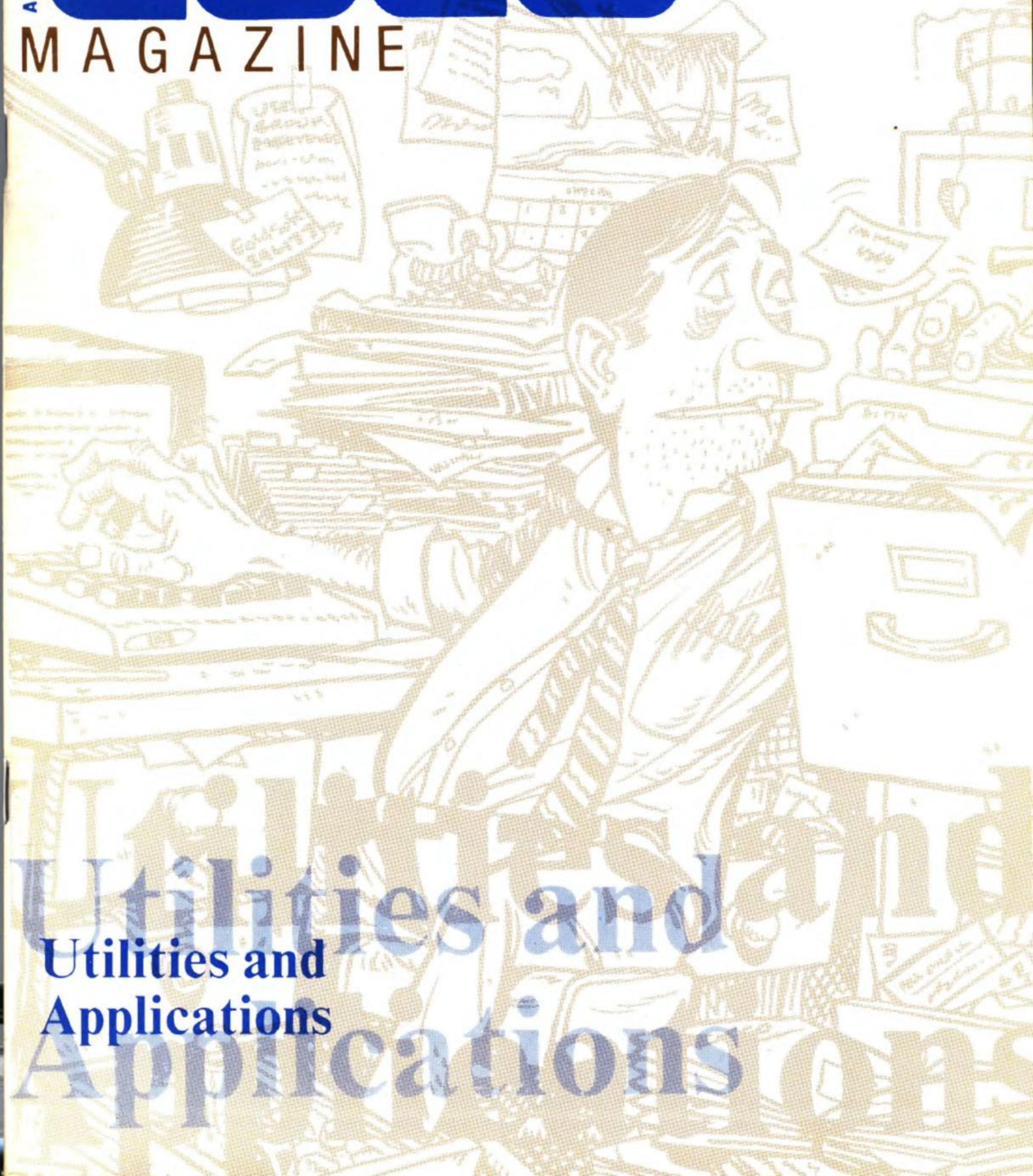
THE Magazine for experienced TANDY Colour Computer Users!

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

$4.50

AUSTRALIAN COCO MAGAZINE

VOL4NO7MAR88



Utilities and Applications

GOLDSOFT

GOLDSOFT Software for your Colour Computer!

Prices current for December, 1987.

Tandy Colour Computer HARDWARE & SOFTWARE

The CoCo Connection
Control models,
alarms – anything
electrical.

$219.95

Video Amp –

with sound	$35
without sound	$25

The Probe –
A temperature sensing unit you plug into the joy stick port.

$49.95

CoCo 3 Tapes & Disks!

#2 #3 #4

Get started with your CoCo3!

$16.00

The Best of Series:

#1	– Education
#2.1	– Games 16K
#2.2	– Games 32K
#4	– Business
#5	– Adventure
#6	– Preschool
#7	– Graphics
#8	– Games 16K
#9	– Games 32K
#10	– Education
#11	– Education

$16.00 each on Tape or Disk!

IBM PC Software

- Boulderdash
  $49.95
- Kings Quest 2
  $69.95
- Winnie the Pooh in the 100 Acre Wood
  $69.95
- The Ancient Art of War
  $96.00
- Lounge Lizard Larry
  $69.95
- Donald Duck's Playground
  $59.95

BITS & PIECES!

Blank Tapes	**12 for**	**$18.00**
	or	**$ 2.00 each**
Tape Cases	**12 for**	**$ 5.00**

BOOKS:

Help (for your CoCo)	$9.95
Mico Help (for your MC–10)	$9.95

BACK ISSUES:

Australian CoCo:

Sep '84 – Dec '85	$2.50
Jan '86 – Feb '87	$4.00

Australian Mico:

Aug '84 – Dec '85	$2.00

"SAY THE WORDZ"
Two Curriculum based speller programs for your Tandy Speech/Sound Pack. Requires 32K + Tandy Speech Pack.

$29.95

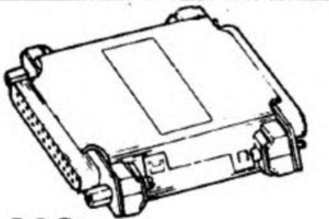
*EASY ORDER! Just fill in the form on page 64.

GIFTS FOR YOUR COMPUTER

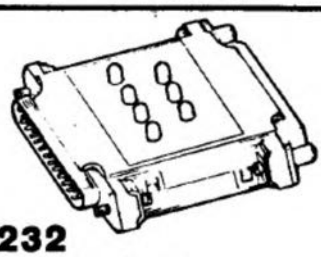
RS232C NULL MODEM ADAPTOR $13.35

RS232 DATA SWITCH WITH TESTER $99.95

RS232 BREAK OUT BOX $94.95



RS232 SURGE PROTECTOR $28.80



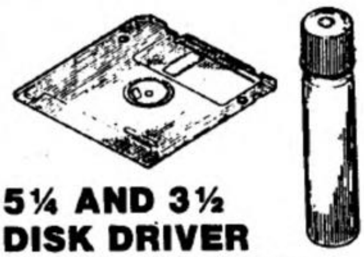
RS232 MINI TESTER $24.70

RS323 GENDER CHANGER $12.45

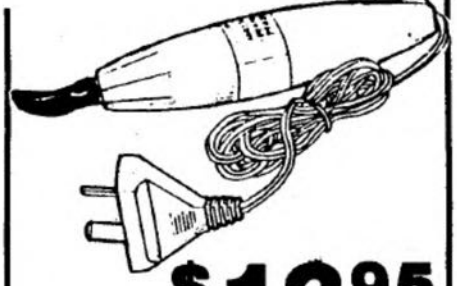
Accessories to keep your system running

MAGNETIC BULK ERASER

$39.95



5¼ AND 3½ DISK DRIVER HEAD CLEANER $6.55 ea



HEAD DEMAGNETISER $13.95

GOLDSOFT DISKETTES 5¼ DS DD

BOX OF 10 $17.99

TAPE HEAD CARE KIT $4.35

CASSETTE TAPE ERASER $13.15

GOLDSOFT

Available from:
GOLDSOFT
P.O. BOX 1742, S'PORT, Q, 4215
TEL: (075) 39 6177 or order on VIATEL '64213#

MasterCard

bankcard welcome here

VISA

Telecom
VIATEL®
AUSTRALIA'S NATIONAL VIDEOTEX SERVICE

APPLICATION FORM

DATE OF APPLICATION / /

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

section 1

PLEASE TICK APPROPRIATE BOX TO INDICATE SERVICE REQUIRED

BUSINESS SERVICE □ NON-BUSINESS SERVICE □

(CHARGES INCURRED ON BUSINESS SERVICES ARE USUALLY TAX DEDUCTIBLE)

SURNAME (OR BUSINESS NAME IF BUSINESS SERVICE) GIVEN NAMES

POSTAL ADDRESS NUMBER/STREET

SUBURB/CITY STATE POSTCODE

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

section 2

CONTACT NAME (IF BUSINESS SERVICE) GIVEN NAMES

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

SUBURB/CITY STATE POSTCODE

CONTACT TELEPHONE NUMBER (INCLUDING STD CODE)

section 3

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

PLEASE INDICATE TYPE OF EQUIPMENT USED TO ACCESS VIATEL

special instructions

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

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

telecom use only

DTE PP VN

BG SC CI

GOLDLINK: The place to be on VIATEL REF

inside coco

72 PAGES OF ADVANCED AUSSIE PROGRAMMING!
(With special thanks to George McLintock)

FOUNDER Greg Wilson

THE CREW
Managing Editor Graham Morphett
Editor Alex. Hartmann
Production Sheryl Bentick
Accounts Karen Court
Advertising Graham Morphett

SUB EDITORS
Submissions Alex. Hartmann
Forth John Redmond
OS9 Jack Fricker
Fred Bisselling

SPECIAL THANKS TO:
Bob Horne, Michael Turk, Johanna Vagg and Martha Gritwhistle.

GOLDSOFT

Published by:
GOLDSOFT, PO Box 1742, Southport, Q'ld, 4215
Registered Publication QBC 4009.
Telephone: (075) 39 6177.

Printed by:
The Warwick Newspaper, 50 Albion St, Warwick.
Telephone: (076) 61 1355.

DEADLINES:
7th of the preceding month.

IN A NUT SHELL

Hi and welcome to this month's Utilities and Applications issue. This month (and in the next few months) we will devote most of the magazine to a choosen subject.

(Just to wet your appetite, next month will be the Games and Adventures issue, packed with, well, like it says - Games and Adventures for your CoCo 1, 2 and 3.

Of Programs, Magazines and Users

As I've said before, and always will, this magazine is a hobbyist based magazine, meaning that whatever programs YOU submit, determines what NEXT month's magazine will contain.

I've sometimes heard, or read, that the programs found in Australian CoCo Magazine and Softgold Magazine are, and I quote, "always the same old programs I'd seen a few months ago, only in a new style".

The example used was this: for one month last year, we published lotto programs till they were coming out of our ears!

Lotto programs can be useful to some people, and they seem the easiest thing to write. Anyway, some people seem to miss the point.

Alright, so it's the same type of program that was published a month ago ... but there will always be somebody out there, busy learning (or trying to) about their new CoCo.

And there's the point - no matter what kind of program it is, there will ALWAYS be somebody out there who will benefit from YOUR program!!

