Registered by Australian Post — Publication No. QBG 4009

NEW GUINEA K5.95

$4.50

For your TANDY Color Computer

AUSTRALIAN RAINBOW

August, 1985

No.50

We produce Software

for

The CoCo and Tandy 1000

Our list of programs from

★ OWLS NEST SOFTWARE	U.S.A.
★ PAL CREATIONS	U.S.A.
★ CoCo CASSETTE (Monthly)	U.S.A.
★ PICOSOFT GAMES	U.S.A.
★ C.C.S.	AUSTRALIA
★ C.H.S.	AUSTRALIA

is too long to print here.

So, for a FREE CATALOGUE, call or write to

Computer Hut Software

21 WILLIAMS ST., BOWEN, QLD., 4805, AUSTRALIA — PHONE (077) 862220

Now, for all versions of COCO, on disk or on cassette, the definitive EXPANDED COLOR BASIC and the BASSEMBLER

EXPANDED BASIC is designed to be added on to Extended Basic and includes the following new features:

- Printing text in all modes and colors
- Special 51x24, 64x24, 85x24 PMODE4 screens
- Scrolling of any screen section any way
- Extra colors in a new mode
- Borders for the text screen
- Extra graphics pages
- User definable sound effects
- REPEAT...UNTIL loops
- Multiline IF...THEN...ELSE statements
- Procedures in addition to subroutines
- Local and lower case variables
- Full ON-ERROR implementation
- Breakdisabling, auto-line numbering
- On-screen editing, 10 function keys
- User-definable characters /printer widths
- Single key entry of most BASIC words
- Ability to execute strings as commands.

The BASSEMBLER is a UNIQUE macro-assembler that allows you to inter-mix assembly language and BASIC in the same program - you can literally have one line assembly and the next line BASIC.
The BASSEMBLER understands the complete 6809 assembly language, macros and other pseudo-ops, labels and LINE NUMBERS too.
A disassembler/moniter Program is included with the BASSEMBLER.
Easy to follow instructions and demonstration programs come with the BASSEMBLER and EXPANDED BASIC.
64K is required. Specify disk or tape (giving model of recorder) and specify COCO model (1 or 2).

The BASSEMBLER	**$15**
EXPANDED COLOR BASIC	**$30**
EXPANDED BASIC + BASSEMBLER	**$40**
DISK USERS add an extra	**$ 3**

THE PROPORTIONER

If you own a DMP200 printer and have at least 32K RAM and would like to be able to print 'book quality' textfiles that are right and left justified in proportional character spacing, then this utility is just what you've been looking for. No matter what your word processor, this program will take any ASCII file and print a hard copy perfectly with an equal gap between words and with proportional spacing. Headers, footers, format markers, page numbering, printer drivers and so on are set up to your specification so that any word processor can produce truly professional documents just as in typeset books.
(This advertisement was produced in such a way).
Specify tape/disk when ordering and 32K/64K.
A bargain at $25 tape, $28 disk for any CoCo.
Address shown below.

MATHEMATICAL FUNCTION DATABASE

For the professional scientist working at home, here is a compendium of the most frequently used mathematical functions. Tables and graphs are provided of BESSEL functions, (POLY)GAMMA functions and ORTHOGONAL polynomials (JACOBI, GEGENBAUER, LEGENDRE, LAGUERRE and HERMITE). Accuracy is as good, if not better, than the handbooks on the subject.
Available for the Model 100 and any CoCo.
Price $30 tape or disk (for CoCo).

Enquiries and orders to:

THE DELBOURGO's,
15 WILLOWDENE AVENUE, SANDY BAY, HOBART,
TASMANIA 7005.
Phone: (002) 253896

AUSTRALIAN RAINBOW

index

UPPER CASE = PROGRAM + ARTICLE
lower case = article only

AUSTRALIAN RAINBOW Publisher and Editor Graham Morphett. Co- editor Kevin Mischewski. Assistant Editor Sonya Young. With grateful assistance from Brian Dougan, Richard and Judy, Bob Thomson, Paul Humphries, Alex Hartmann, Michael Horn, Jim and Sheryl Bentick, Annette Morphett. Cover Art Jim Bentick.
ADVERTISING DEADLINES: The 7th of the preceeding month of publication. All advertising is arranged through ToToAdvertising, PO Box 8730, Gold Coast Mail Centre, Qld, 4217.
OS 9: Kevin Holmes is the contact for os 9 information. He also has access to OS 9 Software from the US. His address is: 39 Pearson St., Harars, NSW, 2250.
All programmes in this issue of AUSTRALIAN RAINBOW are available on Rainbow on Tape.

Telephone: 075 51 0015 Voice; 075 32 6370 CoCoLink
Printed by: Australian Rainbow Magazine PO Box 1742, Southport, Qld, 4216.
Founder GREG WILSON.

It is a year since we took over from Greg, but is about FOUR years since the magazine got going! I've made the comment before, but it is worth repeating - there are few five year old computers still being sold. There are even fewer which show hope of a considerably longer life span than CoCo. With the software available and planned in the forseeable future for CoCo, we are confident CoCos will still be selling in 1990.

To mark the occasion, we have departed for this month only, from American Rainbow as the source of the greater percentage of our material.

Reflecting the growing skills of our contributors, a number of recent articles and programs have been submitted which were outstanding in their own right. They do not fit the content guidelines of Australian CoCo, but more closely resemble the style of material we receive regularly from the US in the Rainbow Magazine.

So in memory of Greg, we present this magazine, which demonstrates the skills that have developed since he gave us all the impetus to get involved. And so you'll not be totally bereft of US input, we've also included an article and a program from a pair of our US readers.

Being one year since we 'moved in', we thought it appropriate to use the anniversary to reassign a couple of columns. The first is OS8.

OS8 has always received a mixed reception where ever it went. Be that as it may, there are certainly enough crazies out there to ensure it lives!

Forth, whilst not able to share many of the benefits of OS8, certainly has its proponents. Now that we've seen Forth working so successfully at CoCoConf, I'm sure many will want to use it themselves.

Forth and OS8 are really not for the beginner, so they come to Rainbow. I hope that you try each.

It always seems strange to me that people will tell you they bought their computer to 'experiment with', and once they've learnt Basic, don't want to know about anything else!

I know learning Basic was hard, and even now you probably don't feel confident with it, but I'll wager that several years ago you knew even less! And I'll bet that since then you've not regretted the time spent.

You need to take a step of faith and accept the possibility of even greater achievement! The support you've felt whilst learning Basic, even if only remotely through this magazine, is there for these other languages and systems - help is at the other end of the phone, so give it a go - make this year your year to branch out and try something different!

Whilst Australian Rainbow Magazine will retain the current commitment to Education and Business, expect the articles this year to centre more on Forth, Pascal (perhaps), Machine Language, OS-9 and advanced Basic techniques.

We want to challenge some of those clubs out there that have almost gone to sleep to wake up and provide some meaty input to the magazine! As an incentive, we are broadening the Games Contest of last year to a Programming Contest. The prizes are still being worked on, but I'm sure you will wont to be in with a chance when you see what they are.

More next month!

Last month I mentioned the 'new' CoCo. A negative mention of it in this magazine, and a simultaneous positive mention of it in American Rainbow raised a few eyebrows.

I'm happy to stand by what I said!

We do however, have some information about this phantom, so if you want to dream, here's what we think it could be like, if it eventuates.

'The new CoCo' appears to be a true 16 bit 68000 machine, with a Basic compatible with CoCo's. It is HIGHLY unlikely that machine language progs or progs with pokes will work in it without an emulator. But the difference between it and your existing CoCo would have a similar 'feel' to the difference between the MC 10 and CoCo. In other words, whilst being able to run some CoCo software, the new computer would be so far in front, you wouldn't want to!

It' is being widely described as 'the OS9 Machine' and there is little doubt that programs written in Basic 09 on CoCo would work on the new CoCo. But this almost certainly also means another 'revision' or even 'version' of OS-9 for Tandy. It could turn out to be that the Version 2 of OS-9 we have been hearing about, and expecting to be released with a future 128K CoCo might well be for the 'new CoCo'!

Price initially would be targetted on US$1000.00. BUT remember what I told you - things are so bad for personal computers in the US at present, that you'd have to be very brave to go this far out on a limb. Tandy might well be satisfied with a copy of another IBM machine (heaven help us!). See Bob Rosen's Article on the RMS chip set.

LETTERS

Dear Graham,

I'm a CoCo 2 owner and once a MICO owner but soon grew out of it.

I'm writing to thank you. When Greg Wilson died I thought Rainbow would go 'Phut'.

I think it's good that CoCo owners share programs. Out of all the programs I have, I've only had to buy one.

When I went to buy my CoCo 2 I was given some programs. After that I got in contact with an ex-contact and he put me onto his brother.

I have some friends that own Commodores (sorry about the language) one of whom has a 'Vic-20'.

He can turn out games - like that! He can get his 'Vic-20' to read the data for each letter at different places.

He can make an 'A' look like a 'Pac Man'. If my CoCo can do it and you know how would you please publish an article about it?

One more question. Why is 'Rainbow' called 'Rainbow'?

Yours Faithfully,
Gary Cazzulino.

Gary,

Thank you for the nice words.

The Commode Vic-20 is limited in a number of ways. In general, the characters you describe are the only ones you'd use under normal circumstances due to the Vic-20's lack of memory.

Your CoCo will draw these characters and more, it can also behave like an Apple, it can play four voice music, it can perform any business or education function, it can communicate with other computers by telephone, it can speak and understand the English language and it can control any external appliance you care to mention.

And that is only as at 12/5/85, because, likely as not, tomorrow we'll find something else exciting and new that CoCo can do!

Rainbow is called Rainbow for a number of reasons:

1. Originally, Lonnie Falk, Editor of American Rainbow, from whom we still obtain much material, called the magazine Rainbow because of the colours available from CoCo. Keep in mind that CoCo is five years old, and five years ago there weren't that many Colour computers around! CoCo was a big break through for its time - we just never understood just how big it was!

2. Greg Wilson, a genius and all round amazing guy decided to reprint Rainbow in Australia. He added tid bits of Australian news and spent a lot of money getting the magazine off the ground.

3. When Greg passed away last year, we wanted to continue his work. We feel that Rainbow is still Greg's magazine, and that we just have custody of it. So a lot of things we keep the same - including the name.

4. The name is appropriate, because as everyone knows, there is a pot of gold at the end of the Rainbow. That pot is CoCo. I know that sounds corny - but we hear every day of someone who is helped by CoCo, some kid who goes forward in her school work because she learns to drive a CoCo, some old man in retirement who gains a new lease of life through owning a CoCo. That is the sort of gold that is worth having!

Graham.

Dear Graham,

Could you advise if with a 128K upgrade there is more readily useable RAM. If I ?MEM do I get greater MEM which can be used without any switching. I have 64K but as you know 32K is virtually unusable. With the Tandy Disk, programs such as 40K, USE64K etc, which supposedly give 40K or 64K, don't operate.

Maybe a tute on the pokes for more useable RAM and a program such as USE64K for disk, would be popular.

Many Thanks,
Mick Gooch.
LOWOOD, QLD.

Mick,

The 128K upgrade will not provide initially, any further RAM addressable by BASIC.

The memory created is available to OS-9 and FLEX, as well as assembly language programs.

I understand that a DOS may be available later in the year which will enable 96K for BASIC programs. It is felt that BASIC would be noticably slow under these circumstances.

You use a computer to do certain tasks because it does them better than the old ways; large quantities of memory need more efficient languages to service them adequately.

Of course, there is always OS8 .....

I'm led to believe that a 40K prog, usable in a disk system is difficult to achieve due to the position of the DOS in RAM.

When one is wanting to run long games, I can understand why one would need large quanitiies of continuous memory, but if the task to be done can be broken into modules, then a disk system provides the opportunity to load program 1, access a set of standard files, accomplish a task, then close and RUN programs 2, 3 and so on, all the time accessing the same files, all the time working from menu to menu. We demonstrated how to do this on the disk version of 'Best of CoCoOz'. Under such a system, you could often get away with 16K!

Graham.

*

Dear Graham,

I would firstly like to commend you on an excellent magazine. I own a 64K EBC CoCo and have been computing for nearly a year. The Australian Rainbow has really been a terrific help. I am in year 11 at Guyra Central School.

Enclosed is a listing of my first attempt at graphics.

Listing one is called "AUZFLAG". It draws up the the Australian flag and plays "Waltzing Matilda". I certainly hope you like it.

With all the controversy about changing the flag, I thought I would also make a program that draws Australia's unofficial flag, the Eureka flag (Listing two).

Finally, I have submitted a one-liner that should be a help to people who can't think of lotto numbers. Simply RUN and press ENTER for another number.

I do hope you print my programs because I realise there are few Australian submissions in the Australian Rainbow. I do not get CoCoOz because the Rainbow is all I need for the computing I do.

I live on a 2000 acre propety on the Northern Tablelands near the town of Guyra, which is just north of Armidale.

We have become interested in micro-computers because we can see their potential as places to store records, but with no printer nor a disk drive it is a little difficult to do it efficiently. In the future something might happen, with any luck, but until then I'll just have to learn more from Australian Rainbow!

Michael Hartmann.
Guyra. NSW.

PS. Could you please enter me as a User-Group contact. My number is (067) 79-7547. Thanks.

Michael,

We've listed your programs elsewhere and used them in the first contents page for Rainbow on Tape.

I always get a thrill when I receive letters from favourite parts of the country. The New England area is special to me, and I have spent many happy hours at Black Mountain - just near you! If Mr Laurence is still a teacher at your school, tell him I said hello!

I don't think you need to wait for disk drives or printers to develop an effective Data Base for the property. Try 'my' one, published in Australian CoCo in September 1984, or adapt the one in Australian Rainbow this month.

You'll find you can obtain a lot of useful information even without the expensive add ons!

Welcome too as contact for your area. Make sure you contact Douglas (the) Barber at Tandy in Armidale to let him know what you are doing. Doug is a good guy and will do his best to help.

Graham.

*

Dear Graham,

In the past I have never bothered to answer your questions on the type of equipment or hardware I have; it is only because I only do it to pass time - I am not a serious hand, yet I do still love to type in the programs from the Rainbow. I become a nail-biter waiting for each issue.

If you are interested I have the Grey case CoCo upgraded to 64K, HJL Keyboard, CCR-81 cassette and now the Super5 CPA80 printer, plus VIP Writer, using it for the first time.

Please be careful - errors are begining to creep into the programs again. I am just too stupid to work them out for myself and spend hours just looking for errors in my typing. This is something I do to excell in and do not make many errors now.

Dean Hodgsons "Tank Math" has a Return without a Gosub in it, in this issue the "Hi-Res Racer" has an OS in line 70. Now I must close, keep up the good work! You feel like part of the family.

D.R.T. Austin.

You may not be aware that most of Rainbow's content is from the US, with the exception of this issue.

Normally therefore, we reprint what they print - if they make an error, we make the same one!

Dean's program is another exception, as he comes from SA. I know his program works, because I demonstrate it often at shows and in schools.

I suggest you take your listings to your Users' Group.

Graham.

OS8

OS has been a part of Australian CoCo since inception. But like FORTH, the high levels of understanding required of both these systems, mean that it is really only those with experience in computing that will be able to fully utilise their functions.

It is with a great pride that we therefore introduce OS8 to those readers that do not get Australian CoCo, and we trust that you derive much benefit from it as the days go by.

We should reiterate some of OS8's advantages here, as there are likely to be some who have not had previous contact with OS8.

OS8 is an operating system similar, in many ways, to many other better known systems such as MS DOS, OS9 and CPM.

However the authors of OS8 have looked carefully at the real needs of the computer owner and have been able to adopt proceedures which enable your CoCo to become most anything you want it to be.

A fairly definitive article on the subject was published in October 1984's issue of Australian CoCo. The Author, Paul Humphries, explained that "OS8 is the ultimate disk operating system. It has no drawbacks, or need for operator intelligence.

"It can be run on" any drive or disk system, and "it comes complete with a diamond tipped razor mode, which shreads the diskettes into 'stringy-floppy' format if required."

Some of the then awaited commands have now become available. For example the 'Chauffeur' command is now available.

'Chauffeur' overcomes the necessity for external drivers.

'Seance' allows for recovery of dead programs.

'Auto help' reads your brain waves and automatically executes the commands you are thinking of at the time.

OS8 has it's own internal modem which can be utilised to access memory from outside computer systems if internal memory requirements are exceeded. This is accomplished by internal microwave link, therefore eliminating the need for the telephone line.

This is done automatically, and it is quite usual when in the MEMSEARCH mode, to find parts of your programs stored in computers around the world, but more especially in the computers of the Treasury Dept of the Queensland Government. For some reason these seem particularly devoid of other use!

Documentation at Levels 1 through 5 is fairly complete, being supplied on microdot, but printed manuals run to 23 volumes, so it is unlikely that these will be readily available until the laser disk is released later this year.

CoCo's memory is an item which OS8 handles quite differently to traditional methods. In fact we had a program in Australian CoCo this past year which under software, provided 128K!

This is no longer a great feat we now find, because Barry Cawley has discovered a way in which to access 1056K, and as long as you use the OS8 loader, you don't even need OS8! The program to accomplish this should appear in next month's Australian Rainbow.

Initially programs were only available for those with OS8, but the ease of use of OS8 means that many have been able to provide programs which appear to run under Basic, but which are being controlled by the author's host system through the microwave link feature of OS8.

One such program was the 'Enable' program of Barry Cawley. I've taken the liberty of reprinting this program here.

Barry's program accesses the original CoCo ROMs and using a special OS8 technique, is able to get those ROMs to provide you with additional help when you make an error in BASIC.

As an example, enter the program, SAVE, then RUN. Then give a NEXT command. The result should astound you. Try other error codes too. Additional help has been found for some 38 commands so far!

Perhaps you're beginning to understand just what a help the full system could be to you.

(Thanks go to Paul Humphries for permission to reuse parts of his previous article here.)

THE LISTING:

```
1 ' ROM ENABLE PROGRAM FOR OS8
      BY BARRY CAWLEY
2 GOTO10
3 SAVE"ENABLE":DIR:STOP
10 CLEAR200,&H7CF3
20 FOR A= 31987 TO 32767
30 READ B
40 POKE A,B
50 NEXT
60 EXEC 31987
70 DATA 48,141,0,4,191,1,146,57,
53,32,189,173,51,189,209,229,52,
36,189,202,59,48,98,189,167,233,
189,169,116,15,111,189,185,92,18
9,185,175,53,4,79,49,141,2,141,4
8,181,189,125,46,48,141,2,208,18
9,125,46,22
80 DATA 47,61,166,128,39,6,173,1
59,160,2,32,246,57,87,79,87,32,8
9,79,85,82,32,83,84,85,80,73,68,
33,13,0,87,72,65,84,63,63,0,70,7
9,82,71,69,84,70,85,76,76,32,65,
82,69,78,84,32,89,79,85,13,0,87,
72,65,84,32
90 DATA 68,65,84,65,63,63,0,68,7
9,78,39,84,32,68,79,32,84,72,65,
84,0,87,72,79,79,80,83,33,33,0,7
8,79,32,87,65,89,32,73,68,73,79,
84,32,0,73,32,87,73,76,76,32,78,
79,84,32,33,33,33,0,78,79,32,87,
65,89,32
100 DATA 77,65,84,69,32,32,0,78,
79,46,46,46,78,79,84,32,65,71,65
,73,78,33,33,32,32,32,13,0,67,65
,78,39,84,32,66,69,32,68,79,78,6
9,32,0,73,76,76,69,71,65,76,32,6
8,73,82,69,67,84,0,66,85,84,84,6
9,82,32,70
110 DATA 73,78,71,69,82,0,84,65,
78,68,89,32,66,82,65,73,78,32,68
,65,77,65,71,69,32,13,0,83,84,82
,73,78,71,32,84,79,79,32,69,65,8
3,89,0,83,84,82,73,78,71,32,84,7
9,79,32,83,84,85,80,73,68,32,13,
0,78,79
120 DATA 32,73,32,87,73,76,76,32
,78,79,84,32,0,70,73,76,69,39,83
,32,65,32,78,79,78,79,0,89,79,85
,32,66,76,69,87,32,73,84,33,0,87
,72,73,67,72,32,79,78,69,63,63,3
2,32,0,84,72,65,84,32,84,73,67,7
5,76,69
130 DATA 83,0,87,72,69,82,69,32,
63,32,32,32,32,32,32,0,78,79,32,
87,65,89,32,76,79,86,69,82,32,0,
72,69,32,87,73,76,76,32,78,79,84
```

```
,32,84,65,76,75,32,84,79,32,77,6
9,13,0,72,69,39,83,32,84,79,79,3
2,66,79,83
140 DATA 89,32,32,32,13,0,68,79,
32,65,32,87,72,65,84,32,63,63,63
,32,32,32,32,32,0,67,65,78,39,84
,32,70,73,78,68,32,73,84,33,0,87
,72,65,78,68,32,87,72,69,82,69,3
2,63,63,32,32,32,13,0,68,73,83,7
5,32,83
150 DATA 65,73,68,32,78,79,32,32
,32,0,65,82,69,32,89,79,85,32,73
,78,32,76,79,86,69,32,63,32,32,1
3,0,84,72,65,84,83,32,83,73,76,7
6,89,32,32,67,65,78,32,89,79,85,
32,84,89,80,69,0,73,84,83,32,73,
78,32,65,32
160 DATA 66,65,68,32,77,79,79,68
,32,13,0,79,72,32,87,69,32,72,65
,86,69,32,84,87,73,78,83,33,33,3
2,13,0,83,80,73,76,76,69,68,32,7
3,84,32,33,32,32,0,73,32,84,72,7
3,78,75,32,73,32,65,77,73,78,32,
76,79,86
170 DATA 69,32,68,69,65,82,33,13
,0,79,79,79,80,83,33,0,73,32,67,
65,78,39,84,32,82,69,68,32,79,82
,32,82,73,84,69,13,32,73,39,86,6
9,32,72,65,68,32,73,84,33,33,0,1
25,57,125,75,125,82,125,104,125,
116,125,130
180 DATA 125,139,125,153,125,168
,125,182,125,203,125,218,125,233
,125,247,126,12,126,28,126,48,12
6,63,126,77,126,90,126,104,126,1
17,126,131,126,145,126,169,126,1
87,126,206,126,221,126,240,127,0
,127,21,127
190 DATA 34,127,47,127,67,127,88
,127,103,127,130,127,137,32,32,3
2,32,32,32,0,255
200 FOR A= 30672 TO 30767
210 READ B
220 POKE A,B
230 NEXT
240 EXEC 30672
250 DATA 142,1,103,204,126,126,1
67,128,51,140,49,239,129,167,128
,51,140,15,239,129,215,148,204,0
,200,52,6,48,140,2,126,174,90,15
,112,13,111,39,3,126,140,241,50,
98,52,20,214,148,231,159,0,136,1
89,161,193
260 DATA 39,251,126,161,185,140,
171,239,38,8,48,140,7,50,102,126
,172,121,126,130,115,71,79,79,68
,65,89,32,84,72,73,83,32,73,83,3
2,79,83,56,13,0
```

THAT OS 8

David Law is a Senior Lecturer in Rethesurmics (an OS 8 based series of studies in Philosphy), at the Sunbury COTAFE in Victoria.

His work shows the special skill that is rarely achieved by programmers in OS8. Having a philosophical basis for programming in OS8 provides one with a broader perspective when problems arise.

THE LISTING:

```
1 '*****THAT OS8*****************
2 GOTO10
3 SAVE"THAT OS8":DIR:STOP
4 '
5 ' ****   THAT **** OS8    ****
   ****                     ****
6 '****A BIT OF FOOLISHNESS****
   ****  PUT TOGETHER BY   ****
   ****                    ****
7 '****     DAVID LAW      ****
   ****     SUNBURY        ****
   ****     VICTORIA       ****
   ****                    ****
8 '****THE MAIN IDEA COMES ****
   ****FROM THE DRAWINGS OF****
   ****     M.C.ESCHER     ****
9 '
10 GOTO280
11 '!!!! ALL REM STATEMENTS !!!!
    !!!!   CAN BE OMITTED   !!!!
19 '****VARIOUS SUB-ROUTINES****
20 AS=RND(133)+122:X=USR0(AS):GO
SUB24:RETURN
21 FORDD=1TO50:IFINKEY$=""THENGO
SUB20:NEXTDD:RETURN
22 RETURN
23 FORDL=1TO300:NEXT
24 FORDL=1TO60:NEXT:RETURN
25 IFPEEK(1473)=106THENRETURNELS
EGOSUB23:PLAY"V31T602L8;6;11;03;
3;L5;6;P24;L7;3;L2;6":RETURN
29 '****CLEAR SPACE AND PUT ****
    ****FANCY BITS IN HIGH  ****
    **** MEMORY            ****
30 CLEAR200,32700:FORA=32739TO32
767:READB:POKEA,B:NEXTA
31 FORC=32700TO32735:READD:POKEC
,D:NEXTC
32 DEFUSR0=32739:DEFUSR1=32700
39 '**** ADAPTED FROM AN  ****
    **** ARTICLE IN THE   ****
    **** AUST. R'BOW OCT83****
    **** BY RAY GAUVREAU  ****
40 DATA 189,179,237,31,152,142,4
,0,16,142,4,31,237,137,1,224,237
,164,49,168,32,237,129,140,4,32,
38,240,57
49 '**** ADAPTED FROM AN  ****
    **** ARTICLE IN THE   ****
    **** AUST. R'BOW JAN84****
    ****BY D.S.LEWANDOWSKI****
50 DATA 142,4,0,52,4,95,90,39,2,
32,251,53,4,124,14,34,39,17,166,
132,129,96,39,1,74,167,128,140,5
,255,39,224,32,240,0,57
59 '****     TITLE PAGE     ****
60 CLS
61 PRINT@75,STRING$(10,":")
62 PRINT@107,":THAT OS8:"
63 PRINT@139,STRING$(10,":")
64 PRINT@354,"THERE IS NO NEED T
O PRESS       ANYTHING, BUT IF Y
OU REALLY     HAVE TO, JUST ABOU
T ANY KEY     WILL DO.";
65 GOSUB21
69 '****  STRINGS FOR OS 8  ****

70 C=RND(8)
71 CLSC
72 A1$=CHR$(128)+CHR$(128+16*(C-
1)+3)+CHR$(128+16*(C-1)+3)
73 A2$=A1$+CHR$(128)
74 A3$=A1$+CHR$(128+16*(C-1)+3)
75 A4$=CHR$(128)+CHR$(128+16*(C-
1)+15)+CHR$(128+16*(C-1)+15)+CHR
$(128)
76 A5$=CHR$(128)+CHR$(128+16*(C-
1)+12)+CHR$(128+16*(C-1)+12)
77 A6$=A5$+CHR$(128)
78 A7$=CHR$(128+16*(C-1)+12)+CHR
$(128+16*(C-1)+12)+CHR$(128+16*(
C-1)+12)
79 A8$=CHR$(128)+A7$
80 A9$=A7$+CHR$(128)
81 B$=CHR$(128+16*(C-1)+3)+CHR$(
128+16*(C-1)+3)+CHR$(128+16*(C-1
)+3)+CHR$(128)
89 '****     FIRST PAGE     ****
90 GOSUB20
91 PRINT@66,"'ENOUGH OF THAT COM
PUTER'she";
92 PRINT@98," said 'I NEED BOOK-
SHELVES!'";
93 PRINT@162,"'NO SWEAT' says i.
 THIS WAS ";
94 PRINT@194,"A GOOD EXCUSE TO I
NSTAL AND ";
95 PRINT@226,"TEST THOSE RECENTL
Y ARRIVED ";
96 GOSUB21
97 GOSUB98:GOTO104
98 GOSUB25
99 POKE1473,106
```

```
100 PRINT@296,A2$;:PRINT@302,A3$
;:PRINT@308,A2$;
101 PRINT@328,A4$;:PRINT@334,A8$
;:PRINT@340,A6$;
102 PRINT@360,A4$;:PRINT@366,B$;
:PRINT@372,A2$;
103 PRINT@392,A6$;:PRINT@398,A9$
;:PRINT@404,A6$;:RETURN
104 PRINT@460," chips ";
105 GOSUB21
109 '****    SECOND PAGE    ****
110 CLSRND(8)
111 GOSUB20
112 PRINT@68,"NO SOONER SAID THA
N DONE!";
113 PRINT@135," OUT WITH THE OLD
 ";
114 PRINT@167,"BLACK & DECKER,AN
D";
115 PRINT@199,"  INTO THE COCO.
 ";
116 PRINT@266,"NOW WITH THE ";:G
OSUB98
117 PRINT@421,"....CHIPS IN PLAC
E....";
118 GOSUB21
119 '****    THIRD PAGE     ****
120 CLSRND(8)
121 GOSUB20
122 PRINT@68,"I HAVE THEIR VAST
POWER";
123 PRINT@130,"(AND INBUILT LUBR
ICATION!!!)";
124 PRINT@201,"  TO HELP ME   ";
125 PRINT@233,"TO DRAW UP THE ";
126 PRINT@265,"NECESSARY PLANS";
127 PRINT@356,"AND SO.....
.....";
128 GOSUB21
129 '****    FOURTH PAGE    ****
    **** HI-RES GRAPHICS ON****
130 PMODE4:PCLS:SCREEN1,1:COLOR1
,0:GOSUB230
139 '**** SPEED UP AND DRAW ****
    **** ON HIDDEN HI-RES  ****
140 POKE65495,0
141 PMODE3,5:PCLS:COLOR2,1:GOSUB
230:GOSUB270
142 POKE65494,0
149 '****    FIFTH PAGE     ****
150 CLSRND(8)
151 PLAY"V1503T30L16D02GD01GT15D
#
152 GOSUB20
153 PRINT@66,"HMMM";
154 PRINT@70,STRING$(24,"!");
155 PRINT@98,"  PERHAPS THE CHIP
S NEEDED  ";
156 PRINT@130,"  MORE VINEGAR, A
ND MAYBE I ";
157 PRINT@162,"  SHOULD NOT HAVE
 USED THE  ";
158 PRINT@194,"big HAMMER TO MAK
E SURE THAT";
159 PRINT@226," THEY WERE FIRMLY
 IN PLACE!!";
160 PRINT@290,"HOWEVER, A BIT OF
 SAWING, A ";
161 PRINT@322,"BIT OF GLUEING, A
 LICK OF   ";
162 PRINT@354,"PAINT, AND.....
.....";
163 GOSUB21
164 PLAY"V20T402L8;6;11;03;3;L5;
6;P24;L7;3"
169 '****    SIXTH PAGE     ****
    **** HIDDEN HI-RES ON  ****
170 SCREEN1,0:PLAY"01;L2;3":GOSU
B21
179 '****    SEVENTH PAGE   ****
180 CLSRND(8)
181 P$="01L10002FFFFFFF01AAAAAA
A"
182 FORLD=1TO12
183 PLAY"XP$;"
184 GOSUB20
185 PRINT@67,"WELL, WHEN SHE SAW
 WHAT I ";
186 PRINT@108,"HAD MADE,";
187 FORP=1250TO1277:POKEP,33:NEX
TP
188 PRINT@237,"strong";
189 PRINT@355,"WORDS WERE USED,
AND ALL I";
190 PRINT@387,"COULD DO WAS.....
.....";
191 NEXTLD
199 '****    EIGHTH PAGE    ****
200 CLS0
201 GOSUB20
202 F$=CHR$(95):G$=F$+F$:H$=G$+F
$:I$=G$+G$:J$=G$+H$
203 K$=CHR$(223):L$=K$+K$:M$=K$+
L$:N$=L$+L$:O$=N$+K$
204 PRINT@69,J$+K$+J$+K$+I$+L$+J
$;
205 PRINT@101,F$+O$+F$+M$+F$+K$+
F$+L$+G$+K$+F$;
206 PRINT@133,H$+M$+J$+K$+F$+M$+
F$+K$+H$;
207 PRINT@165,F$+O$+F$+M$+F$+K$+
F$+L$+G$+K$+F$;
208 PRINT@197,F$+O$+F$+M$+F$+K$+
I$+L$+J$;
209 PRINT@293,J$+K$+F$+M$+F$+K$+
J$+K$+F$+M$+F$;
210 PRINT@325,F$+M$+F$+K$+F$+M$+
F$+K$+F$+M$+F$+K$+F$+M$+F$;
211 PRINT@357,J$+K$+F$+K$+F$+K$+
F$+K$+J$+L$+H$;
212 PRINT@389,F$+M$+F$+K$+F$+K$+
F$+K$+F$+K$+F$+M$+F$+M$+F$;
213 PRINT@421,F$+M$+F$+K$+J$+K$+
F$+M$+F$+M$+F$;
214 FORLD=1TO20:GOSUB20:NEXT
215 POKE1535,106:X=USR1(0)
216 GOSUB23:PRINT@235,"GOOD-BYE"
217 PLAY"V25":FORV=25TO1STEP-1:P
LAY"V-T9L8A":NEXTV
218 X=USR1(0)
219 GOTO219
229 '****   DRAWING PLANS   ****
230 LINE(0,146)-(45,138),PSET
231 LINE(69,132)-(94,128),PSET
232 LINE(153,118)-(163,117),PSET
233 LINE(199,109)-(206,107),PSET
234 LINE-(254,146),PSET
235 LINE-(49,185),PSET
236 LINE-(0,146),PSET
237 LINE-(0,152),PSET
238 LINE-(49,191),PSET
239 LINE-(255,152),PSET
240 LINE-(255,146),PSET
241 LINE(49,185)-(49,191),PSET:G
OSUB23
242 LINE(45,159)-(45,3),PSET
243 LINE-(61,0),PSET
244 LINE-(223,128),PSET
245 LINE-(223,138),PSET
246 LINE-(57,168),PSET
247 LINE(45,159)-(57,168),PSET:G
OSUB23
248 LINE-(57,104),PSET
249 LINE-(112,139),PSET
250 LINE(124,136)-(69,145),PSET
251 LINE(69,112)-(69,156),PSET
252 LINE-(134,144),PSET:GOSUB23
253 LINE-(134,76),PSET
254 LINE-(193,125),PSET
255 LINE-(153,132),PSET
256 LINE(153,107)-(153,141),PSET
257 LINE-(223,128),PSET
258 LINE(176,126)-(137,92),PSET:
GOSUB23
259 LINE-(87,100),PSET
260 LINE(137,144)-(57,91),PSET
261 LINE-(57,29),PSET
262 LINE-(105,69),PSET
263 LINE(45,3)-(120,66),PSET
264 LINE-(69,75),PSET:GOSUB23
265 LINE(134,76)-(57,91),PSET
266 LINE(69,39)-(69,89),PSET
267 LINE-(124,124),PSET
```

continued on P 15

EDUCATION PAGE

A project which has finally come to fruition is the CoCoConnection. This is the unit which allows one to connect CoCo to external devices.

