFPPOINT(A,B)=0 THE
N PMODE4,5:SCREEN1,S:LINE(K,L)-(
K+MG,L+MG-1),PRESET,BF
940 K=K+MG
950 IF K>254 THEN K=0:A=T+M1
960 NEXT:L=L+MG:K=0:IF L>192 THE
N SOUND1,1:B=E+M2-MG
970 NEXT
980 EXEC44539
990 GOTO 750
1000 PMODE4,5:SCREEN1,S
1010 T=0:E=0:LINE(T,E)-(T+M1,E+M
2),PRESET,B
1020 A$=INKEY$:IF A$="" THEN 102
0
1030 IF A$="^" AND E>0 THEN E=E-
1:GOSUB 1100
1040 IF A$=CHR$(10) AND E+M2<192
 THEN E=E+1:GOSUB1100
1050 IF A$=CHR$(8) AND T>0 THEN
T=T-1:GOSUB1100
1060 IF A$=CHR$(9) AND T+M1<255
THEN T=T+1:GOSUB1100
1070 IF A$=CHR$(13) THEN 910
1080 IF A$="E" THEN 900
1090 GOTO 1020
1100 PCOPY1 TO 5:PCOPY2 TO 6:PCO
PY3 TO 7:PCOPY4 TO 8:LINE(T,E)-(
T+M1,E+M2),PRESET,B:RETURN
1110 A=VAL(RIGHT$(A$,1)):A=INT(A
):IF A<1 OR A>5 THEN PRINT"-?- I
LLEAGL FUNCTION CALL":RETURN
1120 SR=A
1130 PMODE4,A:SCREEN1,S:EXEC4453
9:RETURN
1140 CLS
1150 W$="Z O O M E R":SY=2:GOSUB
 1340:W$="BY":GOSUB1340:W$="DENN
IS C. MELLICAN":GOSUB1340:SY=SY+
1:W$="DO YOU WANT TO...":GOSUB13
40
1160 SY=SY+1
1170 W$="1. DISPLAY PICTURE
  ":GOSUB1340
1180 W$="2. DRAW OR CHANGE PICTU
RE":GOSUB1340
1190 W$="3. ZOOM IN ON PICTURE
  ":GOSUB1340
1200 W$="4. REVERSE VIDEO PICTUR
E ":GOSUB1340
1210 W$="5. DISPLAY GRAPHICS PAG
E ":GOSUB1340
1220 W$="6. COPY GRAPHICS PAGES
  ":GOSUB1340
1230 W$="7. SAVE RESULTS ON TAPE
  ":GOSUB1340
1240 W$="8. LOAD FROM TAPE
  ":GOSUB1340
1250 A$=INKEY$:IF A$="" THEN 125
0
1260 A=VAL(A$):A=INT(A):IF A<1 O
R A>8 THEN 1250
1270 ON A GOTO 830,100,860,770,8
20,780,1290,850
1280 GOTO 1250
1290 CLS:INPUT"HOW MANY PAGES (1
-8)";PG
1300 PG=INT(PG):IF PG<1 OR PG>8
THEN 750
1310 LINEINPUT"FILENAME:";FF$
```

continued on p 54

continued from p 38

```
300 PRINT:PRINT" THIS PROGRAM VI
LL CONVERT A      DIM STATEMENT F
ROM A COCO2       PROGRAM TO A HB
UFF STATEMENT     WHEN CONVERTING
 FROM COCO2 TO    COCO3."
310 PRINT:PRINT" YOU CAN ALSO CA
LCULATE YOUR      HBUFF SIZE WHEN
 WRITING YOUR     OWN PROGRAMS FO
R THE COCO3."
320 PRINT:PRINT" <1> CALCULATE H
BUFF SIZE."
330 PRINT" <2> CONVERT DIM STATE
MENT."
340 PRINT:PRINT"       ENTER <1>
OR <2>"
350 A$=INKEY$:IF A$=""THEN350
360 IF A$="1" THEN CLS5:GOSUB 40
:GOTO80
370 IF A$="2" THEN CLS5:GOSUB40:
GOTO390
380 GOTO350
390 X=0:Y=0
400 PRINT:PRINT:INPUT"  FIRST AR
RAY DIMENSION";X1
410 INPUT" SECOND ARRAY DIMENSIO
N";Y1
420 GOTO160
```

continued from p 42

```
5 POKE248,50:POKE249,98:POKE250,
28:POKE251,175:POKE252,126:POKE2
53,173:POKE254,165:POKE410,126:P
OKE411,0:POKE412,248
6 CLEAR200,31000:FORX=32742TO327
67:READ I:POKEX,I:NEXT:EXEC32762
:DATA58,142,58,18,16,222,33,48,1
40,246,159,166,28,175,127,255,64
,126,173,192,48,140,236,159,114,
57
7 CLS3
8 FOR TL=1 TO 10
9 PRINT@41," W A R N I N G ";
10 PRINT@96,"THIS PROGRAM WILL S
ELF DESTRUCT ";:PRINT@132,"IF IN
-CORRECT PASSWORD IS";:PRINT@172
,"<ENTERED>";
11 FOR X=1 TO 300:NEXT X
12 PRINT@41," - - - - - - - ";
13 FOR X=1 TO 300:NEXT X
14 NEXT TL
15 CLS7
16 PRINT@192,"PLEASE ENTER PASSW
ORD";
17 INPUT PW$
18 IF PW$="PASS-1"THEN POKE383,0
 :LIST:ELSE GOTO 19
19 IF PW$="PASS-2"THEN 28
20 CLS4
21 FOR TL=1 TO 5
22 PRINT@288,"A C C E S S  T O
P R O G R A M ";
23 PRINT@361," D E N I E D ";
24 FOR Z=1 TO 300:NEXT Z
25 PRINT@361," - - - - - - ";
26 FOR Z=1 TO 300:NEXT Z
27 NEXT TL:POKE113,0:EXEC40999
```

WHERE DID THAT PROGRAM COME FROM?

By Allan Thompson
ARTICLE

HOW MANY TIMES have you used a program from your collection and it needed some modification to suit your purpose? It might simply need different printer codes inserted or it may need variables changed or DIM statements. What do those variables mean? Where are they in the long program?

You need to consult the authors accompanying article but which magazine was it in? Like me, you probably have quite a collection of both mags dating way back. "Been there and done that" eh?

Isn't it frustrating? After searching for ages and probably not finding it you are called for tea .... or you see the sun on the horizon (grin).

You give up. Right? You use something else but that program was THE ONE you wanted. Hmm ...

As always, there is a solution. In fact there are two of them. The first invariably needs the second one first if you have a large collection like me. "Cut the bull and get to the point" you say. OK.

SOLUTION ONE

Put REM statements at the top of all your programs providing reference material. eg.

```
1 GOTO10
2 REM ************************
3 REM *      FILENAME/BAS      *
4 REM * BILL BLOGGS ADELAIDE  *
5 REM *  SEE P15,XX/86 A/COCO *
6 REM ************************
7 SAVE"FILENAME/BAS":END
10 CLS:PCLEAR....etc
```

SOLUTION TWO

Record all magazine information and program articles in a suitable database. These should be entered under key words or "fields" and sorted into alphabetical order.

The key word categories used would depend largely upon what your collection consisted of, how much detail you want to record and of course, the capacity of the the CoCo and database you choose to use.

To cover a wide range of topics such as those included in our Australian Softgold and CoCo, you would need to break the data up into major fields and sub-fields under them.

You could use:

```
* ARTICLES      * COMMUNICATION
* PROGRAMS      * REVIEWS
* TUTORIALS     * MODS
```

... for the major fields. These in turn, could be broken down.

* ARTICLES: COCOCONF, COCO HOUSEKEEPING, HINTS, USERS GROUPS and OTHER.

* COMMUNICATION: BBS, HARDWARE, SOFTWARE, GOLDLINK, VIATEL.

* PROGRAMS: APPLICATION, GAMES, GRAPHICS, MUSIC, ROBOTICS, SIMULATION, SOUND, UTILITIES and OTHER.

* REVIEWS: SOFTWARE and HARDWARE.

* TUTORIALS: ASSEMBLY, BASIC, EDUCATION, OS9 and OTHER.

* MODS: HARDWARE and SOFTWARE.

Once again, each of these could be broken down further. eg.

* GAMES: ADVENTURE, ARCADE, EDUCATION.

The database would need to cater for these fields and have sufficient space for each entry and the number of entries anticipated for each field and sub-field.

It should have a FAST sort (not like VIP) and most importantly, a search facility.

Most databases fall down in the area of capacity. VIP has a database which is a good one except for one thing.

For the life of me I cannot understand why they spoilt an otherwise versatile program with an agravatingly s-l-o-w sort facility.

It literally takes hours to sort a decent batch of input. They put bicycle wheels and pedals on a Rolls!! Why ??

Good grief there are fast M/L sorts. (There are some good sorts over here too. Coming over for the Grand Prix ?)

If you sort your data at regular intervals it helps but with a task like a cross-reference index of all our mags, it becomes tiresome.

I eventually sorted after each magazine was done. I even found that I had to break it up into separate data files for each year.

This was less than convenient because the search had to be done separately in each program. For some reason it also split the sort into two so you ended up with two alphabetical listings one under the other.

There must be something better.

Has anyone put their A/Softgold and A/CoCo mags into a cross- reference index database?

What program did you use? What are its features? If you know of a good one you would recommend, how about review of it in the Australian Softgold or CoCo. Has any smart cookie written one that could do this?

Do you have a memory saving technique which would help (coded entries)? Don't keep these things a secret. Let us in on it too.

If someone can come up with a decent program I can use, I would gladly make my Australian Softgold and CoCo cross-reference indexes produced thereon available to Graham for distribution through GOLDSOFT.

continued on p 61

DISK STRUCTURES (AND OTHER WAR STORIES)

By Alex Hartmann
ARTICLE

DISKS ARE A great device. They store and retrieve information quickly, and a fair amount as well. In fact, a disk, if totally filled with data, can hold as much as two C20 tapes, both sides.

But how many of us actually know what makes a disk tick? And how many more of us know how to restore a disk file that has crashed?

Not very many I bet - sure, there are those who have done that sort of thing before ... but like the old cliche goes, "Where are they when you need 'em?".

In the next few months, I hope to be presenting articles of a disk nature. In short, understanding a disk structure (this months special feature) and eventually being able to restore a disk file that previously crashed to working order.

Part One:
Understanding Disk Structures

To get started, you will need to get a disk of some sort (back-up'ed, of course), for your subject. We won't be changing any data on the disk, just having a look.

(Murphys law says that if anything can go wrong, it will. Therefore, make SURE you have a backup of your disk.)

Most people would have a Tandy drive of some sort, wheather it be the old grey or white vertical case or the slightly older FD500 and/or FD501 drive.

Therefore I will be talking about DOS'es that use 35 tracks, not some custom made 40 track disk drive (I'm not criticizing them, it's just that there are more 35 trackkers around).

The Physical Disk

Your disk (when formatted to a standard Tandy DOS) has been formatted to a specific way so that the computer can understand the data that was written to the disk.

A "disassembly" of a disk shows the following:

* Each disk contains 35 tracks (numbered 0 to 34),
* Each single track contains 18 sectors (numbered 1 to 18), and
* Each single sector contains 256 bytes of data (256 characters).
* Therefore a full disk could contain (256 bytes * 18 sectors * 35 tracks) 161,280 bytes.

Now the above sounds alright ... so far. But it goes deeper than that!

* There are 70 granules to a disk.
* Each granule is 9 sectors, or 2,304 bytes in length.

Granule? Wasn't that the word they described the alien with in a movie I saw last week?

* Not quite. As you know, a granule is 2,304 bytes in length.
* So when the computer saves a file that is 890 bytes long, it allocates 2,304 bytes (or a granule) of disk space to it.
* If that particular file grows to (say) 2,305 bytes in length (one byte more than 2,304 bytes), it gets allocated 4,608 bytes (or two granules) disk space.

But what of the disk directory?

* The disk directory is two granules in length.
* It is located conveniently in the 'middle' of the disk, or track 17 (which is half of 35).
* It really only uses the first half of track 17 for filenames and file allocation.
* So you really only get 68 granules (or 156,672 bytes) of disk space for your data and programs.

More on ... 'Track 17'

Track 17 is divided into two lots - the granule allocation table (the GAT) and the File Allocation Table (the FAT).

The GAT is located on sector 2. This sector is the 'contents page' for the entire disk. It knows where any piece of your data is located on the disk.

This particular sector is only 68 bytes in length (even though it can hold up to 256 bytes at a time).

If the 68 bytes are examined, they will show that they are made up of only three types of characters:

* A hexadecimal value of FF (CHR$(255)) - this states that this is a free sector.
* A hexadecimal value between 00 and 43 (decimal 0 to 67) - this states that the granule is part of a disk file. If the value is converted to decimal, it points to the next granule in line to be loaded into the computer.

For example, if the value of a byte is 12, then granule 12 is the next granule to be loaded off the disk.

* A hexadecimal value between C0 and C9 (or 192 to 201) - this means that this is the last granule in the file to load. The idea here is to convert the value to a binary number, and then taking only bytes 0 to 5 and converting them to decimal, ie:

Val	Binary	Altern:	Dec:
192	11000000	00000000	0
193	11000001	00000001	1
194	11000010	00000010	2
195	11000011	00000011	3
196	11000100	00000100	4
197	11000101	00000101	5
198	11000110	00000110	6
199	11000111	00000111	7
200	11001000	00001000	8
201	11001001	00001001	9

So if a byte contained the value of 195, then (according to

the above chart) the computer would have to load in an extra 3 full sectors from that granule.

The FAT is located on sectors 3 to 11. Disk file descriptors (the 'filename' plus the list below) are 32 chracters in length and contain information like ...

* the name of the file
* the extension of the file
* what the file is, eg
  - Basic program
  - Basic data file
  - Machine language program
  - Other
* If the file is in ASCII or not
* Where it can find the first granule of the file
* and the number of bytes the program uses in the last sector of the file.

That's basically it, descriptive-wise.

Simulated loading of a file

Let's put all of the above into practice!

There is a file on a disk called "myprog/bas". The specifications are:

* it is two granules long,
* it has been saved in ASCII,
* it is a basic program,
* we know that it is 3,000 bytes long.
* it is the first file in the directory.

So we type (don't really type this!):