And just how did YOU learn to use your CoCo?

The bottom line is that no matter how small or insignificant your program is, someone will learn from it.

Guaranteed.

It always makes my day to see someone who has learned something new that will benefit them.

So don't delay - that program will go to good use - I know it will. Remember, I'm the submissions editor as well, and I'm usually the one on the end of the telephone who hears what other users have to say.

So send in your program - all you do is read the page on "How to Submit your Program" in this magazine.

Ah - and also, please don't send anymore lotto checker programs ... pretty please?

The Coffs Harbour Trip

On the weekend of the 20-21st February, I went to Coffs Harbour to give a small talk on "Beginning with Computers".

Before I go on, Coffs Harbour is a 'small' town with a population of 21,000 people.

It is 180 north of Kempsey (halfway between Sydney and Brisbane) and is located right on the coast.

Anyway, quite a number of people attended, and not all were CoCo owners. One owned a Microbee, another with a Tandy 1000.

All over, we had learned something of beneficial value (yes - even I learned something!).

The contact for Coffs Harbour is Bob Kenny, and he can be called on (066) 512-205.

Melbourne

In early March, Graham and I are going to Melbourne.

And on the night of the 9th, we are to visit the Ringwood User Group. If you live in the Ringwood area, and are interested, then come along! If you want to know more, then read the Magazine (heard that somewhere before...).

For up-to-date information, ring the Ringwood User Group contact, and in particular, an Ivor Davies on (03) 758 4496.

We try to please everyone.

On the night of the 10th (the following night), we will appear at the Morwell User Group, doing one of the things we do best - demonstrate!

Again, for up-to-date information, ring either Jeff Sheen on (051) 33 9904 or a Joe Hester on (051) 27 7817.

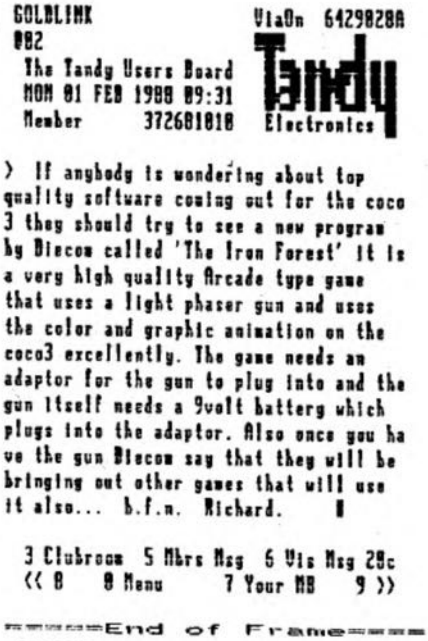
We're going to have a great night - showing the latest in software, hardware and information.

Just to whet your appetites, we're going to bring down with us a 20 megabyte hard drive, running OS-9.

So, if you're interested in hearing what's new, or seeing what you can get for your CoCo, or just coming to have fun, then we'll see you there!

Alex

COM • STATION 642

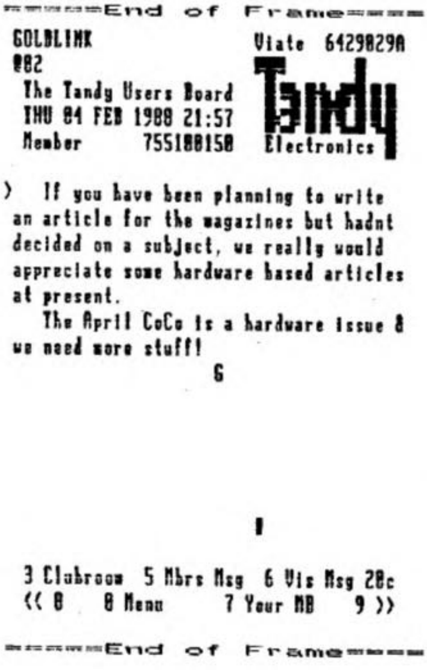


GOLDLINK ViaOn 6429028A
#02
The Tandy Users Board
MON 01 FEB 1988 09:31
Member 372601010
Tandy Electronics

> If anybody is wondering about top quality software coming out for the coco 3 they should try to see a new program by Diecom called 'The Iron Forest' it is a very high quality Arcade type game that uses a light phaser gun and uses the color and graphic animation on the coco3 excellently. The game needs an adaptor for the gun to plug into and the gun itself needs a 9volt battery which plugs into the adaptor. Also once you have the gun Diecom say that they will be bringing out other games that will use it also... b.f.n. Richard.

3 Clubroom 5 Mbrs Msg 6 Vis Msg 20c
<< 0 0 Menu 7 Your MB 9 >>

=====End of Frame=====

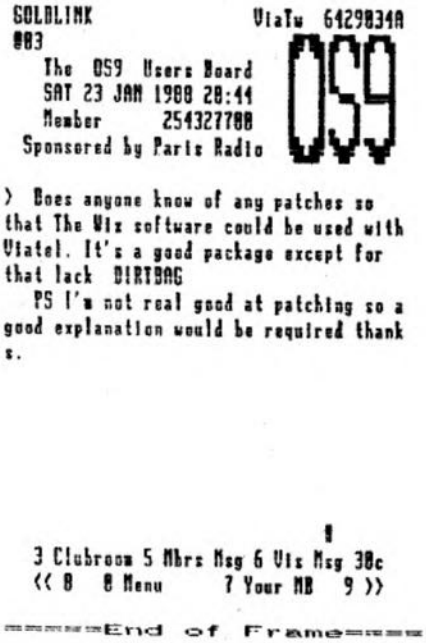


GOLDLINK Viate 6429029A
#02
The Tandy Users Board
THU 04 FEB 1988 21:57
Member 755100150
Tandy Electronics

> If you have been planning to write an article for the magazines but hadnt decided on a subject, we really would appreciate some hardware based articles at present.
The April CoCo is a hardware issue & we need more stuff!
G

3 Clubroom 5 Mbrs Msg 6 Vis Msg 20c
<< 0 0 Menu 7 Your MB 9 >>

=====End of Frame=====

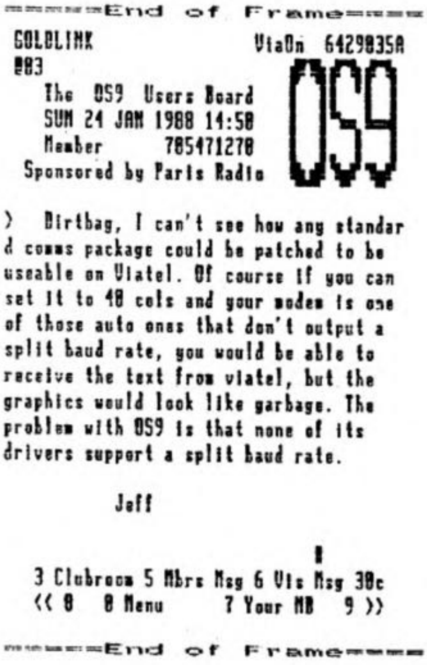


GOLDLINK ViaTw 6429034A
#03
The OS9 Users Board
SAT 23 JAN 1988 20:44
Member 254327700
Sponsored by Paris Radio
OS9

> Does anyone know of any patches so that The Wiz software could be used with Viatel. It's a good package except for that lack DIRTBAG
PS I'm not real good at patching so a good explanation would be required thanks.

3 Clubroom 5 Mbrs Msg 6 Vis Msg 30c
<< 0 0 Menu 7 Your MB 9 >>

=====End of Frame=====



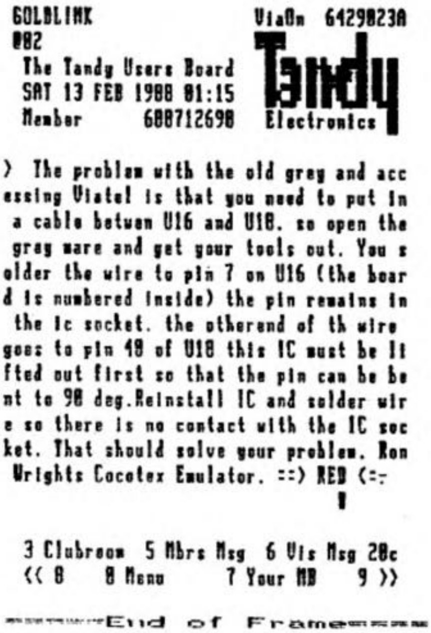
GOLDLINK ViaOn 6429035A
#03
The OS9 Users Board
SUN 24 JAN 1988 14:50
Member 705471270
Sponsored by Paris Radio
OS9

> Dirtbag, I can't see how any standard comms package could be patched to be useable on Viatel. Of course if you can set it to 40 cols and your modem is one of those auto ones that don't output a split baud rate, you would be able to receive the text from viatel, but the graphics would look like garbage. The problem with OS9 is that none of its drivers support a split baud rate.

Jeff

3 Clubroom 5 Mbrs Msg 6 Vis Msg 30c
<< 0 0 Menu 7 Your MB 9 >>

=====End of Frame=====

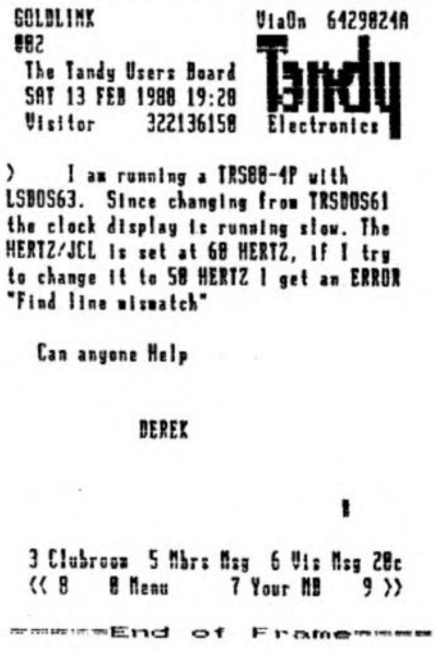


GOLDLINK ViaOn 6429023A
#02
The Tandy Users Board
SAT 13 FEB 1988 01:15
Member 600712690
Tandy Electronics

> The problem with the old grey and accessing Viatel is that you need to put in a cable betwen U16 and U18. so open the grey mare and get your tools out. You solder the wire to pin 7 on U16 (the board is numbered inside) the pin remains in the ic socket. the otherend of th wire goes to pin 40 of U18 this IC must be lifted out first so that the pin can be bent to 90 deg.Reinstall IC and solder wire so there is no contact with the IC socket. That should solve your problem. Ron Wrights Cocotex Emulator. ::) RED (:-

3 Clubroom 5 Mbrs Msg 6 Vis Msg 20c
<< 0 0 Menu 7 Your MB 9 >>

=====End of Frame=====



GOLDLINK ViaOn 6429024A
#02
The Tandy Users Board
SAT 13 FEB 1988 19:20
Visitor 322136150
Tandy Electronics

> I am running a TRS80-4P with LSDOS63. Since changing from TRSDOS61 the clock display is running slow. The HERTZ/JCL is set at 60 HERTZ, if I try to change it to 50 HERTZ I get an ERROR "Find line mismatch"

Can anyone Help

DEREK

3 Clubroom 5 Mbrs Msg 6 Vis Msg 20c
<< 0 0 Menu 7 Your MB 9 >>

=====End of Frame=====

Letters

Dear Graham,

Would you please thank Mal McLauchlan for his program "Printer Graphics" and ask him to keep up the good work. I, for one, will be very interested in any other graphics programs he creates for the printer.