Its release was held up because we redesigned it! Again!

We found a way to make the connections either inputs or outputs. So we now have 32 connections which are configured in the software to be inputs, or outputs. You can change them in mid program too!

One of the ways we think this unit will be used with CoCo in the classroom will be as a quiz master. You could connect 32 buttons to the CoCoConnection, and CoCo could be easily programmed to tell you which button was pushed first.

Another way the unit will be used is for the control of Physics and Chemistry experiments. CoCoConnection could measure the pH of a solution every x minutes for 48 hours and draw a graph of the result.

Or it could be used to simulate a number of real life situations, reacting to random inputs.

Robot control is easy, although Tandy's Armatron has proved to be a poor test bed - its movements are mechanically controlled, rather than electrically, so whilst in theory, with servo motors the job could be accomplished, there has to be a more suitable robot arm around - any suggestions?

We are very honoured to have the help of John Redmond and John Poxon on this project. These guys are Forth experts and have already demonstrated that Forth can handle the tasks typically required of a unit like CoCoConnection many times more efficiently than Basic.

Basic will do the job where port check speeds of less than 120 times per second are required - quite adequate for trains and many robots, but Forth will improve performance in other instances. John Poxon will be discussing both Forth and Turbo Pascal in relation to the unit, and we are sure you'll find it a most effective tool to have in the classroom.

Stars has taken off! As one teacher said to me recently, "if it stays where it is, it would be adequate, but the information is under constant review, and more articles are being added monthly - it just gets better!"

Next month there will be more news on Stars, so I wont pre-empt that, but I recommend you call on your Tandy shop and have a look at Stars soon!

If Stars has taken off, CoCo Max is in full flight. The big problem with CoCo Max is supply! Peter Collison just can't get sufficient units into the country - so order early and be patient - CoCo Max is worth the wait.

We were part of Tandy's display at the "Computers in Education" exhibition held in Brisbane, just before we went to press.

Six thousand people went through the turnstyles, enough to bounce a few theories off!

The exhibition demonstrated just how important support is becoming to new purchasers of computers. Once it was hard to convince folk on the basis of the number of Tandy outlets and the quality of CoCo; these days people seem to understand the need for support.

The easy bit is the purchase of the computer - when you own the thing - that's when you need help, and if there is no back up for you, then you'll waste your money.

More than one guy in the eastern states has been caught with a problem at 11.00PM and needed an answer before the morning. If he had a CoCo, he called the guys in the Users' Groups in W.A. who are still naturally awake at that time of night, and got an expert answer (because they are expert!). But I bet there isn't much that most other owners can do in like circumstances!

Back to the exhibition - we had the CoCo Max there with green and amber monitors. Upset the Apple people a bit, but they've had enough fun with their overpriced computers - its time the world knew about CoCo!

I'm off to Toowoomba and the New England / Darling Downs area this month. There is a rumour around about me also being in Tara (!) too - we'll see! If I get that far I'll be doing well!

I'm looking forward to meeting with the folk from each of these places. If you are part of one of these comunities and want to come along to one of the meetings, call the local Tandy store or myself for more details.

AREA and VOLUME

by Les and David Thurbon

We're combining the replies to two recent requests by teachers with this month's Education programs.

Les & David Thurbon have provided us with a very nice Area and Volume program, which draws the item you want to measure, and then calculates the area or volume as required. Relatively straight forward programming.

But they have combined with this, a new screen character set, and you need to carefully note the following instructions:

1. Extract the DOS if you have a disk system connected.

2. Type in listing 1 and save as a Machine Language program - if you do not know how to do this then consult the section on POKES later in this magazine.

3. Type in listing 2, CSAVE and RUN.

The A & V program loads the Machine Language program and this provides you with a new set of letters on the screen.

Next month David shows how to generate your own character set with another program in this series.

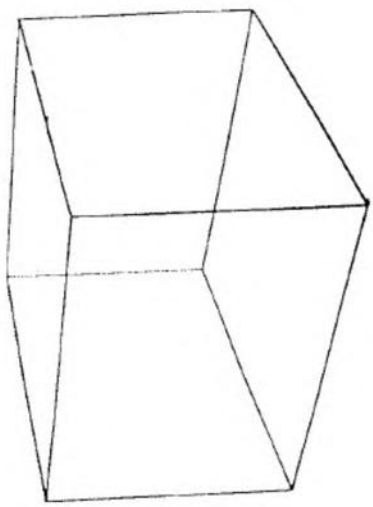
LISTING 1:

```
1 REM CLEAR MEMORY IF NECESSARY
2 FOR I= 17224TO 18243
3 READ Q$:POKEI,VAL("&H"+Q$):NEX
TI
4 EXEC 15364
 5DATA 8E,3C,B,BF,1,68,39,34,36,
81,8,10,27,0,8E,81,D,27,1C,C6,40
,10,8E,3C,D1,A1,A0,27,38,31,27,5
A,26,F7,C6,1F,A1,A0,27,13,31,29,
5A,26,F7,35,B6,DC,88,10,83,5,E0,
24,2,20,F4,20,75,8D,1E,C6,9,F7,3
C,8E,C6,39,F7,3C,9D,8D,34,C6,7
 6DATA F7,3C,8E,C6,A7,F7,3C,9D,2
0,D8,8D,4,8D,24,20,D2,DC,88,10,8
3,5,FF,27,46,34,4,44,56,54,54,54
,54,86,C,3D,86,20,3D,1F,1,35,4,C
4,1F,3A,30,89,6,0,39,C6,2,86,FF,
A7,84,30,88,20,5A,26,F8,C6,7,A6,
A0,A7,84,30,88,20,5A,26,F6,86
 7DATA FF,A7,84,A7,88,20,39,8D,B
C,30,1F,10,8E,3C,D2,8D,D6,20,84,
32,62,8E,1C,9F,8D,CD,8E,7,80,A6,
80,A7,89,FE,7F,8C,1D,FF,23,F5,8E
,1C,60,86,FF,A7,80,8C,1D,FF,23,F
9,16,FF,60,20,FF,FF,FF,FF,FF,FF,
FF,21,E7,E7,E7,E7,E7,FF,E7,22,D7
,D7,D7
 8DATA FF,FF,FF,FF,23,D7,D7,83,D
7,83,D7,D7,24,EF,C3,AF,C7,EB,87,
EF,25,9B,9B,F7,EF,DF,B3,B3,26,EF
,D7,D7,EF,D5,BB,C5,27,E7,E7,E7,F
F,FF,FF,FF,28,F7,EF,DF,DF,DF,EF,
F7,29,DF,EF,F7,F7,F7,EF,DF,2A,EF
,AB,C7,EF,C7,AB,EF,2B,FF,EF,EF,8
3,EF,EF
 9DATA FF,2C,FF,FF,FF,E7,E7,F7,E
F,2D,FF,FF,FF,81,FF,FF,FF,2E,FF,
FF,FF,FF,FF,E7,E7,2F,FB,FB,F7,EF
,DF,BF,BF,30,C7,BB,B3,AB,9B,BB,C
7,31,EF,CF,EF,EF,EF,EF,C7,32,C7,
BB,FB,F7,EF,DF,83,33,C7,BB,FB,E7
,FB,BB,C7,34,F7,E7,D7,83,F7,F7,F
7,35,83
 10DATA BF,87,FB,FB,BB,C7,36,F7,
EF,DF,87,BB,BB,C7,37,83,FB,F7,EF
,DF,BF,BF,38,C7,BB,BB,C7,BB,BB,C
7,39,C7,BB,BB,C3,F7,EF,DF,3A,FF,
E7,E7,FF,E7,E7,FF,3B,E7,E7,FF,E7
,E7,F7,EF,3C,FB,F7,EF,DF,EF,F7,F
B,3D,FF,FF,81,FF,81,FF,FF,3E,DF,
EF,F7,FB
 11DATA F7,EF,DF,3F,C7,BB,FB,F7,
EF,DF,EF,40,C7,BB,6D,55,63,BF,C3
,41,EF,D7,BB,BB,83,BB,BB,42,87,D
B,DB,C7,DB,DB,87,43,C7,BB,BF,BF,
BF,BB,C7,44,87,DB,DB,DB,DB,DB,87
,45,83,BF,BF,87,BF,BF,83,46,83,B
F,BF,87,BF,BF,BF,47,C7,BB,BF,B3,
BB,BB,C7
 12DATA 48,BB,BB,BB,83,BB,BB,BB,
49,C7,EF,EF,EF,EF,EF,C7,4A,C3,FB
,FB,FB,FB,BB,C7,4B,BB,B7,AF,9F,A
F,B7,BB,4C,BF,BF,BF,BF,BF,BF,83,
4D,BB,93,AB,BB,BB,BB,BB,4E,BB,9B
,AB,B3,BB,BB,BB,4F,C7,BB,BB,BB,B
B,BB,C7,50,B7,BB,BB,87,BF,BF,BF,
51,C7,BB
 13DATA BB,BB,AB,B7,CB,52,87,BB,
BB,87,AF,B7,BB,53,C7,BB,BF,C7,FB
,BB,C7,54,83,EF,EF,EF,EF,EF,EF,5
5,BB,BB,BB,BB,BB,BB,C7,56,BB,BB,
BB,D7,D7,EF,EF,57,BB,BB,BB,AB,AB
,93,BB,58,BB,BB,D7,EF,D7,BB,BB,5
9,BB,BB,D7,EF,EF,EF,EF,5A,83,FB,
F7,EF,DF
 14DATA BF,83,5B,FF,FF,FF,81,DB,
DB,DB,5C,C3,BD,66,6E,66,BD,C3,5D
,FF,FF,C7,BB,BB,D7,93,5E,EF,C7,A
B,EF,EF,EF,EF,5F,E7,DF,EF,D7,EF,
F7,CF,60,EF,D7,EF,83,6D,AD,EF,D7
,BB,61,FF,FF,CF,F7,C7,B7,CB,FF,F
F,62,BF,BF,87,BB,BB,BB,87,FF,FF,
63,FF,FF
 15DATA C7,BB,BF,BB,C7,FF,FF,64,
FB,FB,C3,BB,BB,BB,C3,FF,FF,65,FF
,FF,C7,BB,83,BF,C7,FF,FF,66,F7,E
B,EF,C7,EF,EF,EF,FF,FF,67,FF,FF,
C7,BB,BB,C3,FB,BB,C7,68,BF,BF,A7
,9B,BB,BB,BB,FF,FF,69,EF,FF,CF,E
F,EF,EF,C7,FF,FF,6A,FB,FF,F3,FB,
FB,FB,FB
 16DATA BB,C7,6B,BF,BF,B7,AF,9F,
AF,B7,FF,FF,6C,CF,EF,EF,EF,EF,EF
,C7,FF,FF,6D,FF,FF,D7,AB,AB,BB,B
B,FF,FF,6E,FF,FF,A7,9B,BB,BB,BB,
FF,FF,6F,FF,FF,C7,BB,BB,B8,C7,FF
,FF,70,FF,FF,A7,9B,BB,BB,87,BF,B
F,71,FF,FF,CB,B3,BB,BB,C3,FB,FB,
72,FF,FF
 17DATA A7,9B,BF,BF,BF,FF,FF,73,
FF,FF,C7,BF,C7,FB,C7,FF,FF,74,EF
,EF,C7,EF,EF,EB,F7,FF,FF,75,FF,F
F,BB,BB,BB,B3,C8,FF,FF,76,FF,FF,
BB,BB,BB,D7,EF,FF,FF,77,FF,FF,BB
,BB,AB,AB,D7,FF,FF,78,FF,FF,BB,D
7,EF,D7,BB,FF,FF,79,FF,FF,BB,BB,
BB,C3,FB
 18DATA BB,C7,7A,FF,FF,83,F7,EF,
DF,83,FF,FF,7B,0,FF,0,FF,0,FF,0,
FF,0,7C,0,FF,0,FF,0,FF,0,FF,0,7D
,0,FF,0,FF,0,FF,0,FF,0,7E,0,FF
```

Listing 2:

```
1 ' *******AREA AND VOLUME******
*   ********PIXEL SOFTWARE******
*
2 GOTO10
3 SAVE"A&V":DIR:STOP
10 CLEAR200,&H3C03:PCLEAR4:PMODE
4:PCLS5:DRAW"C0":COLOR0,5
20 PRINT@416,"READING MACHINE LA
NGUAGE PART..."
30 LOADM"CHRSET":EXEC
40 SCREEN1,1:PCLS0:FORX=0TO191ST
EP2:LINE(X,C)-(255-X,191-C),PRES
ET,B:C=C+2:NEXT:LINE(0,96)-(255,
96),PSET:PAINT(128,96),5,5
50 DRAW"BM20,86ND20R17F3D4G3L17B
D10BR50NR5R3U20NL2R3BR100NR20D10
NR15D10R20BR40NU20R20BM128,96NE1
7NF17NG17NH17":PRINT@461,"Softwa
re":PRINT@493,"Presents";
60 A$=INKEY$:IFA$=""THEN60
70 PCLS:CLS:PRINT@235,"Area & Vo
lume":PRINT@262,"Designed by L.
Thurbon":PRINT@293,"Programmed b
y D. Thurbon":PRINTTAB(9)"\ Copy
right 1985"
80 A$=INKEY$:IFA$=""THEN80
90 PCLS5:CLS:INPUT"Do you want a
rea or volume (A/V)";A$
100 IFA$="V"THEN690
110 PCLS5:CLS:PRINT"1:Triangle
    2:Square        3:Rectangle
    4:Parallelogram 5:Trapezoid
    6:Trapezium     7:Pentagon
    8:Hexagon"
120 PRINT"9:Octagon        10:Cir
cle        11:Segment      12:Sec
tor        13:Ring         14:Ell
ipse       15:Sphere       16:Cub
e          17:Rect. prism  18:Con
e"
130 PRINT"19:Cylinder      20:Tor
us"
140 PRINT:INPUT"Enter your selec
tion";A
150 ON A GOTO220,250,270,290,310
,330,370,400,430,450,470,510,530
```

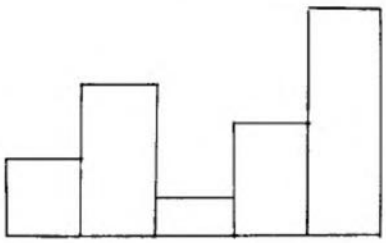
```
,550,570,580,600,630,650,670
160 GOTO110
170 PRINT@416,"Enter the value f
or '"+A$+"'";:RETURN
180 PRINT@416,STRING$(63,32):RET
URN
190 PCLS5:CLS:PRINT@16-(LEN(TI$)
/2),TI$:RETURN
200 PCLS5:CLS:PRINT@257,"The are
a of this "TI$" is":PRINTA"units
"
210 A$=INKEY$:IFA$=""THEN210ELSE
90
220 TI$="Triangle":GOSUB190:DRAW
"BM50,120M+150,0M-75,-75M-75,+75
BM50,120BL5L5R3U75NL3NR75BM50,12
0BD3D5U3R150ND3U3":PRINT@228,"h"
:PRINT@368,"b"
230 A$="b":GOSUB170:INPUTB:A$="h
":GOSUB180:GOSUB170:INPUTH
240 A=(B*H)/2:GOTO200
250 TI$="Square":GOSUB190:DRAW"B
M90,120R75U75L75D75BD3D3ND3R75NU
3D3":PRINT@368,"b"
260 A$="b":GOSUB170:INPUTB:A=B^2
:GOTO200
270 TI$="Rectangle":GOSUB190:DRA
W"BM90,120R75U50L75D50BD3D3ND3R7
5ND3U3BU3BR3NR6R3U50NR3L3":PRINT
@368,"b":PRINT@246,"a"
280 A$="a":GOSUB170:INPUTA1:A$="
b":GOSUB180:GOSUB170:INPUTB:A=A1
*B:GOSUB200
290 TI$="Parallelogram":GOSUB190
:DRAW"BM90,88NE40R50E40NL50BR3NR
6R3D40NR3L40BL6BD3ND6D3L50NU3D3"
:PRINT@269,"b":PRINT@184,"h"
300 A$="b":GOSUB170:INPUTB:A$="h
":GOSUB180:GOSUB170:INPUTH:A=B*H
:GOTO200
310 TI$="Trapezoid":GOSUB190:DRA
W"BM90,88M+10,-50R50M+10,+50M90,
88BD3ND6D3R70ND3U3BU3BR3L1R4L3U5
0NR3L10BL3BU3NU6U3L50NU3D3":PRIN
T@79,"a":PRINT@181,"h":PRINT@271
,"b"
320 A$="a":GOSUB170:INPUTA1:A$="
b":GOSUB180:GOSUB170:INPUTB:A$="
h":GOSUB180:GOSUB170:INPUTH:A=H/
2*(A1+B):GOTO200
330 TI$="Trapezium":GOSUB190:DRA
W"BM90,90NR70M+10,-30M+30,-10M+3
0,+40BR3NR6R3U40NR3L27BM90,90BL3
L6R3U30NL3R7BM90,90BD3D6U3R10NU3
ND3R30NU3ND3R30D3U6"
340 DRAW"BM100,90U3BU3U3BU3U3BU3
U3BU3U3BM130,90U3BU3U3BU3U3BU3U3
BU3U3BU3U3BU3U3":PRINT@201,"h":P
RINT@182,"H":PRINT@299,"a  b  c
"
350 A$="h":GOSUB170:INPUTH:GOSUB
180:A$="H":GOSUB170:INPUTH1:GOSU
B180:A$="a":GOSUB170:INPUTA1:GOS
UB180:A$="b":GOSUB170:INPUTB:GOS
UB180:A$="c":GOSUB170:INPUTC
360 A=.5*(B*(H1+H)+A1*H+C*H1):GO
TO200
370 TI$="Pentagon":GOSUB190:LINE
(128,96)-(128,146),PRESET:FORT=0
TO5:A=(2*3.1415926535)*T/5:LINE-
(50*SIN(A)+128,50*COS(A)+96),PSE
T:NEXT
380 DRAW"BM99,51U6D3R58U3D6":PRI
NT@111,"b"
390 A$="b":GOSUB170:INPUTB:A=1.7
2*B^2:GOTO200
400 TI$="Hexagon":GOSUB190:LINE(
128,96)-(128,146),PRESET:FORT=0T
O6:A=(2*3.1415926535)*T/6:LINE-(
50*SIN(A)+128,50*COS(A)+96),PSET
:NEXT
410 DRAW"BM176,71R6L3D50L3R6":PR
INT@247,"b"
```



```
420 A$="b":GOSUB170:INPUTB:A=2.5
98*B^2:GOTO200
430 TI$="Octagon":GOSUB190:DRAW"
BM100,130R43E25U43H25L43G25D43F2
5BD3D6U3R43U3D6":PRINT@399,"b"
440 A$="b":GOSUB170:INPUTB:A=4.8
28*B^2:GOTO200
450 TI$="Circle":GOSUB190:CIRCLE
(128,96),50:DRAW"BM128,96NU3ND3R
50":PRINT@243,"r"
460 A$="r":GOSUB170:INPUTR:A=3.1
415926535*R^2:GOTO200
470 TI$="Segment":GOSUB190:CIRCL
E(128,96),50,0,1,.54,.97:DRAW"BM
78,87R100M128,96M78,87":CIRCLE(1
28,96),30,0,1,.54,.97
480 PRINT@176,"]":DRAW"BM78,87BU
3U50D3R100U3D50BM78,87BL3L6R3U41
L3R40":PRINT@168,"h":PRINT@80,"C
"
490 A$=CHR$(8)+"the angle  ']":
GOSUB170:INPUTCA:GOSUB180:A$="C"
:GOSUB170:INPUTC:GOSUB180:A$="h"
:GOSUB170:INPUTH:GOSUB180:A$=CHR
$(8)+"the radius":GOSUB170:PRINT
CHR$(8):INPUTR
500 A=3.1415926535*R^2*(CA/360)-
(C*(R-H)/2):GOTO200
510 TI$="Sector":GOSUB190:CIRCLE
(128,96),50,0,1,.625,.875:DRAW"B
M128,96H35BH3H4BM128,96E35BE3E3B
M128,96BF3F6H3E36NH3F4":CIRCLE(1
28,96),56,0,1,.625,.875
520 PRINT@79,"b":PRINT@243,"R":A
$="b":GOSUB170:INPUTB:GOSUB180:A
$="R":GOSUB170:INPUTR:A=(B*R)/2:
GOTO200
530 TI$="Ring":GOSUB190:CIRCLE(1
28,96),30:CIRCLE(128,96),50:DRAW
"BM128,96NU3ND3NL50NR30":PRINT@2
42,"r":PRINT@235,"R"
540 A$="r":GOSUB170:INPUTR:GOSUB
180:A$="R":GOSUB170:INPUTR1:A=3.
1415926535*(R1^2-R^2):GOTO200
550 TI$="Ellipse":GOSUB190:CIRCL
E(128,96),70,0,.6:DRAW"BM128,96N
R70NU41NL3ND3":PRINT@244,"b":PRI
NT@209,"c"
560 A$="b":GOSUB170:INPUTB:GOSUB
180:A$="c":GOSUB170:INPUTC:A=3.1
415926535*B*C:GOTO200
570 TI$="Sphere":GOSUB190:CIRCLE
(128,96),50:DRAW"BM128,96R50":PR
INT@244,"R":A$="R":GOSUB170:INPU
TR:A=4*3.1415926535*R^2:GOTO200
580 TI$="Cube":GOSUB190:DRAW"BM9
0,130R50U50L50ND50E25R50NG25D50G
25BD3D6U3L50U3D6":PRINT@366,"b"
590 A$="b":GOSUB170:INPUTB:A=6*B
^2:GOTO200
600 TI$="Rect. box":GOSUB190:DRA
W"BM70,130R70U40L70ND40E60R70NG6
0D40NG60BR3R6L3G60R3L6BL3BD3D6U3
L70D3U6BU3BL3L6R3U40L3R6"
610 PRINT@295,"a":PRINT@397,"b":
PRINT@279,"c"
620 A$="a":GOSUB170:INPUTA1:GOSU
B180:A$="b":GOSUB170:INPUTB:GOSU
B180:A$="c":GOSUB170:INPUTC:A=2*
(A1*B+B*C+A1*C):GOTO200
630 TI$="Cone":GOSUB190:CIRCLE(1
28,96),20,0,2:DRAW"BM128,56M218,
96M128,136BD3ND6D3R90U3D6BM221,9
6NR6R3U40R3L90"
640 PRINT@220,"r":PRINT@405,"h":
A$="r":GOSUB170:INPUTR:GOSUB180:
A$="h":GOSUB170:INPUTH:A=3.14159
26535*R*SQR(R^2+H^2):GOTO200
```

continued on P 15

64K



EXAM GRAPH

by David Law

I wrote this program to allow my two children to draw graphs of their school exam marks. I hoped it would give them a pictorial idea of which subjects needed more effort.

I realise that there are lots of bar-graph programs available, but I wanted to do my own, and maybe it will encourage others to try things for themselves.

Strictly speaking, this is not all my own work, as I have applied a lot of things that I have picked up whilst reading Australian Rainbow and Australian CoCo.

I have an old grey case, upgraded to a 64K, cassette and printer, and I rely on the magazines as my major sources of information, as I am too far from the nearest user group.

THE LISTING:

```
1 '****GRAPH OF EXAMINATION****
   ****      RESULTS          ****
   ****    BY DAVID LAW       ****
2 GOTO10
3 SAVE"EXAMGRAF":DIR:STOP
10 CLEAR370:PCLS:DIM N$(45):Y1=1
65
11 '****NUMBER STRINGS 0-9****
12 N$(1)="U6R3D6L3BR7"' 0 A.R JU
N'83 P.9
13 N$(2)="BU6BR2D6BR4"' 1
14 N$(3)="BU6R4D3L4D3R4BR4"'2
15 N$(4)="BU6R4D3NL2D3L4BR8"'3
16 N$(5)="BU2UE3RD4NL4D2BR4"'4
17 N$(6)="R4U3L4U3R4BD6BR4"'5
18 N$(7)="NU6R4U3NL4D3BR4"'6
19 N$(8)="BU6R4D6BR4"'7
20 N$(9)="U6R4D3NL4D3NL4BR4"'8
21 N$(10)="BR4U6L4D3R4D3BR4"'9
22 '****  SYMBOL STRINGS  ****
23 N$(11)="BU3BR2DBDDBR2"':
24 N$(12)="BU3BR2DBDDLBR3"';
25 N$(13)="BR2H2E2BD4BR3"'<
26 N$(14)="BR2BU2R8BD2BR2"'-
27 N$(15)="E2H2BD4BR5"'>
28 N$(16)="BU3BRUERFDGBDDBR2"'?
29 N$(17)="BRHU3E2R4F2D4L5HU2ER2
FD3BR5"'@
30 '**** ALPHABET STRINGS ****
31 N$(18)="BRU6R4D4NL4D2BR2"'A
32 N$(19)="BR2U6R4D2GNL3FD2L3BR5
"'B
33 N$(20)="BR2NR4U6R4BD6BR3"'C
34 N$(21)="BR2U6R2F2D2G2NL2BR4"'
D
35 N$(22)="BR2NR4U3NR3U3R4BD6BR2
"'E
36 N$(23)="BR2U3NR3U3R4BD6BR2"'F
37 N$(24)="BR3U6R4BD4NL2D2NL4BR3
"'G
38 N$(25)="BR2U3NR4U3BR4D6BR2"'H
39 N$(26)="BR3NU6BR3"'I
40 N$(27)="BR2R3U6NL3R3BD6BR3"'J
41 N$(28)="BR2U6BR4G3NLF3BR3"'K
42 N$(29)="BR2NU6R4BR2"'L
43 N$(30)="BR2U6F3E3D6BR2"'M
44 N$(31)="BR2U6F4U4D6BR4"'N
45 N$(32)="BR2U6R4D6NL4BR3"'O
46 N$(33)="BR2U6R4D3NL3BD3BR3"'P
47 N$(34)="BR2U6R4D6NL4NH2NF2BR5
"'Q
48 N$(35)="BR2U6R4D3L3F3BR3"'R
49 N$(36)="BR2R4U3L4U3R4BD6BR2"'
S
50 N$(37)="BR3U6NL3R3BD6BR"'T
51 N$(38)="BR2NU6R4NU6BR2"'U
52 N$(39)="BR2BU6D4F2E2U4BD6BR2"
'V
53 N$(40)="BR2BU6D4F2E2NU2F2E2U4
BD6BR2"'W
54 N$(41)="BR2UE2H2UBR4DG2F2DBR2
"'X
55 N$(42)="BR4U3L2U3BR4D3NL2BD3B
R2"'Y
56 N$(43)="BR2NR4UE4UNL4BD6BR2"'
Z
57 N$(44)="BR2E6BL6F6BL3U3NL3NR3
U3BR5BD6"'*
58 N$(45)="BRBU4NU2BD4BR"''
59 U$=CHR$(63)+CHR$(63)+CHR$(63)
60 GOSUB132:GOTO91
61 '**** CHANGE KEYBOARD  ****
    **** INPUT TO GRAPHICS****
62 PMODE3:PCLS
63 FORZ=1TOLEN(Z$):A=ASC(MID$(Z$
,Z,1))-47:IFA>43ORA<1THENDRAW"BR
4":NEXTELSE DRAW N$(A):NEXT:RETU
RN
64 '**** DRAW GRAPH AND  ****
    ****NUMBERS DOWN SIDE****
65 DRAW"S4BM40,15NL2D15NL2D15NL2
D15NL2D15NL2D15NL2D15NL2D15NL2D1
5NL2D15NL2D15R215"
66 DRAW"C8S8BM2,22;XN$(2);XN$(1)
```

```
;XN$(1);BM8,37;XN$(10);XN$(1);BM
8,52;XN$(9);XN$(1);BM8,67;XN$(8)
;XN$(1);BM8,82;XN$(7);XN$(1);BM8
,97;XN$(6);XN$(1);BM8,112;XN$(5)
;XN$(1);BM8,127;XN$(4);XN$(1);BM
8,142;XN$(3);XN$(1);BM8,157;XN$(
2);XN$(1);":RETURN
67 '**** DRAW GRAPH FOR  ****
    **** ALL SUBJECTS    ****
68 COLOR7,5:X=X+11:Y2=Y1-(VAL(A$
)*150)/100:FORX=X TO X+10:LINE(X
,Y2)-(X,Y1),PSET:NEXTX:RETURN
69 '****DRAW GRAPH FOR   ****
    ****SPECIFIC SUBJECTS****
70 COLOR7,5:X=X+4:Y2=Y1-(VAL(A$)
*150)/100:FORX=X TO X+2:LINE(X,Y
1)-(X,Y2),PSET:NEXTX:RETURN
71 '****PRINT YEARS AND  ****
    ****    TERMS        ****
72 PRINT"YEAR :-";:RETURN
73 PRINT"TERM :-";:GOSUB83:RETUR
N
74 PRINT"        FIRST ";:RETURN
75 PRINT"       SECOND ";:RETURN
76 PRINT"        THIRD ";:RETURN
77 PRINT"       FOURTH ";:RETURN
78 PRINT"        FIFTH ";:RETURN
79 PRINT"        SIXTH ";:RETURN
80 PRINT:PRINT"MARKS FOR "Z$" >>
>>":GOSUB74:GOSUB73:GOSUB75:GOSU
B73:GOSUB76:GOSUB73:GOSUB77:GOSU
B73
81 CLS:PRINT@35,"'"X$"'":PRINT:P
RINT"   (LAST YEAR WAS "Z$")":PR
INT:RETURN
82 '**** INPUT OF MARKS  ****
83 LINEINPUTA$:IFA$="0"THENRETUR
NELSEIFVAL(A$)<10RVAL(A$)>100THE
NPRINT:GOSUB86:PRINTTAB(18)":-";
:GOTO83ELSEIFC$="1"THENSOUND VAL
(A$)*2,1:RETURNELSESOUND VAL(A$)
*2,1:GOSUB70:RETURN
84 '**** ERROR MESSAGES  ****
85 PRINT"  subject"CHR$(128)"nam
e"CHR$(128)"must"CHR$(128)"be"CH
R$(128)"at"CHR$(128)"least   thr
ee"CHR$(128)"letters"CHR$(128)"l
ong":GOSUB88:RETURN
86 PRINT"marks"CHR$(128)"must"CH
R$(128)"be"CHR$(128)"between"CHR
$(128)"0"CHR$(128)"and"CHR$(128)
"100"CHR$(128):GOSUB88:RETURN
87 PRINT"   please"CHR$(128)"use
"CHR$(128)"four"CHR$(128)"number
s":GOSUB88:FORDL=1TO500:NEXT:IFX
=45GOTO113ELSEIFX=80GOTO114ELSEI
FX=115GOTO115ELSEIFX=150GOTO116E
LSEIFX=185GOTO117ELSEIFX=220GOTO
118
88 FOR ER=1TO3:PLAY"L20002FFFFFF
F01AAAAAAA":NEXT:RETURN
89 CLS:PRINT" GOOD-BYE":END
90 '****  SCREEN INPUTS  ****
91 CLS:PRINT@40,"***MARKS GRAPH*
**":PRINT@102,"PLEASE TYPE YOUR
NAME":PRINT:LINEINPUT"   :- ";ID
$:PLAY"L10002AGAG":IFLEN(ID$)<1T
HENID$=CHR$(63):Z$=ID$ELSE Z$=ID
$:DRAW"C6S4BM100,14;":GOSUB62:GO
SUB65
92 PRINT:PRINT@33,ID$","PRINT"
YOU CAN DRAW A GRAPH SHOWING
RELATIVE MARKS IN :-":PRINT:GOTO
94
93 PRINT:PRINT@33,ID$","PRINT"
WHICH ONE WOULD YOU LIKE TO DO
NOW :-":PRINT:PRINT@353,"or":POK
E1379,44:PRINT:PRINT" 3) END";
94 PRINT@160," 1) ALL SUBJECTS,
FOR ANY ONE       TERM OR YEAR":
PRINT" or":POKE1251,44
95 PRINT:PRINT" 2) ANY ONE SUBJE
CT FOR UP TO 6     YEARS (4 TERM
S/YEAR)":IFPEEK(1379)=44THENPRIN
T@449,"ENTER (1) OR (2) OR (3) "
;:INPUTC$:ELSEPRINT@449,"ENTER (
1) OR (2) ";:INPUTC$
96 PLAY"L10004AGAG":IFC$="1"ORC$
="2"ORC$="3"THEN97ELSECLS:GOTO93
97 IF C$="1"THENZ$=ID$:DRAW"C6S4
BM60,14;":GOSUB62:GOSUB65ELSEIFC
$="3"THEN89ELSE110
98 '**** ROUTINE FOR ALL ****
    ****    SUBJECTS     ****
99 CLS:PRINT:PRINT@7,"***ALL SUB
JECTS***":PRINT:PRINT"  ENTER TH
E FIRST SUBJECT NAME    (AN ABBR
EVIATION WILL BE        GIVEN FO
R  IT).":PRINT:PRINT"  THEN ENTE
R THE PERCENTAGE        MARKS FOR
 THE SUBJECT.":PRINT
100 PRINT"  CARRY ON IN  THE SAM
E WAY FOR   THE NEXT SUBJECT.":P
RINT:PRINT"  ^ AND <ENTER> AT TH
E SAME TIME  WILL DRAW GRAPH IF
THERE ARE    LESS THAN NINE SUBJ
ECTS.":PRINTTAB(11)"<ENTER>";
101 DRAW"S4C6BM50,173;":GOSUB127
102 X=39:FORSJ=1TO9
103 PRINT:PRINTTAB(3)"SUBJECT NA
ME"SJ":-";:LINEINPUTX$:IFX$="^"T
HENPLAY"L200CDEFGAB":Z$=U$:GOTO1
08ELSEIFLEN(X$)<3ORVAL(X$)>0THEN
GOSUB85:GOTO103ELSEPLAY"L3503CDC
DC":X$=LEFT$(X$,3):PRINT"   MARK
S FOR("X$") :-";
104 Z$=LEFT$(X$,1):GOSUB63:DRAW"
BM-4,+9;"
```

continued on P 27

We mean business...