```
LOAD"MYPROG"
```

CoCo will do the following:

1. Go to track 17, sector 3 (beginning of the File Allocation file)
2. Search for a file called "myprog/bas" (the extension is "BAS" by default).
3. When "myprog" has been found, it notes the following information:
   - "myprog" is a basic program
   - "myprog" is saved in ASCII
   - "myprog" is located on granule 34
   - "myprog"'s number of bytes used in the last sector of the file equal 184.
4. It then goes to the granule allocation table (the GAT on sector 2 of the same track) and reds in byte number 34 (granule number 34).
5. Granule number 34 contains a value of 35. So CoCo knows to load in granule number 34 as well as what granule 35 will say.
6. Granule number 35 is read - it contains a value of 'C2' or 194. Our table above shows that the value of 194 equals 2 sectors.
7. Off CoCo goes to load in:
   - granule number 34 (which is track 18, sectors 1-9),
   - 2 sectors off granule number 35 (which is track 18, sectors 10-18)
   - and 184 extra bytes off sector 3,

... to complete the loading of the program. It then comes back to give the 'OK' prompt, which completes the process!

Getting into it - user style

Although one disk file has been alloted 32 characters in length, the actual data involved is really only 16 characters long.

A typical disassembly of a disk file is as follows:

* bytes 1 - 8: Contains the filename of the program.
* bytes 9 - 11: contains the extension of the above program.
* bytes 12: describes the file; values for this byte can only be one of the following:
  0 - means the file is a Basic program.
  1 - means the file is a Basic data file.
  2 - means that the file is a Machine-Language program.
  3 - Usually means source code from some program, eg Disk EDTASM.
* byte 13: The ASCII flag. If the value is ...
  255 - then the file is in an ASCII format.
  0 - then the file is saved in binary format.
* byte 14: represents the first granule in the file.
* bytes 15-16: added together will get you the number of bytes to load in the last sector of the file.

Commands you'll need to know

To fully utilize all the above information, there are two commands that you could get to master before next month's tutorial. They are DSKI$ and DSKO$. They are two very powerful commands, and they're purpose is to (in respective order) read a disk sector and write a disk sector.

Their syntax is:

DSK<'I' for input, 'O' for output>$(drive number),(track number),(sector number),(string one),(string two)

To apply: if I wanted to read track 17, sector 3 on drive 1, I would type in:

```
DSKI$1,17,3,A$,B$
```

Now, as you will notice, there are two strings here: **A$ and B$.**

Each disk sector, don't forget, is 256 characters in length, and you can't allot more than 255 characters to one string.

So the disk sector had to be split in two ways, each 128 characters in length.

The end, for now ...

Remember, always work on a disk that has been backup-ed - failure in doing so could cost you your disk - if you do something wrong.

Also remember Murphys law - if something can go wrong, it will.

Until next month, where we attempt to resurrect a KILLED file, ie you've just killed a file on the disk you didn't want to kill.

⊕

A HINT

For those owners of the TP10 Printer who have, no doubt, found that the thermal paper rolls is quite hard to find in Tandy Stores, here is a solution.

Teletype rolls of paper are thermal operated and are exactly twice the width of the TP10 rolls and are also about twice the length.

Thus, by making up a V type jig of wood and using a fine toothed band saw it is possible to slice these rolls in half and you then have a plentiful supply of printing paper.

Save the old spools from the original rolls and hand roll the amount you need on to these spools. The cost of these rolls is about $15 which gives you the equivalent of more than eight rolls priced at $32.

Another small hint to assist loading the rolls in the printer is to snip off the corners of the end of the roll before feeding it in.

REVIEW

Software:
ZONE RUNNER and CAVE WALKER

ZONE RUNNER IS not your average run of the mill or "kamakazi shoot-em up 'till I die" games; quite the opposite.

Zone (short for "Zone Runner") actually requires some brains, and when brain is put into gear, this game is VERY enjoyable!

What's it all about?

In Zone, you are an interplanetary trader, the basic idea being to buy goods from one planet and sell the goods for a higher price to another planet and thus make money.

When you start out, you have the following:

* 100 credits (as I call them, ie 100 credits to your name), and ...
* a space ship, capable of interplanetary travel.

Now, what you have to do may sound pretty basic. But upon your shoulders lies the fate of the universe. For example, if you can't (or don't) supply some goods to one planet (these goods being life supporting), this particular planet may perish, and before you know it, you can't trade with that planet anymore.

If that happens to a few more planets, then there's the threat of your own death. How? You need to buy fuel, energy for shields and weapons (yes, there is some shooting involved) from planets as well.

If a particular planet sells what YOU need dies, then you only have the energy you have left (so far I've only found one planet that sells what I need for my ship).

So it's a catch-22 situation: if you don't supply some planets with what they need, then you may not get what you need and consequentially both die.

And you thought it was going to be easy, eh?

Of course there are other ways to make money - you can steal it - how else?

In the game, there are also:

* cargo ships who, if you happen to plunder them (by shooting them) only get you a few hundred credits;
* pirate ships. These are r-e-a-l-l-y friendly ships (sarcasm plus here). Unlike every other ship in the universe (besides yourself), they are the only thing that will attack and rob you of your wealth!

And you thought you were the only one who could do that ... !

By the way, if you can shoot the pirates, there's an additional reward of a few hundred credits here.

* other Zone Runners. You are known as a "Zone Runner" - pretty classy title, eh? These other Zone Runners have the same purpose in life and that is to make money. Attacking these will result in an extra 1,000 to 14,000 credits, depending how much they have made.
* and last, but not least, patrol ships. The idea of these is that they're "supposed" to keep law and order in the universe ... the only thing I ever get from the patrol ships is trouble! (Like the Queensland police!)

Ship controls

The whole game (which is a great idea) can be run from a single joystick or mouse.

All your controls and status reports, etc are all on the one screen. So if I wanted to increase my speed, I'd position my cursor over the "Energy" panel, select a level and press a button ...

Alternatively I can keep the button pressed and move the controls up or down by moving the joystick/mouse up or down. Nifty!

Choosing a direction? Simple! Position the joystick/mouse over the arrow and pick a direction. The "arrow" is actually like a compass; it is made of a needle that rotates inside a circle, and whatever way the arrow is pointing, the ship will go too!

The main screen is actually split into three different modes:

* the galaxy mode: an overall view of known universe and general position of where you are. Good to see where you are only, and when crossing what's known as the "Neutral Zone".
* Magnification One: this lets you see where you are in comparison to other planets in the area. Ideal when you're blasting pirate ships, cargo ships and other riff-raff (like patrol ships!).
* Magnification Two: highest magnification possible. Lets you see the immediate surrounding area. Ideal mode when trading with other planets.

A nice feature I found is that you can determine the foreground and background colours you want; so I can choose any one of sixty-four colours for the background and foreground.

Disadvantages

... hmmm, this is a tough one. For the tpye who like quick games, this is not for you. When I reviewed Zone, I played for about 2 hours - and I was nowhere near finishing!

Overall comments

Zone is great! It's about:

* learning a little about buying and selling (economics),
* strategy,
* quick-thinking,
* sharp shooting,
* outwitting the "other" ships
* having fun

I'd give Zone, out of 10 a 9! Well worth it!

Specifications

Title: Zone Runner
Cat no: 26-3286
Source: Every Tandy store
Price: $59.95, approx.

System requirements

... or what you need to run this sotware:

* CoCo 3
* Disk Drive
* Joystick/Mouse
* (optional) OS-9 Level 2

****

CAVE WALKER IS A game ... it requires no great thinking (like "Zone Runner") or quick reflexes (like "Polaris") or imagination (like "Dungeons of Daggorath").

Instead, it fits into none of the above categories. It is in a class of its own.

What's it all about?

In Cave Walker, you are an adventurer with the aim of:

* finding treasures,
* avoiding obstacles to ...
* get to the right exit, and
* get to the next level.

... in a maze-like cave.

Now the scenerio runs as follows:

"The great cave of the Wizards was the place where all great fortunes were kept, as well as many dangers, like 'steam jets', 'fire pits', 'cannons' and 'vanishing earths'."

Wait for it - there's more!

"... there is even the legend of the 'Great White Bat', created by the Wizards to defend and protect their possessions."

Now, for an introduction like that, I'd put the software back on the shelf right away, mainly because the introduction would tell me, "A-ha, they're using excessive descriptions of the game to sell the game instead of letting the game sell itself."

Like "Danger Ranger".

In "Danger Ranger", the introduction was too 'oooh - aaaahed', ie in "Cave Walker's" case, the 'steam jets', 'fire pits', 'Great White Bat', and so on.

With introductions like that, to me it usually means that the game can (and probably will) be a bore.

And that's what this is.

Okay, but what exactly do you do in "Cave Walker"?

Alright, enough criticism.

There are over 25 caves, each with their own objects, dangers and so on, with the ultimate goal of getting to the Wizards cave and be overly rich.

To gain access to this room, you must open the treasure chest with a key, and to get the key, you need to find the spell books. Once all three parts of the key have been found, the treasure chest will open, revealing the treasures of the Wizards.

The end!

Advantages

Cave Walker, even though it's really only meant for a CoCo 2 can be adapted for a CoCo 3. By pressing a sequence of keys, you can play the game in colour instead of black and white.

The booklet gives a thorough description on how to back up your disk, "What happens if...?" and so on.

Overall rating

I didn't really think much of Cave Walker. Sure, it has it's good points, but the idea of "find the treasure, watch out for the baddies and get extremely rich" is similar to flogging a dead horse!

If they "flogged the horse" in a more extravagant setting ie, same concept, more challenging, though provoking, it'd be great!

The only real person I would give it to is a small child of about 6-9 years - then again, maybe that's who Tandy are trying to sell it to, in relation to the way the introduction is written.

I'd give it a 5 out of 10.

Specifications

Title: Cave Walker
Cat. No: 26-3246
Source: Any Tandy store
Price: $59.95, approx

System requirements

* CoCo 2 (CoCo 3 for colour)
* Disk Drive
* One Joystick

continued from p 45