Also, thank Johanna Vagg for her program, "Letterhead Pictures" - it makes your letter heads look much better.

Arthur Williams,
Harrington, NSW

Arthur,

Mal's work goes from strength to strength - he's a very clever man. And as for Johanna, well, we all realise what a wonderful asset she is for the CoCo community.

Graham.

*

Dear Graham,

re: Backyard Drives.

This is just a note in relation to the page long article by a Mr. John Poxon, in the November Softgold magazine about backyard drives. There's a couple of things I'd like to say in relation to this matter.

We are all sorry to hear that he had a bad experience with a couple of drives.

There was five of us who read the article and did start to panic, as we all had "backyard drives". We all took our drives and got them SEQEB approved which, I might add, cost a few dollars - only to find out that the drives were by all means up to scratch and that they all had the safety requirements that were needed.

All I can say as a result, is that we trust our local expert, and have no hesitation in getting him to do our drives or controllers.

I would recommend him to anybody who asks as I know he is reliable.

So Mr. Poxon, maybe you haven't got very reliable contacts, because as we found out, there are some good quality drives from the backyard experts, and not all of them are rip-off merchants, with outrageous prices.

People that are interested in drives: I would suggest that you contact your local user group as most of them are well up in the CoCo scene and know someone that is reliable.

Rosemary Canhill.

Rosemary,

I'm with you!

Unfortunately, it is possible to obtain drives which are not up to scratch, but since we've been doing a magazine, we've not seen too many that were really bad.

In fact the drives put together by the Brisbane groups over the years have been quite well made.

In other states, the Western Australian Group, I believe, has done some excellent work in this area, from time to time; the New South Wales people have come to swear by the drives supplied by Blaxland Computer Services and in South Australia/Victoria, people have been making drives for a long while, and we've not heard a single complaint about them.

I think what John's article DOES do, is just to remind us all that in the race to obtain a good product at a reasonable price, there is a bottom line - the really cheap article rarely provides long-term satisfaction, whether it's a computer OR a drive. It is a fact of life that you must pay for quality.

Graham.

*

Dear Graham,

It appears you haven't heard of a great BBS down here run by the Peninsula Club!

That's right, a BBS run by a User Group member.

It is also now totally devoted to the CoCo 1/2/3 and OS9 systems as well.

The details are as follows:

Phone:	03-551-5131
Baud:	300 bps only
Hours:	21:30 to 07:30
DOS:	TRS-DOS
Access:	Public
O.System:	Basic!
Sysop:	Stan Blazejewski

Maybe this will give some other user groups some incentive to follow suit elsewhere!

Darren Reed
Watsonia, VIC.

Darren,

This is great news, Darren.

Unfortunately CoCo Groups have not interested themselves as much as one would have though in the area of communication.

Getting computers to "talk" to each other is one of the really interesting aspects of computing.

In the early days here at Goldsoft, we ran a board called "CoCoLink", which was replaced eventually by our service on Viatel.

I well remember all the problems we had!

Running a Bulletin Board is tough work, so our sympathy goes out to Stan and we hope that users will make his board an interesting, fun place to be.

Graham.

*

Dear Graham,

Just a few lines to let you know that I'm still alive and kicking. As per usual, work has been absolutely flat strap. That's the unfortunate part about this job - don't get much time on the CoCo. Over Christmas things "should" quieten down a lot - then I'll look into CoCo. It won't know what's hit it!

Is Karen as good looking as she sounds on the phone? - you have got a top lady on the staff, mate.

Maybe she would like to come to PNG and be my secretary?

Well mate, that's enough waffle from me for now.

Best wishes and every success mate.

Fred Bisseling,
Up North Somewhere in PNG.

Fred,

First things first.

Karen is 5'3" blue-eyed blonde who looks better than she sounds, but you can't have her because she's the first female we've ever had in the office that can handle Martha!

We'll look forward to seeing you on the Coast in April, although knowing you, I would have expected you to turn up on April 1.

See you in April!
Graham

Access 64K of RAM

64K of RAM for data storage.

by George McLintock

64K CoCo/CoCo3 UTILITY

I have developed a series of ML routines which allow a Basic program to make use of 64K of RAM for data storage. This submission covers one of them.

The utility also allows a procedure for sorting and searching a much larger file on disk, and this is described later under the heading of extending the row and column concept to larger files on disk.

INDXCARD and ONEARRAY

These are two different programs.

ONEARRAY is a series of ML routines which allow you to access data in a buffer by reference to a row and or column coordinates.

In effect, instead of holding data in a normal two dimensional Basic array, it is held in a common buffer area and accessed from the Basic program as required.

In terms of the amount of data which can be held in memory for a Basic program, this provides two major advantages:

- it eliminates a large amount of the 'overheads' required by Basic to define the array
- and it allows the buffer area to occupy the full 64K of RAM available in a 64K CoCo with a similar amount of RAM being available with a 128K CoCo 3.

While this utility is submitted in association with a particular Basic program, it is not restricted to this purpose.

It can be used with any other Basic program which accesses data in a two dimensional way.

INDXCARD is a Basic program which has been set up to provide an index card type of system, using the facilities provided by ONEARRAY.

It also uses other ML routines which I have submitted previously for certain functions

- ASORT for sorting data
- ASEARCH for searching the array
- PCLEAR 0 to provide extra memory

The ONEARRAY utility has been coded so that ASORT and ASEARCH can be used without modification.

In effect, INDXCARD, is a basic framework for such a program. You have to modify various parts of the program to suit your own specific requirements, but none of these are difficult.

Modifications are required for the parameters which define the file structure, and to the code used to print reports.

Other modifications to the Basic program are possible, and fairly straight forward.

The program itself demonstrates the amount of extra data that can be held in memory with this approach. INDXCARD is set up to handle the equivalent of an array dimensioned as A$(10,500).

As a normal Basic array this would require over 27K of memory just to define the array, and before putting any data in the array.

Whereas the approach used here in fact provides around 40K for actual data storage, after allowing for all Basic overheads (including the program code as well).

OVERVIEW OF DATA STRUCTURE

It is a common feature of most programs that any specific module or part of that program will only access a limited portion of the total amount of data available to the whole program.

For example, with a two dimensional array (like with INDXCARD) any processing of the data in the main array (ie, A$(10,500)) will involve either a single row or a single column, but not the whole array at any one time.

To take this example a little further, if you want to find a particular record, or if you want to sort all records into a specific sequence, then that particular part of the program which is performing this specific function will only require access to the single column of the array that you are using as the key for the operation eg surname, suburb etc.

Likewise, once you have selected a particular record to do something with it, like print it, then all you require access to at that time is the row in the array which represents that record.

The functions performed by ONEARRAY are based on this basic concept.

In effect, all data for the main array (A$(10,500)) is held in the buffer in a form which is not directly accessable by Basic.

The ML routines provide an interface between the Basic program and the data in the buffer which allows the Basic program to access a single row and a single column at any one time.

In concept, the ML routines move the data for a selected row and column between the buffer area and the normal Basic program area as required to allow Basic to access the data normally.

The ML routines operate at sufficient speed to have no real impact on processing time.

In fact, for INDXCARD, only the data for each row is actually moved between the buffer and the program area.

For this particular program, the only operations actually performed on columns is a sort or search, and for these specific functions, other ML routines are used to access the actual data.

Hence it is sufficient to simply set up the array VARPTR's in the program area to point to the data in the buffer for these operations, without having to move the actual data.

Both ASORT and ASEARCH were coded specifically to allow this type of use.

However, to allow ONEARRAY to be used for other purposes, an option is provided to actually move the data for each column back to the program area if required. If using this option, sufficient memory must be provided in the program area to hold this data as well.

NUMBERS AND STRINGS

All data in the buffer is handled as string variables, and if you want to process numeric variables in this way they must be converted to strings for moving to and from the buffer

For a disk system this can be done quite simply with the MKN$ and CVN commands. These commands will operate on any variables, and are not restricted to variables in a direct access file.

It is possible to use numeric variables with these routines by following other procedures in the Basic program and these are explained later. These other procedures will work with any system.

PROGRAM OPERATION

The general logic for the calling sequences and other operational aspects for this routine are similar in many ways to those used for ASORT and ASEARCH.

It uses a parameter array to initialise variables within the ML routine, and uses the direct page register to point to working storage areas. it has been designed to use the same working storage area as ASORT and ASEARCH, and this is set for the cassette buffer area.

It also requires that all simple variables used by the program are defined before initialising the routine, and that no new simple variables are defined between the call to initialise it and any subsequent calls by the EXEC command.

For more information about these general aspects, you should refer to these earlier submissions.

The actual operations performed by the ML routines are relatively simple, but are closely integrated with corresponding operations to be performed by the Basic program which uses it

ENTRY POINTS

The operations performed, and entry points to do them are as follows.

The entry points are defined as offsets from the variable N, which is the start of the ML code N+57 - initialises the routines N+105 - move a row of data from the program area to the buffer area N+205 - move a row of data from the buffer area to the program area N+318 - move a column of data from the buffer area to the program area.

- with an option to set the VARPTR's of the column array to point to the data in the buffer area instead N+401

- delete a row of data from the buffer area N+427

- set sort switches for a multi-level sort N+179

- reset string space.

INITIALISATION

The routine is normally initialised at the start of the program only, and then called by the EXEC 'address' command to perform the functions required

The routine requires 11 parameters for its initialisation, and these are included in a parameter array set up with a DIM P(10) command.

When the values for each parameter are set, the routine is entered with a call like X=USR(VARPTR(P(0))).

An example of the parameters required is as follows. These parameters are the ones used with INDXCARD.

A detailed description of their use is provided later. Different parameters are used for different purposes.

P(0)=VARPTR(Q) - Q is variable for this record number

P(1)=VARPTR(N) - N is variable for number of records in the buffer

P(2)=VARPTR(B) - B is variable for this field number

P(3)=NF - number of fields in each record

P(4)=VARPTR(A$(0)) - start of record array in program

P(5)=VARPTR(TB(0)) - start of string space for record in program area

P(6)=VARPTR(T$(1)) - start column array in program area

P(7)=0 or &HE00 - start string space for column array

- if P(7)=0 then string data is not moved to the program area

P(8)=0 or &H2600 - end string space for column array

P(9)=VARPTR(N(1)) - start pointer array

P(10)=VARPTR(TF(0)) - start of sort switches.

The actual value of some of these parameters will vary according to the way in which the Basic program is organised.

The parameters P(0) to P(2) are the VARPTR's of variables within the program area which are nominated to perform specific functions.

The Basic program sets the values of these as required, and the ML routine accesses them to find the value of the actual parameter for each operation.