We have started to use Dynacalc here for stock recording, and I must say it is very easy to get to know. There are help files at your disposal all the time, and things like cell formulas go in simply and quickly.

Dynacalc has a 51 column screen like Telewriter 64, and it has some neat features like the ability to 'scroll protect' whatever size top and left hand margins you wish, instant recalculation of the sheet upon each data entry, and logical movement around the sheet.

During the conference, a user group was started for users of Pro Color, Dynacalc and Telewriter 64. Yet another reason to consider very carefully the purchase of these products.

Business users not using 51 column screens may not be aware that such things are available commercially, and have been for some time. The V.I.P. range of products use an internal one also. But products like Rainbow Writer, Screen Machine and others (all available from Software Spectrum and their agents) provide 51 columns for programs you write yourself.

This change of screen format can be very handy for invoice input, and for some file reporting programs.

FISHING

by Gordon Levis

Gordon Levis sent a program he calls 'Fishing'. Would you therefore be surprised to learn that Gordon is a fisherman? He is one of the lucky ones who fish for those beautiful W.A. crays!

Gordon's program illustrates what I have always said. If you can program, you do not need commercial spreadsheets or data bases, because you can tailor something to meet your special needs.

I really wanted to print this program this month, but I can't justify the space - it takes up 10 granules on a disk! (The program is designed for a 32K tape based system.)

"Fishing" will be included on RAINBOW on TAPE this month, and if you are involved in the business use of CoCos then I suggest you look at the very simple way Gordon has approached the task he had for his computer. Much of what he has done can be adapted for other businesses too.

To give you some guide values to allow you to see the program do its thing, Gordon has provided some additional information.

You will first be asked what weight of cray fish you have caught over a seven day period. A bag of crays weigh about 44 kilos. You are asked for the current price, presently $13.50 a kilo.

The next inputs relate to manpower. The skipper and crew usually work on a percentage of the gross value of the catch. Typical percentages for crew members are 25%, 11% and 8% as you go down the pecking order.

Next is bait and fuel. Salmon heads cost about $20.00 a box, other types of fish heads cost $15.00 a box, whilst hides and hocks are $10.00 a bag.

Diesel is $0.36 per litre and petrol is $0.54 per litre.

Then there are electrical and mechanical costs for three conditions. The first is for on board costs, the second is for utility vehicles used to haul gear etc, and the third is for miscellaneous costs.

You then get a screen print for profit and loss on each account. Adjustments can be made, and then a print out can be obtained.

As you can see, the conditions of Gordon's work call for a special program, but the principles are similar to many businesses.

This column is about the programs we use in our day to day business lives. If like Gordon, you have a program you use in your business, and you would like to share it, we'd love to include it in this part of the magazine.

TAPE DATABASE

by Sam Robinson

(Sam is a teacher at the Correspondance School in Brisbane. We have appreciated his friendship over the last year. G.)

This DATABASE program has its origins in the fact that a customized database was needed to file brief anecdotal details on a continuing basis and to be able to retrieve this data to either screen or printer (the simple TP-10 in my case).

The program is menu driven and fairly self-explanitory. It first asks for the DATE of inputting of data, then for the COMMENT, which includes a simple wrap-around technique straight from the Tandy manual, and finally a reminder prompt to end with the initials of the person inputting the information.

Data can be SAVEd to tape and LOADED from tape in the usual way. The PRINT command is split over two sections - PRINT TO SCREEN and PRINT TO PRINTER. On selection of either, you are asked whether you wish All the files or just One, if you select ONE you are then prompted for the file number that you wish to have.

I hope it can be of some use to some of your readers in the same way that I have gained all my limited knowledge from the works of other readers.

May I, in conclusion, pass on my sincere thanks to the many talented contributors of your magazine. Without the help of people like the Delbourgos, Les Thurbon, Dean Hodgson and hundreds of others, many more of us would never have got off the starting blocks.

THE LISTING:

```
1 '*****A DATABASE***************
   *****BY SAM ROBINSON*********
   *****17/6/85*****************
2 GOTO10
3 SAVE"DBASE":DIR:STOP
10 CLS(0)
15 PRINT@0,"NAME OF ORGANIZATION
 GOES HERE."
20 PRINT@128,"ALWAYS ENTER DATE
(**/**/**)":PRINT@160,"AT BEGINN
ING OF EVERY COMMENT.":PRINT@224
,"ALWAYS END WITH INITIALS.":PRI
NT@256,"(SER) ETC."
25 PRINT@448,'  PRESS (ENTER) T
O CONTINUE  '
30 A$=INKEY$:IF A$="" THEN 30
35 CLEAR 3000: DIM S$(20)
40 REM     COMMENTS FOLLOW
45 CLS0
50 PRINT @ 0,"NAME OF ORGANIZATI
ON GOES HERE."
55 PRINT @ 67,"DO YOU WANT TO--"
;
60 PRINT @ 131,"(1) START A NEW
FILE"
65 PRINT @ 163,"(2) REPLACE A CO
MMENT"
70 PRINT @ 195,"(3) ADD TO EXIST
ING FILE"
75 PRINT @ 227,"(4) DELETE A COM
MENT"
80 PRINT @ 259,"(5) PRINT COMMEN
TS ON SCREEN"
85 PRINT @ 291,"(6) PRINT COMMEN
TS TO PRINTER";
90 PRINT @ 323,"(7) SAVE COMMENT
S TO TAPE"
95 PRINT@355,"(8) LOAD COMMENTS
FROM TAPE"
100 PRINT @ 452,"*PLEASE MAKE YO
UR SELECTION";
105 PRINT@495,"(1-8)";
110 L$=CHR$(13)
115 A$=INKEY$: IF A$="" THEN 115
120 M=VAL(A$)
125 IF M<1 OR M>8 THEN 115
130 CLS
135 ON M GOSUB 150, 215, 160, 27
0, 335, 610, 485, 545
140 GOTO 40
145 REM
150 REM    INPUT/ADD COMMENTS
155 Y=1
160 CLS: PRINT @ 7,"INPUT/ADD CO
MMENTS"
165 PRINT @ 34, "PRESS (ENTER) W
HEN FINISHED"
170 PRINT: PRINT "DATE(**/**/**)
";:INPUT D$(Y)
175 PRINT"COMMENT"Y:PRINT"**MUST
 BE SHORTER THAN 8 LINES**"
180 GOSUB 630
185 PRINT:PRINT"INITIALS (***)";
:INPUT I$(Y)
190 CLS
195 IF D$(Y)="" THEN RETURN
200 Y=Y+1
205 GOTO 170
210 REM
215 REM     REPLACE COMMENTS
220 N=0
225 CLS: PRINT @ 8, "REPLACE COM
MENTS"
230 PRINT @ 34,"PRESS (ENTER) WH
EN FINISHED"
235 PRINT: INPUT "COMMENT NO. TO
 REPLACE";N
240 IF N=0 THEN RETURN ELSE 245
245 PRINT"REPLACEMENT COMMENT:":
PRINT"**MUST BE SHORTER THAN 8 L
INES**":GOSUB 675
250 PRINT
255 GOTO 215
260 RETURN
265 REM
270 REM     DELETE COMMENTS
275 CLS: PRINT @ 8,"DELETE COMME
NT"
280 PRINT @ 34,"PRESS (ENTER) WH
EN FINISHED"
285 PRINT:INPUT "COMMENT TO DELE
TE";N
290 IF N=<0 THEN RETURN
295 IF N> Y-1 THEN 285
300 FOR X=N TO Y-2
305 D$(X) = D$(X+1):S$(X) = S$(X
+1):I$(X) = I$(X+1)
310 NEXT X
315 D$(X) = "":S$(X) = "":I$(X)
=""
320 Y=Y-1
325 GOTO 270
330 REM
335 REM      PRINT COMMENTS
340 IF M=5 THEN P=0
345 IF M=6 THEN P=2
350 CLS0:PRINT@193,"WOULD YOU LI
KE TO PRINT -";:PRINT@259," aLL
COMMENTS OR oNE COMMENT?";
355 A$=INKEY$:IF A$="" THEN 355
360 IF A$()"A" AND A$()"O" THEN
355
365 IF A$="A" THEN 385
370 IF A$="O" THEN PRINT@352,"PR
INT WHICH COMMENT?";
375 B$=INKEY$:IF B$="" THEN 375
380 X=VAL(B$)
385 CLS:IF P=2 THEN 390 ELSE 400
390 CLS0:PRINT@236,"PRINTING NOW
";
395 IF P=2 THEN GOSUB 615
400 IF A$="O" THEN 410
405 FOR X=1 TO Y-1 STEP 1
410 IF P=0 THEN GOSUB 615
415 PRINT#-P,X".   ";D$(X);L$;S$
(X);L$;I$(X);L$;STRING$(32,"-")
420 IF P=2 THEN GOTO 440 ELSE IF
 P=0 THEN GOTO 425
425 PRINT"  PRESS (ENTER) TO CON
TINUE"
430 C$=INKEY$: IF C$="" THEN 430
435 CLS
440 IF A$="O" THEN GOTO 45
445 NEXT X
450 RETURN
455 IF EOF(-1) THEN 475
460 PRINT #-P,D$(Y);S$(Y);I$(Y);
465 Y=Y+1
470 GOTO 455
475 CLOSE #-1: RETURN
480 REM
485 REM  SAVE COMMENTS ON TAPE
490 CLS: PRINT @ 133,"SAVE COMME
NTS ON TAPE"
495 PRINT @ 234,"POSITION TAPE"
500 PRINT @ 294, "PRESS PLAY AND
 RECORD"
505 PRINT @ 388, "PRESS (ENTER)
WHEN READY"
510 A$=INKEY$:IF A$="" THEN 510
515 OPEN "O", #-1, "COMMENTS"
520 FOR X=1 TO Y-1
525 PRINT #-1, D$(X),S$(X),I$(X)
530 NEXT X
535 CLOSE #-1: RETURN
540 REM
545 REM      LOAD COMMENTS FROM
TAPE
550 CLS: PRINT @ 134,"LOAD COMME
NTS FROM TAPE"
555 PRINT @ 235, "REWIND TAPE"
560 PRINT @ 300, "PRESS PLAY"
565 PRINT @ 388, "PRESS (ENTER)
WHEN READY"
570 R$=INKEY$:IF R$="" THEN 570
575 OPEN "I", #-1, "COMMENTS"
580 Y=1
585 IF EOF(-1) THEN 605
590 INPUT #-1, D$(Y),S$(Y),I$(Y)
595 Y=Y+1
```

```
600 GOTO 585
605 CLOSE #-1: RETURN
610 GOTO 335
615 PRINT#-P,"     NAME OF ORGAN
IZATION  "
620 PRINT#-P,"     ---- -- -----
-------"
625 RETURN
630 A$=INKEY$
635 IF A$="" THEN PRINT CHR$(191
);CHR$(8);:GOTO 630
640 IF A$=CHR$(13) THEN B$="":RE
TURN
645 IF POS(P) >22 THEN IF A$=CHR
$(32) THEN A$=CHR$(13)
650 PRINTA$;
655 B$=B$+A$
660 S$(Y)=B$
665 GOTO 630
670 RETURN
675 C$=INKEY$
680 IF C$="" THEN PRINT CHR$(191
);CHR$(8);:GOTO 675
685 IF C$=CHR$(13) THEN D$="":RE
TURN
690 IF POS(P) >22 THEN IF C$=CHR
$(32) THEN C$=CHR$(13)
695 PRINTC$;
700 D$=D$+C$
705 S$(N)=D$
710 GOTO 675
715 '**************************
720 '*    THIS PROGRAM WAS    *
725 '*          BY            *
730 '*       DESIGNED         *
735 '*     SAM ROBINSON       *
740 '*    4, FARR STREET      *
745 '*     EAST IPSWICH       *
750 '*      QUEENSLAND        *
755 '*         4305           *
760 '**************************
765 FOR X=1TO4:PRINTX"/";:CSAVE"
DATABASE":MOTOR ON:FOR DLY=1TO10
00:NEXT DLY:MOTOR OFF:NEXT X
770 'FOR 4 COPIES TYPE RUN 765
```

continued from P 7

```
268 LINE(124,94)-(124,136),PSET:
RETURN
269 '****  COLOURING PLANS  ****
270 PAINT(3,150),2,2
271 PAINT(50,98),2,2
272 PAINT(141,91),2,2
273 PAINT(100,39),3,2
274 PAINT(113,125),3,2
275 PAINT(113,146),3,2
276 PAINT(53,189),4,2
277 PAINT(61,66),4,2
278 PAINT(61,132),4,2
279 PAINT(105,91),4,2:RETURN
280 PCLEAR8:CLS
299 '****  A LITTLE BIT OF  ****
    ****SELF-AGGRANDISEMENT****
300 C=245
301 Q=1039:R=1040:S=1519:T=1520
302 POKEQ,C+4:POKER,C+1:POKES,C+
1:POKET,C+4:Q=Q+31:R=R+33:S=S-33
:T=T-31
303 IFQ()1411GOTO302
304 POKEQ,C+8:POKER,C+9:POKES,C+
2:POKET,C+6:Q=Q+1:R=R-1:S=S+1:T=
T-1
305 IFQ()1424GOTO304
306 POKE1123,C+1:POKE1132,C:POKE
1139,C+5:POKE1148,C+4:POKE1411,C
+4:POKE1420,C:POKE1427,C+5:POKE1
436,C+1
307 C=C-16:IFC)132GOTO301
308 POKE359,60:SCREEN0,1
309 PRINT@173,"david's";
310 FORDL=1TO3
311 ST=238
312 PRINT@ST,"star";:PT=ST+1024:
POKEPT-2,42:POKEPT+5,42
313 IFPEEK(1504)=42GOTO317ELSEPR
INT@ST,"    ";:POKEPT-2,96:POKEP
T+5,96
314 ST=ST+32:IFST<335GOTO312
315 NEXTDL
316 POKE1504,42:ST=ST-32:GOTO312
317 PRINT@503,"presents";:PLAY"V
1":FORV=1TO30:PLAY"V+T9L8A":NEXT
V:CLS5
318 POKE359,126:GOTO30
319 GOTO30
```

continued from P 10

```
650 TI$="Cylinder":GOSUB190:CIRC
LE(78,96),20,0,1.5:CIRCLE(178,96
),20,0,1.5,.75,.25:DRAW"BM78,66R
100BM78,126R100BM78,96L3R6L3D30B
D3D6U3R100U3D6":PRINT@400,"h":PR
INT@264,"r"
660 A$="r":GOSUB170:INPUTR:GOSUB
180:A$="h":GOSUB170:INPUTH:A=2*3
.1415926535*R*(R+H):GOTO200
670 TI$="Torus":GOSUB190:CIRCLE(
128,96),30:CIRCLE(128,96),50:DRA
W"BM128,96U3D6U3NL30NR50":PRINT@
238,"r    R"
680 A$="r":GOSUB170:INPUTR:GOSUB
180:A$="R":GOSUB170:INPUTR1:A=4*
3.1415926535^2*R1*R:GOTO200
690 PCLS:CLS:PRINT"1:Sphere
   2:Cube          3:Rect. box
   4:Cone          5:Cylinder
   6:Torus"
700 PRINT:INPUT"Enter your selec
tion";A
710 ON A GOTO 750,760,770,790,81
0,830
720 GOTO690
730 PCLS:CLS:PRINT@257,"The volu
me of this "TI$" is":PRINTV"unit
s"
740 A$=INKEY$:IFA$=""THEN740ELSE
90
750 TI$="Sphere":GOSUB190:CIRCLE
(128,96),50:DRAW"BM128,96R50":PR
INT@244,"R":A$="R":GOSUB170:INPU
TR:V=(4/3)*3.1415926535*R^3:GOTO
730
760 TI$="Cube":GOSUB190:DRAW"BM9
0,130R50U50L50ND50E25R50NG25D50G
25BD3D6U3L50U3D6":PRINT@366,"b":
A$="b":GOSUB170:INPUTB:V=B^3:GOT
O730770 TI$="Rect. box":GOSUB190
:DRAW"BM70,130R70U40L70ND40E60R7
0NG60D40NG60BR3R6L3G60R3L6BL3BD3
D6U3L70D3U6BU3BL3L6R3U40L3R6":PR
INT@295,"a":PRINT@397,"b":PRINT@
279,"c"
780 A$="a":GOSUB170:INPUTA1:GOSU
B180:A$="b":GOSUB170:INPUTB:GOSU
B180:A$="c":GOSUB170:INPUTC:V=A1
*B*C:GOTO730
790 TI$="Cone":GOSUB190:CIRCLE(1
28,96),20,0,2:DRAW"BM128,56M218,
96M128,136BD3ND6D3R90U3D6BM221,9
6NR6R13U40R3L90"
800 PRINT@220,"r":PRINT@405,"h":
A$="r":GOSUB170:INPUTR:GOSUB180:
A$="h":GOSUB170:INPUTH:V=1.047*R
^2*H:GOTO730
810 TI$="Cylinder":GOSUB190:CIRC
LE(78,96),20,0,1.5:CIRCLE(178,96
),20,0,1.5,.75,.25:DRAW"BM78,66R
100BM78,126R100BM78,96L3R6L3D30B
D3D6U3R100U3D6":PRINT@400,"h":PR
INT@264,"r"
820 A$="r":GOSUB170:INPUTR:GOSUB
180:A$="h":GOSUB170:INPUTH:V=3.1
415926535*R^2*H:GOTO730
830 TI$="Torus":GOSUB190:CIRCLE(
128,96),30:CIRCLE(128,96),50:PRI
NT@416:INPUT"What is the small r
adius";R:GOSUB180:PRINT@416:INPU
T"What is the large radius";R1:V
=2*3.1415926535*R1*R^2:GOTO730
```

16K

'GREG'

by Andrew White

(Andrew is a regular contributor to Australian CoCo, and also supplies us with "The Adventures of CoCo" seen in that magazine too.

He also supplied the brief resume of CoCoConf you'll find inside!

Greg touched all our lives, but when a young fellow like Andrew remembers with work like this, one can't help but be grateful that we were able to share Greg's life for just a few years. G.)

To the new reader of Australian Coco or Rainbow, and those who have been with our "family" for some time, you will probaly have heard of our founder, Greg Wilson. He has influenced the Coco's use to an almost inestimable degree in Australia.

Back in the dark days of computing, in 1981, Greg bought one of the first Cocos. After much experience with "build-your-own" computers and the infamous Tandy Model I he was greatly impressed. In those days Australian users were getting every possible scrap of information from the Americans, and a journalist called Lonnie Falk, started printing a newssheet for the Coco called "Rainbow". Greg arranged Australian licensing, and since he was forced to retire because of heart problems, he devoted all his energies to the magazine.

When Lonnie Falk had about 1000 subscribers, Greg was hoping to reach 100. He talked frequently with Lonnie on the telephone and devoted his energies to starting a network of user groups that no other computer has even come close to today.

Then, over a year ago, Greg died. Although I never knew him, I felt I did through his magazines, and from what I had heard of others, and I was shocked and saddened, emotions I know others felt as well.

Luckily for ourselves Graham and his friends offered to take on the arduous task of continuing Greg's work in both Rainbow and Coco (I believe much of their material was in Greg's first Aust. Coco)

So in memorial to Greg I wrote this program, which, I quicken to add, with things like the Greg Wilson Award, is the type of thing he would be totally against and tell me so in no uncertain (but highly entertaining) terms.

Thanks Greg.

NOTES ON USE

1) 64K users may type in the program "as is" or CLOAD or LOAD with no modifications.

2) 16K disk users must unplug their controller before typing/loading the program.

3) 16K ers who receive Rainbow on Tape must POKE 25,6:NEW then CLOAD, BEFORE making the changes below.

*Type DEL100-, or alternately, only type up to and including line 99.

*Change line 1 to read FORA=0TO97 (leave the rest of the line alone.)

*Save the program.

*If you typed POKE25,6:NEW now type PCLEAR4.

*Run the program.

*Type NEW.

*Type lines 100 on of the 64K version, or alternately, POKE25,6:NEW. CLOAD the Cocooz version, and type DEL-99

*Type new lines 98-99, as follows.

```
98 POKE65495,0:PMODE4,1:COLOR0,1
99 FORA=0TO12:READP,H,V:FORD=1TOP:READX,Y:PSET(
   H+X,V+Y):NEXTD,A:SCREEN1,0
```

*Save the second program.

*If you typed POKE25,6:NEW now type PCLEAR4.

*Run the second program.

From now on just CLOAD the first copy, type RUN, type NEW, CLOAD the second copy, and type RUN.

THE LISTING:

```
1 '**********GREG***************
  *****ANDREW WHITE*************
2 GOTO5
3 SAVE"GREG":DIR:STOP
5 CLEAR225:CLS3:PRINT@234,"PLEAS
E WAIT";:PMODE4,1:PCLS1:COLOR0,1
:POKE65495,0:LINE(0,0)-(256,192)
,PSET,B
6 FORA=0TO110:READP,H,V:FORD=1TO
P:READX,Y:PSET(H+X,V+Y):NEXTD,A:
POKE65494,0:SCREEN1,0
7 DATA5,50,60,9,12,10,12,10,11,1
2,11,11,11
8 DATA19,62,60,1,11,2,7,2,8,2,9,
2,10,2,11,3,3,3,4,3,5,3,6,4,11,5
,11,6,11,7,11,8,11,9,11,10,11,11
,11,12,11
9 DATA12,74,60,1,11,2,11,3,11,4,
11,5,11,6,11,7,11,8,11,9,11,10,1
1,11,11,12,11
10 DATA12,86,60,1,11,2,11,3,11,4
,11,5,11,6,11,7,11,8,11,9,11,10,
11,11,11,12,11
11 DATA12,98,60,1,11,2,11,3,11,4
,11,5,11,6,11,7,11,8,11,9,11,10,
11,11,11,12,11
12 DATA11,110,60,1,11,2,11,3,11,
4,11,5,11,6,11,7,11,8,11,9,11,10
,11,11,12
13 DATA17,50,72,7,6,7,7,7,8,7,9,
8,3,8,4,8,5,8,9,8,10,8,11,8,12,9
,1,9,2,9,8,10,7,11,7,12,6
14 DATA33,62,72,1,6,2,5,3,1,3,2,
3,3,3,4,3,5,3,6,3,7,3,8,3,9,4,7,
4,8,4,10,4,11,4,12,5,7,5,11,6,7,
6,11,7,11,7,12,8,7,9,6,10,6,11,6
,11,9,11,10,11,11,12,6,12,9,12,1
0,12,11
15 DATA20,74,72,1,7,2,7,3,7,4,7,
4,12,5,7,5,10,5,11,6,8,6,9,7,8,9
,12,10,10,10,11,11,7,11,8,11,9,1
2,4,12,5,12,6
16 DATA19,86,72,1,3,2,3,3,3,4,3,
5,3,6,3,7,3,8,3,9,3,10,3,10,4,10
,5,10,6,10,7,11,8,11,9,12,10,12,
11,12,12
```

```
17 DATA13,98,72,6,6,7,7,7,8,8,5,
8,9,9,5,9,10,10,5,10,11,11,5,11,
12,12,5,12,12
18 DATA30,110,72,1,5,1,8,1,9,1,1
0,2,5,2,8,2,9,2,10,3,5,4,5,5,5,5
,12,6,6,6,9,6,10,6,11,7,7,7,8,12
,1,12,2,12,3,12,4,12,5,12,6,12,7
,12,8,12,9,12,10,12,11,12,12
19 DATA5,50,84,9,1,10,2,10,3,11,
4,12,5
20 DATA29,62,84,1,5,2,5,3,5,4,1,
4,2,4,3,4,4,4,5,4,10,4,11,4,12,5
,4,5,5,5,6,5,7,5,8,5,9,6,5,7,6,8
,1,8,6,9,1,9,6,10,1,10,6,11,1,11
,6,12,1,12,6
21 DATA21,74,84,1,1,1,6,2,1,3,1,
3,6,4,1,4,6,4,11,4,12,5,5,5,9,5,
10,6,5,6,6,6,7,6,8,7,4,7,5,8,2,8
,3,9,1
22 DATA22,98,84,1,1,1,2,1,3,2,4,
2,5,3,4,3,6,3,7,3,8,4,5,4,9,4,10
,4,11,4,12,5,5,6,5,7,5,8,5,9,6,1
0,6,11,6,12,6
23 DATA16,110,84,1,1,1,6,2,1,2,6
,3,1,3,6,4,1,4,6,5,5,6,5,7,5,8,4
,9,4,10,3,11,1,11,2
24 DATA16,62,96,2,5,2,6,3,2,3,3,
3,4,3,7,3,8,3,9,4,1,4,10,4,11,4,
12,10,11,11,9,11,10,12,8
25 DATA16,74,96,1,7,2,2,2,3,2,4,
2,5,3,1,3,6,4,7,5,7,6,8,7,8,8,8,
9,9,10,9,11,10,12,10
26 DATA5,86,96,8,9,9,8,10,7,11,7
,12,6
27 DATA7,98,96,1,6,2,5,3,5,4,4,5
,1,5,2,5,3
28 DATA13,62,108,4,1,4,2,4,3,4,4
,5,5,5,6,5,7,6,8,6,9,7,10,7,11,7
,12,12,12
29 DATA16,74,108,5,7,5,12,6,7,6,
12,7,7,8,7,9,7,10,7,10,8,10,9,10
,10,11,7,11,10,11,11,11,12,12,7
30 DATA17,86,108,1,7,2,7,3,7,4,7
,5,7,6,7,7,8,8,8,9,8,9,11,9,12,1
0,8,10,9,10,11,11,8,11,12,12,8
31 DATA19,98,108,1,8,2,8,3,8,4,8
,5,8,6,7,6,8,6,12,7,7,7,8,7,9,7,
10,7,11,8,7,8,12,9,7,10,7,11,6,1
2,6
32 DATA23,110,108,1,6,2,6,2,11,2
,12,3,6,3,7,3,8,3,9,3,10,4,6,4,1
1,4,12,5,6,6,5,9,12,10,9,10,10,1
0,11,11,6,11,7,11,8,12,5,12,6
33 DATA13,62,120,8,1,8,2,9,3,9,4
,9,5,10,6,10,7,10,8,11,9,11,10,1
1,11,11,12,12,1
34 DATA22,74,120,1,1,1,2,1,3,2,4
,2,5,2,6,3,7,3,8,3,9,4,10,4,11,4
,12,5,7,6,1,6,6,7,2,7,3,7,4,7,5,
12,1,12,2,12,12
35 DATA25,86,120,1,3,2,3,2,4,3,3
,3,5,3,6,4,3,4,7,5,3,5,8,6,3,6,8
,6,9,7,3,7,9,8,2,8,10,9,1,9,10,1
0,11,11,1,11,11,12,1,12,2,12,11
36 DATA23,98,120,1,2,1,11,2,3,2,
11,3,3,3,11,4,2,4,11,5,2,5,11,6,
1,6,10,7,10,8,1,8,10,9,2,9,9,10,
2,10,9,11,2,11,9,12,2,12,8
37 DATA23,110,120,1,1,1,2,1,8,2,
7,3,7,4,6,5,1,5,5,6,1,6,4,7,1,7,
4,8,1,8,2,8,3,9,1,11,11,11,12,12
,7,12,8,12,9,12,10,12,11
38 DATA2,62,132,12,1,12,2
39 DATA12,74,132,1,3,1,4,1,5,1,6
,1,7,1,8,1,9,1,10,1,11,1,12,12,1
,12,2
40 DATA12,86,132,1,3,2,4,3,5,4,6
,5,7,6,8,7,9,8,10,9,11,10,12,11,
12,12,12
41 DATA12,98,132,1,12,2,12,3,12,
4,12,5,12,6,12,7,12,8,12,9,11,10
,11,11,11,12,11
42 DATA13,110,132,1,11,2,10,3,10
,4,9,5,9,6,8,7,7,8,6,8,5,9,3,9,4
,10,1,10,2
43 DATA14,74,144,1,1,1,2,2,3,3,4
,3,8,3,9,3,10,3,11,3,12,4,4,4,5,
4,6,4,7,5,5
44 DATA7,98,96,6,1,7,2,8,3,9,3,1
0,4,11,4,12,5
45 DATA12,110,96,1,5,2,6,3,6,4,7
,5,8,6,8,7,9,8,10,9,10,10,11,11,
12,12,12
46 DATA9,50,132,6,12,7,10,7,11,8
,9,9,8,10,7,11,6,12,6,12,5
47 DATA10,62,132,1,5,2,4,3,4,4,3
,5,3,6,3,7,2,8,2,9,1,10,1
48 DATA12,122,60,1,12,2,12,3,12,
4,12,5,12,6,11,7,11,8,11,9,11,10
,11,11,10,12,10
49 DATA12,134,60,1,10,2,10,3,10,
4,9,5,9,6,9,7,9,8,9,9,8,10,8,11,
8,12,8
50 DATA23,146,60,1,8,1,12,2,7,2,
10,2,11,3,7,4,7,5,7,6,7,7,6,8,6,
8,11,8,12,9,6,9,9,9,9,10,10,6,10,8
,10,9,11,6,11,7,12,5,12,6
51 DATA31,158,60,1,4,1,5,1,12,2,
4,2,8,2,10,2,11,3,3,3,9,4,3,4,10
,4,11,5,2,5,9,6,2,6,9,7,1,7,9,8,
1,8,9,9,2,9,8,10,2,10,8,10,9,10,
10,11,3,11,10,11,11,12,4,12,12
52 DATA8,170,60,1,5,1,6,2,7,2,8,
3,9,3,10,3,11,4,12
53 DATA12,122,72,1,7,2,7,3,6,3,7
,3,8,4,5,4,7,4,9,5,4,5,7,5,10,6,
7
54 DATA2,134,72,11,3,12,2
55 DATA11,146,72,1,1,7,2,7,3,7,4
,8,1,11,4,11,5,11,6,12,2,12,3,12
,4
56 DATA21,158,72,1,1,1,5,1,6,1,7
,1,8,2,4,2,9,2,10,3,11,5,1,5,2,6
,3,6,4,6,5,6,6,6,7,6,8,6,9,6,10,
6,11,6,12
57 DATA24,170,72,1,1,1,2,1,3,1,4
,1,5,1,6,1,7,1,8,1,9,1,10,1,11,1
,12,4,2,4,3,4,4,4,5,4,6,4,7,4,8,
4,9,4,10,4,11,4,12,4,1
58 DATA12,134,84,8,12,9,9,9,10,9
,11,10,6,10,7,10,8,11,3,11,4,11,
5,12,1,12,2
59 DATA26,158,84,4,6,5,4,5,5,6,3
,6,12,7,1,7,2,7,10,7,11,8,9,9,7,
9,8,9,12,10,6,10,10,10,11,11,3,1
1,4,11,5,11,7,11,8,11,9,12,1,12,
2,12,6,12,7
60 DATA12,134,96,1,8,2,9,3,10,4,
9,5,8,5,7,6,6,7,5,7,4,7,3,8,1,8,
2
61 DATA5,170,84,1,4,1,5,2,2,2,3,
3,1
62 DATA11,146,96,7,1,7,2,7,3,8,4
,8,5,9,6,10,7,11,8,12,9,9,11,9,1
2
63 DATA13,158,96,1,9,1,10,2,9,2,
11,2,12,3,8,4,7,5,6,6,5,7,4,8,3,
9,2,9,1
64 DATA14,122,108,1,1,2,2,2,3,3,
4,3,5,3,6,4,7,4,8,5,8,5,9,5,10,5
,11,6,10,6,11
65 DATA13,146,108,5,9,5,10,5,11,
5,12,6,6,6,7,6,8,7,1,7,2,7,3,7,4
,7,5,8,1
66 DATA16,158,108,2,1,3,2,3,3,4,
4,4,5,4,6,5,7,5,8,5,9,6,10,6,11,
6,12,7,11,8,11,9,12,10,12
67 DATA12,146,120,3,8,3,9,3,10,3
,11,3,12,4,4,4,5,4,6,4,7,5,1,5,2
,5,3
68 DATA13,158,120,3,12,4,11,4,12
,5,4,5,5,5,6,5,7,5,8,5,9,5,10,6,
1,6,2,6,3
69 DATA38,170,120,1,2,1,8,1,9,1,
10,1,11,1,12,2,2,2,3,2,4,2,5,2,6
,2,7,2,9,3,2,3,6,3,9,4,3,4,7,4,9
,5,3,5,7,5,9,6,3,6,7,6,10,7,3,7,
7,7,11,8,2,8,8,8,11,9,2,9,8,9,12
,10,2,10,12,11,1,12,1
70 DATA18,182,120,1,1,2,1,3,2,4,
2,5,3,6,3,7,4,8,4,9,5,10,5,11,6,
12,6,5,10,6,11,7,11,8,11,9,12,10
,12
71 DATA7,134,132,10,11,10,12,11,
9,11,10,12,6,12,7,12,8
72 DATA9,146,132,1,4,1,5,2,1,2,2
,2,3,11,12,12,9,12,10,12,11
```

```
73 DATA20,158,132,1,6,1,7,1,8,2,
3,2,4,2,5,3,1,3,2,8,11,8,12,9,9,
9,10,10,7,10,8,11,4,11,5,11,6,12
,1,12,2,12,3
74 DATA2,170,132,11,1,12,1
75 DATA14,182,132,1,3,2,4,3,5,4,
5,6,6,7,1,7,7,8,2,8,7,9,3,9,7,10
,7,11,7,12,7
76 DATA12,134,144,4,11,4,12,5,10
,6,8,6,9,7,6,7,7,8,4,8,5,9,2,9,3
,10,1
77 DATA12,146,144,4,11,4,12,5,10
,5,8,5,9,6,7,7,6,8,4,8,5,9,2,9,3
,10,1
78 DATA12,158,144,2,12,3,11,4,9,
4,10,5,6,5,7,5,8,6,4,6,5,7,2,7,3
,8,1
79 DATA11,122,156,7,11,7,12,8,10
,8,11,9,10,9,9,10,9,10,8,11,7,11
,8,12,6
80 DATA17,134,156,1,5,2,3,2,4,3,
1,3,2,7,11,7,12,8,9,8,10,9,8,9,9
,10,7,10,8,11,5,11,6,12,4,12,5
81 DATA14,146,156,1,3,2,2,3,1,6,
12,7,12,7,11,8,10,8,9,9,7,9,8,10
,6,11,4,11,5,12,3
82 DATA3,158,156,1,2,1,3,2,1
83 DATA8,98,168,1,10,2,10,3,11,4
,11,5,12,6,12,7,12,8,12
84 DATA12,110,168,1,12,2,12,3,12
,4,12,5,11,6,11,7,10,8,9,9,9,10,
8,11,7,12,6
85 DATA13,122,168,1,5,2,4,2,5,3,
4,4,2,4,3,5,1,5,2,6,1,9,12,10,11
,11,10,12,9
86 DATA21,134,168,1,7,1,8,2,6,2,
7,3,5,3,6,4,4,4,5,5,2,5,3,6,1,6,
2,8,12,9,11,9,12,10,10,10,11,11,
9,11,10,12,8,12,9
87 DATA11,146,168,1,7,1,8,2,6,2,
7,3,5,4,3,4,4,5,1,5,2,5,3,6,1
88 DATA12,98,180,5,1,6,2,7,3,8,4
,9,1,9,5,10,1,10,5,11,1,11,6,12,
1,12,7
89 DATA15,110,180,1,1,1,8,2,9,2,
10,3,11,4,12,5,1,6,1,6,2,7,2,8,3
,9,4,10,5,11,6,12,7
90 DATA18,122,180,1,8,1,9,1,10,1
,11,1,12,2,7,2,8,3,6,4,5,5,4,6,3
,7,2,8,1,10,12,11,11,11,12,12,10
,12,11
91 DATA12,134,180,1,9,2,8,3,7,4,
5,4,6,5,3,5,4,6,2,6,3,7,1,7,2,8,
1
92 DATA15,74,156,2,2,2,3,3,1,3,4
,4,5,5,6,6,7,8,7,8,8,9,9,10,10,1
1,10,11,11,12,11,12,12
93 DATA15,86,168,1,1,2,2,2,3,3,4
,4,5,5,6,6,7,7,8,7,9,8,9,9,9,10,
9,11,10,12,10,12,11
94 DATA8,86,0,6,12,7,12,8,11,9,1
1,10,11,11,10,12,10,12,9
95 DATA12,98,0,1,9,2,8,3,8,4,8,5
,7,6,7,7,7,8,6,9,6,10,6,11,6,12,
6
96 DATA12,110,0,1,7,1,8,2,7,3,7,
4,7,5,7,6,7,8,8,9,8,10,8,11,8,12
,8
97 DATA10,74,12,5,12,6,10,6,11,7
,9,8,8,9,7,9,6,10,6,11,5,12,5
98 DATA18,86,12,1,4,1,5,2,3,2,5,
3,2,3,5,4,2,4,4,5,1,5,4,6,5,7,5,
8,4,9,4,10,4,11,4,12,3,12,6
99 DATA17,98,12,1,3,1,5,1,7,2,3,
2,4,2,6,3,6,4,5,5,5,6,4,6,5,7,5,
8,5,9,6,10,6,11,6,12,5
100 DATA5,74,24,1,5,2,3,2,4,3,2,
4,1
101 DATA8,62,24,9,11,9,12,10,10,
11,8,12,6,12,7
102 DATA12,62,36,5,11,5,12,6,6,6
,7,6,8,6,9,6,10,7,4,7,5,8,1,8,2,
8,3
103 DATA15,62,48,4,4,4,5,4,6,4,7
,4,8,4,9,4,10,4,11,4,12,4,13,4,1
4,5,1,5,2,5,3,5,4
104 DATA12,122,0,1,8,2,7,3,6,4,6
,5,6,6,6,7,6,8,7,9,7,10,7,11,7,1
2,8
105 DATA11,134,0,1,8,2,8,3,9,4,9
,5,9,6,10,7,10,8,10,9,11,10,11,1
1,12
106 DATA15,122,12,1,5,2,4,2,6,3,
4,3,5,3,7,4,7,5,6,6,6,7,5,8,5,9,
5,10,5,11,5,12,4
107 DATA12,110,12,1,6,2,6,3,6,4,
7,5,8,6,8,7,7,8,7,9,6,10,6,11,5,
12,5
108 DATA17,134,12,1,4,1,6,2,4,2,
5,2,7,3,7,4,6,5,6,7,5,8,5,9,5,10
,5,11,4,12,1,12,4,12,7,12,8
109 DATA21,146,12,1,1,1,4,1,6,1,
8,1,9,1,10,1,11,1,12,2,2,2,5,2,1
0,3,3,4,3,5,4,6,4,7,5,8,6,9,7,10
,7,11,8,12,9
110 DATA3,158,12,1,10,2,11,3,12
111 DATA10,134,24,12,3,12,4,12,5
,12,6,12,7,12,8,12,9,12,10,12,11
,12,12
112 DATA11,146,24,1,1,1,2,2,10,2
,11,2,12,3,10,4,9,5,9,6,10,6,11,
6,12
113 DATA12,158,24,4,1,4,2,5,3,6,
4,6,5,7,6,7,7,8,8,8,9,9,10,9,11,
10,12
114 DATA13,146,36,1,1,3,4,4,3,4,
5,4,6,4,7,4,12,5,1,5,2,5,8,5,9,5
,10,5,11
115 DATA12,158,36,10,1,10,2,10,3
,10,11,10,12,11,4,11,5,11,6,11,7
,11,8,11,9,11,10
116 DATA12,146,48,3,1,3,2,3,3,4,
4,5,5,6,6,7,6,8,7,9,7,10,8,11,8,
12,9
117 DATA17,158,48,1,9,2,10,3,10,
4,11,4,12,8,9,8,10,8,11,8,12,9,5
,9,6,9,7,9,8,10,1,10,2,10,3,10,4
118 POKE223,0:FORD=1TO33:READP$:
O$=O$+P$:NEXT:PLAYO$+O$:FORD=1TO
96:READP$:PLAYP$:NEXT
119 DATA L4,C#,E,A,L8.,B,04,L16,
C#,L2,C#,03,L4,D,E,A,A,L4.,B,02,
L8,B,03,L4,C#,E,G#,L2,A,L4,E,L1.
,F#,L4,F#,04,C#,03,F#,04,C#,03,F
#,04,C#,03,F#,04,C#,03,F#,04,C#,
03,L2,F#,L4,G#,G#,G#,B,G#,E,L1.,
G#,L4,D,F#,A,G#,L4.,A,L8,E,L4,E,
G#,B,G#,L4.,A#,L8,C#,L4,D,E,A
120 DATA 04,L8,C#,L4.,C#,03,L4,B
,L2.,A,L4.,B,L8,A,L4,B,L4.,04,C#
,03,L8,B,L4,A,L2,E,L4,E,L4,F#,L8
,G#,L4,A,L4.,E,L8,A,L4,G#,L1.,A
121 EXEC44539:PRINT@234,"THANKS
GREG.";:EXEC44539:CLS:END
```