```
815 '**OUTPUT TO PRINTER**
816 GOSUB 890:PRINT@451,"PRESS <
X> TO STOP PRINTING";
817 PE=PEEK(65314) AND 1:IF PE=0
 THEN 819
818 PRINT@193,"**please"+S$+"tur
n"+S$+"printer"+S$+"on**":GOTO 8
17
819 PRINT@193,"PRINTING "+IN$
820 PRINT#-2,CHR$(14)+IN$
830 PRINT#-2:PRINT#-2
840 FOR I=1 TO L
845 IF INKEY$="X" THEN 870
850 PRINT#-2,A$(I)
860 NEXT I
870 GOTO 510
875 '**EXIT TO MAIN MENU**
880 GOSUB 890:GOTO 100
882 GOSUB 981:GOSUB 989:GOTO 530
885 '**CLEAR SCREEN ROUTINE**
890 FOR J=64 TO 448 STEP 32
900 PRINT@J,""
910 NEXT J
920 RETURN
925 '**SCAN KEYS/FLASH CURSOR**
930 X=0
940 I$=INKEY$
950 IF I$="" THEN 960 ELSE CH=VA
L(I$):RETURN
960 IF X=5 THEN PRINT@LC,CHR$(14
1); ELSE IF X=10 THEN PRINT@LC,C
HR$(142);
970 IF X>10 THEN 930 ELSE X=X+1:
GOTO 940
975 '**WARNING!! - COLD START**
980 POKE113,0:EXEC40999
981 GOSUB 890:PRINT@257,"ENTER D
RIVE NO. "
982 LC=274:GOSUB 930
983 IF CH<0 OR CH>4 THEN SOUND 1
00,4:GOTO 982
984 DN=CH
985 PRINT@273,DN
986 DIR DN:PRINT:PRINT@488,"<PRE
SS ANY KEY>";
987 LC=503:GOSUB 930
988 CLS:RETURN
989 PRINT@32,"ascii"+S$+"data"+S
$+"file"+S$+"transfer"+S$+"utili
ty":RETURN
990 PCLEAR 1:CLEAR 23500:GOTO 40
```

HINT.....

POKEs

New POKES for 6ms second step rate and double sided drive access for the CoCo 3.

```
POKE &HD7C0,0     6ms
POKE &HD016,&H14 6ms
POKE &HD09F,&H41 Double Sided
POKE &HD0A0,&H42 Double Sided
POKE &HD7C0,&H2  Tandy drive
                 20ms Step rate
```

COPYROM

By David Thurbon
UTILITY
32K DECB

I AM SORRY ABOUT the "Copy ROM" program (April Aust. CoCo, p38) which doesn't work. It seems that I sent a bad copy of it to the CoCo magazine.

To rectify the problem I have sent another copy of the program which will now work.

The reason it didn't work originally was that I hadn't turned off the 60 Hz internal interrupts of the 6809.

Since the copyrom program takes longer than a sixtieth of a second to complete its job, the computer would interrupt the program.

This is basically what happens. If you didn't understand all that, don't worry.

This new program is guaranteed to work. Just type in the basic listing, save it, and RUN it.

Then you can save a machine language version by typing:

```
(C)SAVEM"COPYROM",&H4000,&H4019
,&H4000
```

The program is written in position independent code so that it may be loaded anywhere into memory.

I would like to thank Frank Rees for mentioning the problem.

(Ed's note: You can find all three programs, ie the basic version, the assembly version and the binary version, on this months CoCoOz on tape/disk.)

The Listing:

```
0 GOTO10
1 '***** "COPYROM"
2 '***** D.W. THURBON
3 SAVE"43:3":END'7
10 PCLEAR4
20 FOR P=&H4000 TO &H4019
30 READA$:POKEP,VAL("&H"+A$)
40 NEXT
50 EXEC&H4000
60 DATA7F,FF,40,1A,50,8E,80,,7F,
FF,DE,EC,84,7F,FF,DF,ED,81,8C,FE
,FE,25,F1,1C,AF,39
```

The Listing:

```
0010 * REVISED VERSION OF COPYROM PROGRAM
0020 * D. THURBON 11.8.87
0040        ORG $4000
0050 START CLR $FF40    TURN OFF DRIVES
0060  ORCC #$50         TURN OFF FIRQ AND IRQ
0070  LDX #$8000        POINT X TO START OF ROM
0080 LOOP CLR $FFDE     GO TO ROM MODE (DEFAULT)
0090  LDD ,X            PICK UP VALUE FROM ROM
0100  CLR $FFDF         GO TO ALL RAM MODE
0110  STD ,X++          DROP OFF VALUE IN RAM
0120  CMPX #$FEFE       END OF ROMPRINTPRINT
0130  BLO LOOP          NO, KEEP GOING
0140  ANDCC #$AF        TURN FIRQ AND IRQ BACK ON
0150  RTS               RETURN TO BASIC
0160  END
```

BOUNCE

Modified by Colin North
UTILITY
CoCo3

BOUNCING BALL IS A utility you can put in your program, if you so require. Right now, the condition of the program is so to give an idea of what you can do with it.

On the screen is a ball. It will bounce around the screen, while you can ...

* enlarge it in size,
* reduce it in size,
* increase its' step rate,
* reduce its' step rate,
* change its' colour.

The Listing:

```
0 GOTO10
3 SAVE"20A:3":END'7
10 ' *** BOUNCE ***
20 ' BY JAMES CLARK
30 '***************
40 ' MODIFIED FOR THE COCO3
50 ' BY COLIN NORTH
60 '***************
70 POKE65497,0
80 POKE&HFFBD,59
90 D=0
100 CLS
110 DIM E(5,5)
120 B=1:C=1
130 ON ERR GOTO370
140 PRINT" USE F1 & F2 TO CHANGE
 COLOUR":PRINT
150 PRINT" <ALT> ENLARGE CURSOR"
:PRINT
160 PRINT" <CTRL> SHRINK CURSOR"
:PRINT
170 PRINT" <CLEAR> CLEARS SCREEN
":PRINT
171 PRINT" <Q> INCREASE STEPPING
 RATE":PRINT
172 PRINT" <A> DECREASE STEPPING
 RATE":PRINT
180 INPUT"(X,Y) CO-ORDINATES";X,
Y
190 Z=2:A=2:F=2:CO=0
240 POKE&HFFBA,CO
250 PMODE4,1:SCREEN1,1:PCLS1:COL
OR0
260 GET(10,10)-(15,15),E,G:PCLS1
270 IF D=1 THEN A=RND(5)
280 IF X<9 THEN B=1
290 IF X>250 THEN B=2
300 IF Y<9 THEN C=1
310 IF Y>187 THEN C=2
320 IF B=1 THEN X=X+A
```

continued overleaf

PARTY TRICK

By Paul Stevenson
APPLICATION
16K CB

PARTY TRICK is a program suited for just that - parties! It seems to amaze those who don't have a computer and are just looking on ...

```
2 GOTO10
3 SAVE"50A:3":END'6
10 '****AGE TELLER****
20 '*BY PAUL STEVENSON
30 CLS
40 PRINT@35,"==================
========"
50 PRINT@77,"AGETELL"
60 PRINT@135,"BY PAUL STEVENSON"
70 PRINT@163,"==========(C) 1987
========"
80 PRINT@192," THIS PROGRAM WILL
 TELL YOU THE   AGE OF ANYONE SI
XTY OR UNDER.  PROVIDING THEY TE
LL THE TRUTH   ABOUT THE NUMBERS
."
90 PRINT@388,"PRESS ANY KEY TO S
TART"
100 P$=INKEY$:IFP$=""THEN100
110 CLS
120 PRINT"      2   3   6   7
10              11  14  15  18
19              22  23  26  27
30              31  34  35  38
39              42  43  46  47
50              51  54  55  58
59"
130 INPUT"DOES YOUR AGE APPEAR I
N THESE    NUMBERS";A$
140 IF A$="N"THENX=0
150 IF A$="Y"THENX=2
160 CLS
170 PRINT"      8   9   10  11
12  13          14  15  24  25
26  27          28  29  30  31
40  41          42  43  44  45
46  47          56  57  58  59
60  13"
180 INPUT"HOW ABOUT THESE NUMBER
S";B$
190 IFB$="N"THENX=X
200 IFB$="Y"THENX=X+8
210 CLS
220 PRINT"      16  17  18  19
20              21  22  23  24
25              26  27  28  30
31              48  49  50  51
52              53  54  55  56
57              58  59  60  19
26"
230 INPUT"AND THESE ONES";C$
240 IFC$="N"THENX=X
250 IFC$="Y"THENX=X+16
260 CLS
270 PRINT"      32  33  34  35
36  37          38  39  40  41
42  43          44  45  46  47
48  49          50  51  52  53
54  55          56  57  58  59
60  33"
280 INPUT"AND DOES YOUR AGE APPE
AR HERE";D$
290 IFD$="N"THENX=X
300 IFD$="Y"THENX=X+32
310 CLS
320 PRINT"      4   5   6   7
13              12  14  15  20
21              22  23  28  29
30              31  36  37  38
39              44  45  46  47
52              53  54  55  60
29"
330 INPUT"HOW ABOUT THESE";E$
340 IFE$="N"THENX=X
350 IFE$="Y"THENX=X+4
360 CLS
370 PRINT"      1   3   5   7
 9   11         13  15  17  19
 21  23         25  27  29  31
 33  35         37  39  41  43
 45  47         49  51  53  55
 57  59"
380 INPUT"LAST TIME! ARE THEY HE
RE";F$
390 IFF$="N"THENX=X
400 IFF$="Y"THENX=X+1
410 CLS
420 FORZ=1TO500:NEXTZ
430 PRINT@199,"=================
=="
440 PRINT@231,"= YOUR AGE IS "X"
 ="
450 PRINT@263,"=================
=="
```

continued from p 47

```
1320 PRINT"READY CASSETTE":EXEC4
4539
1330 CSAVEMFF$,1536,(PG+1)*1536-
1,0:GOTO750
1340 SX=(32-LEN(W$))/2:PRINT@SX+
(32*(SY-1)),W$;:SY=SY+1:RETURN
```

DATAGAN & EDTASM

By Frank Rees
UTILITY
32K ECB

I ENJOY HAVING TO use EDTASM less with each usage. Currently I am doing a teleprinter program for the MC10.

A Basic loader which makes an ML tape for where you want to locate the 'Printer2' program in memory. I decided to use the CoCo to do the job using EDTASM+.

I was trying to make up a tape which 'Datagen' would convert into data statements Basic tape which, then in turn, would be coverted to MC10. As Datagen is at 6A00-7F32, I made the tape to be done at 5000-5112.

All efforts to load this tape for EDTASM+ to Datagen resulted in a crash. Suspecting the problem I mentioned in my last commentery on EDTASM+, I used CLOADM to load the tape, then used ...