For example, when moving a record from the buffer to the program area, the Basic program would set Q equal to the record number required and EXEC N+205.

The ML routine would then access the value contained at the address specified by P(0) to find the record number required. setting P(0)=VARPTR(Q) establishes the link required for using this method of passing parameters to the ML routine.

When moving a record from the program area to the buffer, the ML routine does not use a record number at all. Each new record is simply added to the buffer in sequence.

The record number simply identifies each record by its position in the buffer.

The Basic program must also maintain the correct value of the number of records in the buffer in the variable nominated in the parameter P(1).

The ML routine itself does not count the number of records in the buffer, but it uses the value at the address specified by P(1) to control its own operations.

STRUCTURE OF DATA IN THE BUFFER

Data is stored in the buffer from the start, upwards in memory. It is held as variable length strings with pointers to identify each separate data element.

For each record, bytes 0 and 1 form a two byte integer to show the total length of the record.

Byte 2 contains the length of the first data field for the record. This is followed by the string data for that field. The next byte contains the length of the second data field, and is followed by the string data for that field.

This pattern continues for the length of the record.

The parameter at P(3) in the initialisation array is retained in working storage, and is used to specify the number of fields in the record.

A parameter in the working storage area also holds the address of the last record in the buffer.

The ML routines use these pointers to find the record required and then to find each data string to be moved back to the program area.

The pointers are set up as each record is moved from the program area to the buffer.

USE OF ARRAYS IN THE BASIC PROGRAM AREA

Various arrays are required to be defined by the Basic program which uses these routines, to allow both of them to operate together.

These arrays are used for two quite different and distinct functions:

- as normal Basic string arrays
- and as a means of reserving memory space in the program area for other uses.

This second function could in fact be provided by any of the other methods available to reserve and protect memory within the program area, eg memory protected by a CLEAR statement etc.

There are various advantages associated with using an area defined for an array in this way, which are described in other partè of this description.

NORMAL BASIC STRINGS REQUIRED

The Basic program must define a normal array to hold a single row (or record) of the main array, eg if there are 11 fields in each record (ie for a main array of A$(10,500)), then the Basic program must establish a single array A$(10) to hold each record as it is moved to or from the buffer.

The starting address for this array is set by P(4) in the parameter array.

If you want to access a single column in the main array, then the Basic program must also provide an array to hold that column, eg an array as DIM T$(500).

If you in fact use the routine for a purpose which does not require access to a column, then you don't require this array.

A column array is also required if you want to sort or search the array on a particular field.

The same column array (eg T$(500)) can be used for both these functions.

The starting address for this array is specified in P(6) of the parameter array.

For INDXCARD, the starting address used is VARPTR(T$(1)). This is used instead of T$(0) to simplify the coding of the Basic program only.

Other arrays are required if you want to sort the columns, and these are described later.

MEMORY RESERVED FOR STRING SPACE

Basic requires two separate areas of memory to define and hold string variables. The first area holds the VARPTR of the variable which holds its name, length, and where the actual string data is located.

The second area of memory is used to hold the actual contents of the string. For a normal Basic program, this second area is reserved by Basic at the top of memory and is called normal string space.

While it is possible to manipulate data in normal string space from a ML routine, it gets a little messy and can slow the program operations down significantly.

The Basic interpreter can in fact access string data from any part of memory for most of its operations. In the context of this routine, any data to be accessed by Basic directly must be in the lower 32K of memory.

The only restriction on this is that any strings created by Basic itself, will have the string data contents set up in normal string space. The single exception to this is if the string contents are defined entirely within the Basic code, eg A$="FRED"

In effect, this means that if the ML routine is setting up string data for Basic to read, then the string data can be located anywhere in the lower 32K of memory. ie where ever it might be convenient to put it.

When doing it this way, the ML routine must also set up the VARPTR area in the correct structure so that Basic can find where the string data is.

The normal way of doing this type of thing is to use the CLEAR command to protect high memory, or to use the graphic screen area or other areas reserved by Basic like the cassette buffer or disk file buffer areas.

However, another convenient way of doing the same thing is to use the DIM command which reserves an area of memory for the VARPTR's of the variables, but this memory can then be used for any other purpose once it has been established.

An advantage of this approach is that the area reserved can be adjusted to suit the particular requirements. eg to get 100 bytes use DIM T1(20), or for 1000 use DIM T1(200) etc. Each VARPTR defined uses 5 bytes.

The parameter at P(5) is used to specify the starting address for the area of memory to be used to hold the string data for each record in the program area.

The ML routines assume that there will always be sufficient memory provided for this purpose, and it performs no check that this is correct.

Given the expected use of the routine this is reasonable. The maximim length of any record should be known and controlled by the Basic program.

INDXCARD performs this function because it is also required for the external file structure as well

The parameters at P(7) and P(8) are used to specify the start and end address of the string storage space for the column array.

Because the total length of these strings cannot be controlled or estimated to the same extent, the ML routine checks that the space provided is not exceeded.

If you are in fact using this option, then the graphic screen area could well be a suitable area of memory for this purpose.

The area reserved can be varied from 1.5K to 12K by using normal Basic commamds.

For other use, you could define another dummy array to provide the space.

Because of the limited functions that are normally performed on columns, the ML routines provide an option to have the VARPTR's of the array specified in P(6) set up to point to the string data as stored in the buffer.

For example, if all you want to do on columns is to sort or search the records, and you use other ML routines for these functions, like ASORT & ASEARCH, then it is not necessary to actually move the string data for the column back to the program area.

The other ML routines can access the string data in the actual buffer itself.

To use this option, simply set the value of P(7)=0 when initialising the routine.

MOVING COLUMNS

The parameter at P(9) is used to specify the sequence in which records are moved from the program to the buffer area

·P(9) is the VARPTR of the first element in a numeric array which contains the record numbers to be moved from the buffer to the program area. It operates in much the same way.

As the pointer array for the move routine with ASORT.

For example, with the parameters as specified, where N(N) is the pointer, and T$(N) is the column array, then if ...

```
N(1)=20
N(2)=15
N(3)=1
```

... then after the move,

T$(1) will contain the field for record 20,

T$(2) will contain the field for record 15,

... and T$(3) for record 1 etc

If moving the column back to search the file with ASEARCH then you have to allow for the sequence in which the records are returned. The easy way to avoid complications is to set the pointer array to its natural sequence before moving the column back, ie use code like

```
FOR X=1TO N:N(X)=X:NEXT X
```

... before returning the column

If you return the column in some other sequence then the subscript value returned by ASEARCH in P(7) wil not be the actual record number found.

The record number required to access the record found wil be N(P(7)) and not P(7).

MOVING COLUMNS WITHOUT THE POINTER ARRAY

The use of a pointer array for moving columns back from the buffer was adopted mainly to facilitate sorting.

However, if you want to use this approach for a large number of small records then it may cause you to run out of memory in the problem area.

For example, if the average record length is only 10 bytes then you could hold around 3,000 records in a 32K buffer. But with this many records, you would require 30K of memory in the program area to hold the two arrays required to access a single column, ie T$(3000) requires 15K and N(3000) would also require 15K.

With this sort of file structure, you can still access a column with ASEARCH for searching the file; but it requires some POKE's to change in the ML code. This allows you to set up the VARPTR's in TR$(MR) in natural sequence, without using a pointer array.

The POKE's required are:

```
POKE M+&H17D,1
POKE M+&H113,32
POKE M+&H114,6
```

... and set P(6)=1 when initializing the routine. P(7) must also be zero for this.

The complication with this approach is that part of the ML routine MUST be restored to normal before you can return a record from the buffer, ie after the EXEC M+318 to move the column back to the program area, you must then ...

```
POKE M+&H113,&H6D
POKE M+&H114,&H84
```

... so that the ML routine will still return a single record correctly.

To use this procedure, you POKE the changes to M+&H113 & 114 each time before you EXEC M+318, and then restore these locations immediately after.

If using this approach, then you cannot do a multi-level sort of the file in the buffer with ASORT, but you can still do a single sort.

You can, of course, always search the file without returning a column. array, it just takes a bit longer.

For example, to search A$(Y) for a match with A$, you can use code like ...

```
10 FOR X=1TO N
20 Q=X:EXEC M+205
21 'search each record.
30 IF A$(Y)=A$ THEN 'match fnd.
40 NEXT X
```

AN ALTERNATIVE SEARCH ROUTINE

As an alternative to the above procedure, I have also developed a special ML search routine, called XSEARCH, which will search any field in the buffer directly, ie it does not require any space in the program area at all.

It is included as a separate 'add on' utility because it will not always be required.

For applications of this nature, it means that the only pace required in the program area is to hold one record, a row of the array.

AN ALTERNATIVE SORT ROUTINE

Following the same concept as for XSEARCH, I have also developed a special ML sort routine, called XSORT, which will sort the records in the buffer directly, without requiring a column array to be established in the program area.

It is also possible to both XSEARCH and XSORT together with ONEARRAY to obtain a more limited sort/search function then that produced by ASORT/ASEARCH.

Hence both basic functions can be obtained without requiring a column array in the program area.