Greg Wilson, the man I wont forget,
Even though we never met.
For with very few words
He inspired me to become an Author,
of programs and articles.

Johanna Vagg.

PUT TOGETHER

by David Law

```
1 'PUT TOGETHER BY david a l law
 SUNBURY
2 ' ADAPTED FROM A PROGRAMME BY
'D.S.LEWANDOWSKI' IN THE JAN.'84
 AUSTRALIAN RAINBOW, PAGE35
3 DATA 8E,4,0,34,4,5F,5A,27,2,20
,FB,35,4,7C,0E,22,27,11,A6,84,81
,60,27,1,4A,A7,80,8C,5,FF,27,E0,
20,F0,0,39
4 GOTO8
5 A=128+RND(127):POKEI,A:I=I+1
6 IFI=>1535THEN8
7 GOTO5
8 FORB=&HE00 TO&HE23
9 READA$
10 POKEB,VAL("&H"+A$):NEXTB
11 DEFUSR=&HE00:Z=USR(0)
12 I=1024:RESTORE:GOTO5
```

16K ECB

MUSIC RECOGNITION

by Darrell Berry

MUSICREC is a music-recognition program written in ECB for the 16K CoCo. Given a piece of music (or whatever) fed into the cassette input port, it will compare a spectrum (see below) of it with other pieces stored in memory, and display a series of comparisons with those stored in memory before printing the title of the piece most closely matching the current piece. It also allows the current piece to be added to the list in memory.

To use the program you will need:

a) a tape recorder with an earphone output (you must have one of those!)

or preferably, if you want to input music from records,

b) a turntable, amp. and tape deck with variable recording level and headphone socket for monitoring recording.

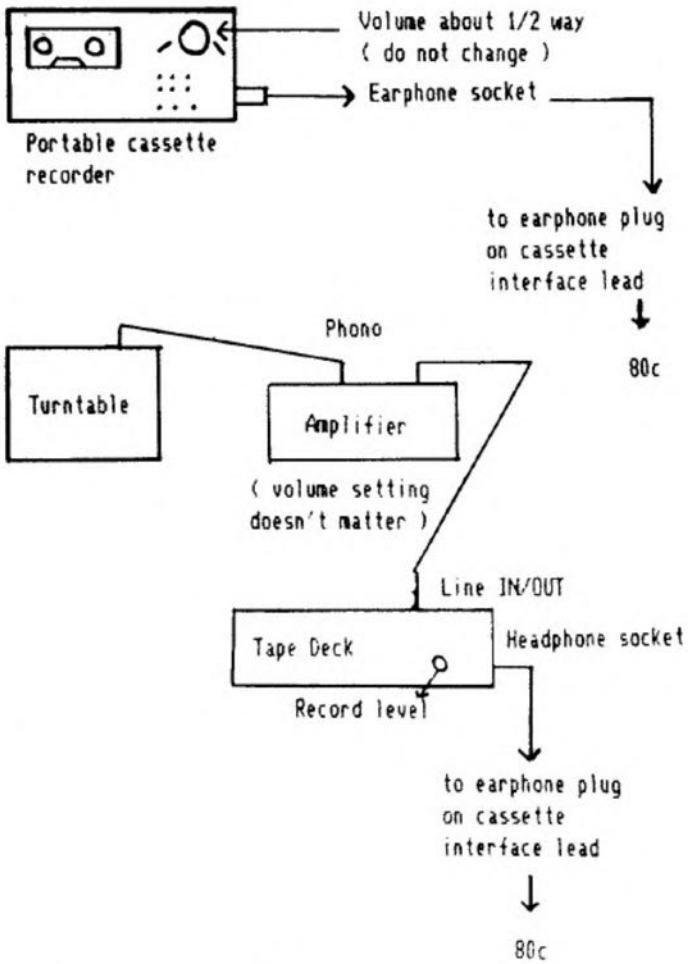
The two set ups are diagrammed later on.

The most important thing to ensure when using MUSICREC is to ensure that the level of the input to the computer remains the same from song to song, otherwise the system will not work. This is why b) above is preferred, as you can listen at whatever volume you like, without upsetting the line to the CoCo from the tapedeck monitor socket.

USING MUSICREC

1. Connect everything up. See the diagrams for details.
2. type RUN. A header followed by a '.' should appear. (after a keypress). The '.' means the program is scanning the input and hasn't heard anything yet.
3. Start the record/tape. The word START. should appear. This means the program has detected some input. If this does not happen, recheck the wiring and levels.

   After about 1.5 minutes, the program should display the first comparison spectrum. The one on the left represents the piece just listened to, the one on the right is from the list in memory. Two are always present, as examples. Others may be added by you. The number in the top centre of the screen represents the 'difference' between the two spectra. THe lower it is, the more they match. The title of the song from memory is also displayed. Press the space bar to see the next comparison. AFter all the songs in memory have been compared the program will say which was the best match (if it has heard the one you played it before, it should be the name of that song), and ask if you want to add the new songs spectrum to the list. Answer YES or NO. If you answer YES, enter it's title when asked.
5. The program will go back to displaying the '.'. Set up the next record/tape and the press any key to read it in.



i) put in a blank tape
ii) set deck to RECORD and PAUSE
iii) adjust levels so that signal peaks at about +3db
iv) DO NOT change the record level either during or between runs. Make a note of it, and always use the same setting, or MUSIC REC WILL NOT WORK!
v) deck should if possible be set to MONO
vi) NOTE: If you can't monitor record level with headphones you will have to use the headphone socket on your amplifier. This means you will not be able to adjust the listening volume.
vii) NOTE: With this set up you can also feed in music straight from tape, without changing the set up at all.

The Listing:

```
10 ' ** MUSIC RECOGNITION PROGRA
M
20 '        -- DARRELL BERRY
30 '
```

```
40 F=-1
50 DIM A(10),SA(10,10),ST$(10)
60 L=&HFF20
70 CLS:MIN=32767
80 PRINT "music"+CHR$(128)+"reco
gnition":PRINT"   [FEB FIVE SOFT
WARE] - 1985"
90 PRINT
100 PRINT "PRESS ANY KEY"
110 IF INKEY$=""THEN110ELSEPRINT
"."";
120 FOR K=1 TO 10:A(K)=0:NEXT
130 P=PEEK(L)
140 IF PEEK(L)=P THEN140
150 PRINT"START."
160 FOR K=1 TO 2000
170 V=PEEK(L)+PEEK(L)+PEEK(L)+PE
EK(L)+PEEK(L)+PEEK(L)+PEEK(L)+PE
EK(L)+PEEK(L)+PEEK(L)-20
180 A(V)=A(V)+1
190 NEXT K
200 CLS0
210 FOR V=1 TO 10
220 A(V)=INT (1000*A(V)/K)
230 PRINT V,A(V)
240 NEXT
250 IF INKEY$=""THEN250
260 IF ST=0 THEN PRINT "NO SONGS
 TO COMPARE...":GOTO550
270 FOR SC=1 TO ST
280 CLS 0
290 S$=LEFT$(ST$(SC),15):L=LEN(S
$)
300 FOR K=1 TO L
310 CH=ASC(MID$(S$,K,1)):IF CH=3
2 THEN CH$=CHR$(128) ELSE CH$=CH
R$(CH+32)
320 PRINT@502-L/2+K,CH$;
330 NEXT K
340 FOR X=1 TO 10
350 R=A(X)/12:IFR>29THENR=29:C=2
 ELSE C=3
360 FOR Y=29 TO 29-R STEP-1
370 SET (X,Y,C):SET(X+1,Y,C)
380 NEXT
390 R=SA(SC,X)/12:IFR>29THENR=29
:C=2 ELSE C=5
400 FOR Y= 29 TO 29-R STEP-1
410 SET (X+40,Y,C):SET(X+41,Y,C)
420 NEXT
430 NEXT
440 T=0
450 FOR K=1 TO10
460 T=T+ABS(SA(SC,K)-A(K))
470 NEXT
480 PRINT @12,T;
490 IF T<MIN THEN MIN=T:MS=SC
500 IF INKEY$=""THEN500
510 NEXT SC
520 CLS0
530 PRINT "BEST MATCH IS ";ST$(M
S)
540 PRINT "DIFF =";MIN
550 INPUT "ADD TO LIST ";Q$:IF Q
$="NO"THEN 60
560 INPUT "WHAT TITLE ";T$
570 ST=ST+1:ST$(ST)=T$
580 FOR K=1 TO 10:SA(ST,K)=A(K):
NEXT
590 GOTO 60
600 ST=ST+1:INPUT "TITLE ";ST$(S
T):FOR K=1 TO 10:INPUTSA(ST,K):N
EXT
610 INPUT "ANOTHER ";Q$:IFQ$="NO
"THEN60 ELSE600
```

You may wonder how reliable the program is. I have stored up to eight songs in it, and it has without fail recognized them all. I would be interested in hearing of others' success rates, especially with the different types of music. I must confess that I held no expectation that the program would work as well as it has for me.

I would also like to hear if anyone can find a use for the program. At the moment it is an interesting curiosity, amusing, but little else. Write to me at:

Jaye Franklin Hall
6 Elboden St,
Sth Hobart,
TAS, 7000.

16K

AUTO SAVE SAVE SAVE

by Peter Knox

Have you ever sufferred a power cut or computer failure and lost a program you have just laboriously typed in. After this happened to me a few times I decided to do somthing about it. This little program is the result.

Type in the 'AUTOSAVE' program from line 59999, CSAVE it to tape, and before you commence typing in that long program, CLOAD"AUTOSAVE" and after each page, type goto 60000 and save your masterpiece as many times as you like.

CoCo will ask you if you wish to verify your saves, if your answer is yes you will be asked to rewind tape, press PLAY and presto, your tape will be checked as a precaution against that dreaded IO ERROR.

If you are one that likes to know somthing is happenning, adjust your volume control so you can hear what is going on.

Another trap for the unwary is trying out a program before it has been completed and if the program has the high speed poke in it, a subsequent save will be wasted as it will not CLOAD. Line 60020 takes care of that by returning CoCo to normal speed.

Though very simple, I have found this little program invaluable and use it as a matter of course in all my programs. It has saved me many hours of time and frustration. Why not give it a go.

The Listing:

```
0 ''''''''''''''''''''''''''''''''''''''
1 '            AUTOSAVE                '
2 '          PETER KNOX                '
3 '                                    '
4 'DELETE -10 AND WRITE YOUR
    PROGRAM. AFTER TYPING A PAGE
,   TYPE GOTO60000 AND CSAVE WHA
T   YOU HAVE DONE.
10 '
59999 STOP
60000 CLS:PRINT:INPUT" HOW MANY
COPIES DO YOU WANT";N
60010 PRINT:INPUT" ENTER FILENAM
E ";F$
60020 CLS3:POKE65494,0:AUDIOON
60030 FORA=1TON:PRINT@226," ON S
AVE NO.";A;"OF ";F$;" ";
60040 CSAVEF$:MOTORON
60050 FORB=1TO900:NEXTB,A:MOTORO
FF
60060 PRINT@226,N;" COPIES OF ";
F$;" SAVED ";
60070 PRINT@290," DO YOU WANT TO
 VERIFY SAVES ";
60080 A$=INKEY$:IFA$=""THEN60080
60090 IFA$="Y"THEN PRINT@356,"
REWIND TAPE TO START  ";:PRINT@3
88," PRESS PLAY AND <ENTER> ";EL
SEGOTO60140
60100 IFINKEY$=""THEN 60100
60110 CLS3:FORA=1TON:PRINT@229,"
 CHECKING NO.";A;" ";:SKIPF:NEXT
A
60120 CLS3:PRINT@229," ";F$;" VE
RIFIED O.K. ";
60130 GOTO60130
60140 CLS4:PRINT@205," O.K.";
60150 GOTO60150
```

16K

PRINTER AID

by Paul Pickthorn

PRINTER AID illustrates very simply an excellent routine enabling you to send all your usual control codes to your printer.

This version has been written for the DMP 200 printer but should work with most TANDY printers and can easily be adapted to suit your own printer. Just load PRINTER AID then turn your printer on. Follow the prompts and make your selection. When you have made your selection press <BREAK> to exit the program. Your printer is now configured ready for you to use.

The Listing:

```
0 CLS
5 CLS4
6 PRINT@40,"$$$$$$$$$$$$$$$$";
7 PRINT@104,"BUDGET SOFTWARE";
8 PRINT@167,"TEL (09) 390 5277";
9 PRINT@235,"PRESENTS";
10 PRINT@297,"PRINTER AIDS";
11 PRINT@360,"TURN PRINTER ON";
12 PRINT@424,"$$$$$$$$$$$$$$$$";
13 FORX=1TO2000:NEXT
14 CLS
15 CLS:PRINT@39,"PRINTER AIDS V-
2.1";
16 PRINT@99,"1* UNDERLINE
     "; 'UNDERLINE ON
17 PRINT@131,"2* DELETE
     ";
18 PRINT@163,"3* WORD PROCESSING
 MODE  ";
19 PRINT@195,"4* ELONGATION MODE
     ";
20 PRINT@227,"5* DELETE
     ";
21 PRINT@259,"6* PROPORTIONAL PR
INT   ";
22 PRINT@291,"7* CORRESPONDENSE
     ";
23 PRINT@323,"8* STANDARD PRINT
     ";
24 PRINT@355,"9* CONDENSED
     ";
25 PRINT@387,"10* COMPRESSED
     ";
26 PRINT@419,"11* BOLD TYPE
     ";
27 PRINT@451,"12* DELETE
     ";:
31 PRINT:PRINT:INPUT"SELECTION";
```

```
Z
32 ONZ GOTO 100,101,102,103,104,
105,106,107,108,109,110,111
100 PRINT#-2,CHR$(15):GOTO15 'UN
DERLINE ON
101 PRINT#-2,CHR$(14):GOTO15 'OF
F
102 PRINT#-2,CHR$(20):GOTO15 'WO
RD PROCESSING MODE
103 PRINT#-2,CHR$(27)CHR$(14):GO
TO15 'ELONGATION MODE
104 PRINT#-2,CHR$(27)CHR$(15):GO
TO15 'ELONGATION OFF
105 PRINT#-2,CHR$(27)CHR$(17):GO
TO15 'PROPORTIONAL
106 PRINT#-2,CHR$(27)CHR$(18):GO
TO15 'CORRESPONDENCE
107 PRINT#-2,CHR$(27)CHR$(19):GO
TO15 'STANDARD
108 PRINT#-2,CHR$(27)CHR$(20):GO
TO15 'CONDENSED
109 PRINT#-2,CHR$(27)CHR$(23):GO
TO15 'COMPRESSED
110 PRINT#-2,CHR$(27)CHR$(31):GO
TO15 'START BOLD
111 PRINT#-2,CHR$(27)CHR$(32):GO
TO15 'END BOLD
```

16K

VARIABLES LISTER

by John Carmichael

Varlist is one of those neat little utilities that I always use when programming in BASIC. When developing a program of any size it is often almost impossible to keep track of what variables I have used and where in the program they have been used. Confusion can reign supreme.

The simple solution to this problem is to tack VARLIST onto the end of the program and anytime one needs to know which variables have been used where in the program simply type in RUN50000.

You are given the option of having the list sent to the printer or the screen and as a bonus a tally of the number of bytes used by REMARKS and SPACES as well as the total number of program lines is displayed.

The Listing:

```
50000 'MERGE ONTO END OF PROGRAM
 AND RUN 50000
50010 'CHANGE DIM VARIABLES IF
?OM ERROR
50015 'IF &H USED,THEN ADD LINE-
50065 IFA=38THENN=N+1:GOSUB50245
:GOTO50060
50020 PCLEAR1:CLEAR2000:DIMA$(40
,50),RE$(30):CLS:C=0:PRINT"LINE
 VARIABLE
50030 S=PEEK(25)*256+PEEK(26)+2
50040 NLA=PEEK(S-2)*256+PEEK(S-1
):IFNLA=0THEN50250 ELSE LS=LS+1
50050 LN=PEEK(S)*256+PEEK(S+1):N
=S+2:IFLN=50000THEN50250
50060 A=PEEK(N):IFA=0 THENS=N+3:
GOTO50040ELSEIFA=134THENS=NLA+2:
GOTO50040'DATA
50070 IFA=34THENGOSUB50160:GOTO5
0060 ELSEIFA=32THENSP=SP+1
50080 IFA=130 ORA=131 THENRE$(R)
=STR$(LN):R=R+1:L=L+NLA-N-1:S=NL
A+2:GOTO50040
50090 IFA>90ORA<65 THENN=N+1:GOT
O50060
50100 A$="":GOSUB50200
50110 FORB=1TOC:IFA$(B,0)=A$THEN
X=VAL(A$(0,B)) ELSE50140
50120 IFVAL(A$(B,X-1))<>LN THEN
A$(B,X)=STR$(LN):A$(0,B)=STR$(X+
1)
50130 B=98
50140 NEXT:IFB=99THEN50060ELSE C
=C+1:A$(C,0)=A$:A$(0,C)="2":A$(C
,1)=STR$(LN)
50150 PRINTLN;A$:GOTO50060
50160 N=N+1:IFPEEK(N)=34 ORPEEK(
N)=0THEN50170 ELSE50160
50170 IFPEEK(N)=0THEN50060 ELSEN
=N+1
50180 RETURN
50200 A$=A$+CHR$(A):N=N+1:A=PEEK
(N)
50210 IFA<90ANDA>47THENIFA<58 OR
A>64THEN50200
50220 IFA=40THENA$=A$+"(":N=N+1:
RETURN
50230 IFA=36THEN50200
50240 RETURN
50245 N=N+1:A=PEEK(N):IFA<71ANDA
>47THENIFA<58 ORA>64THEN50245
50246 RETURN
50250 INPUT"PRINTOUT/SCREEN (P/S
)";A$:IFA$="P"THEND=-2 ELSED=0
50260 FORZ=1TOC:PRINT#D,CHR$(13)
"     "A$(Z,0)TAB(15);:A=VAL(A$(0
,Z))-1
50270 FORB=1TO A:PRINT#D,A$(Z,B)
;:NEXT
50280 NEXTZ:PRINT#D,CHR$(13)"REM
ARKS"TAB(15);:FORB=0TOR:PRINT#D,
RE$(B);:NEXT:PRINT#D,CHR$(13)L"B
YTES USED BY REMARKS"CHR$(13)SP"
BYTES USED BY SPACES"CHR$(13)LS"
PROGRAM LINES":GOTO50250
```

ADVANCED BASIC

by Tino Delbourgo

(The following article is a copy of Tino's notes from CoCoConf '85. For those who couldn't attend, we thought you might like to see a bit of what you missed! G.)

String Functions:

The only types of data that BASIC can handle are numbers and strings (in contrast to languages like PASCAL which can handle records, sets, etc. as well). Actually, BASIC is very good at dealing with strings. We'll have a close look at strings now and the general ways of manipulating and modifying them. Then we will go through a few examples which will help to show the useful ways in which these string expressions can be used.

The most common ways that one would want to change strings would be to:

1. Delete the first N characters of a string A$:MID$(A$,N+1)
2. Delete the last N characters of a string A$:LEFT$(A$,LEN(A$)-N)
3. Delete the space preceding a positive number N when converted to a string using STR$(N):MID$(STR$(N),2)
4. Delete the space preceding a number N if it is positive: MID$(STR$(N),INSTR(STR$(N)," ")+1)
5. Delete M characters of a string A$ starting at the position N: LEFT$(A$,N-1)+MID$(A$,N+M)
6. Insert B$ before the Nth character in A$:LEFT$(A$,N-1)+B$+MID$(A$,N)
7. Delete the first occurrence of B$ from A$:LEFT$(A$,INSTR(A$,B$)-1)+MID$(A$,INSTR(A$,B$)+LEN(B$))
8. Replace the first occurrence of B$ in A$ by C$:LEFT$(A$,INSTR(A$,B$)-1)+C$+MID$(A$,INSTR(A$,B$)+LEN(B$))
9. Add sufficient B$'s (B$ must be a single character here) onto the end of A$ so that the length of A$ is N characters long: STRING$(N-LEN(A$),B$)
10. Check if B$ is included in A$ (useful with ON...GOTO):INSTR(A$,B$)<>0
11. Get the character at the Nth position of A$:MID$(A$,N,1)

Here is an example of each of these 'standard' string expressions:

1. X$=MID$(X$,3) will change X$="Extend" to X$="tend".
2. Y$=LEFT$(Y$,LEN(Y$)-4) will change Y$="enjoyment" to Y$="enjoy".
3. N$=MID$(STR$(N),2)) will make N$="53.5" when N=53.5 .
4. N$=MID$(STR$(N),INSTR(SR$(N)," ")+1) will make N$="-53.5" if N=-53.5 and N$="53.5" if N=53.5
5. J$=LEFT$(J$,10)+MID$(J$,15) will change J$="This is a CoCoconference" to J$="This is a conference".
6. K$=LEFT$(K$,9)+"OS8 "+MID$(K$,10) will change K$="Have the chips arrived?" to K$="Have the OS8 chips arrived?".
7. C$=LEFT$(W$,INSTR(W$,"ferro")-1)+MID$(W$,INSTR(W$,"ferro")+5)) will make C$="potassium cyanide" if W$="potassium ferrocyanide".
8. R$=LEFT$(A$,INSTR(A$,"US")-1)+"Australian"+MID$(A$,INSTR(A$,"US")+2)) will make R$="Australian RAINBOW" if A$="US RAINBOW".
9. G$=G$+STRING$(15-LEN(G$),".") will change G$="Get ready" to G$="Get ready.....".
10. ONINSTR("NSEW",INKEY$) GOTO100,200,300,400 will cause the computer to go to the appropriate line if the N, S, E, W key is pressed.
11. IFMID$(A$,5,1)="*"THEN540 will cause the computer to jump tp line 540 when the fifth character in A$ is a "*".

Memory Organization

One of the main reasons why CoCo shines over other small home computers is because there are many different ways that the memory in a 64K CoCo can be arranged. This is due to the 6883 SAM (synchronous address multiplexer) chip. Before we look at what this very important chip does, we'd better really understand what memory is.

I like to think of memory locations (in computerese they are called 'bytes') as P.O. boxes a number from 0 to 255 can be stored. Because CoCo has 65536 bytes of RAM memory and 32768 bytes of ROM memory, then a total of 96304 P.O. boxes would seem to be required. However, because CoCo is a 64K addressable computer, there are only 65536 lables that can be put on the outside of these boxes. The job of the SAM chip is to 'unstick' some of the lables from the P.O. boxes and to 'stick' the lables on other P.O. boxes. It just so happens that the SAM chip is designed to maintain three memory arrangements:

1. Bytes 0 to 32767 are RAM memory.
   Bytes 32768 to 65535 are ROM memory.
2. Bytes 0 to 32767 are the other 32K of RAM memory.
   Bytes 32768 to 65535 are ROM memory.
3. Bytes 0 to 65535 are RAM memory.

Arrangement 1 is the default arrangement used when the computer is first switched on. Many long 64K-only programs such as OS9, VIP writer and DEFT Pascal use Arrangement 3. It is also a common technique to make bytes 32768 to 65535 of Arrangement 3 the same as the ROM memory. This allows easy modifications and enhancement of the BASIC interpreter. The 32K CoCo has the same memory arrangement as Arrangement 1 for the 64K CoCo. The same is true for the 16K CoCo except memory locations 16384 to 32767 are unused.