```
CSAVEM"PR2-DATA",&H5000,
 &H5112,&H5000
```

... to remake the tape. You guessed it. Loaded okay!

from previous page

```
330 IF B=2 THEN X=X-A
340 IF C=1 THEN Y=Y+A
350 IF C=2 THEN Y=Y-A
360 IF F=2 THEN PUT(X-Z,Y-Z)-(X+
Z,Y+Z),E,NOT
370 IF F=1 THEN PSET(X,Y,0)
380 IF PEEK(343)=191 THEN CO=CO+
1:IF CO=64 THEN CO=0
390 POKE&HFFBA,CO
400 IF PEEK(344)=191 THEN CO=CO-
1:IF CO=0 THEN CO=63
410 IF PEEK(341)=191 THEN Z=Z+1
420 IF Z>40 THEN Z=40
430 IF PEEK(342)=191 THEN Z=Z-1
440 IF Z<1 THEN Z=1
450 IF PEEK(339)=191 THEN PCLS1
451 IF PEEK(339)=251 THEN A=A+1
452 IF PEEK(339)=254 THEN A=A-1
453 IF A<1 THEN A=1
455 IF PEEK(342)=253 THEN 500
460 GOTO 270
500 POKE65281,0:GOTO270
```

COCO 3 COMMENTS

By Brian Bere–Streeter
TUTORIAL
CoCo3

THERE IS NO NEED to repeat the new features of the CoCo 3, as these are, by now, well known.

However some of the new commands and functions have a great deal of power and demand exploration. Amongst the new or revised commands are PALETTE, ATTR, CLS and WIDTH.

PALETTE.

Unlike the Coco1 or 2 where the colour range was restricted to 8 plus black, the CoCo 3 has a total of 64 colours available.

At any one time a maximum of 16 colours are immediately available, and on power-up and on using the new command PALETTE RGB (or just RGB), the standard set of 16 colours (8 foreground and 8 background) are loaded into 16 'slots' ready for use.

The 8 background colours are the standard green, yellow, blue, red, buff, cyan, magenta and orange.

The foreground colours only use black, green, buff and orange, This way maintaining compatability with the old PMODEs and 32 x 16 text screens.

Now the good news: you are not restricted to these 8 colours. Using the new PALETTE command (PALETTE slot no.,colour no.) you can load any of the 64 colours into any of the 16 slots in any combination. Not only can you run a program using your custom set of 16 colours, but during the program run you can, at any time, by calling PALETTE substitute a colour in a particular slot for another colour, and in fact use all 64 colours in the one program, but display only 16 at any one time.

Also calling PALETTE in a new program line at the start of old programs, will enable a colour change to most existing CoCo 1 & 2 programs in both text and graphic modes.

The tables in the back of the manual will show which slots for which colours in the old text and Pmodes, and the new coloured text looks nice as cyan or yellow on black, and PMODE 4 as blue on cyan or brown on yellow.

ATTR

This is a new command which lets you put a range of coloured texts and coloured backgrounds on the screen, and uses the standard 8 foreground and 8 background colours, to create in essence, 64 different text colour combinations.

For obvious reasons some combinations are either blank (eg, blue on blue) or virtually unreadable (eg, orange on red).

ATTR also provides underlining of text and blinking of text. Again you are not restricted to the 8 background and 8(4) foreground colours. Using PALETTE you can set up any of the 64 colours for foreground or background, and the program 'COLRTEXT' demonstrates all combinations of the 64 foreground colours on the 8 standard background colours, all combinations of the 8(4) foreground colours on 64 background colours and a combination of 8 custom foregrounds on the 64 backgrounds.

The 8 custom foregrounds require pre-loading of 8 new colours into the relevant foreground slots (examining the program will show you how) before invoking the ATTR command.

CLS

This is a modified version of the old CLS command, which on the CoCo 1 or 2 clears the screen to the selected colour, but retains the black border around the screen. The modified form of this command in the CoCo 3 clears the entire screen to the selected colour, in the 40 & 80 column modes.

You can give your text intensive programs a colourful lift by calling a CLS colour then an ATTR combination to have, say, blue text on a cyan screen with a red border.

The program 'COLRTEXT' also demonstrates all combinations of the standard 8 colours used in CLS with the standard 8 background colours of the default ATTR 0,x (black text).

Similarly you can modify your programs to display any of the 64 colours in foreground, background or border by using PALETTE to load a new colour into the relevant slot.

WIDTH

This is also a new command and will set either 40 column or 80 column text with full upper and lower case. Note that the 80 column text screen may not be usable on some TV screens.

Samples of both screen formats are shown in the program 'COLRTEXT'.

General

The program 'COLRTEXT' is more than just a demonstration program, but can be used as a reference when you want to find a new colour combination to put into a program, as the relevant CLS, ATTR and PALETTE numbers are shown on screen with the colours selected.

Take note that when using a colour TV set some colours 'bleed' into each other and therefore some combinations are unusable.

When using the RGB monitor, all colours are crisp & sharp and the improvement over a standard TV is certainly worthwhile. In fact, before I bought my CoCo 3, I even went through the motions of evaluating other alternative computer systems for my CoCo 2 replacement (perish the thought !!), and in doing so evaluated other RGB monitors.

Whilst the price of $700 for the Tandy CM- 8 seems high on

the surface, a check of other magazine adds shows that RGB monitors range from about $600 to $1200 for digital style and about $750 to $1400 for analogue style (the CM-8 is analogue), so Tandy's price seems competative.

I certainly feel the price is justified to get the best out of the CoCo 3. If using a monochrome TV, try using:

```
PALETTE 0,0:PALETTE 8,63:CLS1
```

... for your 40 or 80 text screen, and adjust the contrast control for clarity. Alternatively, using standard power-up colours, try using CLS5 to clear to a uniform shade, before using text.

Finally when you use any of the new commands, and want to run a new program without powering down, remember to reset the default colours with PALETTE RGB before running the new program, or your previous colour selection may do unexpected things to the new program.

Enjoy your CoCo 3, and if you discover other new things about it that are undocumented, please send them in to Australian CoCo, for all to share.

The Listing:

```
0 GOTO10
1 '***** "COLRTEXT"
2 '***** BRIAN BERE-STREETER
3 SAVE"17A:3":END'7
10 WIDTH80:ON BRK GOTO 5000
20 PALETTE0,0:PALETTE8,18:CLS1
30 LOCATE0,0:ATTR7,0:PRINT STRIN
G$(80,127)
40 FOR X=1TO21:LOCATE0,X:PRINT C
HR$(124):LOCATE 79,X:PRINT CHR$(
124):NEXT
50 LOCATE0,22:PRINT CHR$(124)+ST
RING$(78,127)+CHR$(124);
60 LOCATE20,2:ATTR1,0,U:PRINT"Mu
lti-colour Text Screen Demonstra
tions";:ATTR3,0
70 LOCATE10,4:PRINT"1. Shows def
ault black text on 8 default bac
kground colours";
75 LOCATE10,5:PRINT"   with 8 st
andard colours for borders.";
80 LOCATE10,8:PRINT"2. Shows tex
t in 64 foreground colours on 8
default";
85 LOCATE10,9:PRINT"   backgroun
d colours.";
90 LOCATE10,12:PRINT"3. Shows te
xt in 8 default foreground colou
rs on 64";
95 LOCATE10,13:PRINT"   backgrou
nd colours.";
100 LOCATE10,16:PRINT"4. Shows t
ext in 8 custom foreground colou
rs on 64";
105 LOCATE10,17:PRINT"   backgro
und colours";
150 LOCATE28,20:ATTR1,0,B:PRINT"
Make a Selection > 1 to 4 ";:ATT
R1,0
160 A$=INKEY$:IF A$=""THEN160
170 IF A$="1"OR A$="2"OR A$="3"O
R A$="4" THEN180ELSE160
180 A=VAL(A$)
190 ON A GOTO 1000,2000,3000,400
0
1000 ON BRK GOTO1190
1010 WIDTH40:PALETTE RGB
1020 FOR X=1TO8:CLS X
1030 FOR Y=0TO7:ATTR0,Y
1040 PRINT:PRINT:PRINT:PRINT:PRI
NT:PRINT:PRINT:PRINT:PRINT
1050 PRINT:PRINT:PRINT:PRINT:PRI
NT:PRINT:PRINT:PRINT:PRINT:PRINT
:PRINT:PRINT:PRINT:PRINT
1060 PRINT"
                ";
1070 LOCATE0,1:PRINT"    ";:ATTR
0,Y,U:PRINT"COLOUR TEXT SCREENS
USING CLS/ATTR";:ATTR0,Y
1080 LOCATE0,7:PRINT"    Now is
the time for all good men to  c
ome to the aid of the party."
1090 LOCATE0,10:PRINT"    The qu
ick brown fox jumps over the
lazy dog."'COPYRIGHT 1986 BERE-S
TREETER.
1100 LOCATE5,15:PRINT"USE CLS";X
;"THEN ATTR 0,";Y
1110 LOCATE5,18:ATTR0,Y,B:PRINT"
PRESS SPACE FOR NEXT TEXT SCREEN
":ATTR0,Y
1120 LOCATE5,20:PRINT"PRESS BREA
K TO EXIT
1130 A$=INKEY$:IF A$=""THEN1130
1140 IF ASC(A$)=3THEN1150
1150 CLSX
1160 NEXT Y
1170 NEXT X
1180 GOTO1020
1190 PALETTE RGB:WIDTH 40:ATTR2,
0:CLS1:GOTO10
2000 ON BRK GOTO2230
2010 WIDTH40:PALETTE RGB:CLS1
2020 PRINT"    ";:ATTR2,0,U:PRI
NT"TEXT COLOURS USING PALETTE/AT
TR";:ATTR2,0:PRINT
2030 FOR X=0TO63:PALETTE 8,X
2040 LOCATE0,3
2050 ATTR0,0:PRINT"          ";:
ATTR0,1:PRINT"          ";:ATTR0
,2:PRINT"          ";:ATTR0,3:PR
INT"          ";
2060 ATTR0,0:PRINT"   ATTR   ";:
ATTR0,1:PRINT"   ATTR   ";:ATTR0
0,2:PRINT"   ATTR   ";:ATTR0,3:P
RINT"   ATTR   ";
2070 ATTR0,0:PRINT"          ";:
ATTR0,1:PRINT"          ";:ATTR0
,2:PRINT"          ";:ATTR0,3:PR
INT"          ";:ATTR2,0:PRINT
2080 PRINT"   0,0       0,1
  0,2       0,3   ":PRINT
2090 ATTR0,4:PRINT"          ";:
ATTR0,5:PRINT"          ";:ATTR0
,6:PRINT"          ";:ATTR0,7:PR
INT"          ";'COPYRIGHT 1986
BERE-STREETER.
2100 ATTR0,4:PRINT"   ATTR   ";:
ATTR0,5:PRINT"   ATTR   ";:ATTR0
,6:PRINT"   ATTR   ";:ATTR0,7:PR
INT"   ATTR   ";
2110 ATTR0,4:PRINT"          ";:
ATTR0,5:PRINT"          ";:ATTR0
,6:PRINT"          ";:ATTR0,7:PR
INT"          ";:ATTR2,0
2120 PRINT
2130 PRINT"   0,4       0,5
  0,6       0,7":PRINT:PRINT
2140 PRINT"    ";:PRINT"USE PAL
ETTE 8,";X;"THEN ATTR 0, X"
2150 PRINT:PRINT:PRINT"    ";:A
TTR2,0,B:PRINT"PRESS SPACE FOR N
EXT TEXT COLOUR":ATTR2,0
2160 PRINT:PRINT"    PRESS BREA
K TO EXIT"
2170 IF X=0OR X=9OR X=18OR X=27O
R X=36OR X=38OR X=45OR X=54OR X=
63 THEN GOTO2180ELSE2190
2180 LOCATE5,17:PRINT"(STANDARD
ATTRIBUTE COLOUR SET)
2190 A$=INKEY$:IF A$=""THEN2190
2200 IF ASC(A$)=3THEN2210
2210 NEXT
2220 GOTO2010
2230 PALETTE CMP:ATTR2,0:WIDTH40
:GOTO10
3000 ON BRK GOTO3230
3010 WIDTH40:PALETTE RGB:CLS1
3020 PRINT"    ";:ATTR2,0,U:PRI
NT"TEXT COLOURS USING PALETTE/AT
TR";:ATTR2,0:PRINT
3030 FOR X=0TO63:PALETTE 1,X
3040 LOCATE0,3
3050 ATTR0,1:PRINT"          ";:
ATTR1,1:PRINT"          ";:ATTR2
,1:PRINT"          ";:ATTR3,1:PR
INT"          ";
3060 ATTR0,1:PRINT"   ATTR   ";:
ATTR1,1:PRINT"   ATTR   ";:ATTR2
,1:PRINT"   ATTR   ";:ATTR3,1:PR
INT"   ATTR   ";
3070 ATTR0,1:PRINT"          ";:
ATTR1,1:PRINT"          ";:ATTR2
,1:PRINT"          ";:ATTR3,1:PR
INT"          ";:ATTR2,0:PRINT
3080 PRINT"   0,1       1,1
  2,1       3,1   ":PRINT
3090 ATTR4,1:PRINT"          ";:
ATTR5,1:PRINT"          ";:ATTR6
,1:PRINT"          ";:ATTR7,1:PR
INT"          ";'COPYRIGHT 1986
BERE-STREETER.
3100 ATTR4,1:PRINT"   ATTR   ";:
ATTR5,1:PRINT"   ATTR   ";:ATTR6
,1:PRINT"   ATTR   ";:ATTR7,1:PR
INT"   ATTR   ";
3110 ATTR4,1:PRINT"          ";:
ATTR5,1:PRINT"          ";:ATTR6
,1:PRINT"          ";:ATTR7,1:PR
INT"          ";:ATTR2,0
3120 PRINT
3130 PRINT"   4,1       5,1
  6,1       7,1":PRINT:PRINT
3140 PRINT"    ";:PRINT"USE PAL
ETTE 1,";X;"THEN ATTR X, 1"
3150 PRINT:PRINT:PRINT"    ";:A
TTR2,0,B:PRINT"PRESS SPACE FOR N
EXT TEXT COLOUR":ATTR2,0
```