The Listing:

```
                    00100 *CALLED  ONEARRAY - DATA 2 DIMENSIONAL ARRAY
                    00110 *
                    00120 *PARAMETER ARRAY TO INITIALISE
                    00130 *P(0)=VARPTR THIS RECORD NUMBER
                    00140 *P(1)=VARPTR TOTAL NUM RECORDS
                    00150 *P(2)=VARPTR THIS FIELD NUMBER
                    00160 *P(3)=NUMBER OF FIELDS IN RECORD
                    00170 *P(4)=VARPTR OF RECORD ARRAY IE A$
                    00180 *P(5)=START STRING SPACE FOR RECORD ARRAY
                    00190 *P(6)=VARPTR OF SORT/SELECT ARRAY
                    00200 *P(7)=START STRING SPACE FOR SS ARRAY
                    00210 *     - IF P(7)=0 THEN DONT MOVE STRING DATA
                    00220 *P(8)=END STRING SPACE FOR SS ARRAY
                    00230 *P(9)=VARPTR POINTER ARRAY
                    00240 *P(10)=START SPACE FOR MULTI-LEVEL SORT SWITCHES
                    00250 *
                    00260 *WORKING STORAGE STARTS FROM 80 & SHARES 9-29 WITH ASORT
                    00270 *
7D00                00280          ORG     32000
                    00290 *
            0050    00300 ARECNO  EQU     80      P(0)
            0052    00310 ANREC   EQU     82      P(1)
            0054    00320 AFLDNO  EQU     84      P(2)
            0056    00330 FIL1    EQU     86      P(3)
            0057    00340 NUMFLD  EQU     87
            0058    00350 ARRAY   EQU     88      P(4)
            005A    00360 STOUT   EQU     90      P(5)
            005C    00370 SRTARY  EQU     92      P(6)
            005E    00380 STSTR   EQU     94      P(7)
            0060    00390 ENDSTR  EQU     96      P(8)
            0062    00400 SPOINT  EQU     98      P(9)
            0064    00410 SORTSW  EQU     100     P(10)
            0066    00420 STBUF   EQU     102     START BUFFER
            0068    00430 ENDBUF  EQU     104     END BUFF
            006A    00440 THISBF  EQU     106     THIS END
            006C    00450 MVSW    EQU     108     MOVE SW
                    00460 *
            0009    00470 ERSW    EQU     9
            000A    00480 ESPACE  EQU     10      END STRINGS POKED
            000C    00490 CNT     EQU     12
            000E    00500 RCNT    EQU     14
            0010    00510 T1      EQU     16
            0012    00520 T2      EQU     18
            0014    00530 I       EQU     20
            0016    00540 IA      EQU     22
            0018    00550 JA      EQU     24
            001A    00560 IL      EQU     26
            001B    00570 JL      EQU     27
                    00580 *
                    00590 *
            0004    00600 BUF3    EQU     4
                    00610 *
            FFDF    00620 RAM     EQU     $FFDF
            FFDE    00630 ROM     EQU     $FFDE
            0050    00640 IM      EQU     $50
            00AF    00650 IU      EQU     $AF
```

```
                        00660 *
                        00670 *PUT MEMORY SET UP AT FRONT
7D00 CC     3031        00680 COCO3     LDD     #$3031
7D03 FD     FFA4        00690           STD     $FFA4
7D06 CC     3233        00700           LDD     #$3233
7D09 FD     FFA6        00710           STD     $FFA6
7D0C 20     15          00720           BRA     SETDP
                        00730 *
7D0E CC     3C3D        00740 COCO3X    LDD     #$3C3D
7D11 FD     FFA4        00750           STD     $FFA4
7D14 CC     3E3F        00760           LDD     #$3E3F
7D17 FD     FFA6        00770           STD     $FFA6
7D1A 20     11          00780           BRA     RBAS1X
                        00790 *
7D1C 1A     50          00800 SETRAM    ORCC    #IM
7D1E 21     E0          00810           BRN     COCO3   CHANGE WITH POKE
7D20 7F     FFDF        00820           CLR     RAM
7D23 86     02          00830 SETDP     LDA     #2
7D25 1F     8B          00840           TFR     A,DP
7D27 39                 00850           RTS
                        00860 *
7D28 21     E4          00870 RBASIC    BRN     COCO3X
7D2A 7F     FFDE        00880           CLR     ROM
7D2D 1C     AF          00890 RBAS1X    ANDCC   #IU
7D2F 4F                 00900 RBAS1     CLRA
7D30 1F     8B          00910           TFR     A,DP
7D32 39                 00920           RTS
                        00930 *
7D33        6D60        00940 TABLE     FDB     28000   START
7D35        7C9C        00950           FDB     31900   END
7D37        6D60        00960           FDB     28000   NOW
                        00970 *
7D39 8D     E8          00980 INIT      BSR     SETDP
7D3B 17     00B5        00990           LBSR    CONVT
7D3E 1F     01          01000           TFR     D,X     GET FROM
7D40 1F     B8          01010           TFR     DP,A    PUT MEMORY
7D42 C6     50          01020           LDB     #ARECNO ADDRESS
7D44 1F     03          01030           TFR     D,U     IN U
7D46 CC     0B03        01040           LDD     #$0B03  NUMBER
7D49 DD     10          01050           STD     <T1     PARAMETERS
7D4B 17     00A5        01060 INIT1     LBSR    CONVT   MOVE PARAMS
7D4E ED     C1          01070           STD     ,U++    FROM ARRAY
7D50 30     05          01080           LEAX    5,X     TO DP
7D52 0A     10          01090           DEC     <T1     TABLE
7D54 26     F5          01100           BNE     INIT1
7D56 30     8C DA       01110           LEAX    TABLE,PCR
7D59 EC     81          01120 INIT2     LDD     ,X++    PARAMS FROM
7D5B ED     C1          01130           STD     ,U++    TABLE
7D5D 0A     11          01140           DEC     <T1+1   AS WELL
7D5F 26     F8          01150           BNE     INIT2
7D61 96     5E          01160           LDA     <STSTR
7D63 9A     5F          01170           ORA     <STSTR+1
7D65 97     6C          01180           STA     <MVSW   SET SW
7D67 20     C6          01190           BRA     RBAS1
                        01200 *
                        01210 *MOVE FROM RECORD TO  BUFFER
                        01220 *
7D69 8D     B1          01230 MRTB      BSR     SETRAM
```

```
7D6B 0F    09      01240          CLR     <ERSW   ERROR SW
7D6D 96    57      01250          LDA     <NUMFLD NO FIELDS
7D6F 97    0C      01260          STA     <CNT    IN RECORD
7D71 9E    6A      01270          LDX     <THISBF CURRENT END BUFFER
7D73 9F    10      01280          STX     <T1     FOR LATER
7D75 CC    0002    01290          LDD     #2      INITIALISE
7D78 DD    0E      01300          STD     <RCNT   REC COUNT
7D7A 30    02      01310          LEAX    2,X     FOR COUNT
7D7C 9F    6A      01320          STX     <THISBF NEW END BUFFER
7D7E DE    58      01330          LDU     <ARRAY  START ARRAY MOVE FROM
                   01340 *
7D80 9E    0E      01350 MRTB1    LDX     <RCNT   CURRENT REC LEN
7D82 E6    C4      01360          LDB     ,U      LEN THIS STR
7D84 3A            01370          ABX             INC FOR STR
7D85 30    01      01380          LEAX    1,X     PLUS LEN BYTE
7D87 9F    0E      01390          STX     <RCNT   NEW LEN
                   01400 *
7D89 9E    6A      01410          LDX     <THISBF
7D8B 3A            01420          ABX             CHECK OVERFLOW
7D8C 30    01      01430          LEAX    1,X
7D8E 9C    68      01440          CMPX    <ENDBUF
7D90 24    37      01450          BHS     ERR1
7D92 9E    6A      01460          LDX     <THISBF
7D94 E7    80      01470          STB     ,X+     LEN THIS STR
7D96 5D            01480          TSTB
7D97 27    0A      01490          BEQ     MRTB3   IS NUL
7D99 10AE  42      01500          LDY     2,U     THIS STR DATA
7D9C A6    A0      01510 MRTB2    LDA     ,Y+     MOVE DATA
7D9E A7    80      01520          STA     ,X+
7DA0 5A            01530          DECB
7DA1 26    F9      01540          BNE     MRTB2
7DA3 9F    6A      01550 MRTB3    STX     <THISBF
7DA5 33    45      01560          LEAU    5,U     TO NEXT VARPTR
7DA7 0A    0C      01570          DEC     <CNT
7DA9 26    D5      01580          BNE     MRTB1   DO ALL FIELDS
                   01590 *
7DAB 9E    10      01600          LDX     <T1     PUT RECORD
7DAD DC    0E      01610          LDD     <RCNT   LENGTH IN
7DAF ED    84      01620          STD     ,X      BUFFER
7DB1 20    13      01630          BRA     RETBAS
                   01640 *
7DB3 17    FF66    01650          LBSR    SETRAM
7DB6 DE    58      01660          LDU     <ARRAY  RESET INPUT
7DB8 D6    57      01670          LDB     <NUMFLD RECORD
7DBA 6F    C4      01680 MRTB5    CLR     ,U      TO NULL
7DBC 33    45      01690          LEAU    5,U
7DBE 5A            01700          DECB
7DBF 26    F9      01710          BNE     MRTB5
7DC1 DC    0A      01720          LDD     <ESPACE RESET BASICS
7DC3 FD    0023    01730          STD     >35     POINTER
7DC6 16    FF5F    01740 RETBAS   LBRA    RBASIC
7DC9 0C    09      01750 ERR1     INC     <ERSW
7DCB 20    F9      01760          BRA     RETBAS
                   01770 *
                   01780 *BRING A RECORD BACK FROM BUFFER TO RECORD ARRAY
                   01790 *
7DCD 17    FF4C    01800 MBTR     LBSR    SETRAM
7DD0 8D    3D      01810          BSR     FINDR   POINT U TO IT
```

```
7DD2 109E 58      01820        LDY    <ARRAY  START ARRAY
7DD5 9E   5A      01830        LDX    <STOUT  START STRING SPACE
7DD7 96   57      01840        LDA    <NUMFLD
7DD9 97   10      01850        STA    <T1
7DDB E6   C0      01860 MBTR1  LDB    ,U+     LEN STR
7DDD E7   A4      01870        STB    ,Y      SET VARPTR
7DDF AF   22      01880        STX    2,Y
7DE1 5D           01890        TSTB
7DE2 27   07      01900        BEQ    MBTR3
7DE4 A6   C0      01910 MBTR2  LDA    ,U+     MOVE STRING
7DE6 A7   80      01920        STA    ,X+     DATA
7DE8 5A           01930        DECB
7DE9 26   F9      01940        BNE    MBTR2
7DEB 31   25      01950 MBTR3  LEAY   5,Y     NEXT VARPTR
7DED 0A   10      01960        DEC    <T1
7DEF 26   EA      01970        BNE    MBTR1
7DF1 20   D3      01980        BRA    RETBAS
                  01990 *
                  02000 *CONVERT FLOATING POINT NUMBER (X) TO INTEGER (D)
                  02010 *
7DF3 6D   84      02020 CONVT  TST    ,X
7DF5 27   15      02030        BEQ    ZERO
7DF7 86   90      02040        LDA    #144    MAX VALID
7DF9 A0   84      02050        SUBA   ,X      EXPONENT
7DFB 97   0C      02060        STA    <CNT
7DFD EC   01      02070        LDD    1,X     MANTISSA
7DFF 8A   80      02080        ORA    #$80    ASSUME POSITIVE
7E01 0D   0C      02090        TST    <CNT
7E03 27   06      02100        BEQ    EXCONV  NO MOVE REQUIRED
7E05 44           02110 CONVT1 LSRA           MOVE TO INTEGER
7E06 56           02120        RORB           POS IN D
7E07 0A   0C      02130        DEC    <CNT
7E09 26   FA      02140        BNE    CONVT1
7E0B 39           02150 EXCONV RTS
7E0C 4F           02160 ZERO   CLRA
7E0D 5F           02170        CLRB
7E0E 39           02180        RTS
                  02190 *
                  02200 *SET U TO RECORD NUMBER
                  02210 *
7E0F 9E   50      02220 FINDR  LDX    <ARECNO VARPTR REC NO
7E11 DE   66      02230 FINDR1 LDU    <STBUF  START BUFFER
7E13 6D   84      02240        TST    ,X
7E15 27   0E      02250        BEQ    FOUND   IS RECORD ZERO
7E17 8D   DA      02260        BSR    CONVT
7E19 1F   01      02270        TFR    D,X     FOR COUNTING
7E1B 30   1F      02280 FINDR2 LEAX   -1,X    COUNT STARTS ONE
7E1D 27   06      02290        BEQ    FOUND
7E1F EC   C4      02300        LDD    ,U      LEN THIS RECORD
7E21 33   CB      02310        LEAU   D,U     INC ADDRESS
7E23 26   F6      02320        BNE    FINDR2
7E25 33   42      02330 FOUND  LEAU   2,U     PAST RECORD LENGTH
7E27 39           02340 FOUND1 RTS
                  02350 *
                  02360 *ADVANCE U TO FIELD NUMBER
                  02370 *
7E28 9E   54      02380 FINDF  LDX    <AFLDNO VARPTR FIELD NO
7E2A 6D   84      02390 FINDF1 TST    ,X
```

```
7E2C 27   F9     02400        BEQ   FOUND1  IS FIELD ZERO
7E2E 8D   C3     02410        BSR   CONVT
7E30 D7   0C     02420        STB   <CNT
7E32 4F          02430        CLRA          FOR REG ADJUST
7E33 E6   C4     02440 FINDF2 LDB   ,U      LEN THIS FIELD
7E35 33   CB     02450        LEAU  D,U
7E37 33   41     02460        LEAU  1,U     TO NEXT LEN
7E39 0A   0C     02470        DEC   <CNT
7E3B 26   F6     02480        BNE   FINDF2
7E3D 39          02490        RTS
                 02500 *
                 02510 *MOVE FROM BUFFER TO SORT/SEARCH ARRAY
                 02520 *
7E3E 17   FEDB   02530 MBTSS  LBSR  SETRAM
7E41 0F   09     02540        CLR   <ERSW
7E43 DC   5E     02550        LDD   <STSTR  START STR SPACE
7E45 DD   12     02560        STD   <T2     SAVE IT
7E47 9E   52     02570        LDX   <ANREC
7E49 8D   A8     02580        BSR   CONVT   COUNTER
7E4B DD   0E     02590        STD   <RCNT   FOR MOVE
7E4D 9E   62     02600        LDX   <SPOINT POINTER ARRAY
7E4F 9F   10     02610        STX   <T1
7E51 109E 5C     02620        LDY   <SRTARY SS ARRAY
                 02630 *
7E54 9E   10     02640 MBTSS2 LDX   <T1     POINTER
7E56 8D   B9     02650        BSR   FINDR1  POINT U TO RECORD
7E58 8D   CE     02660        BSR   FINDF   TO FIELD REQ
7E5A 9E   12     02670        LDX   <T2     START STRING SPACE
7E5C E6   C0     02680        LDB   ,U+     LEN THIS STR
7E5E 0D   6C     02690        TST   <MVSW
7E60 27   29     02700        BEQ   MBTSS4  DONT MOVE DATA
7E62 3A          02710        ABX
7E63 9C   60     02720        CMPX  <ENDSTR END SPACE
7E65 1024 FF60   02730        LBHS  ERR1
7E69 9E   12     02740        LDX   <T2     RESET
7E6B E7   A4     02750        STB   ,Y      SET VARPTR
7E6D AF   22     02760        STX   2,Y     THIS STRING
7E6F A6   C0     02770 MBTSS3 LDA   ,U+     MOVE STRING
7E71 A7   80     02780        STA   ,X+     DATA
7E73 5A          02790        DECB
7E74 26   F9     02800        BNE   MBTSS3
7E76 9F   12     02810        STX   <T2     RESET TO NEW
7E78 31   25     02820 MBTSS6 LEAY  5,Y     TO NEXT VARPTR
7E7A 9E   10     02830        LDX   <T1     RESET TO
7E7C 30   05     02840        LEAX  5,X     POINTER
7E7E 9F   10     02850        STX   <T1     ARRAY
7E80 9E   0E     02860        LDX   <RCNT
7E82 30   1F     02870        LEAX  -1,X
7E84 9F   0E     02880        STX   <RCNT
7E86 26   CC     02890        BNE   MBTSS2
7E88 16   FE9D   02900 RETB2  LBRA  RBASIC
                 02910 *
7E8B E7   A4     02920 MBTSS4 STB   ,Y      SET VARPTR
7E8D EF   22     02930        STU   2,Y     ONLY
7E8F 20   E7     02940        BRA   MBTSS6
                 02950 *
                 02960 *DELETE A RECORD FROM MEMORY
                 02970 *
```

```
7E91 17     FE88      02980 DELETE  LBSR   SETRAM
7E94 17     FF78      02990         LBSR   FINDR    POINT U TO IT
7E97 33     5E        03000         LEAU   -2,U     BACK TO START
7E99 1F     31        03010         TFR    U,X
7E9B EC     84        03020         LDD    ,X       LEN THIS ONE
7E9D 30     8B        03030         LEAX   D,X      TO NEXT
7E9F A6     80        03040 DEL1    LDA    ,X+
7EA1 A7     C0        03050         STA    ,U+      MOVE REST OVER TOP
7EA3 9C     6A        03060         CMPX   <THISBF
7EA5 25     F8        03070         BLO    DEL1
7EA7 DF     6A        03080         STU    <THISBF
7EA9 20     DD        03090         BRA    RETB2
                      03100 *
                      03110 *ROUTINE TO SET SWITCHED REQUIRED FOR MULTI LEVEL SORT
                      03120 *
7EAB 17     FE6E      03130 MLSRT   LBSR   SETRAM
7EAE 0F     09        03140         CLR    <ERSW    USE AS SWITCH
7EB0 9E     52        03150         LDX    <ANREC
7EB2 17     FF3E      03160         LBSR   CONVT    COUNTER TO
7EB5 DD     0E        03170         STD    <RCNT    FIND END
7EB7 109E 64          03180         LDY    <SORTSW START MEMORY
7EBA 9E     5C        03190         LDX    <SRTARY SET UP
7EBC 9F     14        03200         STX    <I       FIRST VARPTR
7EBE A6     84        03210         LDA    ,X       FOR LOOP
7EC0 97     1B        03220         STA    <JL      THAT FOLLOWS
7EC2 EC     02        03230         LDD    2,X
7EC4 DD     18        03240         STD    <JA
7EC6 6F     A0        03250         CLR    ,Y+      FIRST SW
                      03260 *
7EC8 DC     18        03270 MLSRT1  LDD    <JA      MOVE THIS
7ECA DD     16        03280         STD    <IA      TO PREVOUS
7ECC 96     1B        03290         LDA    <JL
7ECE 97     1A        03300         STA    <IL
7ED0 9E     0E        03310         LDX    <RCNT
7ED2 30     1F        03320         LEAX   -1,X
7ED4 27     B2        03330         BEQ    RETB2    FINISHED
7ED6 9F     0E        03340         STX    <RCNT
                      03350 *
7ED8 9E     14        03360         LDX    <I       MOVE TO NEXT
7EDA 30     05        03370         LEAX   5,X      VARPTR
7EDC 9F     14        03380         STX    <I
7EDE A6     84        03390         LDA    ,X       SET POINTERS
7EE0 97     1B        03400         STA    <JL      FOR THIS ONE
7EE2 EC     02        03410         LDD    2,X
7EE4 DD     18        03420         STD    <JA
7EE6 8D     02        03430         BSR    DOCOMP
7EE8 20     DE        03440         BRA    MLSRT1
                      03450 *
                      03460 *DO COMPARASION OF STRINGS AND SET SWITCH FOR
                      03470 *
7EEA D6     1A        03480 DOCOMP  LDB    <IL
7EEC D1     1B        03490         CMPB   <JL
7EEE 26     15        03500         BNE    NO
7EF0 5D               03510         TSTB
7EF1 27     0D        03520         BEQ    YES      BOTH NULL
7EF3 9E     16        03530         LDX    <IA      TO STRING
7EF5 DE     18        03540         LDU    <JA      DATA
7EF7 A6     80        03550 DCOMP1  LDA    ,X+
7EF9 A1    C0      03560          CMPA     ,U+
7EFB 26    08      03570          BNE      NO
7EFD 5A            03580          DECB
7EFE 26    F7      03590          BNE      DCOMP1
                   03600 *
7F00 96    09      03610 YES      LDA      <ERSW    SET SW FOR THIS
7F02 A7    A0      03620          STA      ,Y+      COMPARASION
7F04 39            03630          RTS
7F05 0C    09      03640 NO       INC      <ERSW    CHANGE SW
7F07 20    F7      03650          BRA      YES      AND SET IT
                   03660 *
           7FF8    03670 ASORT    EQU      ZZEND+3+219
           81B8    03680 ASERCH   EQU      ZZEND+3+583+84
7F09 17    FE10    03690          LBSR     SETRAM
7F0C 17    00E9    03700          LBSR     ASORT
7F0F 16    FE16    03710 RETB4    LBRA     RBASIC
                   03720 *
7F12 17    FE07    03730          LBSR     SETRAM
7F15 17    02A0    03740          LBSR     ASERCH
7F18 20    F5      03750          BRA      RETB4
           7F1A    03760 ZZEND    EQU      *
           7D39    03770          END      INIT
 00000 TOTAL ERRORS
```