The MC10 has a very different memory arrangement. Bytes 57344 to 65535 are ROM memory; bytes 16384 to 36863 are RAM memory in a 20K MC10, only bytes 16384 to 20479 are used in the 4K MC10. Also bytes 128 to 255 are RAM memory: all other bytes are unused.

Machine Language Injections

No computer language is complete without the ability to

call up machine language subroutines. But CoCo not only lets you do this; it also lets you pass a piece of information (parameter) to the subroutine. The subroutine has the ability to pass data back to BASIC; it 'returns an answer'. Therefore all ML subroutines must be functions.

You can define 10 such ML functions: written as USR0 to USR9 (USR stands for user). The parameter that is passed to the function must be placed in brackets. It can either be a number or a string. Since a USR subroutine is a function, it must be placed on the right-hand side of an equals sign. The value returned can either be a number or a string and the variable type on the other side of the equals sign must correspond. One would write for example:

A=USR1(3276)

In this example the parameter passed to the function is 3276. The value returned is put in the variable A and hence must be a number since A is a numeric variable.

However, before any USR function is activated in a program, the start address in memory of the ML function must be defined. To do this you use the DEFUSR command. For example to define function USR1 to start at byte &H7F00, one would type:

DEFUSR1=&H7F00

If you do not wish to pass parameters to an ML subroutine or get a value returned then you can just use the EXEC command. Simply put the address where the ML subroutine starts after the command.

e.g. EXEC&H700

If no address is placed after the EXEC command, then the last address specified is used.

However, before you even think about defining any USR functions or ML subroutines, you must first place the actual ML subroutine in memory. To do this, the subroutine must either be loaded form tape or disk using the CLOADM or LOADM command, or it must be put into memory via the POKE command. To protect memory from BASIC you use the CLEAR command. The second parameter of the CLEAR command specifies the address of the last byte that BASIC can 'touch'. (The first parameter is of course the amount of string space that should be reserved: the default value is 200 bytes).

e.g. In order to protect byte &H700 onwards from BASIC one would type CLEAR200,&H7EFF. The sample program below shows a ML subroutine in action: its function is to cause the computer to go into memory arrangement 3 and to cause bytes 32768 to 65535 to be the same as ROM memory: this enables you to use, modify and enhance BASIC since BASIC is now in RAM memory.

Here's the listing:

```
10 DATA1A,10,8E,80,00,B7,FF,DE,A
6,80,B7,FF,DF,A7,1F,8C,E0,00,26,
F1,39
20 CLEAR200,&H7FBF
30 FORI=&H7FC0 TO&H7FD4:READB$:P
OKEI,VAL("&H"+B$):NEXTI
40 EXEC&H7FC0
```

Here's another program that defines function USR0 to be an ML subroutine that clears the screen to the character code of the specified parameter:

```
10 DATABD,B3,ED,7E,A9,2A
20 CLEAR200,&H7FF0
30 FORI=&H7FF1 TO&H7FF6:READB$:P
OKEI,VAL("&H"+B$):NEXTI
40 DEFUSR0=&H7FF1
50 FORC=0TO255:A=USR0(C):NEXTC
60 IFINKEY$=""THEN50
```

Writing BASIC programs using BASIC programs

There are some programs that seem very repetitive to create and type in. It would be much easier to get CoCo to do some of the work for you. Well CoCo is able to do just that for you, due to its very advanced BASIC. Any ASCII file can be loaded as a BASIC program and any BASIC program can be saved as an ASCII file, simply by placing a ",A" after the file. e.g. to save a program in ASCII format one would type:

SAVE"filename",A or CSAVE"filename",A

This means that one can use BASIC programs to create other BASIC programs. The file would be created using the standard file output commands: namely OPEN, PRINT and CLOSE. For example to create DATA statements consisting of the contents of a block of memory, one could use this program:

```
10 CLEAR1000
20 INPUT"NAME OF PROGRAM";E
30 OPEN"O",#1,F$
40 INPUT"START ADDRESS OF BLOCK"
;S
50 INPUT"END ADDRESS OF BLOCK";E
60 L$="":LN=0:X=S
70 L$="":LN=LN+1
80 IFX>E THENGOSUB140:GOTO120
90 L$=L$+MID$(STR$(PEEK(X)),2)+"
,":X=X+1
100 IFLEN(L$)>220THENGOSUB140:GO
TO70
110 GOTO80
120 CLOSE#1
130 END
140 L$=MID$(STR$(LN),2)+" DATA"+
LEFT$(L$,LEN(L$)-1)
150 PRINTL$:PRINT#1,L$
160 RETURN
```

Using ML Injections

ML programming is difficult, and it is therefore handy if you are able to use an already programmed ML subroutine. Using ML subroutines is more practical and economical than making them yourself. You can get an ML expert to write useful subroutines for you and all you have to worry about is providing the necessary DATA statements (one method is described in the previous section). Memory is protected using the CLEAR command and the start address of the ML subroutine defined using the DEFUSR command. I will show

you how any BASIC programmer can use ML subroutines in his program, without worrying about the the actual ML. The example I will use is a series of graphic pages that are flipped through quickly in order to create animation. 36 pages will be used: this is more than is possible on a 16K/32K CoCo, but can just be done on a 64K CoCo. However two special ML subroutineswould be required: one to copy the contents of one page to another. The other routine would change the start of the screen so that the SCREEN command would display that page.

Here's the program:

```
10 CLEAR200,&H7FA0:DEFUSR0=&H7FA
1:DEFUSR1=&H7FC7
20 CLS*PRINT"SPINNING EARTH -- T
HE DELBOURGOS":PRINT" HAVE YOU R
EMEMBERED TO ENTER"
30 PRINT" POKE28160,0:POKE25,110
:NEW      for disk  OR":PRINT" P
OKE26112,0:POKE25,102:NEW      f
or tape?"
40 PRINT:PRINT" IF YOU HAVE, PRE
SS (ENTER);     OTHERWISE (BREAK
), ENTER POKES, AND RELOAD THE P
ROGRAM."
50 IFINKEY$()CHR$(13)THEN50
60 PRINT:PRINT" LOAD FROM (T)APE
 OR (D)ISK?"
70 A$=INKEY$:IFA$()"T"ANDA$()"D"
THEN70
80 PRINT" GET YOUR TAPE/DISK REA
DY...     AND THEN HIT ANY KEY."
90 IFINKEY$=""THEN90
100 FORI=&H7FA1 TO&H7FFF:READD:P
OKEI,D:NEXTI
110 DATA189,179,237,52,4,31,137,
141,50,31,1,53,4,141,44,31,3,16,
142,6,0,26,80,127,255,223,236,12
9,237,193,49,62,38,248,127,255,2
22,57,189,179,237,77,38,48,141,1
3,31,1
120 DATA220,183,147,186,159,186,
48,139,159,183,57,90,193,35,34,2
8,150,188,192,20,36,4,203,20,134
,128,52,2,134,6,61,235,224,79,30
,137,77,43,4,145,25,36,1,57,126,
180,74
130 PMODE2,15:PLCS:SCREEN1,0
140 FORP=1TO35STEP2
150 IFA$="D"THENLOADM"PAGE"+MID$
(STR$((P+1)/2),2),21504
160 IFA$="T"THEN:CLOADM"PAGE"+MI
D$(STR$((P+1)/2),2),21504
165 A=USR0(35*256+P):A=USR0(36*2
56+P+1)
170 NEXTP
180 FORP=1TO35STEP2:A=USR1(P):SC
REEN1,1:FORT=1TO40:NEXTT,P:GOTO1
80
```

Faults In BASIC

Rarely is any program perfect! The BASIC interpreter is no execption. However, there are no major bugs in the language. There is only 1 bug in Colour BASIC - the floating (real number) point subtraction operation. Extended Colour BASIC again only has one major bug. This bug is in the PCLEAR command routine. Whenever the PCLEAR command is used in the program, it causes program execution to be halted (CoCo returns to the 'OK' mode) or it causes an error message. This only happens when the program is first run. In order to avoid this bug, one must put a GOTO statement after the PCLEAR statement that simply orders the computer to go to the next line. Only do this if the number of graphic pages that you wish to reserve is greater than the default value (greater than 4).

In Disk Extended BASIC there are a few more mistakes. The major mistake is that the FILES command behaves like the PCLEAR command in that it causes faulty execution of a program when first run. However, the FILES command also causes incorrect matching of the start of the graphics screen as far as BASIC is concerned with the actual start in memory as far as the SAM chip is concerned. This results in an improper vertical alignment of the graphics screen. In short, choose your numbers with care and use the same GOTO command technique that was described with the PCLEAR command.

e.g. you would write:

```
1 FILES3,768:GOTO2
2 .....etc.
```

Some other mistakes of Disk BASIC are the opening of different files in different disk drives at the same time. As I onyl have one disk drive I don't know much about it, but I know it is a problem for multi-drive users. All these above mistakes have been fixed in CB1.2, ECB1.1, and DECB1.1 . The DOS command was added in DCB1.1 .

GET and PUT Commands with AND, OR, NOT

The GET and PUT commands are extremely powerful since they allow one to put any shape of any size on the screen. However, if you look closely at the chapter about GET and PUT in the Extended Colour BASIC manual, you will see that if you use the full graphic option when you get an area of the screen, you can use the AND, OR NOT operators when you put that area somewhere on the screen. In order to understand about these special operators, we need to know what AND, OR, NOT really mean.

```
AND Operation
0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1

OR Operation
0 OR 0 = 0
0 OR 1 = 1
1 OR 0 = 1
```

```
1 OR 1 = 1

NOT Operation
NOT 0 = 1
NOT 1 = 0
```

Notice how these operators work for only the numbers 0 and 1. This means that in two-colour modes, if one puts a green dot on a black dot using the AND option, the colour of the dot on the screen will become green since 1 OR 0 = 1. For four-colour modes the codes for the colours must be written as binary numbers and the appropriate operation then performed. The binary numbers for the colours are:

00 = green, buff
01 = yellow, cyan
10 = blue, magenta
11 = red, orange

Here's a sample program using these options:

```
10 DIMA(63)
20 PMODE3,1:PCLS:SCREEN1,0
30 COLOR2:LINE(0,0)-(49,49),PSET
,BF
40 GET(0,0)-(49,49),A,G
50 PCLS
60 CIRCLE(128,96),80,3:PAINT(128
,96),3,3
70 PUT(103,71)-(146,120),A,OR
80 GOTO80
```

Saving BASIC Programs with A Title Screen

BASIC has a very useful command called CSAVEM. This command can be used to save a block of memory to tape. This memory block could either be an ML program, a graphic/text screen or a BASIC program or any combination of the above list. This means that you can have a title screen displayed as the program is loading. In order to do this you need to know a little about the memory locations that control where the BASIC program is located in memory.

From our knowledge of how memory is arranged, we know that there are 65536 addressable memory locations at one time. Each byte can however hold a number from 0 to 255, i.e. it can hold 256 possible numbers. It is not difficult to see that two bytes can hold together 256 squared possible numbers, i.e. 65536 numbers. This happens to coincide (not by accident) with the number of addressable memory locations. This means that each address takes two bytes. The first byte contains the most-significant part of the number and the next byte contains the least-significant part. Therefore to work out the address stored in any two bytes, simply use this simple formula:

PEEK(X)*256+PEEK(X+1) where X is the address of the first byte of the two bytes.

Bytes 25 and 26 contain the addresses of the start of the BASIC program in memory. Bytes 27 and 28 contain the end address of the program. Bytes 29 and 30 contain the start address of the variables, and bytes 31 and 32 contain the start address of the arrays.

In order to make a title screen, simply save the memory from the text screen to the end of your BASIC program on tape via the CSAVEM command. You may also end up saving the graphic screens on tape since these are situated in memory after the end of the text screen but before the start of the BASIC program. The execution address used will be the entry address for the RUN command (byte 44661). The start address will be the end of the BASIC program (found out by a: PRINTPEEK(27)*256+PEEK(28)).

However, if you try and do that and then run the saved program by typing CLOADM:EXEC, it won't work. This is because the computer doesn't know where the BASIC program ends. To solve this little difficulty, add this line to the start of the program:

```
0 POKE27,XXX:POKE28,YYY:POKE29,X
XX:POKE30,YYY:POKE31,XXX:POKE32,
YYY
```

This will tell BASIC where the end of the program is and where the variable and array storage starts (this will be at the end of the program). After typing in the line do a PRINTPEEK(27) and replace all the XXX's with the number you get. Remember you'll have to put appropriate spaces after the number if it is less than 3 digits. Likewise PRINTPEEK(28) and replace all the YYY's with the number that you get. To save the program and the title text together do PRINTPEEK(27*256)+PEEK(28) and type: CSAVEM"filename",1024,777,44661 where ZZZ is the number printed. The title screen used is the current text screen and hence you will need to make the commands that create that text e.g. you would type:

```
CLS2:PRINT@228,"WELCOME TO COCO
CONF'85";:CSAVEM"HELLO",1024,9792,44661
```

To load your saved program simply type:

```
CLOADM:EXEC
```

You'll notice the block at the top left corner of the screen blink as the program is loading. The graphic screen at the time of saving could contain a pretty picture and that picture would also be loaded along with the title screen and BASIC program.

Merging Programs Together

Using a technique similar to that described in the previous section, cassette programs can be appended to one another (this is often inaccurately called 'merging'). The idea is very simple. A BASIC program is loaded from tape.

Then the pointer to the start of the program is moved past the end of it, a NEW is done, and the second program is loaded. The process is repeated untill all the programs are loaded. Then the pointer to the start of the BASIC program is moved back to the start of the first program. The programs are thus appended.

Obviously all the line numbers in the first program must be less than all the line numbers in the next program, etc. Failure to have this done will lead to improper order of execution. To move the pointer to the end of the next program simply:

POKE26,PEEK(28)-2 if PEEK(28))2 or if PEEK(28)(2 then type instead:

POKE25,PEEK(27)-1:POKE26,254;PEEK(28).

Then enter NEW and the computer is ready to load the next program from the tape.

Autostarts

The computer has a special subroutine in RAM memory that returns the next character in a BASIC program or command line. If the ML program you are loading starts at byte &H9F, the start address of the subroutine, it will autostart since the first thing the computer does after loading an ML program is to call that special subroutine. The only problem with this way of autostarting is that your ML program can only be 16 bytes long. Generally the ML program's job is to load the main ML program from tape which is then executed. The procedure is used in most of the commonly autostarting programs.

continued from P 12

```
105 Z$=MID$(X$,2,1):GOSUB63:DRAW
"BM-4,+8;"
106 Z$=MID$(X$,3):GOSUB63
107 GOSUB83:GOSUB68:DRAW"C6S4BM+
10,+8":CLS:NEXTSJ
108 GOSUB122:GOTO93
109 '****    ROUTINE FOR    ****
     ****SPECIFIC SUBJECTS****
110 CLS:PRINT@5,"***SPECIFIC SUB
JECT***":PRINT:PRINT" WHICH SUBJ
ECT":PRINT@97,"";:INPUTX$:PLAY"L
7503CCDDEEFF":IFX$=""THEN110ELSE
Z$=ID$+CHR$(61)+X$:DRAW"C6S4BM60
,14;":GOSUB62:GOSUB65
111 CLS:PRINT@35,"***"X$"***":PR
INT:PRINT"    ENTER YEAR FOLLOWED
 BY           PERCENTAGE MARKS FO
R            EACH TERM.":PRINT:P
RINT" ENTER '0' IF YOU HAVE NO M
ARKS  FOR ANY TERM(YOUR SCHOOL M
IGHT  ONLY HAVE THREE TERMS).":P
RINT
112 PRINT"    PRESS ^ AND <ENTER>
 TO DRAW      THE GRAPH IF YOU HA
VE MARKS      FOR LESS THAN SIX Y
EARS.":PRINT@491,"<ENTER>";:GOSU
B127113 X=45:PRINT@35,"'"X$"'":P
RINT:GOSUB74:GOSUB72:LINEINPUTZ$
:GOSUB120:GOSUB80:DRAW"C6S4BM45,
180;":GOSUB63  '1ST YEAR
114 X=80:GOSUB81:GOSUB75:GOSUB72
:LINEINPUTZ$:GOSUB120:GOSUB80:DR
AW"C6S4BM80,180;":GOSUB63  '2ND
YEAR
115 X=115:GOSUB81:GOSUB76:GOSUB7
2:LINEINPUTZ$:GOSUB120:GOSUB80:D
RAW"C6S4BM115,180;":GOSUB63   '3
RD YEAR
116 X=150:GOSUB81:GOSUB77:GOSUB7
2:LINEINPUTZ$:GOSUB120:GOSUB80:D
RAW"C6S4BM150,180;":GOSUB63   '4
TH YEAR
117 X=185:GOSUB81:GOSUB78:GOSUB7
2:LINEINPUTZ$:GOSUB120:GOSUB80:D
RAW"C6S4BM185,180;":GOSUB63   '5
TH YEAR
118 X=220:GOSUB81:GOSUB79:GOSUB7
2:LINEINPUTZ$:GOSUB120:GOSUB80:D
RAW"C6S4BM220,180;":GOSUB63   '6
TH YEAR
119 GOSUB122:GOTO93
120 IFZ$="^"THENPLAY"L200CDEFGAB
":GOTO119ELSEIFLEN(Z$)(>4 OR SGN
(VAL(Z$))(>+1THENGOTO87ELSEPLAY"
L5504DEFDEF":RETURN
121 '****  PRESENT GRAPH  ****
122 SOUND250,1:PMODE3:SCREEN1,1:
GOSUB130
123 DRAW"S4BM10,180;XN$(18);XN$(
31);XN$(42);BM10,190;XN$(28);XN$
(22);XN$(42);"
124 SOUND155,1:PMODE4:SCREEN1,1:
GOSUB130
125 IFINKEY$=""GOTO122
126 CLS:PLAY"L2502AG":RETURN
127 IFINKEY$=""GOTO127
128 CLS:RETURN
129 FORDL=0TO150:NEXTDL:RETURN
130 FORDL=0TO550:NEXTDL:RETURN
131 '****OPENING GRAPHICS ****
132 PMODE4,1:PCLS:SCREEN1,1
133 FORX=32TO223STEP2:LINE(128,0
)-(X,145),PSET:LINE(128,191)-(X,
46),PSET:NEXT
134 PLAY"V31T402L8;6;11;03;3;L5;
6;P24;L7;3;L4;6"
135 NA$=N$(21)+N$(18)+N$(39)+N$(
26)+N$(21)+N$(45)+N$(36):ST$=N$(
36)+N$(37)+N$(18)+N$(35)
136 DRAW"S8BM120,100;XN$(26);BM-
4,+0;XN$(26);S4C6BM104,46;XNA$;S
16BM68,106;XST$;C5BM12,106;XN$(4
4);BM204,106;XN$(44);"
137 GOSUB122:RETURN
```

Tandy
ELECTRONICS

CoCo For Beginners Or Professionals

FROM ONLY

199.95

- **Create Exciting Color Graphics Plus Great Sound Effects**
- **Large Selection of Instant-Loading Program Paks™**
- **Expandable Memory**

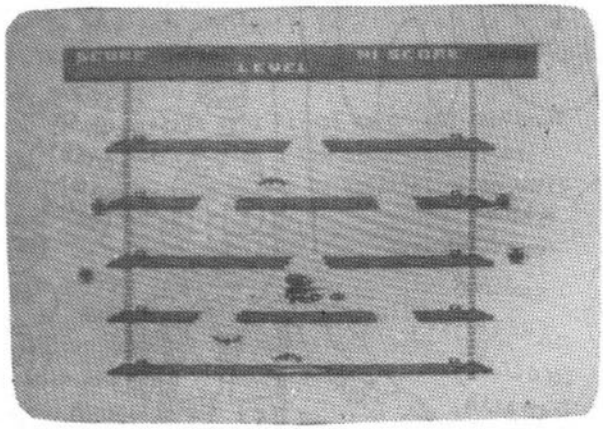
The ideal way to introduce your family to the world of color computers — at a price that won't break the budget! Simply plug into your TV and use the BASIC language to create full-color programs. Add realism with sound effects and music. Features 16,000 characters of memory for home budgets, investment information, recipes, etc. Help the kids develop their maths skills with educational software, or relax with sports and space games. Perfect for learning to program too. To upgrade the system simply add more memory — up to 64K, cassette player and ROM Paks. 26-3134 Was 239.95 . . . . . **Now 199.95**

The 16K Extended model goes one step further with Extended BASIC programming for complex geometrical shapes and elaborate graphics, with sound. Features POKE, PEEK and USR commands, string arrays up to 255 characters and multi-character variable names. This model can be expanded with disk drives, more memory and a printer. 26-3136
Was 299.95 . . . . . **Now 249.95**

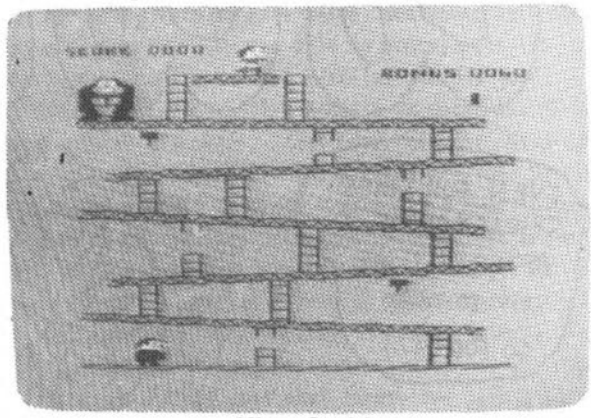
Top-of-the-line **64K Extended** version is for the serious programmer. Create color business charts, even animation. Add word processing and electronic filing software, disk drives and a printer for an advanced computer system. 26-3127
Was 449.95 . . . . . **Now 349.95**

You Can Afford to Upgrade Your System with Tandy Accessories and Software Using the Money You Save!

Hours of Fun With New Computer Games



Time Bandit



King Cuthbert

A. Time Bandit. Create your own game from more than 300 variations! Destroy Evil Guardians, Killer Smurfs and more before your time runs out. 26-7352 ..... **29.95**

B. King Tut. Search for treasure in King Tut's tomb with only a small candle to light your way. Look out for snakes and King Tut's ghost. 26-7313 ..... **29.95**

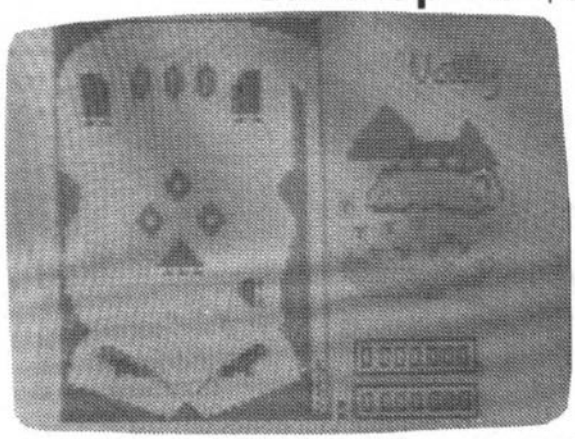
C. Danger Ranger. Collect 10 keys guarded by evil creatures, then get treasure chests from the Acid Chamber — but watch out for the demons. 26-7315 **29.95**

D. King Cuthbert. Climb ladders and ramps to save Cuthbert's lady love from the Evil Gorilla. Try to avoid rolling barrels and fireballs. 26-7320 ..... **29.95**

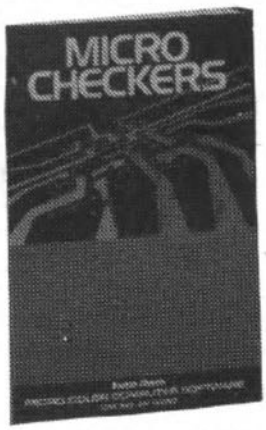
E. Devil Assault. Defend yourself against attacks by Vampire Bats, Robots and nasty Sproings. Face the Devil in higher skill games. 26-7316 ..... **29.95**

F. Buzzard Bait. Fly from cloud to cloud on your "bird" jousting the enemy. Try to avoid the deadly Perodactyl, Lava Pit and Lava Hand. 26-7351 ..... **29.95**

Save Up To $30 on MC-10 Software



A.



B.

A. Micro Math Design

B. Micro Checkers

A. Micro Color Checkers. Challenge the computer in this all-time favourite. 26-3360 Was 14.95 . . . . **Now 4.95**

B. Micro Color Games Pak. Five arcade-style games in one! Lunar Lander, Horse Race, Breakout, Egg Hunt and Pong. 26-3361 Was 16.95 . . . . . **Now 7.95**

C. Micro Color Math Design. Two programs — Math functions/plot graphics. 26-3362 Was 16.95 . . **Now 4.95**

D. Micro Color Pinball. Stone-age pinball, two cavemen try to bounce boulders off dinosaurs. Requires 16K RAM module. 26-3363 Was 16.95 . **Now 4.95**

E. Micro Compac. Configure your MC-10s serial port for communications. Access up-to-date news and financial data. Requires optional modem and cable. Includes user ID numbers. 26-3350 Was 49.95 . **Now 19.95**


DONE BY THE MAD AND DEPRAVED ANDREW WHITE.
LEO DE VINCI HE 'AINT
COCOCONF
THAT'S CROOKED!
EVEN GOD LIKES THIS LANGUAGE .... AFTER ALL HE SAID TO NOAH "GO FORTH..."
JHN REDMOND
WOOSH
CRACK!
SPURK
ZAP!
GAMES PLAYERS SAVING THE EARTH, COCOCONF, AUSTRALIA, YOUR PET DOG.
DAZED BEWILDERED BATTERED MAGAZINE EDITOR
HEY ALEX?
YEH ANDREW?
DUCK!
PAIR OF FROGS
FLASH!
MAD MICK THE FLASHER.
IT'S OKAY - BUT I DON'T LIKE THE SMILE...
TASTELESS CRITIC
M-M-MARTHA GRITWHISTLE!
"TAROT" "DOS" "WHAT!"
ARTISTIC COMPUTER NOVICE
DATA 100,30
OPEN #path
WHILE tst DO
ENDWHILE
INPUT x$
ALL I SAID WAS "DID YOU GET YOUR OSB CHIPS"!
ARRGH!
I KNOW MY PIZZA'S HERE SOMEWHERE
BRAIN DOUGAN
HELLO, PRONTO PIZZA'S?
KEVIN THE KOOK
IT'S M-MARTHA GRITWHISTLE!
SCARED COMPUTER NOVICE
GAMES!
"HAPPY JACK" FRICKER.
CUTE COMPUTER
ANNOYING PERSON
THANK GOD IT'S ONLY ONCE A YEAR!
AND WHAT'S THE COLOUR MONITOR FOR??
BLATT!
BEEP!
COMPUTER NOVICE
NO! YOU CAN'T BUY A 300 METRE TRAIN SET!
ANNETTE

I had a call from the states the other night, and one of the first things said was " I hear that CoCoConf wasn't successful."

Well I guess it depends how you measure success.

We had 75 people at CoCo Conf. It's true I figured on 200, but we achieved everything I wanted to achieve with 75.

We wanted to give some worthwhile CoCo instruction - we achieved that with classes in Hardware, MS DOS, Forth, OS9, BASIC and Advanced BASIC.

We wanted you to see the very latest hard and software for CoCo - it was there - with a few minor exceptions.

We wanted to honour Greg Wilson by instituting an award in his name - we did it.

We wanted to reward some of the superior games programmers of this year - we did it.

Lastly, we wanted to spend some time with you - get to know you better, and get to understand your needs. We achieved that too!

So what was unsuccessful?

Nothing - we had a great time!

The lasting memories will be Tino Delbourgo (15 yrs old) leading two three hour courses in Extended Colour BASIC, his classes consisting of 25-30 mature aged people each time; the look of suprise on Andrew Simpson's face when he was awarded first prize in the games competition; the pride on his father's face; the pride of the very capable John Redmond in his son, as James was awarded second prize in the games competition; and the friendliness and warmth of all participants.

It is clear that the hobby brings parents and kids together and I find that very exciting.

There are some people who need to be thanked publicly for their efforts over the weekend.

Firstly to the crew of Rainbow - Annette, my wife, Sheryl and Jim Bentick, Sonya and Rod Young, Kevin Mischewski, Alex Hartmann, Michael Horne, Andrew White, and Bill, my father - thank you for all for the hard work you put in. You people are responsible for CoCoConf being such an unqualified success. We haven't done one of these before, but despite this, the organizational side of things flowed absolutely smoothly - without even a hint of trouble, reflecting the excellent planning you gave to your individual responsabilities.

Next, thank you to:

Ron Wright for travelling so far, to see us for so short a period, to give so much.

Ron and family have given us all a lot over the past four years. We want them to know they are appreciated.

The Dougan family. Bev & Natasha helped in the kitchen. Natasha was on a 40 hour famine fast - she must have found the going hard - especially with so many examples of her mother's fine culinary art around her! Jason did odd jobs (some VERY odd) for me all weekend, and his father Brian - or Brain as he is known around the office, talked himself hoarse. All weekend, I never saw Brain with less than 10 people around him, all wanting to know about hardware mods!

John Redmond, John Poxon and Rod Young came to show us the possibilities of FORTH. John R. has had a column in Australian CoCo for some time now - but most of us have ignored it. John demonstrated bubble sorts that are 35 times faster in FORTH than in BASIC, and drew things on the screen at lightening speed and in many other ways demonstrated that we ignore FORTH at our own peril. He also showed just how easy FORTH is to master. John P. showed the CoCoConnection, its possibilities, and how FORTH can bring this machine alive. Rod Young uses FORTH in his work everyday. He was able to demonstrate its use in day to day problem solving - fellas you did good!

Paul Humphries had committed himself to doing the MS DOS tutorial, then discovered that one of the Mainframe Computers he is responsible for was not working. Despite this he raced down to CoCoConf - gave his 3 hour tutorial and raced back to his computer - and then wrote an article for the magazine later that morning. He is, needless to say, a special friend of this magazine.

The OS9 guys - Jack Fricker, Kevin Holmes, Fred Bisseling, Jacky Cockinos, Kevin Mischewski, et al, don't need thanking - they just shut themselves into a corner all weekend, emerging only for dinner Saturday night! Mind you, OS9'ers do tend to be like that!

I've already talked about Tino - and the special effort his family made to get him to Southport is very much appreciated. But the Delbourgos are a family we have gained much inspiration from for some time now, so our special thanks go to them.

The first 'Greg Wilson Award' for services to the Computing Comunity went to the Delbourgo family. I have subsequently spoken with Helga Wilson, and she agrees that whilst Greg would have disapproved of his name being used - he could not have argued with the choice of recipient.

I need to apologise to Tasmania - once more Queenslanders have left you off the map of Australia. The excuse is that the map on the award is figurative - but we will undertake to put a Tasmania on the map when we get the award back from you next year!

This award is signified with a mantle shield which is kept by the recipient for a year, and with a smaller wall plaque, which is retained by the recipient.

In awarding the games prizes, it was felt that although his programs and articles didn't directly come under the category of games, both the quality of the articles and program and the basis they provided for some exciting games in the future, made it necessary to bestow a special award on John Redmond.

John received the Amber Yangen Monitor supplied by Blaxland Computer Services.

Our thanks go to Bruce and Roger at Blaxland for your support and for the monitor. We use several of these monitors at Rainbow and we are very satisfied with them. The Yangen monitor will make John Redmond's work easier.

The first prize in the games competition went to Andrew Simpson for his program "SABRE". Sabre is a real time space pilot / shoot 'em up game, where you as commander of your ship have to pilot the ship around the galaxy and shoot the enemy, using your computer's guidance system. We chose this game because it achieves its aim, even though it is written entirely in BASIC. In fact, it is a fine example of what can be achieved with CoCo's graphics

in BASIC.

Andrew received a Disk Drive and DOS from Software Spectrum, who wanted to do something special for Andrew and provide one of the new units they are about to release. Unfortunately, our timing was a bit off, and the new unit wasn't to be available for several weeks, but Software Spectrum provided a stand-in unit anyway! Very nice John and Ken - that is the kind of service one has to respect!

Second prize went to James Redmond whose graphics adventure "Labyrinth" is scathingly clever. Desite his father's interest in FORTH, James still prefers BASIC for his style of work. John tells me that James planned the entire program before he went near the keyboard. The actual typing only took one night! THAT is intelligent programing!

James received the Tandy Speech - Sound Pak from Bayne and Trembath. Bayne and Trembath have been friends of this magazine for a long time. Their support to us and to Tandy computer users is legendary. We thank them for their help this year.

The third prize was awarded to Kevin Gowan for his program "Move About". This prize was awarded for a program which showed good structure. "Move About" shows much planning, good structure and plays very well. We specially like the way the letters move.

Kevin, who comes from Port Pirie, received a Platinum "Worksaver" from Geoff Tolputt, the Australian importer of that product. Geoff's offer of this prize came late but fitted nicely into or plan for prize night. We thank him very much. "Worksaver" is a valuable tool to own.

To those of you who were coming but decided against it at the last minute - you blew it! You missed a nine course chinese banquet on Saturday night that even stopped Andrew White and Tino Delbourgo! You missed meeting some of the most capable computer users in the country.

But never mind because the news is that CoCoConf will be on again next year and will be bigger and better! As will the contests and the prizes.