```
3160 PRINT:PRINT"      PRESS BREA
K TO EXIT"
3170 IF X=0OR X=9OR X=18OR X=27O
R X=36OR X=38OR X=45OR X=54OR X=
63 THEN GOTO3180ELSE3190
3180 LOCATE5,17:PRINT"(STANDARD
ATTRIBUTE COLOUR SET)
3190 A$=INKEY$:IF A$=""THEN3190
3200 IF ASC(A$)=3THEN3210
3210 NEXT
3220 GOTO3010
3230 PALETTE CMP:ATTR2,0:WIDTH40
:GOTO10
4000 ON BRK GOTO4240
4010 WIDTH40:PALETTE RGB:CLS1
4020 PALETTE8,45:PALETTE9,38:PAL
ETTE10,0:PALETTE11,27:PALETTE12,
36:PALETTE13,54:PALETTE14,18:PAL
ETTE15,9
4030 PRINT"      ";:ATTR2,0,U:PRI
NT"TEXT COLOURS USING PALETTE/AT
TR";:ATTR2,0:PRINT
4040 FOR X=0TO63:PALETTE 1,X
4050 LOCATE0,3
4060 ATTR0,1:PRINT"          ";:
ATTR1,1:PRINT"          ";:ATTR2
,1:PRINT"          ";:ATTR3,1:PR
INT"          ";
4070 ATTR0,1:PRINT"   ATTR   ";:
ATTR1,1:PRINT"   ATTR   ";:ATTR2
,1:PRINT"   ATTR   ";:ATTR3,1:PR
INT"   ATTR   ";
4080 ATTR0,1:PRINT"          ";:
ATTR1,1:PRINT"          ";:ATTR2
,1:PRINT"          ";:ATTR3,1:PR
INT"          ";:ATTR2,0:PRINT
4090 PRINT"   0,1       1,1
  2,1       3,1   ":PRINT
4100 ATTR4,1:PRINT"          ";:
ATTR5,1:PRINT"          ";:ATTR6
,1:PRINT"          ";:ATTR7,1:PR
INT"          ";'COPYRIGHT 1986
BERE-STREETER.
4110 ATTR4,1:PRINT"   ATTR   ";:
ATTR5,1:PRINT"   ATTR   ";:ATTR6
,1:PRINT"   ATTR   ";:ATTR7,1:PR
INT"   ATTR   ";
4120 ATTR4,1:PRINT"          ";:
ATTR5,1:PRINT"          ";:ATTR6
,1:PRINT"          ";:ATTR7,1:PR
INT"          ";:ATTR2,0
4130 PRINT
4140 PRINT"   4,1       5,1
  6,1       7,1":PRINT:PRINT
4150 PRINT"      ";:PRINT"USE PAL
ETTE 1,";X;"THEN ATTR X, 1"
4160 PRINT:PRINT:PRINT"      ";:A
TTR2,0,B:PRINT"PRESS SPACE FOR N
EXT TEXT COLOUR":ATTR2,0
4170 PRINT:PRINT"      PRESS BREA
K TO EXIT"
4180 IF X=0OR X=9OR X=18OR X=27O
R X=36OR X=38OR X=45OR X=54OR X=
63 THEN GOTO4190ELSE4200
4190 LOCATE5,17:PRINT"(STANDARD
ATTRIBUTE COLOUR SET)
4200 A$=INKEY$:IF A$=""THEN4200
4210 IF ASC(A$)=3THEN4220
4220 NEXT
4230 GOTO4010
4240 PALETTE RGB:PALETTE15,38:AT
TR2,0:WIDTH40:GOTO10
5000 END
```

MENU MAKER

By Brian Bere–Streeter
UTILITY
CoCo3

MENUMAKER IS A simple program that takes advantage of the CoCo 3's 80 column text screen to set-up a menu selection screen for up to 8 subroutines or other programs.

The colour selections for text can be changed with the Palette and Attr commands, if you don't like my choice.

Line 60 should be edited to change the heading text, and lines 70-140 to change the selections text, but check the centreing of the heading text for best effect.

Change lines 1000-8000 to contain the starting addresses of your subroutines or programs. You can use these to either, load and run other programs, which can also be modified to re-load and run Menumaker for further selections, or to call-up programs set-up as subroutines within the main program, which will return to the menu selection screen.

A sample of the later use is shown in my CoCo 3 colour text demonstration program 'COLRTEXT' in which Menumaker has been changed to only 4 selections, but note in the 80 column mode, much longer descriptive selection headings can be used.

The Listing:

```
0 GOTO10
1 '***** "MENUMAKR"
2 '***** BRIAN BERE-STREETER
3 SAVE"17:3":END'7
10 WIDTH80
20 PALETTE0,0:PALETTE8,18:CLS1
30 LOCATE0,0:ATTR7,0:PRINT STRIN
G$(80,127)
40 FOR X=1TO21:LOCATE0,X:PRINT C
HR$(124):LOCATE 79,X:PRINT CHR$(
124):NEXT
50 LOCATE0,22:PRINT CHR$(124)+ST
RING$(78,127)+CHR$(124);
60 LOCATE28,2:ATTR1,0,U:PRINT"Me
nu Selection System";:ATTR3,0
70 LOCATE10,4:PRINT"1. Program N
umber 1";
80 LOCATE10,6:PRINT"2. Program N
umber 2";
90 LOCATE10,8:PRINT"3. Program N
umber 3";
100 LOCATE10,10:PRINT"4. Program
 Number 4";
110 LOCATE10,12:PRINT"5. Program
 Number 5";
120 LOCATE10,14:PRINT"6. Program
 Number 6";
130 LOCATE10,16:PRINT"7. Program
 Number 7";
140 LOCATE10,18:PRINT"8. Program
 Number 8";
150 LOCATE28,20:ATTR1,0,B:PRINT"
Make a Selection > 1 to 8 ";:ATT
R1,0
160 A$=INKEY$:IF A$=""THEN160
170 IF A$="1"OR A$="2"OR A$="3"O
R A$="4"OR A$="5"OR A$="6"OR A$=
"7"OR A$="8" THEN180ELSE160
180 A=VAL(A$)
190 ON A GOTO 1000,2000,3000,400
0,5000,6000,7000,8000
1000 CLS1:PRINT"1":END
2000 CLS1:PRINT"2":END
3000 CLS1:PRINT"3":END
4000 CLS1:PRINT"4":END
5000 CLS1:PRINT"5":END
6000 CLS1:PRINT"6":END
7000 CLS1:PRINT"7":END
8000 CLS1:PRINT"8":END
```

Hint ...

CoCo at work

For something really trivial, type in ...

```
POKE359,57:POKE65480,0
```

This will show the system variables used by the computer. Try typing in a few commands, like CLS, TIMER=0, etc. Make a typing mistake.

Although you won't see what you're typing, the computer will still understand what you've typed.

To get out of this mode, type...