The Listing:

```
                   00100 *CALLED XSEARCH - SPECIAL SEARCH FOR ONEARRAY BUFFER
                   00110 * WHICH SEARCHES THE BUFFER WITHOUT A SORT/SELECT
                   00115 *ARRAY IN THE PROGRAM AREA
                   00120 * IS INSTALLED AT THE END OF A BASIC PROGRAM BEFORE
                   00125 *ONEARRAY
                   00130 *PARAMETERS USED ARE AS INITIALISED BY ONEARRAY
                   00140 *P(0)=VARPTR THIS RECORD
                   00150 *P(6)=VARPTR OF SEARCH KEY
                   00160 *
7D00               00170          ORG      32000
           0050    00180 ARECNO   EQU      80
           0062    00190 SPOINT   EQU      98
           006A    00200 THISBF   EQU      106
           0009    00210 SW1      EQU      9
           000A    00220 STREC    EQU      10
           000C    00230 SW       EQU      12
           000D    00240 IL       EQU      13
           000E    00250 IA       EQU      14
           0010    00260 JA       EQU      16
           0012    00270 CNT      EQU      18
                   00280 *
           7DAC    00290 SETRAM   EQU      ZZEND+3+$1C
           7E83    00300 CONVT    EQU      ZZEND+3+$F3
           7EA1    00310 FINDR1   EQU      ZZEND+3+$111
           7EB8    00320 FINDF    EQU      ZZEND+3+$128
           7E5D    00330 MPTR     EQU      ZZEND+3+$CD
                   00340 *
                   00350 *SEARCH FIELD AS SET UP FOR
                   00360 *
7D00 0F    0C      00370 SERCH    CLR      <SW
7D02 A6    C0      00380          LDA      ,U+         LEN THIS STRING
```

```
7D04 0D   09     00390        TST   <SV1    TYPE SERCH
7D06 26   04     00400        BNE   S1
7D08 91   0D     00410        CMPA  <IL     IF EQUAL
7D0A 26   26     00420        BNE   NO
7D0C 91   0D     00430 S1     CMPA  <IL     LEN KEY
7D0E 25   22     00440        BLO   NO      SERCH STR > THIS ONE
7D10 90   0D     00450        SUBA  <IL     DIFFERENCE
7D12 4C          00460        INCA          ADJ FOR COUNT
7D13 97   12     00470        STA   <CNT
7D15 DF   10     00480 T0     STU   <JA     SAVE IT
7D17 9E   0E     00490        LDX   <IA     ADDR KEY
7D19 D6   0D     00500        LDB   <IL     LEN
7D1B A6   80     00510 T1     LDA   ,X+     COMPARE DATA
7D1D A1   C0     00520        CMPA  ,U+
7D1F 26   05     00530        BNE   NOTYET
7D21 5A          00540        DECB
7D22 26   F7     00550        BNE   T1
7D24 20   0A     00560        BRA   YES
                 00570 *
7D26 0A   12     00580 NOTYET DEC   <CNT
7D28 27   08     00590        BEQ   NO
7D2A DE   10     00600        LDU   <JA
7D2C 33   41     00610        LEAU  1,U     TO NEXT BYTE
7D2E 20   E5     00620        BRA   T0      DO IT
7D30 0C   0C     00630 YES    INC   <SV
7D32 39          00640 NO     RTS
                 00650 *
                 00660 *ENTRY FOR ROUTINE
                 00670 *SV1 IS SET BY A POKE FROM BASIC
                 00680 *
7D33 8D   77     00690 ENTRY  BSR   SETRAM
7D35 9E   62     00700        LDX   <SPOINT
7D37 A6   84     00710        LDA   ,X      SET POINTERS
7D39 97   0D     00720        STA   <IL     TO SERCH
7D3B AE   02     00730        LDX   2,X     STRING
7D3D 9F   0E     00740        STX   <IA
                 00750 *POINT U TO STARTING RECORD
7D3F 9E   50     00760        LDX   <ARECNO
7D41 17   013F   00770        LBSR  CONVT
7D44 1F   02     00780        TFR   D,Y     FOR COUNTER
7D46 17   0158   00790        LBSR  FINDR1
                 00800 *SEARCH SELECTED FIELD IN EACH RECORD
7D49 DF   0A     00810 X1     STU   <STREC  START THIS ONE
7D4B 17   016A   00820        LBSR  FINDF   POINT TO FIELD
7D4E 8D   B0     00830        BSR   SERCH
7D50 0D   0C     00840        TST   <SV
7D52 26   13     00850        BNE   FOUND
7D54 31   21     00860        LEAY  1,Y     INC COUNT
7D56 DE   0A     00870        LDU   <STREC  INC POINTER
7D58 EC   5E     00880        LDD   -2,U    TO NEXT
7D5A 33   CB     00890        LEAU  D,U     RECORD
7D5C 1193 6A     00900        CMPU  <THISBF END BUFFER
7D5F 25   E8     00910        BLO   X1
                 00920 *RECORD NOT FOUND SET REC NUM TO ZERO
7D61 9E   50     00930        LDX   <ARECNO
7D63 4F          00940        CLRA
7D64 5F          00950        CLRB
7D65 20   1D     00960        BRA   ZFP
```

```
                    00970 *RECORD FOUND
                    00980 *CONVERT COUNTER IN Y TO FP NUMBER IN RECORD NUMBER
7D67 9E     50      00990 FOUND   LDX     <ARECNO
7D69 86     90      01000         LDA     #144
7D6B A7     80      01010         STA     ,X+
7D6D 109F   10      01020         STY     <JA
7D70 DC     10      01030         LDD     <JA
7D72 27     10      01040         BEQ     ZFP
7D74 58             01050 TOFP1   ASLB
7D75 49             01060         ROLA
7D76 25     04      01070         BCS     DONE
7D78 6A     1F      01080         DEC     -1,X
7D7A 20     F8      01090         BRA     TOFP1
7D7C 44             01100 DONE    LSRA
7D7D 56             01110         RORB
7D7E ED     81      01120         STD     ,X++
7D80 4F             01130         CLRA
7D81 5F             01140         CLRB
7D82 20     04      01150         BRA     ZFP1
                    01160 *SET TO ZERO
7D84 A7     1F      01170 ZFP     STA     -1,X
7D86 ED     81      01180         STD     ,X++
7D88 ED     81      01190 ZFP1    STD     ,X++
                    01200 *EXIT TO BASIC BY RETURNING RECORD AS WELL
                    01210 *IF RECORD NOT FOUND WILL RETURN RECORD NUMBER 1
                    01220 *
7D8A 16     00D0    01230         LBRA    MPTR
                    01240 *
            7D8D    01250 ZZEND   EQU     *
            7D33    01260         END     ENTRY

00000 TOTAL ERRORS
```