This coming year, the programming contest will be divided into Games, OS8, Education, Adventure Games and General Programming. Prizes will be announced later, but do expect something very special for the grand prize winner. All programs appearing in either of our magazines between August 1985 and July 1986 are eligible for consideration, and this therefore includes all those programs we have received but not yet printed. You have a very high standard to surpass, so get into it now!

Special mentions:

"Cattle Baron"	-	John Day
"Fire Fox"	-	Max Betteridge
"OS8"	-	Barry Cawley
"OS8"	-	Jim Rogers
Various	-	Mike Turk
Various	-	Mark Rothwell
Education	-	Bob Horne
"Orbquest"	-	Tony Parfitt
"Karbah"	-	Keith Roach
"Alphabet"	-	Glenn & Patricia Kentwell
Ham Radio Progs	-	Roy Lopez
ML Progs	-	Greg Watson
Various	-	Tom & Matthew Lehane

Andrew Simpon & Rainer Horn wondering how to get it all back in!

One of Tino's groups.

Tino receives the Greg Wilson Award for Services to the Computing Community.

John Redmond explains Forth for the forth time!

Paul Humphries demonstrates the Tandy 1000.

ASSEMBLY FILE

BY KEVIN

This month we will start dabbling with some short Assembly Language programs. Believe it or not you already know enough to start reading the listings (admitedly with alot of thumbing through the instruction set tables) and what better program to start with than that terror I gave you last month.

I copped some minor abuse from a few of those who actually typed in that program exactly as listed and I guess I deserved all I got. Good on you for trying but I must admit to leaving out some minor details so why don't we now look at that program from the point of view of the author.

Since I have only ever had experience with TANDY's EDTASM we will use that for this excercise. Also, for the first and I hope the last time I shall use a BASIC program as a direct comparison with our Assembly program. Look at Listing 1: This is the BASIC version of our machine language program. The logic should be quite obvious and if the program looks a bit too simple it is because I have tried to make it directly comparable to the Assembly version.

Before we begin to write any code we must decide what we are actually aiming to achieve. If you are to avoid writing cumbersome and unmanagable programs you must plan your attack BERFORE you start writing your code. Some people advocate the use of flow charts at this point but I find drawing a flow chart more confusing than programming so lets keep things simple and think out our plan of attack.

Now, our main plan is to write a short routine to fill the screen with the character A which as we know is the ASCII character 65 decimal ($41 hexadecimal). I guess the best way to do this is to start at the top left corner and working from left to right and from the top row to the bottom row finish up at the bottom right hand corner.

LISTING 1:

```
10 REM MY PROGRAM NAME HERE
110 SC=&H0400
120 A=65
130 B=1
140 X=0
150 POKE SC+X,A
160 X=X+B
170 IFX=512 THEN Z=1 ELSE Z=0
180 IF Z=1 THEN GOTO200
190 GOTO150
200 GOTO200
210 END
```

How the heck do we put a letter A at the first screen location, let alone all the rest. Some of you may remember and others may have figured out that the CoCo uses a portion of its memory to store the data to be printed on the screen. Just to confuse the issue CoCo manipulates memory in a multitude of ways to get a multitude of screen types.. HI-RES, PMODE4 etc..... But for now just let us worry about the comparatively simple normal text screen since that is the one we want to fill with the letter A. RUN listing 1. The screen should slowly fill up with the letter A just as we wanted it to. The first memory location we POKEd to put an A in the top right hand corner was $0400 hexadecimal (1024 decimal). (Pretty soon I'm going to get tired of writing both Decimal and Hex values and use whatever comes to mind at the time. So be warned and remember convention has it that Hex numbers be preceeded with a $ symbol and if you are talking to CoCo in BASIC he wants you to use The symbol &H in front of HEX numbers.) Since there are 512 memory locations used up by the text screen then we have one by one filled up the entire screen with 512 letters A.

Lets start to think about writing the code for our program. What we want is a loop to poke the letter A into 512 memory locations each memory location being one higher than the previous, ie. an increment of 1. But we also need to define the first memory location where we plan to store our letter A. Less obvious is the need to define the ASCII value of the letter A and also to define the increment by which we wish to increase the memory location where we store that A at each pass of the loop. Finally we have to branch out of the loop when we have filled the 512 memory locations otherwise CoCo will blindly carry on filling memory locations with the letter A until finally he puts one A too many right where he shouldn't and the poor old thing gets totally confused.

We are now ready to code our program. I will not explain the BASIC version but its line numbers do match directly with those of our Assembly version.

Start up your EDTASM and type in our Assembly program exactly as listed in Listing 2:. Once you have typed in line 210 BREAK from the editor and type in A/IM/WE. The program will scroll up the screen and if there are no syntax errors you will see the TOTAL ERRORS message at the bottom of the screen complete with the list of labels exactly as in listing 1. Once your program is error free type in A and the program will assemble to to the cassette tape. This is your EXECutable copy of the program. Don't forget to also save a copy of the Source code (all that typing you just did) using the W command.

Let's examine the program itself. You should already know the meaning of the instructions so we will just concentrate on the program logic.

Line 100 overcomes that problem a few of you struck when trying to EXEC last months listing. ORG is a

LISTING 2:

```
                    00010 *MY PROGRAM NAME HERE
2000                00100         ORG     $2000
          0400      00110 SC      EQU     $0400
2000 86   41        00120 START   LDA     #65
2002 C6   01        00130         LDB     #01
2004 8E   0000      00140         LDX     #0000
2007 A7   89 0400   00150 LOOP1   STA     SC,X
200B 3A             00160         ABX
200C 8C   0200      00170         CMPX    #512
200F 27   03        00180         BEQ     LOOP2
2011 7E   2007      00190         JMP     LOOP1
2014 7E   2014      00200 LOOP2   JMP     LOOP2
          2000      00210         END     START
00000 TOTAL ERRORS

LOOP1   2007
LOOP2   2014
SC      0400
START   2000
```

Pseudo-Op which tells the assembler where in memory to commence assembling the program. If ORG is not included the assembler takes the course of least resistance and sets the ORG to 0000 which in CoCo is not a real good place for YOU to locate a machine language program. 0000 can be a useful default value allowing you to LOAD the Machine Langage program from BASIC using a simply calculated offset and thus put the program anywhere is memory that you wish. In this case I have been lazy and set the ORG to $2000 simply because it is a location acceptable to 16K, 32K, Cassette and disk systems, but be warned, it is unprotected memory and any BASIC program is most likely to play havoc. Ideally I should have set the ORG at somewhere around $3F00 for a 16K system or $7F00 for a 32/64K system. Before loading the program you would be required to protect the Machine Code program by entering CLEAR 200,&H3F00 or CLEAR 200,&H7F00 respectively. If the ORG had been set to eg. $7F00 then the first byte used by the Machine code program would have been $7F00 instead of $2000. Using the memory location $7F00 in BASIC's CLEAR command would reserve all the user memory from $7F00 to $7FFF, ie. the top of user RAM memory, for Machine Code programs. That amounts to 256 bytes which is more than enough for our program, it being only about 17 bytes long. Obviously a fair amount of planning must go into selecting your ORG and in the future I'll show you some tricks you can play and find some gaps in memory you can use to preserve that precious user RAM space.

Line 110 sets the variable SC to be equal to $0400 which as we know is the first location where we want to put an A.

Line 120 we have labelled as the START of the program and it appears the assembler has decided the same thing. Examine the left column of numbers and you will see 2000 as the first memory location of this line which as you can see is the exact location we defined as the start address in our ORG statement. But what does line 120 do? Thats easy, because all it does is load the A register with the value 65 which just happens to be the ASCII value for the character A which is the very character we want to put on the screen. And of course 130 loads register B with the value 1 and line 140 loads register X with a whole heap of zeros.

Take special note of the symbol to denote that all these last three operands are data and not memory locations. If they did not include the # symbol (shifted 3 on your keyboard) the respective registers would be loaded with the contents of those memory locations and in this case that would cause total confusion to our poor CoCo. Note also that since there are no $ symbols these values are decimal.

Now that we have initialised our variables so to speak we can enter our loop and start filling the screen with the letter A.

When writing Assembly code we do not generally use line numbers as destinations marking the beginning of a loop or location of subroutines etc. Instead we have the label field which is normally only used to indicate those lines that need to be referred to by other parts of the program. This program uses four labels within the program SC, START, LOOP1 and LOOP2. Labels are only used by the Assembler program at the time the program is being assembled into Machine code. They provide a reference point the Assembler can use to determine how many bytes or to which byte in memory the program must move to.

Lets now start at LOOP1 and follow the course of the program. Line 150 tells us to store the contents of register A somewhere. But how do we interpret the operand. We know that SC refers to memory location $0400 and X refers to the contents of the X register. What the whole instruction tells us to do is store the contents of register A at the memory location pointed by register X at an offset of $0400 bytes. Or in plain english on the first pass through the loop: put the value 65 into memory location 0000+$0400. Easy Eh!

Our next instruction adds register B to register X and thus in this case increments the value of X by 1. Since X when added to SC points to the memory location where we are storing our character A, ABX has effectively moved us on to the next screen location.

But before we do that we must check to see that we have not reached the end of the screen. When we have reached the end of the screen X will have a value of 512 so all that is required is that we compare the value of X with the data value 512. This is done admirably with the CMPX instruction. The comparison is carried out by subtracting 512 from the contents of X if the result is zero then the Z bit is set in the Condition Code register. Line 180 tests the Z bit of the Condition Code register which if set activates the relative branch called for by the BEQ instruction. Here we use the label LOOP2 as the destination for the branch.

If the branch is not activated then we fall through to the next line of our program which calls for a jump back to the start of our loop at the label LOOP1 ready to put an A at the new memory location pointed to by X.

Had the branch suceeded then we would have ended up at

line 200. What could be more obvious. The instruction tells us to jump to LOOP2 which tells us to JMP to LOOP2 which tells us to jump to LOOP2 which tells us to jump to LOOP2 which tells us..... The only way out of this is to hit the RESET button at the rear of the computer.

The final line, 210 is another Pseudo-Op which fills a very important function with EDTASM. You have already used ORG to decide where in memory you will locate your program. When you LOAD a Machine Language program from BASIC and type EXEC the program must start executing at the right memory location and often this is not the same as the ORG address although in our case it is. END START tells the assembler to set the EXEC address at the address of START. Remember the format of BASIC's SAVEM command: SAVEM"name" ,start address, end address, exec address.

The last few lines thrown out by the EDTASM listing simply list the addresses where EDTASM has assembled the labelled instructions.

Now that we have examined the Assembly code lets relate it to the Machine code.

Look to the very left of line 120 and you will find the value 2000. Here we come across a clear reason for developing a working familiarity with Hex numbers as all of the 3 left hand columns of listing 2 are in hex. The assembler ignores convention and simply assumes that you know that these columns always contain hex numbers. So this 2000 is our memory location $2000. This is the location of the first byte of the Machine code for line 120. LDA translates to the value $86 and the decimal value 65 translates to $45. This consumes two bytes as does line 130. Line 140 consumes 3 bytes, one for the istruction and two for the operand dealing as we are with register X which is a two byte register.

Address $2007 contains an $A7, the value for STA but the next three bytes are a little confusing. Here the $89 refers to our X register and of course the $0400 is still our offset from the value of our register X.

Line 180 is the interesting one. At address $200F the BEQ instruction translates to $27 which further translated to plain english means Branch Relative. In eatablishing the operand the assembler has not simply inserted the address of LOOP2 but instead calculated how many bytes away from the current address pointed to by the Program Counter Register we must move to reach the address of LOOP2. In this case we are moving forward 3 bytes. $2011 + $03 = $2014. The CPU has read the instruction and operand of line 180 and before acting on the instruction sets the Program Counter Register to the address of the next instruction, hence the branch actually moves the PC register from address $2011 and only needs to move 3 bytes.

In contrast line 190 jumps to the absolute address of $2007 just as line 200 jumps to the absolute address of $2014.

Run both the BASIC program and the Machine Language program. Do you notice much difference?

DIVERSIONS:

As a bonus this month I will include a comprehensive reply to a letter received from a chap by the name of David Colvin. He has two questions. The first asks why he always gets a SEARCH FAILS reply when trying to use the F (find) command of TANDY's EDTASM. Before using F you must be sure that the string you are searching for occurs somewhere in the listing AFTER the current line otherwise you will most likely get the SEARCH FAILS error or alternatively find the wrong occurrence of the string you are looking for. The easy way to reposition the edit pointer to the start of the listing is to enter the command P#.

Secondly we have a rather common question. "What is all RAM mode and how do I get into that mode?"

When your CoCo is upgraded to 64K I rather imagine that, unless forewarned, most of you would have been a little disappointed to find that you only had somthing less than 32K of USER RAM available for your use from BASIC. When you first switch on CoCo the MEMORY MAP is configured such that the CPU addresses the ROM's (Read Only Memory) when it reads an address within the upper 32K of memory, ie. addresses from $8000 to $7EFF Only when addressing memory below $8000 will it read or write to RAM (Read and Write Memory). There is a small 256 byte block of memory from $FF00 to $FFFF (top of memory) which is used for control of special registers one of which includes the infamous High Speed Poke ($FFD5-high speed $FFD4-normal speed). $8000 bytes =32K and $FFFF bytes =64K and we have 64K RAMS installed. It should now be clear that on startup we address only the lower half of the 64K of RAM we have spent all that good money on.

LISTING 3:

```
                        00100 *GOTO ALL RAM
7F00                    00110          ORG     $7F00
7F00 1A   50            00120 START    ORCC    #$50
7F02 8E   8000          00130          LDX     #$8000
7F05 A6   84            00140 LOOP     LDA     ,X
7F07 B7   FFDF          00150          STA     $FFDF
7F0A A7   80            00160          STA     ,X+
7F0C B7   FFDE          00170          STA     $FFDE
7F0F 8C   FF00          00180          CMPX    #$FF00
7F12 2D   F1            00190          BLT     LOOP
7F14 B7   FFDF          00200          STA     $FFDF
7F17 BE   ABEE          00210          LDX     $ABEE
7F1A 8C   4F4B          00220          CMPX    #$4F4B
7F1D 26   06            00230          BNE     RETURN
7F1F 8E   3E3E          00240          LDX     #$3E3E
7F22 BF   ABEE          00250          STX     $ABEE
7F25 39                 00260 RETURN   RTS
          7F00          00270          END     START
00000 TOTAL ERRORS

LOOP    7F05
RETURN  7F25
START   7F00
```

Contained within the ROMs is all the Machine Code needed by our CoCo to interpret all of the BASIC commands. If you don't have Extended Basic or Disk then quite simply you also have a big hunk of unused

inaccessible memory. Even if you do have Disk and ECB there is still an 8K gap in memory from $E000 to $FEFF. Tough!? Not really as we will soon see.

By POKEing a random value into the Control Register located at address $FFDF we can cause CoCo to ignore the ROMs and for the complete range of addresses from $0000 to $FEFF look to the 64K of RAM we have availiable. We can reset the Control Register by POKEing a random value to address $FFDE.

That is all very well but as yet we have nothing that means anything within the upper 32K of RAM and as we have already found out CoCo packs his bags and takes a holiday if he gets confused. That is until you press reset to jolt him back to the real world. But pressing reset also resets $FFDE and we're back half RAM half ROM.

What we must do put somthing useful into the top 32K of RAM and that is just what listing 3 does.

What we are aiming to do is to copy byte for byte the contents of ROM into RAM at the same addresses as used by ROM. We will then switch to all RAM and lo and behold you will be back in BASIC and CoCo won't have flown off to the Barrier Reef.

ORCC will carry out a logial OR operation on the CC register. Oh boy! here comes some binary. Look up the binary value of $50 in your binary tables. If you're more clever than I you can calculate it out. You should find the binary value %01010000 (the % denotes a binary value). Look at the bit arrangment of the Condition Code Register. They are:

BIT	FLAG	ORCC $50
7	Entire,	0
6	FIRQ,	1
5	Half Carry,	0
4	IRQ,	1
3	Negative,	0
2	Zero,	0
1	Overflow,	0
0	Carry.	0

The only bits that are set and therefore the only bits that are CHANGED by the logical OR are the two interrupts FIRQ and IRQ. Setting the FIRQ and IRQ bits disables these two interrupts and this must be done in our program as any interrupt while we are in in the all RAM mode before we have finished copying our ROMs accross will upset the apple cart (as usual).

The X register is being used for Indexed Addressing as is denoted by the comma before the X in the operand field. We have loaded the X register with the first address we want to copy, $8000. To get our value for A we use X register to point to the location we wish to read. Storing any value to $FFDF switches to all RAM ready for us to put the data in A into RAM at the same address. The plus sign indicates that after we carried out that instruction then the X register is to be incremented by 1.

Go back to ROM/RAM by storing a random value at $FFDE.

The first address we don't want to copy is $FF00 (remember, this is where we find our control registers) so we will only Branch back to LOOP if X is Less than $FF00. Its interesting to note that we will have made approx 32000 passes through the loop before we reach $FF00 so don't be too suprised if it takes a couple of seconds.

We make one final switch to all RAM since that is where we want to be in the end. Then those of you who have one of the BASIC 1.1 CoCo's compatable with mine will find the OK prompt changed to)). The address where my OK prompt is held is $ABEE.

Finally we must return control of CoCo back to BASIC but a special copy of BASIC nonetheless. For it is all in RAM and our ROMs are having a well earned rest.

Before you EXEC the assembled Machine Code DON'T FORGET to CLEAR200,&H7F00.

And now I must retire for a well earned rest???

ST. GEORGE AND THE DRAGON

FIRED BY (ST) GEORGE & ELLEN AFTAMONOW

And now, as they say, for something completely different. A friend of Tom Lehane's, from the US, has sent this excellent game.

32K ECB ARCADE

THE LISTING:

```
1 'ST GEORGE AND THE DRAGON BY
 (ST) GEORGE & ELLEN AFTAMONOW
2 GOTO10
3 POKE65494,0:SAVE"STGEORGE":DIR
:STOP
10 CLEAR1000:CLS
20 DIMR(1,8),S(1,7),B(1,8),N(1,7
),C(1,7)
30 PCLS8:PMODE3,1:GET(0,0)-(29,2
1),B,G:GET(0,0)-(25,16),N,G
40 PRINT@170,"BE WARNED.":PRINT@
192," YE WHO ARE ABOUT TO  ENTER
 THE  REALM OF THE DRAGON  MAY N
EVER  RETURN."
50 PMODE3,1:PCLS7
60 HH$="UE3U5E3R2NDR ER4FR2FR U2
EUNR2L11UR9NFUENRLUER UHU2ERHG3D
LDLGLEUEUE2R3FRF2NL2NR62FNL3 BGB
DG2R2EFU2F2RNH4RNR9ERNFENF2R2NF2
R2F2R256L22 NL2GF62D2GHUE2UBLDLU
RBRHEL3NL7GHDL2NHDLH2 LG2UNU2LNH
DFBL4BG FRFDNE2NR2FDFDFDR4"
70 HI$="HLHUHUNHEH2U2ENE2F2D2F2N
D2EUE2NEFDGDREDFURNUGBG NL5FDG6U
3RNFE2UL4EUBG2BDNG2D3 G2D2F2NLF2
L4UNEHL2EU3EU2NR62L8HLNL2D FD3NH
D3H3ENRU2HUHLNL2D2GD2FDNLF2L4 UN
EULUGU3E2U2LUE3U3HU2NED6D3GD62LG
L"
80 DR$="HUNHLGBUERF2R4ER H4UEBDB
FF4R2NDBUBHH4L2H2 NR2L3H3G2UEL2E
R3F2RENE U4H2G2L2URURU2LDLDGU2NU
```

```
LURENRURNDEU FED2R2GFED2RDRDRDRD
2FD2F2D3F3E2U2H5U4E4RNRG4D3F6D4G
4L2NEL10"
90 PMODE3,1:PCLS7
100 DRAW"BM4,150;S12C5XHH$;XHI$;
BM182,168;S16XDR$;":PAINT(184,16
5),6,5:PAINT(10,10),5,5:SCREEN1,
1
110 FORX=1TO2000:NEXTX
120 NB$(0)="BRHU2ERFD2GLBR4":NB$
(1)="BU3ED4NLRBR2":NB$(2)="UEREH
LGBD3R3BR2":NB$(3)="BU3ERFGNLFGL
HBDBR5":NB$(4)="BU2NR3U2BR3D4BR2
":NB$(5)="R2EHL2U2R3BD4BR2":NB$(
6)="BRHUNR2UERFBD2NHGNLBR3":NB$(
7)="BRUE2UNL3BD4BR2":NB$(8)="BRH
ENRHERFGFGNLBR3"
130 NB$(9)="BUFREUL2HERFDBD2BR2"
140 PMODE2,1:PCLS:SCREEN1,1
150 S$="U2E2R3FEU4HL2HU4EFR2E2U2
G2L2HG3D5FR2E2R2FD5G3HL3G2 BR17B
U":T$="LH3RU10LE3D2R3DL3D11FRNEG
 BR6":P$="HEFGBR15":G$="L3H4U8E4
FR3E2D2G2L3HG2D8F2R3EU3HL3E2R2F2
D4G3 BR11"
160 E$="L2H2U6E3RF3GUH2G2D5NE4DF
R2NEG BR10":O$="L3H2U7E2R3F2D7G2
BUL2HU7ER2FD7GBD BR8":R$="H3RU6H
LEF2DE3F2GNH2BL4D5F2NEG BR8":GG$
="H2U7E2R3FRGD15HLHL2G2LE3R3FU13
HLG2D6FRNEG BR11"
170 A$="LH2EU3H2DGD5RNE6H2U3E3F2
U3HL3NGER3FD8FR6 BR11":N$="HU8HN
EG3NU3D6GHEU7HLERNFBR5FD8FNEG BR
11":D$="HNU2BL2GH3U5E3R2G3D5F2E2
U8NLH5RF5D9FNEG BR12"
180 H$="G4L2E5U6H3G3D6GHEU13NHRN
EDND8BF4NGF3D8 BR8":DD$="L9NGE2U
5HRU4ENED11R3NU12R3U4NL3U4NL3U4L
7H2NHR9F2D12G2BR10"
190 DRAW"BM30,50;S8"+S$+T$+P$+G$
+E$+O$
200 DRAW"BM169,50;"+R$+GG$+E$
210 DRAW"BM58,90;"+A$+N$+D$
220 DRAW"BM135,90;"+T$
230 DRAW"BM165,90;"+H$+E$
240 DRAW"BM60,134;"+DD$+R$
250 DRAW"BM110,134;"+A$
260 DRAW"BM122,134;"+GG$
270 DRAW"BM151,134;"+O$
280 DRAW"BM178,134;"+N$
290 PLAY"T5L4O3C02AO3C02AB-L2.GO
3L4C02AO3C02AB-O3CL1DL4CDCO2B-AG
F"
300 R1$="H2U3E2RUE2R2EHUR2UE2R4R
DRDRDRDR3F2D4FDG2DL21"
310 R2$="H4U4ER2FRU2HUEU3E2UE2R9
FRDF2GDFRFRFDGFDFDFD2LD3L23"
320 Q$="U4BR2D2NL2D2BR3 U4BR2 R2
NR2D4BR7 NR3U4R3D4NHNF2BR7 U4NL2
R2BR2 NR3D4R3NU4BR7 NR3U4R3D4NHN
F2BR3 NR2U4BR2D4BR3 NU4BR5 U4NL2
R2"
330 Q1$="U4NL2R2BR2 ND4BR3D2NL3D
2BR3 U4BR2NR2D2R2D2NL2BR8 R2U2L2
U2R2BR2 NR2D4R2BR2 U4R2D2L2F2BR2
 NR2U2NRU2R2BR2 NR2D2NRD2R2BR2 U
4FDRDFU4"
340 PP$="NU4RU4NLR2D2L2BF2BR2":P
I$="U4RD4LBR3":PC$="NU4RNR3U4NLR
3BD4BR2":PK$="NU4RU4NLBR3G2NLF2B
R2":PA$="NU3RU2NR2U2NGR2ND4FD3LB
R3":PR$="BRNU4LU4R3D2L2NLF2BR2"
350 PD$="RNR3U4NLRND4R2D4BR2":PG
$="BRHU2ENR2D4R2U2LBD2BR3":PS$="
R3NU2RU2L3NU2LU2R4BD4BR2":PT$="U
4NL2RNR2D4LBR5":PO$="BRNR2NU4HU2
ER2ND4FD2GBR3"
360 PN$="NU4RU3NHF3U4LD3BFBR2":P
E$="NU4RNR3U2NR2U2NLR3BD4BR2":PW
$="NRU4RD4R2NU2R2U4RD4LBR3":PV$=
"BR2H2U2RD3FE2U2LD3BFBR2":PX$="U
NLNR3U2NLNUR3BHD4":PB$="NRU4RND4
RFDNL2D2L2BR4":PU$="NRU4RD4R2U4D
4LBR3"
370 UR$="RNR2U4NLR2D4BR2 U2NRU2R
3D2L2F2BR2 U2NR3UERFD3BR2 BRHU2E
RBD2RD2NL2BR2 BRHU2ERFD2GNLBR3 U
4FDF2NU4BR2"
380 UW$="BR5NU4RU2RD2RNU4BR2 U2N
R3UERFD3BR2 R3U2L3U2R3BR2BD2"
390 UU$="NU4R3NU4BR2 U4FDF2NU4BR
2 RNR2U4NLR2D4BR2 U2NR2U2R3BD4NL
3BR2 U2NRU2R3D2L2F2BR8"
400 CLS0:PRINT@168,"  CREATED BY
  ";:PRINT@200,"GEORGE & ELLEN";
:PRINT@232,"  AFTAMONOW   ";:SCR
EEN0,1:FORX=1TO600:NEXTX
410 FORZ=168TO246:PRINT@Z,CHR$(1
28);:SOUNDZ,1:NEXTZ
420 CLS:PRINT@99,"";:INPUT"DO YO
U NEED INSTRUCTIONS";Z$
430 IF Z$="N" THEN470
440 CLS:PRINT"THE OBJECT IS TO F
IND AS MUCH OFTHE HIDDEN GOLD AS
 YOU DARE. THEONLY PROBLEM IS TH
AT YOUR SEARCHIS OVER ONCE YOU F
IND A DRAGON."
450 PRINT"BY PRESSING THE LETTER
 THAT IS  MARKED ON THE STONE, Y
OU WILL BEABLE TO SEE WHAT'S UND
ER IT."
460 INPUT"PRESS (Q)UIT TO STOP S
EARCHING  ONE SCREEN AND GO ON T
O THE NEXT   HIT 'ENTER' TO STAR
T";Z$
470 CLS4:PRINT@37,"CHOOSE A LEVE
L";:PRINT@99,"1. SIR THOMAS THE
TIMID";:PRINT@163,"2. SIR RICHAR
D THE BRAVE";:PRINT@227,"3. SIR
GEORGE THE DRAGON";:PRINT@262,"S
LAYER";
480 I$=INKEY$:IF I$="" THEN480
490 IF I$="1" THENXX=3 ELSEIF I$
="2" THENXX=2 ELSEIF I$="3" THEN
XX=1 ELSE GOTO480
500 PCLS:PMODE3,1:SCREEN1,1:PCLS
510 LINE(0,0)-(40,40),PSET:LINE(
216,40)-(256,0),PSET:LINE(40,40)
-(216,40),PSET:LINE-(216,88),PSE
T:LINE-(40,88),PSET:LINE-(40,40)
,PSET:LINE(40,88)-(0,132),PSET:L
INE-(256,132),PSET:LINE-(216,88)
,PSET:PAINT(120,120),8,8
520 DRAW"BM10,160;S8C3"+Q$
530 DRAW"BM140,160;"+Q1$
540 F=F+1:IFF)1 THEN600
550 DRAW"BM44,105;S4C3XR1$;BM30,
130;XR2$;":PAINT(45,100),6,7
560 PAINT(32,128),6,7
570 GET(26,109)-(55,130),R,G:GET
(42,89)-(67,105),S,G
580 GET(70,109)-(99,130),B,G:GET
(70,89)-(95,105),N,G
590 GET(85,0)-(135,10),C,G
600 FORX=26TO240STEP44:PUT(X,109
)-(X+29,130),R,PSET:NEXTX
610 FORX=42TO170STEP34:PUT(X,89)
-(X+25,105),S,PSET:NEXTX
620 PUT(191,89)-(216,105),S,PSET
630 X=RND(6)
640 ONX GOTO650,660,670,680,690,
700
650 DRAW"BM40,88;S16C8U3R15E4FE2
BL2BGU2EF5DE3RFEF2UEBL6E3 F6R7":
GOTO710
660 DRAW"BM40,88;S16C8U6F2R4ER3F
2R4ERFRFRER2E2F2RE2RE2RERDFEREBL
4E2F3R":GOTO710
670 DRAW"BM40,88;S16C8ER4E2FR5E4
F2E2FE2BL6BGUE3F2RF3RE4FRF2RFR3F
3R":GOTO710
680 DRAW"BM40,88;S16C8E2RURUE2F2
UEBL3U3F3D3F2E3R2EREFRF2R5E2REF3
RFRFRE4":GOTO710
690 DRAW"BM40,88S16C8U4E4F2E2BL4
E2F7R2DE4FEF2E2BL6E3F3R":GOTO710
700 DRAW"BM40,88S16C8E2RERE5F2E2
FEBL6E3F6R2F4R19"
710 PAINT(60,86),6,8
720 PAINT(42,42),7,8
730 DRAW"BM53,102;S8C5"+PP$+"BM8
8,102;"+PI$+"BM121,102;"+PC$+"BM
155,102;"+PK$+"BM200,102;"+PA$+"
BM34,124;"+PS$
740 DRAW"BM82,124;"+PT$+"BM124,1
24;"+PO$+"BM167,124;"+PN$+"BM212
,124;"+PE$
750 DRAW"BM20,10;S8C7"+PS$+PC$+P
O$+PR$+PE$
```

```
760 GOSUB1160
770 DRAW"BM81,30;C7"+PD$+PR$+PA$
+PG$+PO$+PN$+PS$
780 DRAW"BM145,10;"+PU$+PA$+PV$+
PE$
790 IFXX=3 AND YY=(1 THEN DRAW"B
M10,135;S4C7R8D6G2C8NH6C7G2H4U6B
R18"
800 IFXX=3 AND YY=0 THEN DRAW"S4
C7R8D6G2C8NH6C7G2H4U6BR18"
810 IF XX=2 AND YY=0 THEN DRAW"B
M10,135;S4C7R8D6G2C8NH6C2C7G2H4U
6BR18BR18"
820 LE=LE+1:D1=0:D2=0:D3=0:CH=1:
X=0
830 D1=RND(10)
840 IFLE)2 THEN850ELSE880
850 D2=RND(10):IFD2=D1 THEN830
860 IFLE)8 THEN D3=RND(10)
870 IFD3=D1 OR D3=D2 THEN860
880 IFLE(3 THENL=1 ELSEIF LE)2 A
ND LE(9 THEN L=2 ELSEIF LE)8 THE
NL=3
890 DRAW"BM180,30;S8C6"+NB$(L):D
RAW"BM212,10;":LE$=STR$(LE):L=LE
N(LE$)-1:FORX=2TO L+1:DRAWNB$(AS
C(MID$(LE$,X,1))-48):NEXTX
900 B1=0:B2=0:B3=0:B4=0:B5=0:B6=
0:B7=0:B8=0:B9=0:B0=0
910 I$=INKEY$:IFI$="" THEN910
920 IF I$="Q" THEN1450
930 IF I$="P" OR I$="I" OR I$="C
" OR I$="K" OR I$="A" OR I$="S"
OR I$="T" OR I$="O" OR I$="N" OR
 I$="E" THEN940ELSE910
940 ZO=RND(5):PLAY"V15T2000=ZO;B
AG":GOTO1050
950 IF D1=VAL(I$) THEN 1220
960 IFLE)2 AND D2=VAL(I$) THEN 1
220
970 IFLE)8 AND D3=VAL(I$) THEN 1
220
980 SC=SC+CH
990 CH=CH*2
1000 GOSUB1160
1010 IFLE(3 AND CH=512 THEN1640
1020 IFLE)2 ANDLE(9 AND CH=256 T
HEN1640
1030 IFLE=)10 AND CH=128 THEN164
0
1040 GOTO910
1050 ON INSTR("PICKASTONE",I$)GO
TO1060,1070,1080,1090,1100,1110,
1120,1130,1140,1150
1060 IFB1=1 THENGOTO910ELSE B1=1
:I$="1":PUT(42,89)-(67,105),N,PS
ET:GOTO950
1070 IFB2=1 THENGOTO910ELSE B2=1
:I$="2":PUT(76,89)-(101,105),N,P
SET:GOTO950
1080 IFB3=1 THENGOTO910ELSE B3=1
:I$="3":PUT(110,89)-(135,105),N,
PSET:GOTO950
1090 IF B4=1 THENGOTO910ELSE B4=
1:I$="4":PUT(144,89)-(169,105),N
,PSET:GOTO950
1100 IFB5=1 THENGOTO910ELSE B5=1
:I$="5":PUT(191,89)-(216,105),N,
PSET:GOTO950
1110 IFB6=1 THEN GOTO910ELSE B6=
1:I$="6":PUT(26,109)-(55,130),B,
PSET:GOTO950
1120 IFB7=1 THENGOTO910ELSE B7=1
:I$="7":PUT(70,109)-(99,130),B,P
SET:GOTO950
1130 IFB8=1 THENGOTO910ELSE B8=1
:I$="8":PUT(114,109)-(143,130),B
,PSET:GOTO950
1140 IFB9=1 THENGOTO910ELSE B9=1
:I$="9":PUT(158,109)-(187,130),B
,PSET:GOTO950
1150 IFB0=1 THENGOTO910ELSE B0=1
:I$="10":PUT(202,109)-(231,130),
B,PSET:GOTO950
1160 PUT(85,0)-(135,10),C,PSET:D
RAW"BM87,10;S8C6"
1170 SC$=STR$(SC)
1180 L=LEN(SC$)-1
1190 FORX=2TO L+1
1200 DRAWNB$(ASC(MID$(SC$,X,1))-
48)
1210 NEXTX:RETURN
1220 FORVZ=30TO1STEP-1:PLAY"V=VZ
;T80O2CAB":NEXT:FORVZ=30TO1STEP-
1:PLAY"V=VZ;T200IC":NEXT:YY=YY+1
1230 CLS:PRINT@97,"YE HATH UNCOV
ERED THE DRAGON!":FORX=1TO500:NE
XTX:IFYY()XX THEN500ELSEPRINT@219
,"YE HATH FOUND";SC:PRINT"   PI
ECES OF GOLD WHICH WILL      LE
T THY DECENDENTS LIVE IN     TH
E STYLE IN WHICH THEY        HA
VE BEEN ACCUSTOMED.";
1240 PMODE3,1:PCLS7
1250 DRAW"BM4,130;S12C5XHH$;XHI$
;BM184,148;S16XDR$;":PAINT(184,1
45),6,5:PAINT(10,10),5,5:SCREEN1
,1
1260 FORX=1TO400:NEXTX
1270 SS$="V20T25501EFGBCAEDAGFC"
1280 FORX=1TO5
1290 DRAW"BM160,76;C4D63L46L3":D
RAW"BM160,76;C4D2L362L6":DRAW"BM
160,76;C4G2L7":DRAW"BM160,76;C4G
L3UL2G2":PLAYSS$
1300 DRAW"BM160,76;C1D63L46L3":D
RAW"BM160,76;C1D2L362L6":DRAW"BM
160,76;C1G2L7":DRAW"BM160,76;C1G
L3UL2G2"
1310 NEXTX
1320 FORX=2TO60STEP4:CIRCLE(60,1
00),X,5:CIRCLE(70,110),X-2,5:NEX
TX
1330 LINE(4,158)-(118,30),PSET,B
F
1340 LINE(120,84)-(188,82),PSET,
BF
1350 FORX=1TO1000:NEXTX
1360 IF SC)HS(1) THEN HS(3)=HS(2
):HS(2)=HS(1):HS(1)=SC:GOTO1390
1370 IFSC)HS(2) THEN HS(3)=HS(2)
:HS(2)=SC:GOTO1390
1380 IFSC)HS(3) THEN HS(3)=SC
1390 CLS0:PRINT@202," GAME OVER
";:PRINT@266," HI SCORES ";:PRIN
T@298,USING"  ##,###   ";HS(1);:
PRINT@330,USING"  ##,###   ";HS(
2);:PRINT@362,USING"  ##,###   "
;HS(3);
1400 PRINT@418," DO YOU WANT TO
PLAY AGAIN? ";
1410 II$=INKEY$:IFII$=""THEN1410
1420 IF II$="N" THEN GOTO1440ELS
E IF II$="Y" THEN1430ELSE1410
1430 XX=0:YY=0:SC=0:WE=0:F=1:LE=
0:GOTO470
1440 CLS:FORVZ=30TO1STEP-3:PLAY"
V=VZ;T2504GDF":NEXT:END
1450 DRAW"BM20,180;S8C7"+UR$+"BM
80,180;"+UU$
1460 DRAW"BM130,180;"+UU$
1470 X=D1:DRAW"BM210,180;S8C7":G
OSUB1530
1480 IFD2=0 THEN1520
1490 X=D2:GOSUB1530
1500 IFD3=0 THEN1520
1510 X=D3:GOSUB1530
1520 FORA=1TO1000:NEXTA:GOTO500
1530 ON X GOTO1540,1550,1560,157
0,1580,1590,1600,1610,1620,1630
1540 DRAWPP$:RETURN
1550 DRAWPI$:RETURN
1560 DRAWPC$:RETURN
1570 DRAWPK$:RETURN
1580 DRAWPA$:RETURN
1590 DRAWPS$:RETURN
1600 DRAW"BR2"+PT$:RETURN
1610 DRAW"BR"+PO$:RETURN
1620 DRAWPN$:RETURN
1630 DRAWPE$:RETURN
1640 DRAW"BM20,180;S8C7"+PB$+PO$
+PN$+PU$+PS$
1650 DRAW"BR10"+NB$(5)+NB$(0)+NB
$(0)
1660 FORX=1TO2:SCREEN1,0:PLAY"04
T75CDEFGABT2501BAGFED":SCREEN1,1
:PLAY"03T75CDEFGAB01BEGFED":SCRE
EN1,0:PLAY"05T75CDEFGCDT2501FEDB
AG":SCREEN1,1:NEXTX:SC=SC+500:GO
TO500
```