```
POKE359,126
```

TANDY ROCKETS TO THE TOP

By Laurie O'Shea

ARTICLE

Have you had that embarrassing feeling as a Tandy - phile (that is someone who really likes Tandy computers) that other people (eg Commodore - philes) seem to be grabbing the limelight?

Haven't you had that patronising, smug wink from people owning the latest "gizmo", who feel that Tandy missed the boat and that maybe you ought to switch to a SEGA, AMSTRAD or COMMODORE?

Well all you Tandy fans straighten your shoulders, put you head up high and smile benevolently at all those other "brand" computer owners. Tandy didn't miss the boat, they're back on top with a vengeance.

1986 and 1987 have been Tandy's most successful years ever, where in the USA they have taken on Apple, IBM and Compaq at their own game and beaten them hands down.

And that has been mainly on the success of the Tandy 1000 series, but there is much more to come. Tandy have released a stunning new range of computers in the USA which has really taken the fight for market supremacy to the mighty IBM castle as has never been done before.

Tandy was a very early successful player in the personal computer field, and on August 3, 1977 released their first micro-computer, the TRS-80 model 1.

Actually history can be a strange story. Charles Tandy, the man behind the move to get Tandy in the microcomputer field approached Jack Tramill the founder of the Commodore to make a machine for Tandy.

Commodore hesitated and Tandy decided to market their own model manufactured in Tandy plants. It was a tremendous gamble because no-one had succeeded in mass marketing such a product, which had not been possible to manufacture anyway till 1975.

Tandy decided to manufacture 5,000 units on their first production run. They got orders for 30,000 additional Model 1's in the first month alone. Thus really the era of the microcomputer for the man on the street was born. Prior to the Model 1 most people had to build their own computer from bits.

In a strange twist of fate, 9 years later Tandy was unable to make a computer as fast as it could be sold. The Tandy 1000 when launched sold faster than Tandy factories could turn them out and again they were 30,000 units behind.

Between 1977 and the release of the IBM PC in 1980, Tandy and Apple fought it out for top place. Then by 1981 IBM had taken about 80% of the market.

Tandy and Apple stuck to their proprietary product and DOS set-ups, and both had remarkable plunges in sales. Both companies were counted out, by a cynical market that had climbed on the IBM-PC train. I guess the cry was..."All the way with IBM".

Apple struck back with the MacIntosh and Lisa models and although non-IBM compatible in a strange gamble paid off with record sales and good profits. Of course as we know Apple got many of their ideas from Xerox, but whatever, they took IBM on and won about 24-26% of the market which they still hold.

Tandy virtually vanished, it seemed, except in the education market where the Tandy Model 3 and Model 4 machines gained second place to Apple with Commodore third, - in the USA which is about half at the work market.

But of course as we know the famous Tandy Color Computer steadily succeeded in winning friends and has sold about 2 million units in seven years.

Last year Tandy decided to try again with aggressive marketing to win a bigger market share. They decided on competitive pricing, good products, excellent support and first-class marketing.

The figures speak for themselves. In 1986 alone Tandy sold 668,000 machines in the retail market. These comprised 276,000 proprietary machines (TRS-DOS and CoCo's), 272,000 PC-DOS machines and 120,000 portables. They gained 25% of the total retail market, Apple also 25% and IBM 17%.

These sales do not include corporate sector sales where Tandy has also adopted an aggressive marketing policy increasing their sales force from 500 to 1500 and soon to go to 2,000 people solely involved in marketing to the top companies.

But look at the progress of the figures. For the first 9 months of 1986 the figures were Apple 26% of market, IBM 20% of market and Tandy 17%. In the final quarter of 1986 the change had been Tandy 37% of retail market, Apple 24%, IBM 12%, leaving 27% to be sliced up between others.

The success story is based on the Tandy 1000SX which has dominated the lower end of the of market and even competed with the Asian Clones. Tandy is now one of the top five computer manufacturers in the USA, and make their own IBM-PC compatibles in the USA.

In fact they have become the top IBM-PC clone manufacturer in the USA, and rapidly outstripping the opposition. Indeed a "potent force" in the fiercely competitive USA computer market place.

Now Tandy is going to strike a chill in the camps of the competition because it has released a competitively priced range of new models.

A Wall Street analyst said "This (new range) gives Tandy the broadest product line in the industry at the best price performance. This offering has something for everyone."

The new equipment is expected to increase Tandy's share of the market even further. The new products are more powerful than those offered by competitors and cost substantially less.

The top of the line is the Tandy 4000 with the Intel 80386 chip and like the new IBM Personal System/2 line includes a 3 1/2 inch 1.44 MB floppy disk drive. It will be ideal for desk top publishing, education and business.

Tandy has its first MS-DOS laptop, model 1400LT priced a about 25% of the price of similar models from IBM and Compaq. It is tipped to give Toshiba a run for their money as the top selling lab-top.

Tandy has its own Laser Printer (LP1000) which emulates Hewlett-Packard and IBM models but does not support the industry standard postcript page description language for desk top publishing, but upgrades to their standard can be obtained from various manufactures.

continued on p 61

The continuing saga of.....

FRICKERS FOLLIES

By Jack Fricker
ARTICLE

THIS MONTHS FRICKERS Follies is likely to be a bit of a dogs breakfast, mainly because I have a couple of things to say.

Well, this years CoCoConf has come and gone and I would venture to say that it was a success, at least from my point of view.

In fact it was better than previous years because this year I managed to hear some of the other talks such as Mike Turks talk on "C" and John Redmonds talk on Assembler. There was even a talk on messy-dos from Farley.

These unfortunately were the only talks that I could attend. I hear the other ones were excellent - unfortunately we were too busy on OS-9 (like last year and the year before that ... -ed.) to attend (you knew OS-9 would get a mention here somewhere).

One of the things that we did look at that runs under OS-9 at the Conf is the word processor Stylograph, otherwise known as Stylo.

There are 2 main word processors that run under OS-9 - one of them being Stylo and the other excellent one being Dynastar.

Dynastar is a work alike to Wordstar, so if you are happy using Wordstar then Dynastar may be for you. Another interesting thing about Dynastar is that it was written entirely in Pascal!

Even though it wasn't written in assembler is is still very fast and compact. In fact the same company that wrote the Dynastar program wrote the Pascal Compiler (Dyna).

There are 2 versions of Stylo that are available for the CoCo 3, and the version that we looked at at the Conf is the standard version. This version is the one that is intended for all versions of 6809 OS-9 and all types of terminals.

The other version of stylo will only run on CoCo's and does not work properly with CoCo Level 2 and an 80 column display. This version is sold at a much cheaper price than the standard version. Incidentally, it was disabled against being changed to work on any other system or terminal.

We spent quite some time changing the standard version to get it to work with the CoCo 3 and managed to get it working with the exception of the scroll screen down function and I promised those present that I would try to get it to work.

For those of you who were there the function that works in my 68000 system is Insert Line which is 1F 30 (N.B. not 1B 30) according to the manual.

The first thing that I want to set right is the myth that OS-9 is a hard dos to learn. It is not hard, mainly because everything has been set out in a logical format and the syntax really is quite simple to follow.

If you think that OS-9 is difficult to follow have a look at MS-DOS (Messy dos to those who use it). Here is an example:

If you want to list the contents of a text file under OS-9 you just type ...

```
LIST <Filename>
```

That seems simple enough, so lets have a look at how you do the same thing under MSDOS. It is called TYPE - its' syntax is...

```
TYPE <Filename>
```

Okay, thats simple enough you say, but under MSDOS there is a command called LIST and what do you think it does? It gives you a directory, thats what!

There are other examples of the non-logical syntax of MSDOS.

I hope this example helps you to see that OS-9 is easier to learn than some other operating systems. MSDOS is not the only non-logical one. Some of the other systems are unfortunately following this trend.

Well thats enough of the soap box preaching, lets get back to cases.

One of the questions most asked is when will OS-9 Level II will be released. Your guess is as good as mine. It was supposed to be released months ago and as I write this, it still officially hasn't been released.

Well it sort-of has been released ... there are some programs (games and the such) that have been released that use Level II, such as "Koronis Rift" and one or two others that use a short form of Level II.

If you have one of these programs they will use all of the available memory and will run the computer faster, which is one of the advantages of Level II over Level 1.

When you first boot these programs you will see the 'OS-9 Level II' message and the copyright messages.

When you see this, type control C and the program should stop - you will then be running level II. However the stripped version lack some of the tools and commands needed to run OS-9.

To get around this, get one of your Level I disks and put it in drive 0. Then format a fresh disk and do a COBBLER on it.

Re-insert your Level I disk and use DSAVE to create a list of files to be copied and then use this to copy to the new Level II disk.

All of the Level I commands should work with Level II - all the ones that I have tried so far work without problems.

This method I have described will give you the speed and memory capabilities of Level II - but it will not give you the windowing which is the major difference between Level I and II.

LOST IN THE WILDS?

by Ozzie OS9

(from the depths of PNG)

WELL, BELIEVE IT or not, you are finally going to get the final part of Ian Lobleys CASH BOOK program.

I must apologise to Ian, and to everyone else for taking so long to come up with the 'goods'.

Just to refresh your memory, the first parts of Ian's BasicO9 program were published in March and April's CoCo magazines. In order to use the following description of his prize winning program, you need to have typed in and SAVEd all modules exactly as printed.

Therefore you should have a SYSTEM disk with the following procedure files in the CMDS Directory:-

(CAPITAL letters indicate the NAME of the file and 'lowercase letters' the procedures that must be in that file)

If files have been PACKed, use RUNB; if you have NOT packed files, then you must use BASICO9.

```
 RUN-CASH
* menu          * read-bank
* title         * title2
* items         * change-chq
* get-chq       * search-chq
* create-chq    * print-t
* print-chq     * set-date
* input-chq

 CREATE-CASH
* create-menu   * create-chq
* create-bank   * create-name
* input-items   * create-items
(*REM  'create-bank'  creates
bank and start-bal)


 PRINT-CASH
* search-month-p * print-chq-p
* title-p       * macau-title
* items

 OTHERS
* search-month   * del-all
* delete-chq-record
* del-all        * see-bank
* recon          * recalc
* change-start-bal
```

A MEMory size of 9000 is required to RUN the procedure RUN-CASH, store data files etc. during operation.

The 'CASH' directory will always be the working data directory (chd /d0/cash).

Now I'll let Ian take you through his program:----

" ... the way I wrote the program was to combine all the procedures that were involved in one function (such as printer routines) into one loadable file and then individually pack each procedure into the cmds directory when the system had been debugged.

I then built a NEW workable systems disk with just the cmds that I needed.

The procedure needed to get our cheques into the file called 'cheque-rec' is 'input-chq'.

This provides us with one continuous list from the first cheque written, to our very last.

Remember, we will be storing our deposits in this file as well.

The 'input-chq' procedure is probably more complicated than it need be, but I wanted an input program that would automatically enter the month and the year of the cheque (and I only had to enter the day), and one that automatically entered the cheque number for me.

I use OPAK (in the 64 character by 19 line mode) and if you are not, then you will have to change some print formats.

I wrote the input-chq procedure some time ago ('my early days of OS9') and looking back I can see changes that can be made; however it works.

The input-chq procedure runs six other procedures:-

```
* items       * read-bank
* title       * print-chq
* print-t     * input-items
```

The 'items' procedure reads what codes and catagories are in the 'items-list' file, and passes these back to the 'input-chq' procedure by use of the PARAM command.

The next procedure, 'read-bank', reads the current bank balance, and returns the result to the 'input-chq' procedure.

The procedure 'title' prints a heading on the screen. Procedure 'print-chq', prints a cheque to the screen. 'print-t' is called by the 'input-chq' procedure if the item code is not found in the current item code list.

It will print a list of the current items to the screen.

Procedure 'input-items' is the main procedure for entering new codes and catagories. It is called by 'print-t' if the user wants to enter new codes and catagories during a posting session.

RUN 'input-chq', the program will first prompt for what month you are doing and will display the last cheque or deposit that you entered during the last session. It will then prompt for the next cheque number.

Enter the last three (3) digits of the next cheque number. The next prompt will be for the date of the cheque, then (under the cheque number column) you will have to enter a 's' for the same cheque number as previously, a 'd' for deposit or a <CR> for the next cheque number.

You are then able to enter the name of the payee and the amount.

Under the the type heading you will be asked to enter the catagory code you want this cheque or deposit to be posted to.

The program will only allow you to enter codes that you have already entered into the system,

continued next page

TAPE LABLER

By Wayne Kely
UTILITY
16k + PRINTER

TAPE LABLER IS A very small utility program designed to write cassette labels.

The program runs under 2400 baud, and if you want to change this baud rate, then change line 10 to whatever you like.

The Listing:

```
1 '*****************************
2 '******CASSETTE LABELLER******
3 '********BY WAYNE KELY********
4 '*******   31/12/86    *******
5 '*****************************
6 GOTO10
8 SAVE"16A:3":END'7
10 POKE 149,0:POKE 150,18
20 CLS:PRINT@8,"CASSETTE LABELLE
R":PRINT:INPUT"ENTER TITLE";A$
30 INPUT"ENTER CLASSIFICATION";B
$
40 INPUT"CLOAD OR CLOADM";C$
50 GOSUB 180
60 PRINT:PRINT:PRINT"  PRESS ANY
 KEY TO PRINT LABEL"
70 EXEC 44539
80 FOR C=1 TO 4:PRINT#-2,"------
-------------------------";CHR$
(13);:NEXT
90 PRINT#-2,CHR$(13);"**********
**********************"
100 PRINT#-2,TAB(INT(32-T)/2);A$
+"-"+B$
110 PRINT#-2,TAB(INT(32-B)/2);C$
120 PRINT#-2,"******************
**************"
125 PRINT#-2,CHR$(13);
130 FOR T=1 TO 12:PRINT#-2,"----
-------------------------";CH
R$(13);:NEXT T
140 PRINT:INPUT"PRINT ANOTHER LA
BEL (Y/N)";A$:IF A$="Y" THEN GOT
O20
150 IF A$="N" THEN GOTO 170
160 GOTO 140
170 END
180 T=LEN(A$+B$)+1
190 B=LEN(C$)
200 IF T>32 THEN GOTO 230
210 IF B=5 OR B=6 THEN GOTO 230
220 GOTO 40
230 RETURN
```

from previous page

or NEW codes that you wish enter at this point.

The remaining procedures allow you to:-

1. change cheques
2. search for cheques
3. get totals of cheques and catagories
4. change codes and catagories
5. print results to the printer
   - monthly
   - year to date
6. reconcile our bank statement
7. list all cheques and deposits

Hope the above information makes sense to you - it should be enough to get you well underway.

However if you strike problems, I will assist as much as possible. Also, if you make changes, could you please let me have a copy

continued from p 58

And finally, Tandy has two very competitive priced additions to its fast selling 1000 family, the Tandy 1000 TX and 1000 HX models which are intended for the education market, where Tandy is No 2 and rapidly gaining on Apple.

When one considers that support for a computer is vital for a number of years, it now makes more sense than ever to purchase Tandy. Where is the support for the SEGA? What about the Commodore Vic 20, Commodore 16, Commodore plus 4? Need I go on?

But what about the Tandy Colour computer Model 1 which was introduced in 1980 and discontinued in 1983? You can still get upgrades and support. You can even do that for the TRS-80 model 1 introduced in 1977 and discontinued in 1980.

So let's support Tandy, this magazine and people like Computer Hut, Paris Radio Electronics, Blaxland Computer Services and others who have enabled Tandy computers to be so successful in Australia.

But also let's talk to everyone with pride about Tandy computers and show how affordable and reliable they are, especially CoCo's and model 1000's. Not one person is unhappy with their choice. So we have a lot to gain and nothing to lose by our enthusiasm.

continued from p 48

It could be sold as a disk (like "The Best Of" series) or an annual special magazine issue!

I would like to take this opportunity to express my sincere appreciation to Graham and the team (and that means all of you who contribute to its success) for doing such a greatjob for us. Moving into Viatel was a brilliant move.

Communication is the name of the game. It's now at your finger tips - literally. Well done GOLDLINK 642.

I have a suggestion for an article in the mags ... or even a special edition for us to order. What about "How to use GOLDLINK" and "How to leave messages on GOLDLINK" (and where)

1) for Graham,
2) for specialist editors etc.
3) for other members.

By the way, I am on GOLDLINK (Viatel) and my Viatel number to quote is [838155830] should you want to reach me. Regards to you all. Allan Thompson (Adelaide SA).

user contacts

(Stop between numbers = h.h. else a.h.; but, hyphen between = both)

ACT:

CANBERRA NTH	JOHN BURGER	062 58 3924
CANBERRA STH	LES THURBON	062 86 9226

NSW:

SYDNEY:

BANKSTOWN	PAT DORSETT	02 646 3619
BLACKTOWN	KEITH GALLAGHER	02-627-4627
CARLINGFORD	ROSCO MCKAY	02 624 3353
CHATSWOOD	BILL O'DONNELL	02 419 6081
COLYTON	HERMAN FREDRIKSSON	02 623 6379
FAIRFIELD	ARTH PITTARD	02 72 2881
GLADESVILLE	MARK ROTHVELL	02 817 4627
HILLS DIST	ARTHUR SLADE	02 674 5620
HORNSBY	ATHALIE SMART	02 848 8630
INGLEBURN	STEPHEN RIDGWAY	02 605 7382
KENTHURST	TOM STUART	02 654 2178
LEICHHARDT	STEVEN CHICOS	02 560 6207
or	GORGE SCHEGARAY	02 560 9665
LIVERPOOL	LEONIE DUGGAN	02-607-3791
MACQUARIE FIELDS	BARRY DARNTON	02 616 1909
MINTO	GRAHAM POLLOCK	02 603 5026
SUTHERLAND	IAN ANNABEL	02 526 3391
SYDNEY EAST	JACKY COCKINOS	02 344 9111

ALBURY	RON DURCAN	060 43 1031
ARMIDALE	DOUG BARBER	067 72 7647
BLAXLAND	BRUCE SULLIVAN	047 39 3903
BROKEN HILL	TERRY MOONAN	080 88 2362
CANDEN	SEAN MURDOCH	047 74 6291
COFFS HARBOUR	BOB KERRY	066 51 2205
COOMA	ROSS PRATT	064 52 3065
COONAMBONG	GEORGE SAVAGE	047 77 1054
COOTAMUNDRA	CHERYLE VILLIS	069 42 2264
DENILIQUIN	WAYNE PATTERSON	058 81 3014
DUBBO	GRAEME CLARKE	068 89 6549
FORBES	JOHANNA VAGG	068 52 2943
GOSFORD	PETER SEIFERT	043 32 7874
GRAFTON	PETER LINDSAY	066 42 2503
GUYRA	MICHAEL J. HARTMANN	067 79 7547
JUNEE	PAUL MALONEY	069 24 1860
KEMPSEY	RICK FULLER	065-62-7222
LEETON	BRETT WALLACE	069-53-2081
LISMORE	ROB HILLARD	066 24 3089
LITHGOW	DAVID BERGER	063 52 2252
MAITLAND	BILL SHOV	049 66 2557
MOREE	ALP BATE	067 52 2465
NAMBUCCA HDS	WENDY PETERSON	065 68 6723
NARROMINE	GRAEME CLARKE	068 89 6549
NEWCASTLE	LYN DAWSON	049 49 8144
NOWRA	ROY LOPEZ	044 46 5449
PARKES	DAVID SMALL	068 62 2662
PORT MACQUARIE	RON LALOR	065 62 2662
SPRINGWOOD,	JIM HOPPITT	047 54 1474
TAMWORTH	GARY SYLVESTER	067 61 9316
UPPER HUNTER	TERRY GRAFOLIE	065 45 1898
URALLA	FRANK MUDFORD	067 78 4391
WAGGA WAGGA	CES JENKINSON	069 25 2263
WYONG	JOHN WALLACE	043 90 0312

NT:

DARWIN	BRENTON PRIOR	089.81.7766

QLD:

BRISBANE:

BIRKDALE	COLIN NORTH	07 824 2128
CANNON HILL	ROSEMARY LITZOW	07 395 0863
CLAYFIELD	JACK FRICKER	07 262 8869
COLL'WOOD PK	ANDREW SIMPSON	07 268 5206
IPSWICH	NICK MURPHY	07 271 1777
PINE RIVERS	BARRY CLARKE	07 204 2806
SOUTH	JOHN FOXON	07 208 7820
SOUTH WEST	BOB DEVRIES	07 372 7816
SCARBOROUGH	PETER MAY	07 203 6723

AIRLIE BEACH	GLEN EVANS	079 46 1264
BIGGENDEN	ALAN MERHAM	071 27 1272
BOWEN	TERRY COTTON C/O	077 86 2220
BUNDABERG	ROB SIMPKIN	071 71 5301
CAIRNS	JEFF LARSEN	070 54 7127
DALBY	MERRICK TARSKY	074.62.3228
GLADSTONE	CAROL CATHCART	079 78 3594
GOLD COAST	GRAHAM MORPHETT	075 51 0577
GYMPIE	BERT LLOYD	071 8219100
HERVEY BAY	LESLEY HORWOOD	071 22 4989
MACKAY	LEN MALONEY	079511333x782
MARYBOROUGH	JOHN EFFER	071 21 6636
MT ISA	JACK RAE	077 43 3466
MURGON	PETER ANGEL	071 68 1626
ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
TARA	DEBBIE DORFIELD	074 65 3177
TOOWOOMBA	LEN GERSHKOVSKI	076 35 8264
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064

SA:

ADELAIDE	JOHN HAINES	08 278 3560
PORT NOARLUNGA	ROB DALZELL	08 386 1647
SEACOMBE HTS	GLENN DAVIS	08 296 7477
PORT LINCOLN	BILL BOARDMAN	086 82 2365
PORT PIRIE	VIC KNAUERHASE	086 32 1230
WHYALLA	MALCOLM PATRICK	086 45 7637

TAS:

DEVONPORT	JEFF BEST	004 24 6759
HOBART	BOB DELBOURGO	002 25 3696
KINGSTON	WIN DE PUIT	002 29 4950
LAUNCESTON	BILL BOVER	003 44 1584
SMITHTON	HARRY CHRISAFIS	004-52-1590
WYNYARD	ANDREW WYLLIE	004 35 1639

VIC:

MELBOURNE:

MELBOURNE CCC	LES LEISHMAN	03 484 0822
DANDENONG	DAVID HORROCKS	03 707 5870
DONCASTER	JUSTIN LIPTON	03 857 5149
FRANKSTON	BOB HATTER	03.783.9748
HARRE WARREN	LEIGH BANES	03 704 6660
NTH EASTERN	PETER WOOD	03 435 2018
N'TON PENNINSULA	GORDON CHASE	059 71 1553
MELTON	MARIO GERADA	03 743 1323
PAKENHAM	JASON HALL	059 41 1398
RINGWOOD	IVOR DAVIES	03 758 4496
SUNBURY	JACK SMIT	03.744.1355
SUNSHINE	IAN BUTTRISS	03 314 6242
UPR P'TREE GLY	RORY DOYLE	03 758 2671

BAIRNSDALE	COLIN LEHMANN	051 57 1545
BALLARAT	MARK BEVELANDER	053 32 6733
DAYLESFORD	DANNY REDJI	054 24 8329
GEELONG	DAVID COLLIS	052 43 2128