The Listing:

```
                    00100 *CALLED  XSORT  SPECIAL SORT FOR ONEARRAY BUFFER
                    00110 *WHICH SORTS THE RECORDS IN THE BUFFER WITHOUT
                    00115 *A SORT/SELECT ARRAY
                    00120 *IS A REPLACEMENT SORT
                    00130 *IT IS INSTALLED IMMEDIATELY AFTER ONEARRAY AND IS
                    00135 *ENTERED AS FOR ASORT, WITH A DIFFERENT OFFSET IN M+526
                    00140 *PARAMETERS USED ARE AS INITIALISED FOR ONEARRAY
                    00150 *P(1)=NUMBER OF RECORDS IN BUFFER
                    00160 *P(2)=THIS FIELD NUMBER
                    00170 *
7D00                00180         ORG     32000
            0052    00190 ANREC   EQU     82
            0054    00200 AFLDNO  EQU     84
            006A    00210 THISBF  EQU     106
            0066    00220 STBUF   EQU     102
            7BD6    00230 CONVT   EQU     ENTRY-$127-3
            7C0B    00240 FINDF   EQU     CONVT+53
                    00250 *
            0009    00260 TL      EQU     9
            000A    00270 IL      EQU     10
            000B    00280 JL      EQU     11
            000E    00290 IA      EQU     14
```

```
          0010     00300 JA       EQU     16
          0012     00310 IST      EQU     18
          0014     00320 JST      EQU     20
          0016     00330 STREC    EQU     22
          0018     00340 CNT1     EQU     24
          001A     00350 CNT2     EQU     26
                   00360 *
7D00 9E   52       00370 ENTRY    LDX     <ANREC
7D02 17   FED1     00380          LBSR    CONVT    NO OF RECORDS
7D05 1F   01       00390          TFR     D,X
7D07 9F   18       00400 S0       STX     <CNT1    START EACH LOOP
7D09 DE   66       00410          LDU     <STBUF   WITH FIRST
7D0B 33   42       00420          LEAU    2,U      RECORD AS
7D0D DF   12       00430          STU     <IST     LOWEST VALUE
7D0F 17   FEF9     00440          LBSR    FINDF    CALLED LOWER
7D12 A6   C0       00450          LDA     ,U+
7D14 97   09       00460          STA     <TL
7D16 DF   0E       00470          STU     <IA
7D18 DE   12       00480          LDU     <IST     RESET POINTERS
7D1A 9E   18       00490          LDX     <CNT1
                   00500 *LOOP TO SEARCH BUFFER FOR LOWEST VALUE
7D1C 9F   1A       00510 S1       STX     <CNT2
7D1E DF   14       00520          STU     <JST
7D20 17   FEE8     00530          LBSR    FINDF    POINT TO FIELD
7D23 A6   C0       00540          LDA     ,U+      IN NEXT RECORD
7D25 97   0B       00550          STA     <JL
7D27 DF   10       00560          STU     <JA
7D29 9E   0E       00570          LDX     <IA      CURRENT LOWEST
7D2B 96   09       00580          LDA     <TL
7D2D 97   0A       00590          STA     <IL
7D2F 0D   0B       00600 ASC      TST     <JL
7D31 26   06       00610          BNE     NZ
7D33 0D   0A       00620          TST     <IL      LEN LOWER
7D35 27   1E       00630          BEQ     NOSWAP   BOTH ZERO
7D37 20   10       00640          BRA     SWAP     UPPER ZERO LOWER NOT
7D39 0D   0A       00650 NZ       TST     <IL
7D3B 27   18       00660          BEQ     NOSWAP   LOWER ZERO UPPER NOT
                   00670 *BOTH NON ZERO COMPARE NEXT BYTE
7D3D 0A   0B       00680          DEC     <JL      DEC COUNTERS
7D3F 0A   0A       00690          DEC     <IL
7D41 A6   80       00700          LDA     ,X+      LOWER BYTE
7D43 A1   C0       00710          CMPA    ,U+      UPPER
7D45 27   E8       00720          BEQ     ASC      STILL EQUAL
7D47 25   0C       00730          BLO     NOSWAP   LOWER LOEVR THAN UPPER
                   00740 *
7D49 DC   14       00750 SWAP     LDD     <JST     MOVE POINTERS
7D4B DD   12       00760          STD     <IST     FOR UPPER STR
7D4D DC   10       00770          LDD     <JA      TO BE POINTERS
7D4F DD   0E       00780          STD     <IA      FOR LOWER ONE
7D51 96   0B       00790          LDA     <JL
7D53 97   09       00800          STA     <TL
                   00810 *MOVE POINTERS FOR UPPER STRING TO NEXT RECORD
7D55 DE   14       00820 NOSWAP   LDU     <JST
7D57 EC   5E       00830          LDD     -2,U     REC LEN
7D59 33   CB       00840          LEAU    D,U      TO NEXT RECORD
7D5B 9E   1A       00850          LDX     <CNT2    COUNTER TO COMPARE
7D5D 30   1F       00860          LEAX    -1,X     ALL RECORDS
7D5F 26   BB       00870          BNE     S1
```

```
                        00880 *
                        00890 *HAVE DONE ALL RECORDS LEFT AND POINTER TO
                        00895 *LOWER STRING NOW DEFINES LOWEST VALUE IN BUFFER
                        00900 *COPY THIS RECORD TO END OF BUFFER, AND THEN
                        00905 * DELETE IT FROM ITS PRESENT POSITION
                        00910 *
7D61 DE   12            00920          LDU     <IST
7D63 33   5E            00930          LEAU    -2,U     START RECORD
7D65 10AE C4            00940          LDY     ,U       FOR COUNTER
7D68 9E   6A            00950          LDX     <THISBF
7D6A A6   C0            00960 X1       LDA     ,U+      MOVE IT
7D6C A7   80            00970          STA     ,X+
7D6E 31   3F            00980          LEAY    -1,Y
7D70 26   F8            00990          BNE     X1
7D72 9F   6A            01000          STX     <THISBF NEW END BUFFER
                        01010 *DELETE OD COPY
7D74 DE   12            01020          LDU     <IST     SET U TO
7D76 33   5E            01030          LEAU    -2,U     START OF RECORD
7D78 1F   31            01040          TFR     U,X      SET X TO
7D7A EC   C4            01050          LDD     ,U       START NEXT
7D7C 30   8B            01060          LEAX    D,X      RECORD
7D7E A6   80            01070 D1       LDA     ,X+      MOVE OVER IT
7D80 A7   C0            01080          STA     ,U+
7D82 9C   6A            01090          CMPX    <THISBF
7D84 25   F8            01100          BLO     D1
7D86 DF   6A            01110          STU     <THISBF NEW END
                        01120 *RESET STARTING NUMBER OF RECORDS
7D88 9E   18            01130          LDX     <CNT1
7D8A 30   1F            01140          LEAX    -1,X
7D8C 1026 FF77          01150          LBNE    S0
7D90 39                 01160          RTS               FINISHED
                        01170 *
          7D91          01180 ZZEND    EQU     *
          7D00          01190          END     ENTRY

 00000 TOTAL ERRORS
```