HIND SIGHT

by Bob Rosen

(Bob Rosen owns the US company, Spectrum Products and is responsible for much of the advanced developments that have taken place over the last four years for the Colour Computer. G.)

In September of 1980 Radio Shack introduced the 4K Colour Computer for US$399. Computers that followed were the 16K Colour Computer, 32K Colour Computer, TDP-100, 64K Colour Computer, Dragon 32, MC-10, Colour Computer II and Dragon 64. What most Colour users have never heard of is a machine that was going to be called the Deluxe Colour Computer.

What is a Deluxe Colour Computer? Good question! I will proceed to tell you about the Computer that wasn't.
The Deluxe Colour Computer was scheduled to hit the market before the 1984 christmas season. The retail price was going to be US$399. It was going to include the following features: true lower case; full screen editing; two RS-232 ports; the second port having an ACIA; 32K RAM disk; keyboard with two additional function keys; 6-pin joystick inputs; sound chip; and colour video output.

The true lowercase was going to be a vast improvement over the current reverse video mode now used for lower case operation. The two RS-232 ports would have been a welcome addition for OS-9 users since the second serial port would have an ACIA chip in it. This would enable communications between two computers to operate up to 19,200 baud instead of the 300 baud limitation provided by the infamous bit-banging PIA chip. It should also be mentioned that the Deluxe's ROM would have had a built-in terminal program. You notice that I said ROM, singular?

The Deluxe was going to contain a single ROM that would have Standard, Extended and Disk BASIC all in one! This would have some problems with existing software, especially from the third party market. After going through the new ROMs trauma for the past year, I certainly wasn't going to lose any sleep about missing out on this feature of the Deluxe.

When I heard the Deluxe was going to have 6-pin joystick inputs I immediately thought of Apple joysticks with two fire buttons. This would have created a new dimension for games. Even Marty Goodman's "Graphicom" would have benefited from two fire buttons on a single joystick.

Colour video output is a feature I am going to miss. Nothing can match the vivid picture of a colour monitor. Many a user has suffered from severe interferance on their TV sets caused by disk drive cables, unshielded 300-ohm TV switch wire boxes and an assortment of hardware gizmos. Hooking a colour monitor straight to the back of your computer would solve any interferance problem. Fortunately, there are third party video drivers available. Still, having a video output on the back would have been nice.

Am I sorry the Deluxe was pulled off the market just before its final stages? The answer is yes and no. The features I mentioned are definitely improvements over the present Colour Computer II machine. But the two most important features that almost every owner would have wanted was more memory and a hi-res screen. The lack of memory expansion to 128K or more and a 80 by 24 screen would have taken some steam out of initial enthusiasm for this new computer. OS-9 software, especially business applications, must have more than the 32 by 16 screen and 64K memory supported by Tandy.

The lack of these two features doesn't supprise me; present Colour technology will not support them. The SAM chip must be redesigned or replaced for "true" 128K or 256K memory, and Motorola's new RMS and VDG chip set will not be available in initial supplies until the fall. Known as the "RMC chip set", it will have the following features: able to support up to 1 meg RAM, vertical resolution from 64 to 640 pixels, horizontal resolution from 64 to 500 pixels, 32 colours simultaneously out of a palette of 4096, smooth scrolling, lower case, up to 60 character rows by 40 or 80 columns, underlined characters, inverse characters, blinking characters, double height characters, double width characters, different coloured characters, and a 6883/6847 chip set emulation mode, too.

New rumors have evolved since the demise of the Deluxe that Tandy is moving up production of another machine. Already dubbed the "09" machine, it will be Tandy's answer to why they came out with OS-9 in the first place. As a follower of Radio Shack computers for the last seven years, all I can say is the only time to believe is when Radio Shack says it is so.

By the time you read this we may even have the answer!

R.M.S. CHIPSET

by Bob Rosen

(SPECTRUM PROJECTS)

(Raster Memory System) This is the new chipset that Motorola has been working on. It provides sophisticated video display and memory management for the 6809E, 68008, or the 68000. It is really two systems in one chipset: machine 1 mode provides the full power of this system and the machine 2 mode gives you the backward compatibility with 6809e systems using the Mc6883-Mc6847 combination.

HIGHLIGHTS

* 1 Megabyte Memory Capability
* Horizontal resolution 64 to 640 pixels
* Vertical resolution 64 to 500 pixels
* Bit-plane mode and six character/object - oriented list modes.
* 32 colours from a pallette of 4096
* 32 to 32000 user-definable characters in the listmodes.
* Text oriented attributes: Underline Flash, Invert, Colour, Double height, and double width
* ASCII and mosaic characters in internal ROM
* Game oriented attributes: Collision, priority, colour offset
* Virtual screen that is larger than the visible screen, smooth scrolling
* 8 hardware objects positioned by X,Y retisters

RMS uses a block of Dram as screen memory as does the CoCo, but by using registers within the RMS this display can be larger than the actual display screen. The Dram can be four banks of: 16K, 64K or 256K of dynamic RAM, these banks can be used one at a time or expanded to the full memory capability. One thing to remember is that when using one bank of memory the system is slowed down somewhat because it is not able to spread out its access to other parts of the system. When there are two banks or more they are time division multiplexed on to the data bus as well as dram refresh and mpu access in the same cycle, also more colours in bit plane can be accessed.

REGISTERS

This mode of operation refers in part to the Machine #2 function of the system. We will refer to this as the CoCo mode as it is an emulation of the 6883(74LS783)-6847 (or the new 6847-T1 chip which has a full 96 chr set and is made for the 6883). It looks like this: FF22 is one of the registers in the 2nd PIA in the CoCo, this is emulated by a register in the RMS chipset, so that FF20-23 are not used by the normal decode that we know, instead the FF24-FF3F is used in place of this if a PIA is used for other things. (list#2) The other registers we know so well are in their usual place, except the memory switch M1, M0, and the speed mode. (FFd6-FFdd). These pokes within are not used, the page flip switch. FFd4-d5 is useable as is the type page mode. FFde-ffdf, the 64K mode is accessable just like our CoCo.

This takes care of what registers we know, we now take a look at the RMS registers, which in this mode are located at: FF80-FFbf. List #2 is the type #0 CoCo mode we would be using if we were running CoCo software. The only problem would be access to FF20-FF21 and FF23. These will have been changed to FF24-FF27.

In fact the whole structure of the PIA's would be very different as there is no interlink to the first PIA as in the CoCo from the VDG chip (the vertical and horizontal interupt lines) as these are now connected between two RMS chips.

Now comes the part that is interesting to those of you who will run out and buy one as soon as it appears. This is the makeup of the Registers at FF80-FFbf in the initialization of this mode. You will note that changes can be made, after initialization, many possible changes could be done depending on your knowledge and expertese in machine language.

Registers	Value	Comments
memory map	$10	Machine #2, folded map page 00
display data mode	$00	
interupt statue	$00	
border colour	$10	Cmr10, video disabled
object available	N/A	read only
paging	$0F	page F
page independent blocks	$00	top or botton or both of 64K memory can be reserved for Basic pointers and screens that are not to be moved when paging (memory management)
vertical scroll	$00	Move the screen one line at a time
horizontal scroll	$00	Move screen horizontally
DRC start address	$0000	Dynamically Redefinable character set (characters that you define within memory)
True object start address	$0000	Objects that can be moved at will around the screen (8 register=8 objects at one time or more in real time, can collide with other objects, or fixed ones, this will set the collision status X,Y location to be read by software.)
Fixed object start address	$0000	as above
Collision status	N/A	read only
Collision enable	$00	

Real time output $00000000 generates an interupt when the CRT beam reaches the specified X,Y RTO setting
Real time input $00000000 an input pulse on this (lightpen) puts the current X,Y location into the register for your software to read. There is an input line for this function

Video operation	$12	256 x 192 sets screen size
Sync mode	$04	composite sync output
Virtual Screen start address	$00000000	with this you set the size of memory you need larger than the screen itself, then move the screen within it (list #3)
Vertical offset	$00000000	
Horizontal offset	$00000000	
Virtual screen size	$00001800	6K screen (CoCo)
Virtual screen width	$00000020	32 bytes (CoCo)

Colour Map Register

CMR 00	$40f0	green this ms-4 keeps the RMS on page one
CMR 01	$0FF0	yellow
CMR 02	$000F	blue
CMR 03	$0F00	red
CMR 04	$0FFF	buff
CMR 05	$00FF	cyan
CMR 06	$0F0F	magenta
CMR 07	$0F80	orange
CMR 08	$0000	black
CMR 09-CMR0F	$0000	
CMR 10	$00F0	green- Alpha/Semigraphics border colour
CMR 11	$0F80	orange - as above
CMR 12	$00F0	green - graphics border colour
CMR 13	$0FFF	buff - as above
CMR 14-CMR 1F	$0000	

This completes the main mapping of the CoCo RMS system the numbers in the registers have what is put there to emulate the CoCo. Take note: this system is in Tandy's future, it will be, as rumour puts it, called the MODEL 9! Like a model 4 with colour and OS-9 level 2.

I hope you have enjoyed this in depth look at this new and complex chip set, I can only give you small glimpes at the capabilities of this system (the users manual is 146 pages long). There are so many things that this can do. The manual is obtainable from:-

Motorola
Semiconductor Products Section,
Consumer Strategic Marketing.
Phoenix, Arizona.
(602)244-6381. User's manual for the:-
Raster Memory System Version 3.00 April 15, 1984.

PEEKS, POKES and EXECS

by Mark Rothwell, Herman Fredrickson & Les Thurbon

(A new HELP book is being prepared at present. Designed to give much more than the information available in the manuals, the new HELP, yet to be found an official title, will include a lot of information for beginners, plus worthwhile reference material for longtimers.

There will be a section on POKES, and the following article forms the nucleus of that section. If your favourite POKE or EXEC is not listed here, when you write next, make sure you tell us about it! G.)

Here is a list of some interesting Peeks, Pokes and Execs that may help with programing. We give the command, then the result with any remarks that should be considered.

BAUD RATES

```
POKE 149,4:POKE 150,88    --  Baud Rate 50
POKE 149,2:POKE 150,227   --  Baud Rate 75
POKE 149,1:POKE 150,246   --  Baud Rate 110
POKE 149,1:POKE 150,153   --  Baud Rate 134.5
POKE 149,1:POKE 150,110   --  Baud Rate 150
POKE 149,0:POKE 150,180   --  Baud Rate 300
POKE 149,0:POKE 150,87    --  Baud Rate 600
(Basic Default)
POKE 149,0:POKE 150,40    --  Baud Rate 1200
POKE 149,0:POKE 150,25    --  Baud Rate 1800
POKE 149,0:POKE 150,23    --  Baud Rate 2000
POKE 149,0:POKE 150,18    --  Baud Rate 2400
POKE 149,0:POKE 150,10    --  Baud Rate 3600
POKE 149,0:POKE 150,7     --  Baud Rate 4800
POKE 149,0:POKE 150,3     --  Baud Rate 7200
POKE 149,0:POKE 150,1     --  Baud Rate 9600
```

If this command is placed at the beginning of the program, the Baud Rate will be set at the speed desired, until the computer is COLD started.

PRINTER POKES

```
POKE 155,16  --  Sets Line Width to 16 Characters
per line.
POKE 155,32  --  Sets Line Width to 32 Characters
per line.
POKE 155,64  --  Sets Line Width to 64 Characters
per line.
POKE  155,132  --  Sets Line Width to 132
```

Characters per line.
POKE 155,255 -- Sets Line Width to 255 Characters per line.
POKE 111,254:DIR -- Prints Disk Directory to Printer.
POKE 111,254:EXEC 46946
Lists a Basic Program in Memory to the Printer, same as LLIST.
PEEK(65314) -- Returns an even number if printer is on or an odd number if printer is off.
10 POKE111,254:DIR:PRINT#-2,"FREE=)";:PRINT#-2,FREE(0)
As a program line this will give a directory of a disk, and at the bottom of the listing the number of freegranules.

DISK SYSTEM ONLY

POKE 25,14:POKE 26,1:POKE 3584,0:NEW
Clears all Graphic Pages and makes use of maximum available memory. Graphics are no longer available.
POKE 25,20:POKE 26,1:POKE 5120,0:NEW
Sets Basic Program Starting Address at PCLEAR 1. Limited Graphics available.
POKE 25,26:POKE 26,1:POKE 6656,0:NEW
Sets Basic Program Starting Address at PCLEAR 2. Limited Graphics available.
POKE 25,32:POKE 26,1:POKE 8192,0:NEW
Sets Basic Program Starting Address at PCLEAR 3. Limited Graphics available.
POKE 25,38:POKE 26,1:POKE 9728,0:NEW
Sets Basic Program Starting Address to PCLEAR 4. This is the Basic default.
POKE 2439,255 -- Turns Verify on.
POKE 2439,0 -- Turns Verify off.
PEEK(2439) -- Returns 255 if Verify is on.
PEEK(2439) -- Returns 0 if Verify is off.
PEEK(49152) -- Returns 68 if DECB is present.
POKE 65344,0 -- Turns all Disk Drive motors off.
EXEC 49364 -- Warmstart Routine DECB 1.0 only.
EXEC 49383 -- Warmstart Routine DECB 1.1 only.
EXEC 52175 -- Displays Directory to Screen DECB 1.0.
EXEC 52393 -- Displays Directory to Screen DECB 1.1.

SPEED UP POKES

POKE 65495,0 -- Double Speed.
POKE 65494,0 -- Return to Normal after above POKE.
POKE 65497,0 -- Triple Speed. Disables Screen, OK with WORDPAK.
POKE 65496,0 -- Returns to Normal after above POKE.

COLD START

POKE 113,0 :EXEC40999 -- Cold Start Routine.

ORANGE SCREEN

POKE359,13:SCREEN0,1

MACHINE LANGUAGE ADDRESSES

START ADDRESS -- PRINT PEEK(487)*256+PEEK(488)
END ADDRESS -- PRINT PEEK(126)*256+PEEK(127)-1
EXEC ADDRESS -- PRINT PEEK(157)*256+PEEK(158)
CSAVEM or SAVEM"FILENAME", START ADDRESS, END ADDRESS, EXEC ADDRESS

AUTOMATIC INKEYS

EXEC 44539 -- Any key except (BREAK) or (SHIFT) no cursor.
EXEC 41393 -- Any key except (BREAK) or (SHIFT) with cursor.

BREAK KEY DISABLE

POKE248,50:POKE249,98:POKE250,28:POKE251,175:POKE252, 126:POKE253,173:POKE254,165:POKE410,126:POKE411, 0:POKE412,248

TO READ PROGRAMS SAVED WITH HIGH SPEED POKE (65495,0)

To read with normal speed:
POKE143,8:POKE144,24:POKE145,4
To read with computer in high speed:
POKE143,13:POKE144,24:POKE145,6

TO STOP BASIC LISTING

POKE383,158 to return to normal POKE 383,57

DISK DRIVES

PEEK(235) - returns current drive number
PEEK(236) - returns current track number
PEEK(237) - returns current sector number

OTHERS

PEEK(274)*256+PEEK(275) - returns value of TIMER
PEEK(49)*256+PEEK(50) - returns current DATA line number
PEEK(207)*256+PEEK(208) returns circle radius of last circle in PMODE4

EXEC 34471 same as TRON
EXEC 34472 same as TROFF
EXEC &HCCA9 same as DIR

FORTH

This article should be read in conjunction with the use of the FORTH compiler John provided in August 1984's CoCoOz.

We invite your reaction to the Forth articles, and details of the discoveries you are making as you use them.

PART 7.

ALL LOOKING NEAT and TIDY

by John Redmond

(Forth has been a part of Australian CoCo for the past six months or so. John Redmond provided a Forth compiler for you in August 1984 CoCoOz, and mostly it can be used in conjunction with the exercises discussed here.

John demonstrated Forth at CoCoConf.

The speed of the language is amazing. Keep in mind that Graphicon is written in Forth - a most worthy demonstration of the power of Forth.

We regret we are unable to reprint the articles printed so far in Australian CoCo, but we would certainly refer you to them, and also to the book "Starting Forth" by L. Brodie (Prentice Hall). This book was reviewed in Australian CoCo in June this year, and John uses it as his text book.

Next month John Poxon takes over from John Redmond, who has earned a well deserved rest, and begins a series of articles dealing with some of the more practical aspects of Forth. John (Poxon) will be using the CoCoConnection to demonstrate the principles involved.

I think we will be in a position to announce a Users' Group for Forth in the near future, so don't worry about Forth - just get into it and have a go! G.)

The more common word is a little more complicated. It needs to determine the sign of the integer and then all the arithmetic is done on the absolute value of the number (obtained with DABS). The definition is:

```
: . DUP 0 DABS
  <# #S SIGN #> TYPE SPACE ;
```

The DUP at the beginning saves the original integer (which has the sign bit) before DABS does its work. SIGN uses this third stack item and places a '-' into the string buffer if it is negative. Starting to click? Double-length number output is even simpler to code for:

```
: UD. <# #S #> TYPE SPACE ;
: D. SWAP OVER
  <# #S SIGN #> TYPE SPACE ;
```

The words use in these definitions are just the tools we need for specialized number output. One more word and the toolkit is complete. The word is HOLD. This permits us to insert any ASCII character, at any time, into the string buffer. It works in well with #.

```
: .$ SWAP OVER
  <# SIGN # #
   ASCII . HOLD
   #S ASCII $ HOLD #>
   TYPE SPACE ;
```

This new word permits us to type a double-length integer as a dollar amount, with a trailing sign (because SIGN was invoked before #). There is no end to what you can do with numeric output. Consider the normally irritating problem of outputting a time in seconds as

hours:mins:seconds.

This requires a few definitions:

```
: SEXTAL 6 BASE ! ;
: .00 # SEXTAL # DECIMAL
  ASCII : HOLD ;
: .SECONDS DECIMAL
  <# .00 .00 #S #>
  TYPE SPACE ;
```

The word .SECONDS prints a double-length unsigned integer in the required form.

As an exercise, define a word .DATE, which will take a double-length integer as a date in the form DD/MM/YY. This is an extremely memory-efficient trick. The date will be printed as a string of 8 characters, even though it is stored in memory as only four bytes. This sort of compression can be very valuable in a database system. The arithmetic is simple: if you have a modest data file of only 256 records, each with a date, and you manage to save 4 bytes per record, you have saved 1K in space. We'll see something of databases later in this series. They're a snack with Forth.

Last month, we talked about scaled integers and gave

```
100,000,000 355 113 M*/
```

as a good approximation to pi.

The trouble was that it was 100,000,000 times larger than it should have been. If we are going to print it, we need to print it with eight figures after a decimal point. See how? The crudest way would be:

```
: .REAL SWAP OVER DABS
  <# # # # # # # # # #
   ASCII . HOLD #S SIGN #>
   TYPE SPACE ;
```

Perhaps we can make this definition cleaner and more general. We printed the result to eight places, but what if it had been six or four? Lets store the current value in a variable called PRECISION:

```
VARIABLE PRECISION
8 PRECISION !
```

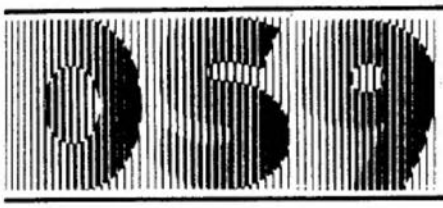
Now we can define a word which uses this current value:

```
: .PLACES PRECISION @ 0
  DO # LOOP ASCII . HOLD ;
```

Now .REAL becomes:

```
: .REAL SWAP OVER DABS
```

continued on P 53



Jacky from Paris Radio demonstrated his 128K upgrade at CoCoConf. It works well with RS DOS, but there were still a few problems with it working under OS-9, which should be fixed by now.

Used as a RAM disk, the unit is especially impressive. Jacky had it hooked up as "drive 2", and it was a simple task to type "BACKUP x TO 2", thereby loading the contents of x disk to the RAM disk. Access speeds were lightening fast, and improve CoCo's performance significantly.

Imagine what it will do for OS-9!

Another Jack, Jack Fricker has been working with Stylograph, attempting to delete a lot of the options you don't normally use. Such things as the ACIA pack, T2, and T1 if you are not using a terminal, can be deleted and free up some memory.

We are honoured to have two programs in Basic 09 this month, submitted by Australians. We would ask that when you send printed OS-9 programs, you use your printer's compressed mode if it has one. This makes for a more standard magazine layout.

HDIR.C

by M.R. Delahunty

In OS9, with all the different layers of directories that can be created on a disk, it can be quite difficult to find all the files that are hiding in all the various sub-directories on one disk.

My version of Hdir (there have been others), will display all the files in all the sub-directories of any directory that you specify on the command line. It also has 5 options that control the format of the displayed information.

There are two main types of displays:

1. Will print one file name per line with filenames in the same directory in the column. It will tab over 3 or more spaces if any file is a directory and print files in that sub-directory starting at the new column position. When it gets to the end of that directory it will backspace 3 or more columns and continue. This format can use a lot of paper if the output is directed to a printer, but there is room to jot down a few comments.

2. The other format is more compressed and needs to know how many columns your output device has. It is similar in that it indents each sub-directory, but more than one filename can be printed on the same line. Each directory name will be preceded by a 'x'. Some error reporting is done.

The 5 options are specified by 5 different letters (n,c,i,n and r) followed immediately by an '=' followed immediately by a decimal number or string. The 'n' option will set the left margin to the value of the number. The 'c' option will activate the compressed listing, and the number should be equal to the number of columns in your output device. The 'i' option will set the amount of indentation for different directory levels. The minimum and default is 3 and maximum is 10. This command will print a title line like "Hierarchical directory of 'DNAME'", where 'DNAME' is the directory name on the command line. The 'n' option can be used to replace 'DNAME' with a more useful namestring. Spaces within this string are not allowed, and it must immediately follow the preceding '='. A lot of backspaces are normally used by this program to control the position of the cursor. Some printers don't like this and also if your output is redirected to a disk file it will not list out correctly. The 'r' option will inhibit the use of backspaces by using 'CR' and 'SPACES' instead. The 'r' option will also set the number of nulls to send after each carriage return (0-255).

This command will individually check each file to see if it is a directory and therefore may take a little time to process a disk with many files on it. It could be useful to redirect the output to a disk file (use 'r' option) called "/D0/contents" for example, this can be listed out a lot quicker. If no options or directory name are given on the command line then 'Hdir' will display a help message. Some command line examples follow:

```
hdir /do c=51 n=5
HRIR n=10 /d1 c=80 )/p
hdir i=5 n=3 c=80 n=BASIC09 r=60
/d1/basic09 )/p
hdir n=20 n=USER GROUP )/p
hdir
hdir  /20 r=0 c=51 n=THIS DISK
)/d2/contents
hdir  /d2 c=51 n=this disk r=0 !
tee /d2/contents
hdir c=51 n=10 c=80 )/p
```

This last example will set the number of columns to 80 (the last option of each type is the one that counts). It will also use the current data directory.

Hdir was written in the C programming language and you will need a C compiler to compile it. The command 'CC1

hdir.c' should compile it without any trouble unless you have any errors in your source file (hdir.c). The program starts off by defining several global variables, these can be manipulated by all the functions within the program.

Main () sets a few default values, scans the command line and set all the options if any, then it prints out the title line and calls pdir(). I have used the write() system call instead of printf() to print output because it reduced the size of the finished program considerable.

Pdir() will open up the first directory, gets the first file name, does a directory test on it, if it is an ordinary file it will print it according to the options. If it is a directory it first prints the name on a new line with a preceding '*' then calls itself (pdir()) with this new file name becoming an argument of pdir. When all file names in the current directory have been read getname() returns 0 for end of file so the main while loop fails and the cursor is moved back a few or more columns, the current directory is closed and depending on how many directory levels have been entered it will either continue reading files in a higher directory or go back to main() and finish.

Getname() will also return -1 if an error has occured reading a file) will reset the high order bit of the last letter tory or not.

Printname() will calculates if a filename, cr() and ftab() to move the cursor.

Finally Help() will as I do.

reduced the size of the finished program considerable.

Pdir() will open up the first directory, gets the first file name, does a directory test on it, if it is an ordinary file it will print it according to the options. If it is a directory it first prints the name on a new line with a preceding '*' then calls itself (pdir()) with this new file name becoming an argument of pdir. When all file names in the current directory have been read getname() returns 0 for end of file so the main while loop fails and the cursor is moved back a few or more columns, the current directory is closed and depending on how many directory levels have been entered it will either continue reading files in a higher directory or go back to main() and finish.

Getname() will also return -1 if an error has occured reading a file name and pdir() will call printname() to print an error message. Getname() will reset the high order bit of the last letter in the filename and add a null at the end.

Dirtest() tests to see if the file is a directory or not.

Printname() will calculates if a filename will fit on the current line and keeps tabs on the cursor position. It calls backup(), cr() and ftab() to move the cursor.

Finally Help() will write a help message if needed.