KAFFRA	MAX HUCKERBY	051 45 4315
MOE	JOSEPH NESTER	051 27 7817
MORNINGTON	MICHAEL MONCK	03 789 7997
MORWELL	JEFF SHEEN	051 33 9904
SHEPPARTON	ROSS FARRAR	058 25 1007
SMYTHESDALE	TONY PATTERSON	053 42 8815
SWAN HILL	BARRIE GERRAND	050.32.2636
TONGALA	TONY MILLIS	058 89 2251
TRARALGON	LEIGH DAVES	051 74 5552
WONTHAGGI	LOIS O'MEARA	056 72 1593

VA:

PERTH	IAIN MACLEOD	09 446 2136
GIRRAWHEEN	HANK WILLEMSEN	09 342 7639
KALGOORLIE	TERRY BURNETT	090.21.5212

CANADA - CoCo:

Ontario	Richard Hobson	416 293 2346
Toronto	Franz Lichtenberg	416 845 2889

special interest groups

TEACHERS' INTEREST GROUP

BRISBANE	BOB HORNE	07 281 8151

BUSINESS:

BRIZBIZ	BRIAN BERE-STREETER	07 349 4696

C64 GROUPS:

NATIONAL C64 USERS' GROUP

	GRAEME NICHOLS	02 451 2954

NSW

SYDNEY

BANKSTOWN	CARL STERN	02 643 3619
CARLINGFORD	ROSKO MCKAY	02 624 3353
GLADESVILLE	MARK ROTHVELL	02 817 4627
SYDNEY EAST	JACKY COCKINOS	02.344.9111

COOMA	ROSS PRATT	064 52 3065

QLD

BRISBANE	JACK FRICKER	07 262 8869

VIC

LATROBE VLY	GEORGE FRANCIS	051 34 5175

WA

KALGOORLIE	TERRY BURNETT	090.21.5212

MC-10 CONTACTS:

LISMORE	BOB HILLARD	066 24 3089
SYDNEY	GRAHAM POLLOCK	02 603 5026

TANDY 1000 / MS DOS:

NSW:

GLADESVILLE	MARK ROTHVELL	02 817 4627

SYDNEY WEST	ROGER RUTHEN	047.39.3903
WYONG	JOHN WALLACE	043 90 0312

QLD:

BRISBANE

NORTH	BRIAN DOUGAN	07 30 2072
SOUTH	BARRY CAVLEY	07 390 7946

GOLD COAST	GRAHAM MORPHETT	075 51 0577

SA

PORT LINCOLN	BILL BOARDMAN	086 82 2365

VIC:

LA TROBE VALLEY	PETER FOLEY	051 74 5791
MELBOURNE	TONY LLOYD	03 882 4664

FORTH:

SYDNEY	JOHN REDMOND	02 85 3751

ROBOTICS:

GOLD COAST	GRAHAM MORPHETT	075 51 0577
SYDNEY	GEOFF FIALA	02 84 3172

CHRISTIAN USERS' GROUP:

COLLIE	RAYMOND L. ISAAC	097 34 1578

MSX

FRANKSTON	ALAN HASSELL	03 786 6290

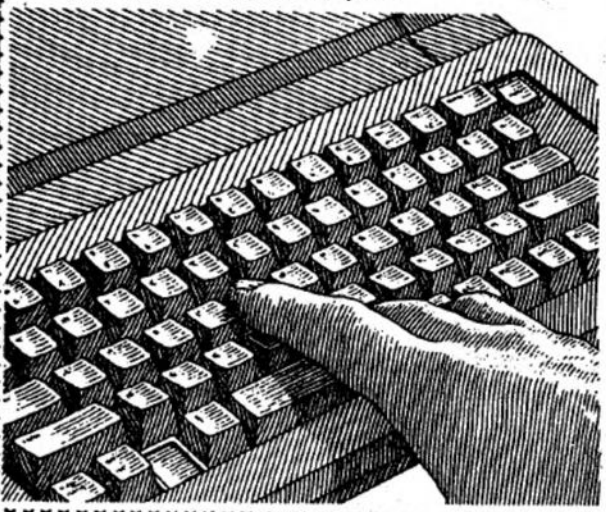
MODEL RAILWAY CLUBS:

BERRIGAN	DAVID PHILLIPS	07 807 2663

COMPETITIONS

As a result of the success of the Tandy programming contest this year, Tandy have agreed to rerun it in 1987–88!

So – get your thinking caps on! Perhaps YOU will be the one receiving that cheque from Tandy next year!



And speaking of cheques, the best ML Game for the CoCo 3 with a BiCentennial theme submitted by 7th November 1987, will win a $300 prize WITH royalties for every program sold from Goldsoft.

The next minor competition – the annual Graphics Competition begins now and ends on 7th November, 1987.

All computer created pictures are elligible, and the competition is divided into a section for Basic pictures, one for CoCoMax & ColourMax pictures, and one for pictures created in some other way.

As with the last Graphics Competition, the judges are looking for animated pictures.

First prize in each category will be 5 boxes of disks or tapes.

GOLDSOFT

P.O. BOX 1742, SOUTHPORT. QLD. 4215 Phone (075) 39-6177

Goldsoft Price List as at November, 1987 Please tick ✓ your requirements.

HARDWARE

CoCoConnection: $219.95 ( )
Video Amp: With sound - $35.00 ( )
Without sound - $25.00 ( )
The Probe: $49.95 ( )

GOLDLINK
Access Goldlink #642# on Viatel with a 1200/75 baud modem. Annual subscription: $44.95 ( )

SOFTWARE
Magazines, Tapes & Disks

Australian oo (Advanced Programs for your CoCo):

Magazines:		Tape ( )	Disk ( )
12 Months	$39.95 ( )	12 Months $123.75	( )
6 Months	$24.95 ( )	6 Months $ 74.25	( )
1 Month	$ 4.50 ( )	1 Month $ 16.50	( )

Softgold (Programs for your CoCo):

Magazines		Tape ( )	Disk ( )
12 Months	$39.95 ( )	12 Months $123.75	( )
6 Months	$24.95 ( )	6 Months $ 74.25	( )
1 Month	$ 4.50 ( )	1 Month $ 16.50	( )

Gold Disk - Available Quarterly:
1 - $16.00 ( )
2 - $16.00 ( )
3 - $16.00 ( )

4 - Coming Soon !!!

The CoCo3 Tape/Disk:
#2 - Tape: $16.00 ( ) Disk: $16.00 ( )
#3 - Tape: $16.00 ( ) Disk: $16.00 ( )
#4 - Tape: $16.00 ( ) Disk: $16.00 ( )

"Say the Wordz":
Two Curriculum based speller programs for your Tandy Speech/Sound pack: $29.95
Req: 32K + Tandy Speech Pack ( )

Best of CoCoOz - $16.00
A selection of programs from Australian CoCo Magazine.

		Tape:	Disk:
# 1	- Education:	( )	( )
# 2.1	- Games 16K:	( )	( )
# 2.2	- Games 32K:	( )	( )
# 4	- Business :	( )	( )
# 5	- Adventure:	( )	( )
# 6	- Preschool:	( )	( )
# 7	- Graphics :	( )	( )
# 8	- Games 16K:	( )	( )
# 9	- Games 32K:	( )	( )
#10	- Education:	( )	( )
#11	- Education:	N/A	( )

BRIC-A-BRAC

Blank Tapes:	12 @	$18.00 ( )
(C-30)	1 @	$ 2.00 ( )
Tape cases:	12 @	$ 5.00 ( )
Disk DSDD:	10 @	$20.00 ( )
	1 @	$ 2.50 ( )

BOOKS
Help (for your CoCo): $ 9.95 ( )
Nico Help (for your MC-10): $ 9.95 ( )

BACK ISSUES
Australian CoCo: Sep 84 - Dec 85: $2.00 ( )
Australian CoCo: Jan 86 - Feb 87: $3.75 ( )
Australian Nico: Aug 84 - Dec 85: $2.00 ( )

ADDITIONAL REQUIREMENTS

.....

.....

.....

New Subscription: ( ) Renewal ( )
Sub No:.....

Name:.....

Address:.....

.....

.....P/Code.....

Phone (...).....

Please find enclosed:
a. Cheque: ( )
b. Money Order: ( )
c. Credit card: ( )

Credit Card Type & Number:

Bankcard ( ):.....-.....-.....-.....

Mastercard ( ):.....-.....-.....-.....

Visa ( ):.....-.....-.....-.....

Expiry Date:...../19...

Authorised Amount: $.....

Signed:.....

NOV 87

DIRectory

Insertions in this Directory cost $160.00 for six months or $300.00 for twelve months per frame. Changes to Insertions incur a further charge.

If you sell Soft or Hardware for Tandy computers, you need to be listed in this quick reference guide.

Remember! Tandy owners READ this magazine!

N.S.W. Central Coast

Computer Wizardry
P.O. Box 979,
Gosford N.S.W. 2250

- ★ *Educational Software*
- ★ *Communicating Software Hardware*
- ★ *Agents for Computer Hut*
- ★ *Agents for Speech Systems*
- ★ *Prompt, Courteous Service*
- ★ *Phone or Write for Catalogue*

Bankcard & Visa Card Welcome

043-24-7293

N.S.W. Gunnedah

Esther's Sports & Electronics
166 Conadilly St.,
Gunnedah N.S.W. 2380
(Tandy Dealer 9223)

Agricultural Computing Specialists
For Friendly Service to the man on the land, or for fast accurate help to the town dwellers

067-42-2230

N.S.W. Lismore

Decro Electronic Services
12 Carrington St.,
Lismore, N.S.W. 2480
(Tandy Dealer 9225)

Best range of Computers and Computing Equipment in Summerland. Whether you live on the North Coast, or are just on holidays, you can't afford not to call and see us!

066-32-1896

South Australia Morphettvale

NATIONAL
1st TIME ADVERTISED
THE PROFIT POTENTIAL IN YOUR PERSONAL COMPUTER
$25,000 PER MONTH IS POSSIBLE
For FREE Brochure
Ph (08) 382 7281
or send SSAE to
Mal. P. Bailey,
6 Bower Crt,
(Dept 7) Morphettvale
South Australia 5162

Victoria Blackburn

D & L Wilson & Co Pty. Ltd.
6 Stafford St.,
Blackburn, Vic. 3130

Serving Melbourne's East
Software — Hundreds of titles
Hardware — Drives, Printers
Service — Upgrades

03-898-4521

Tasmania Hobart

The Delbourgos
15 Willowdene Ave.,
Sandy Bay
Hobart. Tas. 7005

- ★ *Expanded Basic — a better Basic for your CoCo*
- ★ *The Proportioner — a utility to provide equal gaps between proportional letters on Tandy's DMP200 printer*
- ★ *Mathematical function database*

002-25-3896

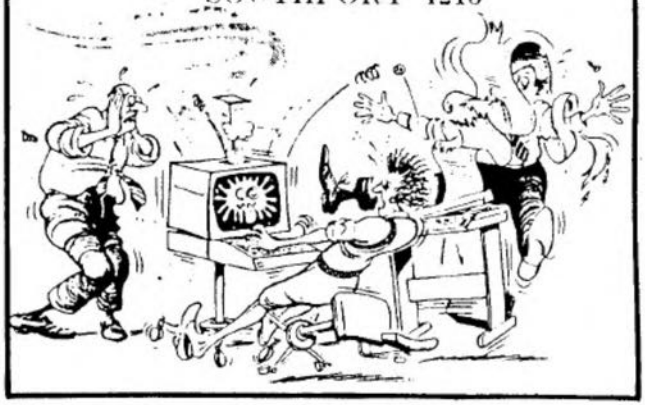
CoCo3 PART 3

DISK/TAPE

ARTIST	3AL	3ONO
CUBES	3HOW?	3YAHTZEE
SUNSET	3SNAKES	3MISSION

TAPE $10.00
DISK $16.00

Available from:
GOLDSOFT
P.O. BOX 1742
SOUTHPORT 4215



Queensland Brisbane

Queensland Colour Software Supplies
P.O. Box 306
Clayfield Qld. 4011

- ☆ *64k & 128K upgrades*
- ☆ *80 column cards*
- ☆ *Y cables*
- ☆ *Games*
- ☆ *Terminal programs*

07 - 262 - 8869
A/H

CANADA

Computer Assembly Manuals
BIG BLUE SEED for IBMtm BUILDERS
* parts list
* placement diagrams
* instructions
for assembling over 75 bare IBM-compat. cards. Now includes guides for 640K, Turbo & AT MthBds!
$17.95 US

Money Order, VISA/MC
NuScope Associates, 1C
P.O. Box 742, Stn B
Willowdale, Ontario