That's about it, I hope you find this command as useful as I do.

```
#include <ctype.h>
int column ;
int colno ;
int dflag ;
int maxcol ;
int rflag ;
int iopt ;
main(argc,argv)
int argc ;
char *argv[] ;
{
int nflag ;
int cflag ;
int ntab ;
char *dflt ;
char *lf ;
char *pointer ;
char *title ;
char namest[33] ;
char dname[128] ;
int i ;
int tab ;
cflag = 0 ;
rflag = 0 ;
nflag = 0 ;
iopt = 3 ;
tab = 0 ;
dflt = "." ;
lf = "\n\l" ;
title = "Hierarchical directory
of " ;
strcpy(dname,dflt);
if(argc == 1)
  {
  help() ;
  exit(0) ;
  }
else
  {
  for(i=1;i < argc ;i++)
     {
     pointer = argv[i] ;
     if(tolower(*pointer)
'm')
        if(*(++pointer) == '-')
           {
           tab = atoi(++pointer
) ;
           if( tab < 0)
             tab = 0 ;
           continue ;
           }
        else
           {
           strcpy(dname,argv[i])
;
           continue ;
           }
     else if(tolower(*pointer)
== 'c')
        if(*(++pointer) == '-')
           {
           maxcol =
atoi(++pointer) -1 ;
           if(maxcol < 30)
             cflag = 0 ;
```

```
            else
               cflag = 1 ;
            continue ;
            }
         else
            {
            strcpy(dname,argv[i])
;
            continue ;
            }
      else if(tolower(*pointer) --
'n' )
        if(*(++pointer) -- ' ')
          {
          strcpy(namest,++pointer)
;
          namest[32] = '\0' ;
          nflag = 1 ;
          continue ;
          }
        else
          {
          strcpy(dname,argv[i]);
          continue ;
          }
      else
if(tolower(*pointer)--'i')
        if(*(++pointer)--'-')
          {
          iopt = atoi(++pointer);
          if(iopt < 3)
            iopt = 3 ;
          if(iopt >10)
            iopt = 10 ;
          continue ;
          }
        else
          {
          strcpy(dname,argv[i]);
          continue;
          }
      else if(tolower(*pointer) --
'r')
         if(*(++pointer) -- '-' )
          {
          rflag= atoi(++pointer) ;
          if(rflag < 1)
             rflag=1 ;
          if(rflag > 255)
             rflag = 255;
          continue ;
          }
         else
           {
           strcpy(dname,argv[i]) ;
           continue ;
           }
      else
         {
         strcpy(dname,argv[i]) ;
         }
      }
   }
if(cflag)
   {
   if(nflag)
     {
     ntab = (maxcol - 26 -
strlen(namest)) / 2 ;
     ftab(ntab) ;
     write(1,title,26);
     printname(namest,0) ;
     }
   else
     {
     ntab = (maxcol -26 -
strlen(dname))/2 ;
     ftab(ntab) ;
     write(1,title,26);
     printname(dname,0);
     }
   }
else
   {
   ftab(tab) ;
   write(1,title,26) ;
   if(nflag)
      printname(namest,0) ;
   else
      printname(dname,0) ;
   }
write(1,lf,2) ;
column = tab ;
ftab(tab) ;
pdir(dname,cflag) ;
write(1,lf,2) ;
}

pdir(dname,cflag)
char *dname ;
int cflag ;
{
int pn ;
int i ;
int getname() , dirtest() ,
printname() ;
char *bs ;
char fname[32] ;
char *error ;
char *parentdir ;
char *did ;
bs = "\b\b\b\b\b\b\b\b\b\b" ;
error = "ERROR in file" ;
parentdir = ".." ;
did = "*" ;
pn = open(dname,129) ;
chdir(dname) ;
ftab(iopt) ;
column += iopt ;

while(i = getname(fname,pn))
   {
```

```
    if(i -- -1)
       {
       dflag = 0 ;
       printname(error,cflag) ;
       break ;
       }
    if(dirtest(fname))
       {
       printname(fname,cflag) ;
       if(rflag)
          ;
       else
          write(1,bs,1) ;
       pdir(fname,cflag) ;
       }
    else
    {
    printname(fname,cflag) ;
    continue ;
    }
    }
    if(rflag)
       ;
    else
       write(1,bs,iopt) ;
    column -= iopt ;
backup() ;
chdir(parentdir) ;
close(pn) ;
}

getname(fname,pn)
char fname[] ;
int pn ;
{
int i ;
do
{
   if((i = read(pn,fname,32)) --
0 )
      return(0) ;
   else if(i -- -1)
      return(-1) ;
}
while(!fname[0]    fname[0] --
'.'    fname[0]    ('.' '\d128') ) ;

for(i=0;fname[i] > 0;i++)
   ;
fname[i] -= 128 ;
fname[++i] = '\0' ;
return(1) ;
}



printname(fname,cflag)
char fname[] ;
int cflag ;
{
int space ;
int len ;
char *did ;
char *bs ;
char *lf ;
int x ;
int i , j ;
did = "*" ;
bs = "\b" ;
lf = "\l" ;
if(cflag)
   {
   len = strlen(fname)  ;
   x = (len / 10) + 1 ;
   space = 10 - (len % 10) ;
   if(dflag || column + (x * 10)
>= maxcol )
      backup() ;
   }
if(dflag)
   write(1,did,1) ;
for(i=0;fname[i] != '\0' ;i++)
   write(1,&fname[i],1) ;
if(cflag)
   {
   ftab(space) ;
   colno += x ;
   column = column + (x*10) ;
   if(dflag)
      backup() ;
   }
else
   {
   if(rflag)
     {
     cr() ;
     ftab(column) ;
     }
   else
     {
     for(j=0;j <i;j++)
       write(1,bs,1) ;
       write(1,lf,1) ;
       }
   }
}



dirtest(fname)
char fname[] ;
{
if(access(fname,128) -- -1)
    {
    dflag = 0 ;
    return(0) ;
    }
dflag = 1 ;
return(1) ;
}
```

```
ftab(tab)
int tab ;
{
int i ;
char *space ;
space = " " ;
for(i=0;i < tab;i++)
    write(1,space,1) ;
}


backup()
{
int i ;
char *bs ;
char *lf ;
bs = "\b";
lf = "\l" ;
if(rflag)
  {
  cr() ;
  ftab(column -(colno*10));
  }
else
 {
 for(i=0;i < colno * 10;i++)
    write(1,bs,1) ;
    if(colno)
       write(1,lf,1) ;
    }
column = column - (colno * 10) ;
colno = 0 ;
}

cr()
{
int i ;
char *null ;
char *r ;
null = "\0" ;
r = "\n" ;
write(1,r,1) ;
for(i=0;i < rflag;i++)
   write(1,null,1) ;
}

help()
{
char *mess ;
char *mess1 ;
char *mess2 ;
char *mess4 ;
char *mess3 ;
char *mess5 ;
mess = "\n\lHDIR [opts] DirName
[opts]\n\loptions are\n\l" ;
mess1 = " m=no. left margin
val\n\l c=no. of columns in out
dev.\n\l      for compressed
format\n\l" ;
mess2 = " n=<namestring> maxlen
- 32\n\l      no spaces, replaces
DirName\n\l" ;
mess3 = " r=no. of nulls after
\"CR's\"\n\l      and no
backspaces\n\l" ;
mess4 = " i=no. to indent
directories by\n\l      min &
default = 3, max = 10\n\l" ;
mess5 = "Hierarchical
Directory\n\l by
M.R.Delahunty\n\l" ;
write(2,mess5,44) ;
write(2,mess,43) ;
write(2,mess1,83) ;
write(2,mess2,63) ;
write(2,mess4,67);
write(2,mess3,55);
}
```

MAILLISTFILEMAN

PART 2

by Tony Ceneviva

We want once more to thank Tony and CoCoPug's Editor, Stuart Hall, for their permission to use this program. This second part involves the actual loading of Data, so if you have completed Part 1 from last month, then into it!

THE LISTING:

```
PROCEDURE DATAFILELOAD
 0000       REM BY A. CENIVIVA 30 HATFIELD WAY BOORAGOON W.A. 6154
 0041       DIM SALNAME:STRING[24]                                TEL 3644557
 004D       MAXBLOCKS=100
 0055       TYPE BLOCKTYPE=SALUT:STRING[9]; NAME:STRING[15]; HOUSENO:STRING
             [5]; STREET:STRING[19]; SUBURB:STRING[16]; TELEPH:STRING
             [10]; FEES:STRING[5]
 00A7       DIM WFILENAME:STRING
 00AE       DIM DISKPOINTER:REAL
 00B5       DIM PRINTERPATH:BYTE
 00BC       DIM PPATH:STRING
 00C3       PPATH="/P"
 00CC       DIM X,Y,Z,W:INTEGER
 00DF       DIM SALUTSTART:BYTE
 00E6       DIM PATH:BYTE
 00ED       DIM WPATH:BYTE
 00F4       DIM BLOCK(100):BLOCKTYPE
 0102       DIM BUFEND:REAL
 0109       DIM CLIENTLISTFILE:STRING
 0110       DIM DISTRBUF(101):STRING[1]
 0121       DIM MRSPOINT:BYTE
 0128       DIM TESTCHR:STRING[1]
 0134       DIM SWAPSWITCH:BOOLEAN
 013B       INPUT "PATHLIST ?",CLIENTLISTFILE
 014D       INPUT "NAME OF WRITEFILE",WFILENAME
 0166       CREATE #WPATH,WFILENAME:UPDATE
 0172       OPEN #PATH,CLIENTLISTFILE:READ
 017E       SEEK #PATH,0
 0187       PRINT CHR$(12)
 018C       PRINT "LOADING FROM DISK ONE BLOCK AT A TIME"
 01B5       REM ***** LOAD FROM DISK ONE CHARACTER AT A TIME
 01E4       SWAPSWITCH=FALSE
 01EA       DISKPOINTER=0
 01F2       Z=0
 01F9       X=0
 0200       Y=0
 0207 100
 020B       REM *** FINDS FIRST '$'
 0221
 0222       X=1
 0229       SEEK #PATH,Z
 0233       LOOP
 0235       EXITIF DISTRBUF(X)="$" OR X>100 OR EOF(#PATH)=TRUE THEN ENDEXIT
```


STOP PRESS: Paris Radio announces the arrival of the first hard disk drives for your CoCo!
They have also received OS-9 system software for their 128K upgrades.

```
0256          X=X+1
0261          GET #PATH,DISTRBUF(X)
026F          Z=Z+1
027A          PRINT DISTRBUF(X);
0283        ENDLOOP
0287        Y=Y+1
0292        IF EOF(#PATH)=TRUE THEN 1000
02A1        IF X>100 THEN
02AD          PRINT CHR$(12)
02B2          PRINT "BLOCK TOO LARGE DISREGARDED "; Y
02D6          Y=Y-1
02E1          W=1
02E8          REPEAT
02EA            PRINT DISTRBUF(W);
02F3            W=W+1
02FE          UNTIL W>50
0309          Z=Z-10
0314          INPUT "PRESS ENTER ",TESTCHR
0328        ENDIF
032A        IF X>100 THEN 100
0339        REM *** LOAD UP TO NEXT $
0351        LOOP
0353          X=X+1
035E          GET #PATH,DISTRBUF(X)
036C          Z=Z+1
0377          PRINT DISTRBUF(X);
0380        EXITIF DISTRBUF(X)="$" OR X>100 OR EOF(#PATH)=TRUE THEN ENDEXIT
03A1        ENDLOOP
03A5        IF EOF(#PATH)=TRUE THEN 1000
03B4        IF X>100 THEN
03C0          PRINT CHR$(12)
03C5          PRINT "BLOCK TOO LARGE DISREGARDED"; Y
03E8          Y=Y-1
03F3          W=1
03FA          REPEAT
03FC            PRINT DISTRBUF(W);
0405            W=W+1
0410          UNTIL W>50
041B          Z=Z-10
0426          INPUT "PRESS ENTER TO CONTINUE",SALNAME
0445        ENDIF
0447        IF X>99 THEN 100
0456        REM *** LOAD UP TO CR.
046B        LOOP
046D          X=X+1
0478          GET #PATH,DISTRBUF(X)
0486          Z=Z+1
0491          PRINT DISTRBUF(X);
049A        EXITIF DISTRBUF(X)=CHR$($0D) OR X>100 THEN ENDEXIT
04B5        ENDLOOP
04B9        PRINT "BLOCK LOADED IS "; Y
04D1        REM *** DISTRIBUTE TO VARIABLES
04EF
04F0        X=0
04F7        LOOP
04F9          X=X+1
0504        EXITIF DISTRBUF(X)="$" THEN ENDEXIT
0517        ENDLOOP
051B        LOOP
051D          X=X+1
```

```
0528      EXITIF DISTRBUF(X)=CHR$($0D) THEN ENDEXIT
053C      ENDLOOP
0540      SALUTSTART=X
0548      LOOP
054A        X=X+1
0555      EXITIF DISTRBUF(X)=CHR$($0D) THEN ENDEXIT
0569      ENDLOOP
056D      LOOP
056F        X=X-1
057A      EXITIF DISTRBUF(X)=CHR$($0D) THEN ENDEXIT
058E      EXITIF DISTRBUF(X)=CHR$($20) THEN MRSPOINT=X
05A6      ENDEXIT
05AA      ENDLOOP
05AE      BLOCK(Y).SALUT=""
05BC      X=SALUTSTART
05C4      LOOP
05C6      EXITIF X>=MRSPOINT THEN ENDEXIT
05D6        X=X+1
05E1        BLOCK(Y).SALUT=BLOCK(Y).SALUT+DISTRBUF(X)
05FD      ENDLOOP
0601      BLOCK(Y).NAME=""
060F      LOOP
0611      EXITIF DISTRBUF(X)=CHR$(13) THEN ENDEXIT
0624        X=X+1
062F        BLOCK(Y).NAME=BLOCK(Y).NAME+DISTRBUF(X)
064B      ENDLOOP
064F      REM *** DISTRIBUTE TO VARIABLES
066D      BLOCK(Y).HOUSENO=""
067B      X=X+1

0686      BLOCK(Y).HOUSENO=DISTRBUF(X)
0698      LOOP
069A      EXITIF DISTRBUF(X)=CHR$(32) OR X>98 THEN ENDEXIT
06B4        X=X+1
06BF        BLOCK(Y).HOUSENO=BLOCK(Y).HOUSENO+DISTRBUF(X)
06DB      ENDLOOP
06DF      BLOCK(Y).STREET=""
06ED      X=X+1
06F8      BLOCK(Y).STREET=DISTRBUF(X)
070A      LOOP
070C      EXITIF DISTRBUF(X)=CHR$(13) OR X>100 THEN ENDEXIT
0726        X=X+1
0731        BLOCK(Y).STREET=BLOCK(Y).STREET+DISTRBUF(X)
074D      ENDLOOP
0751      BLOCK(Y).SUBURB=""
075F      X=X+1
076A      BLOCK(Y).SUBURB=DISTRBUF(X)
077C      LOOP
077E      EXITIF DISTRBUF(X)=CHR$(13) OR X>100 THEN ENDEXIT
0798        X=X+1
07A3        BLOCK(Y).SUBURB=BLOCK(Y).SUBURB+DISTRBUF(X)
07BF      ENDLOOP
07C3      BLOCK(Y).TELEPH=""
07D1      X=X+1
07DC      BLOCK(Y).TELEPH=DISTRBUF(X)
07EE      LOOP
07F0      EXITIF DISTRBUF(X)=CHR$(13) OR X>100 THEN ENDEXIT
080A        X=X+1
0815        BLOCK(Y).TELEPH=BLOCK(Y).TELEPH+DISTRBUF(X)
0831      ENDLOOP
0835      BLOCK(Y).FEES=""
```

```
0843        X=X+1
084E        BLOCK(Y).FEES=DISTRBUF(X)
0860        LOOP
0862        EXITIF DISTRBUF(X)=CHR$(13) OR X>100 THEN ENDEXIT
087C          X=X+1
0887          BLOCK(Y).FEES=BLOCK(Y).FEES+DISTRBUF(X)
08A3        ENDLOOP
08A7        REM *** CHECKING END OF MAXIMUM BLOCKS
08CC        Z=Z-10
08D7        IF Y<MAXBLOCKS THEN 100
08E8        REM *** WRITE TO DISK
08FC
08FD        PRINT CHR$(12)
0902        PRINT "WRITE NEW FILE TO DISK"
091C        SEEK #WPATH,DISKPOINTER
0926        Y=0
092D        LOOP
092F          Y=Y+1
093A          PUT #WPATH,BLOCK(Y)
0948          PRINT BLOCK(Y).NAME
0953        EXITIF Y>MAXBLOCKS-1 THEN ENDEXIT
0968        ENDLOOP
096C        DISKPOINTER=DISKPOINTER+Y*79
097C        Y=0
0983        GOTO 100
0987 1000
098B        REM *** LAST RAMBLOCKS TO DISK
09A8        MAXBLOCKS=Y
09B1        PRINT CHR$(12)
09B6        PRINT "WRITE LAST RAMBLOCK TO DISK"
09D5        SEEK #WPATH,DISKPOINTER
09DF        Y=0
09E6        LOOP
09E8          Y=Y+1
09F3          PUT #WPATH,BLOCK(Y)
0A01          PRINT BLOCK(Y).NAME
0A0C        EXITIF Y>MAXBLOCKS-2 THEN ENDEXIT
0A21        ENDLOOP
0A25        CLOSE #WPATH
0A2B        END
```

continued from P 44

```
    <# .PLACES #S SIGN )#
    TYPE SPACE ;
```

Finally for this month, right justification of output, i.e., typing a string in a field of a particular width.

If the string is less than this width, it is padded at the left with the required number of spaces. Consider a word like FIELD which expects on the stack: string addr, string length, field width. It can be defined as

```
  : FIELD ( STR LEN, FLD LEN--)    2DUP (
    IF OVER - SPACES
    ELSE DROP THEN ;
```

To place a string to the right of a specified field, we code

```
 (addr, length1, length2) FIELD TYPE.
```

What could be simpler? We can also use FIELD to define a syntactically clean string output word, such as

```
  : UD.R  ( DOUBLE INT, WIDTH)
    )R ( SAVE WIDTH ON RET STACK)
    <# #S #) R) ( GET IT BACK)
    FIELD TYPE SPACE ;
```

As a simple exercise, I suggest that you define other words using FIELD to justify different sorts of numeric strings. There is no end to the possibilities. If you play around enough, you will find yourself starting to warm to Forth. I hope so.

STOP PRESS * STOP PRESS * STOP PRESS * STOP PRESS * STOP PRESS *
STOP PRESS * STOP PRESS * STOP PRESS * STOP PRESS * STOP PRESS *

SUPER SPECIAL - while they last! Twin disk drives, 35 track, SS, full height, power supply and case. $190.00.
Complete with controller and cable $430.00 + cartage $8.00 - STILL CHEAP!

NOW! is the time to subscribe to Australian RAINBOW

Copies of back issues can be obtained, subject to the availability of stocks, by using this order form and marking clearly which issues you require to be sent to you.
Each issue costs $4.50 including postage and packing. Please enclose your cheque / money order made payable to: Australian Rainbow Magazine, PO Box 1742, Southport, 4215.

RATES

AUSTRALIAN CoCo / MiCo / softgold		AUSTRALIAN RAINBOW
☐ $3.45	Latest per copy	$4.50 ☐
☐ $19	6 months	$27.75 ☐
☐ $31	12 months	$39.95 ☐

BACK ISSUES

MiCo — First Issue Oct '83 ☐ $3 / copy

CoCo / MiCo / softgold First Issue Aug '84 ☐ $3.45 / copy

Feb '85-Apr '85 ☐	$3.95
Aug '84-Jan '85 ☐	$3.25
to July '84 ☐	$3.00

DISKS & TAPES

Rainbow on Tape (programme listings) $12 for month of ☐

Please note that RAINBOW on TAPE is issued irregularly

or Annually $144 ☐ or Debit my Credit Card Monthly ☐

Tape Monthlies

CoCoOz (Aust CoCo on Tape)		MiCoOz (MiCo on Tape)
☐	Latest $8	☐
☐	6 months $42	☐
☐	12 months $75	☐
First Issue Mar '83		Dec '83

or Debit my Credit Card Monthly

Blank tapes 12 for $18 or $1.70 ea ☐

Cassette Cases 10 for $5 ☐

Disks — $3.50 ea ☐ 10 for $29.99 ☐

BOOKS

Byte
Elementary $5.95 ☐

Help
Medium $9.95 ☐

Facts
Advanced $11.95 ☐

MiCo Help
Medium $9.95 ☐

MiCo Exposed
SORRY OUT OF STOCK ☐

BULLETIN BOARD

Let your computer talk to ours!
CoCoLink — Annual Sub $29 ☐

☐ VISA ☐ BANKCARD ☐ MASTERCARD

Signature _____

☐ CASH
☐ CHEQUE
☐ MONEY ORDER

Authorised

Amount $ _____

BLOCK CAPITALS PLEASE

If you already subscribe to either Australian Rainbow or Australian CoCo please place Subscription No

Complete the section below with one letter, figure or space per square

FIRST NAME SECOND NAME

Address

PC

Telephone Number STD Code Local Number

Subscription ☐ Renewal ☐

PARIS RADIO ELECTRONICS

CALL
(02) 344 9111 VOICE
(02) 344 9511 BBS
TO ORDER

CoCo OS-9™FLEX ™
Free Diskette with Each $50 Purchase

COMPILERS

K-BASIC — A BASIC language to MACHINE language Compiler; includes an Extended Macro Assembler. CCF — $292.00

PL/9 — by Graham Trott. A combination Editor/Compiler/Debugger; Structured Programming at the "almost Assembly Language" Level. CCF — $292.00

INTROL C Compiler — Full Featured C Compiler, Linking Loader includes full Library Manager. CCF — $549.00

•Forth — Forth language for CoCo CCR — $99.95 CCF — $99.95

MICROWARE BASIC09 — for CoCo OS-9 systems. CCO — $149.95

MICROWARE C Compiler — for CoCo OS-9 Systems. CCO — $149.95

DYNA C — C Compiler from the authors of Dynastar and Dynaforms. CCO — $99.95 CCF — $99.95

DATA BASE'S

XDMS — "Mainframe" User's say: "We don't have anything NEARLY as powerful as XDMS:" pure Assembly Language, F·A·S·T and small enough to operate on a single-sided 5" disk.
XDMS (V) I — CCF — $199.95 XDMS (V) II — CCF — $299.95
XDMS (V) III — CCF — $$399.95

RMS — Machine Language Data Base Manager — Super Fast. CCF — $220.00 CCO — $180.00

OPERATING SYSTEMS

XEX — The latest Flex Operating System from Frank Hogg Laboratory USA. Includes hi res screen drivers, user definable keyboard, supports 128k upgrades, and 35, 40, and 80 track drives. CCF — $149.95

OS-9 for the CoCo. The hotest selling software to hit Australia. CCO — $99.95

DISASSEMBLERS

SUPER SLEUTH — Interactive; extremely POWERFUL!! Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly, XREF Generator, Label "Name Changer," and Files of "Standard Label Names" for different Operating Systems included. CCF, Obj. Only $89.95
CCF, w/Source $145.00 CCO, Obj. Only $89.95

DYNAMITE + — Excellent standard "Batch Mode" Disassembler Includes XREF Generator and "Standard Label Names" Files.
CCF, Obj. Only $89.95 CCO, Obj. Only $89.95

WORD PROCESSING

STYLOGRAPH III — A full Screen-oriented WORD PROCESSOR (what you see is what you get); also supports Daisy Wheel proportional printers and WORDPAK CCF — $165.00 CCO — $165.00

STYLO-SPELL Spelling checker to suit stylo. CCF — $99.95 CCO — $99.95

STYLO-MAIL Mail merge program for Stylograph. CCF — $89.95 CCO — $89.95

DYNASTAR & DYNAFORMS — Another full screen Editor and Word Processor for the CoCo. CCF — $149.95 CCO — $149.95

XWORD — A powerful word processor that can be run under O-PAK, XSCREEN hi res drivers and WORD-PAK 80. CCO — $99.95

SPREAD SHEETS

DYNACALC — CoCo's best and fastest Spread Sheet system. CCR — $149.95

UTILITIES

UTILIX is a unix like utilities package for OS-9. It includes 15 different utilities to aid you in the manipulation of text files. CCO — $75.95

FILTERS KIT No. 1 — Eleven utilities used as filters for OS-9 CCO — $39.95

FILTERS KIT No. 2 — Ten more utilities, APPEND, CONFIRM, FF FORCERROR, MACGEN, NULDEVICE, REP, SIZE, TOUCH, and unload. CCO — $39.95

SDISK and BOOTFIX — Use 40 or 80 track, single or double sided drives with OS-9. Create a bootable double sided system disk. CCO — $59.95

HACKERS KIT No. 1 — This package contains a set of programs useful to anyone working with assembly language, trying to unravel the operating system or other assembly language programmes, or customizing their own. CCO — $59.95

XSCREEN — Hi Res Screen driver for OS-9 CCO — $39.95

O-PAK — Hi Res Screen driver for OS-9 CCO — $56.95

SEARCH AND RESCUE — disk utilities to search and retrieve lost data. CCO — $56.95

XTERM — A communications program for use with XSCREEN. CCO — $82.95

MEMORY MINDER — Disk diagnostics program. The most comprehensive program available to analyse your disk drives DOUBLE SIDED CCR — $149.95 SINGLE SIDED CCR — $139.95

XMENU — This is a program that creates a menu driven environment for the coco under OS-9 CCO — $69.95

RAM-DISK — This program simulates a disk drive, utilizing spare RAM memory. Very fast execution of commands. For use with 64K or 128K memory, Flex, XEX, or OS-9. CCF — $49.95 CCO — $49.95

MAGAZINES and BOOKS

AMERICAN RAINBOW — Direct from the USA the same month of publication for the same price as the newsagents. Why wait! $6.80

AMERICAN HOT COCO — Airfreighted direct each month. Subscription available. $5.80

THE OFFICIAL BASIC09 TOUR GUIDE $29.95

HARDWARE

PBJ WORKPAK 80 — an 80 column card to fit in the expansion port of the CoCo. Will run under Flex or OS-9. $199.95

PBJ WORDPAK II — The ultimate wordpak. Includes smooth scrolling, software controlled video switch, improved character set, hardware inverse video and up to 8K video RAM. $249.95

PBJ CC-BUS — A six slot software selectable expansion bus. Will take your disk controller, WORDPAK 80, GAMES Cards etc. $219.95

PBJ PC-PAK — This is a dual function cartridge which contains a centronics compatible parallel printer port and/or a battery backed real time clock. w/ real time clock $199.95 w/o real time clock $ 99.95

PBJ 2SP-PAK — A 2 port RS-232 serial interface with programmable baud rates up to 19.2K. Each port can also be interrupt driven. $119.95

DISK CONTROLLERS

J&M Disk Controller with JDOS Disk Basic $219.95

RADIO SHACK Disk Controller Ver 1.1 $199.95

MISC

CC-VID Monochrome video output board	$ 35.00
MC6809EP Processor chip	$ 24.00
MC6883 SAM chip	$ 24.00
4164-150 nSec, 64K chips	each $ 7.80

VISA

TM FLEX is a trademark of Technical Consultants
TM OS9 is a trademark of Microware

PRICES SUBJECT TO ALTERATION WITHOUT NOTICE

PARIS RADIO ELECTRONICS
161 BUNNERONG RD, KINGSFORD 2032
P.O. BOX 380, DARLINGHURST 2010
Ph: 344 9111

Availability Legends —
CCF — Color Computer FLEX
CCO — Color Computer OS-9
CCR — Color Computer RS-DOS

PRICES DO NOT INCLUDE POSTAGE AND PACKAGING

user group CONTACTS

' (Stop between numbers = b.h. else a.h.; but, hyphen between = both.)

ADELAIDE	JOHN HAINES	08 278 3560
ADELAIDE NTH	STVN EISENBERG	08 250 6214
ALBURY	RON DUNCAN	060 43 1031
ARMIDALE	TOM STUART	067 72 8162
BAIRNSDALE	COLIN LEHMANN	051 57 1545
BALLARAT	MARK BEVELANDER	053 32 6733
BANKSTOWN	KEN HAYWARD	02 759 2227
BLACKTOWN	KEITH GALLAGHER	02-627-4627
BLACKWATER	ANNIE MEIJER	079.82.6931
BLAXLAND	BRUCE SULLIVAN	047 39 3903
BOWEN	TONY EVANS	077 86 2220
BRASSALL	BOB UNSWORTH	07 201 8659
BRIGHTON	GLENN DAVIES	08 296 7477
BRISBANE EAST	ROB THOMPSON	07 848 5512
BRISBANE SH	PATRICK SIMONIS	07 209 3177
BRISBANE SW	GRAHAM BUTCHER	07 376 3400
BRISBANE WEST	BRIAN DOUGAN	07 30 2072
BROKEN HILL	DEAN PARADICE	080 6701
BUNDABERG	JIM McPHERSON	071 72 8329
CAMBERWELL	TONY BALDWIN	03 728 3676
CAMDEN	KEVIN WINTERS	046.66.8068
CAMPBELLTOWN	LEO GINLEY	02 605 4572
CANBERRA NTH	JOHN BURGER	062 58 3924
CANBERRA STH	LES THURBON	062 88 9226
CAULFIELD	JEFF SHEEN	03 528 3724
CHATSWOOD	BILL O'DONNELL	02 411 3336
' or	MARK ROTHWELL	02 817 4627
CHURCHILL	GEOFF SPOWART	051 22 1389
COLYTON TEENS	DWAYNE MANSON	02 623 5805
COOMA	ROSS PRATT	0648 23 065
DANDENONG	DAVID HORROCKS	03 793 5157
DARWIN	BRENTON PRIOR	089.81.7766
DENILIQUIN	WAYNE PATTERSON	058 81 3014
DONCASTER	JUSTIN LIPTON	03 857 5149
DUBBO	GRAEME CLARKE	068 89 2095
EMERALD	LEIGH EAMES	059 68 3392
FORBES	JOHANNA VAGG	068 52 2943
FORSTER	GARY BAILEY	065 54 5029
FRANKSTON	BOB HAYTER	03.783.9748
GIPPSLAND STH	PAT KERMODE	056 74 4583
GLADSTONE	ALBERT VAN GORKUM	079 72 2353
GOLD COAST	SHERYL BENTICK	075-39-2003
GOSFORD	PETER SEIFERT	043 32 7874
GOULBURN VALLEY	TONY HILLIS	058 59 2251
GRAFTON	DAVID HULME	066.42.0627
GREENACRES	BETTY LITTLE	08 261 4083
HASTINGS	MICHEAL MONCK	059.86.8288
HERVEY BAY	LESLEY HORWOOD	071 22 4989
HILLS DIST	DENNIS CONROY	02 671 4065
HOBART	BOB DELBOURGO	002 25 3896
HORNSBY	ATHALIE SMART	02 848 9830
IPSWICH	MILTON ROWE	07 281 4059
JUNEE	PAUL MALONEY	069 24 1860
KALGOORLIE	TERRY BURNETT	090.21.5212
KENMORE	GRAHAM BUTCHER	07 376 3400
LEETON	CHRIS NAGLE	069 53 2969
LITHGOW	DAVID BERGER	063 52 2282
LIVERPOOL	LEONIE DUGGAN	02-607-3791
MACKAY	LEN MALONEY	079511333x782
MACLEOD	ROBIN ZIUKELIS	03 450211x465
MacQUARIEFIELDS	KIETH ROACH	02 618 2858
MAFFRA	MAX HUCKERBY	051 45 4315
MAITLAND	LYN DAWSON	049 49 8144
MARYBOROUGH	NORM WINN	071 21 6638
MELBOURNE	JEFF SHEEN	03 528 3724
MELTON	MARIO GERADA	03 743 1323
MILDURA	SCOTT HOWISON	050 23 6016
MORPHETTVALE	KEN RICHARDS	08 384 4503
MOREE	ALF BATE	067 52 2465
MORWELL	GEORGE FRANCIS	051 34 5175
MT ISA	PAUL BOUCKLEY-SIMONS	077 43 6280
MUDGEE	BRIAN STONE	063-72-1958
MURGON	PETER ANGEL	071 68 1628
NAMBUCCA HDS	WENDY PETERSON	065 68 6723
NEWCASTLE	LYN DAWSON	049 49 8144
NOWRA	ROY LOPEZ	044 48 7031
ORANGE	STEVE LOVETT	063.62.4025
' or	JIM JAMES	063 62 8625
PARKES	DAVID SMALL	068 62 2682
PENRITH	ALEX SCHOFIELD	047 31 5303
PERTH	IAN MACLEOD	09 448 2136
PORT LINCOLN	JOHN BOARDMAN	086 82 2385
PORT MacQUARIE	RON LALOR	065 83 8223
PORT NOARLUNGA	ROB DALZELL	08 386 1647
PORT PIRIE	KEVIN GOMAN	086 32 1368
RINGWOOD	ANDREW RAWLINGS	03 726 6521
ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
ROSEVILLE	KEN UZZELL	02 467 1619
SALE	BRYAN McHUGH	051 44 4792
SANDGATE	MARK MIGHELL	07 269 5090
SEACOMBE HTS	GLENN DAVIS	08 296 7477
SMYTHESDALE	TONY PATTERSON	053 42 8815
SPRINGWOOD	DAVID SEAMONS	047 51 2107
STURT	MARY DAVIS	08 296 7477
SUNBURY	JACK SMIT	03.744.1355
SUTHERLAND	IAN ANNABEL	02 528 3391
SWAN HILL	BARRIE GERRAND	050.32.2838
SYDNEY TEENS	ROD HOSKINSON	02 48 5948
SYDNEY EAST	JACKY COCKINOS	02 344 9111
TAMWORTH	ROBERT WEBB	067 65 7256
TAHMOOR	GARY SYLVESTER	046 81 9318
TONGALLA	TONY HILLIS	058 59 2251
TOOWOOMBA	GRAHAM BURGESS	076 30 4254
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064
TRARALGON	MORRIS GRADY	051 66 1331
UPPER HUNTER	TERRY GRAVOLIN	065 45 1698
WAGGA WAGGA	BRUCE KING	069 25 3091
WHYALLA NORRIE	CHRIS HUNTER	086 45 3395
WONTHAGGI	PAT KERMODE	056 74 4583
YARRAWONGA	KEN SPONG	057 44 1488

SPECIAL INTEREST GROUPS

' BUSINESS

BRIZBIZ	BRIAN BERE-STREETER	07 349 4696

' TDP - TELEWRITER, DYNACALC PROCOLOR

BRISBANE	GEOFF TOLPUTT	07 44 6084

' OS9 GROUPS

BRISBANE	JACK FRICKER	07 262 8869
KALGOORLIE	TERRY BURNETT	090.21.5212
MONARO	FRED BISSELING	0648 23263
PENRITH	BOB THOMSON	042 30 2468
SYDNEY EAST	JACKY COCKINOS	02.344.9111
SYDNEY NTH	MARK ROTHWELL	02 817 4627
BLAXLAND 128K	BOB THOMSON	047 30 2468

' MiCo GROUPS

CARLISLE	STUART HALL	08 361 1922
LITHGOW	DAVID BERGER	063 52 2282
PORT LINCOLN	JOHN	086 82 2385
ROCKHAMPTON	TIM SHANK	079 28 1846
SYDNEY	RAJA VIJAY	02 519 4106

' TANDY 1000 / MS DOS

BRISBANE	BRIAN DOUGAN	07 30 2072
SYDNEY	ROGER RUTHEN	047.39.3903

AUSTRALIAN RAINBOW MAGAZINE
REGISTERED BY AUSTRALIAN POST — PUBLICATION NO. QBG 4009
AUSTRALIAN COCO/softgold
REGISTERED BY AUSTRALIAN POST — PUBLICATION NO. QBG 4007.
PO BOX 1742,
SOUTHPORT. QLD. 4215.

POSTAGE PAID AUSTRALIA

AUSTRALIAN

THIS MONTH

* COCO AGRO by Max Bettridge
* UFO by Gavin Unsworth
* PHONE DIRECTORY by Dung Ly
* GRAPHICS GALORE by Graeme Bennetts
* TOWER OF BEWARS by Stuart Sanders
* LETTERS by Jack Finnen
* AND LOTS MORE FOR YOU TO ENJOY SO WRITE TO US OR POP INTO YOUR LOCAL TANDY STORE OR NEWSAGENT FOR YOUR COPY.