The Listing:

```
1 'LISTING 1 - CONTAINS CODE OF
MODULES THAT ARE DIFFERENT BETWE
EN XCARD AND INDXCARD
3 SAVE"116CA/BAS:3":END'7
2500 'SEARCH FILE
2510 CLS:PRINT "SEARCH FILE"
2530 PRINT "ENTER COLUMN NUMBER
TO SEARCH"
2540 GOSUB 710:INPUT B:GOSUB 310
: EXEC V3
2550 P(0)=VARPTR(T$(0)) 'SEARCH
ARRAY
2560 P(1)=1 'START
2570 P(2)=N 'END
2580 P(3)=VARPTR(B$) 'KEY
2590 P(4)=1:P(5)=0
2600 P(6)=0 'NO GAP
2620 PRINT "ENTER VALUE TO FIND"
2630 INPUT B$
2640 X=USR2(VARPTR(P(0)))
2650 IF P(9)=0 THEN PRINT "VALUE
 NOT IN ARRAY":PRINT "ENTER  Q
TO QUIT":PRINT "OR ENTER TO TRY
AGAIN":GOSUB 380:IF A$="Q" THEN
110 ELSE 2620
2660 Q=N(P(7)):EXEC V2:GOSUB 510
2670 IF A$="Y" THEN Q=P(7):GOTO
110
2680 P(1)=P(7)+1:GOTO 2640
2800 'SORT FILE
2810 CLS:PRINT "SORT LIST"
2820 PRINT "ENTER COLUMN NUMBER
FOR FIRST   SORT - WHERE";:GOSUB
 710
2830 INPUT B:GOSUB 310
2840 P(0)=N 'NUMBER TO SORT
2850 P(1)=VARPTR(T$(1)) 'SORT AR
RAY
2860 P(2)=VARPTR(N(1)) 'INDIRECT
2870 P(3)=0:P(4)=1:P(5)=255
2880 FOR X=0 TO N 'SET POINTER A
RRAY
2890 N(X)=X:NEXT X
2900 EXEC V3 'MOVE FOR SORT
2910 X=USR(VARPTR(P(0))) 'SORT
2920 EXEC V5 'SET KEYS
2930 'MULTI LEVEL SORTS
2940 PRINT "ENTER COLUMN FOR NEX
T SORT":PRINT "OR  -1  TO FINISH
"
2950 PRINT "WHERE SORT KEY IS":G
OSUB 720
2960 INPUT B:IF B<0 THEN 110 ELS
E GOSUB 310
2970 EXEC V3 'MOVE FOR NEXT SORT
2980 'SORT NEXT SUB-GROUP
2990 P(0)=1:Y=1:A=VARPTR(TF(0))-
1
3000 FOR X=2 TO N
3010 IF PEEK(A+X)=PEEK(A+X-1) TH
EN P(0)=P(0)+1:GOTO 3060
3020 IF P(0)=1 THEN 3050
3030 P(1)=VARPTR(T$(Y)):P(2)=VAR
PTR(N(Y))
3040 P=USR(VARPTR(P(0))) 'SORT
3050 P(0)=1:Y=X 'RESET
3060 NEXT X
3070 IF P(0)>1 THEN P(1)=VARPTR(
T$(Y)):P(2)=VARPTR(N(Y)):P=USR(V
ARPTR(P(0)))
3080 EXEC V5:GOTO 2940
3200 'PRINT REPORTS
```

```
5000 'INITIALISE VARIABES
5010 DIM A$(NF),N(MR),T$(MR),P(1
0),C$(NF),TE(INT(ML/5)+1),TF(INT
(MR/5)+1)
5020 PRINT "SETTING UP"
5030 DIM X,Y,B,Q,A$,B$,A,P,N,N$,
A1,A2,A3,V0,V1,V2,V3,V4,V5,V6,V9
,DP,M,M1,M2,Q1$,Q2$
5040 A=PEEK(27)*256+PEEK(28)-55
5050 B=A-389:X=B-583:M=X-541
5060 DEFUSR0=M+521 'SORT
5070 DEFUSR2=M+530 'SEARCH
5080 IF PEEK(&HE301)=87 THEN POK
E M+30,32:POKE M+40,32 ELSE POKE
 M+30,33:POKE M+40,33 'FOR COCO
3 OR OLD ONE
5090 'SET TITLES FOR FIELDS
5100 C$(0)="FIRST NAME"
5110 C$(1)="TITLE / INITIAL"
5120 C$(2)="SURNAME"
5130 C$(3)="STREET NUMBER"
5140 C$(4)="STREET NAME"
5150 C$(5)="SUBURB"
5160 C$(6)="POSTCODE"
5170 C$(7)="PHONE NUMBER"
5180 C$(8)="MEMBERSHIP NUMBER"
5190 C$(9)="EXTRA 1"
5200 C$(10)="EXTRA 2"
5210 '
5220 GOSUB 5280
5230 FOR X=0 TO MR
5240 N(X)=X
5250 NEXT X
5260 RETURN
5270 'INITIALISE BUFFER
5280 DP=PEEK(M+36)*256
5290 V0=M+57 'INITIALISE
5300 V1=M+105 'MOVE TO BUFFER
5310 V2=M+205 'FROM BUFFER TO RE
CORD AREA
5320 V3=M+318 'BUFFER TO SORT AR
RAY
5330 V4=M+401 'DELETE A RECORD
5340 V5=M+427 'SET SORT SWITCHES
5350 V6=M+179 'RESET STRING SPAC
E
5360 V9=DP+9 'ERSW
5370 '
5380 P(0)=VARPTR(Q) 'THIS RECORD
 NO
5390 P(1)=VARPTR(N) 'NO OF RECOR
DS
5400 P(2)=VARPTR(B) 'THIS FIELD
NO
5410 P(3)=NF+1 'NO FIELDS IN REC
ORD
5420 P(4)=VARPTR(A$(0)) 'RECORD
ARRAY
5430 P(5)=VARPTR(TE(0)) 'STRING
SPACE RECORD AREA
5440 P(6)=VARPTR(T$(1)) 'SORT AR
RAY
5450 P(7)=0
5460 P(8)=0
5470 P(9)=VARPTR(N(1)) 'POINTER
ARRAY
5480 P(10)=VARPTR(TF(0)) 'SORT S
WITCHES
5490 '
5500 DEFUSR3=V0:X=USR3(VARPTR(P(
0)))
5505 POKE M+526,14 'ADJUST ENTRY
 FOR XSORT
5510 FOR X=0 TO 10:P(X)=0:NEXT X
5520 RETURN
```

The Listing:

```
1 '**  XSEARCH - SPECIAL SEARCH
       FOR ONEARRAY BUFFER
2 GOTO 51000
3 SAVE"116CB:3":END'7
6 'CAN ONLY BE USED IN ASSOCIATI
ON WITH ONEARRAY
9 '
51000 LN=51500:FOR X=0 TO 141 ST
EP 25:IF X<124 THEN N=25 ELSE N=
16
51010 GOSUB 51030:NEXT X
51020 RESTORE:GOTO 51110
51030 PRINT LN;:A=0:FOR Y=0 TO N
-1
51040 READ C$:B=VAL("&H"+C$):A=A
+B
51050 NEXT Y:READ C$:IF A<> VAL(
"&H"+C$) THEN PRINT "ERROR IN LI
NE NO";LN:STOP
51060 LN=LN+10:RETURN
51070 '
51080 FOR Y= 0 TO N-1:READ C$:PO
KE A,VAL("&H"+C$)
51090 A=A+1:NEXT Y:READ C$:RETUR
N
51100 '
51110 M$="9E1B3089008D6F806F806F
809F1B39":Y=&H01DA
51120 B=0:FOR X=1 TO 30 STEP 2:N
=VAL("&H"+MID$(M$,X,2)):B=B+N:PO
KE Y,N:Y=Y+1:NEXT X
51130 IF B <> &H5BF THEN PRINT "
ERROR IN LINE NO 51110":STOP
51140 EXEC &H1DA:CLEAR
51150 A=PEEK(27)*256+PEEK(28)-14
4:LN=51500
51160 FOR X=0 TO 141 STEP 25:IF
X<124 THEN N=25 ELSE N=16
51170 GOSUB 51080:NEXT X
51180 '
51190 PRINT "XSEARCH ML NOW ADDE
D TO END OF  BASIC PROGRAM": PRI
NT "AND EXTRA BASIC CODE DELETED
"
51200'
51210 DEL 51000-51550
51500 DATA F,C,A6,C0,D,9,26,4,91
,D,26,26,91,D,25,22,90,D,4C,97,1
2,DF,10,9E,E,6BD
51510 DATA D6,D,A6,80,A1,C0,26,5
,5A,26,F7,20,A,A,12,27,8,DE,10,3
3,41,20,E5,C,C,800
51520 DATA 39,8D,77,9E,62,A6,84,
97,D,AE,2,9F,E,9E,50,17,1,3F,1F,
2,17,1,58,DF,A,827
51530 DATA 17,1,6A,8D,B0,D,C,26,
13,31,21,DE,A,EC,5E,33,CB,11,93,
6A,25,E8,9E,50,4F,8EB
51540 DATA 5F,20,1D,9E,50,86,90,
A7,80,10,9F,10,DC,10,27,10,58,49
,25,4,6A,1F,20,F8,44,858
51550 DATA 56,ED,81,4F,5F,20,4,A
7,1F,ED,81,ED,81;16,0,D0,71E
```

The Listing:

```
1 '**  XSORT   - SPECIAL SEARCH
       FOR ONEARRAY BUFFER
2 GOTO 52000
3 SAVE"116CC:3":END'7
6 'CAN ONLY BE USED IN ASSOCIATI
ON WITH ONEARRAY
9 '
52000 LN=52500:FOR X=0 TO 145 ST
EP 25:IF X<124 THEN N=25 ELSE N=
20
52010 GOSUB 52030:NEXT X
52020 RESTORE:GOTO 52110
52030 PRINT LN;:A=0:FOR Y=0 TO N
-1
52040 READ C$:B=VAL("&H"+C$):A=A
+B
52050 NEXT Y:READ C$:IF A<> VAL(
"&H"+C$) THEN PRINT "ERROR IN LI
NE NO";LN:STOP
52060 LN=LN+10:RETURN
52070 '
52080 FOR Y= 0 TO N-1:READ C$:PO
KE A,VAL("&H"+C$)
52090 A=A+1:NEXT Y:READ C$:RETUR
N
52100 '
52110 M$="9E1B308900916F806F806F
809F1B39":Y=&H01DA
52120 B=0:FOR X=1 TO 30 STEP 2:N
=VAL("&H"+MID$(M$,X,2)):B=B+N:PO
KE Y,N:Y=Y+1:NEXT X
52130 IF B <> &H5C3 THEN PRINT "
ERROR IN LINE NO 52110":STOP
52140 EXEC &H1DA:CLEAR
52150 A=PEEK(27)*256+PEEK(28)-14
8:LN=52500
52160 FOR X=0 TO 145 STEP 25:IF
X<124 THEN N=25 ELSE N=20
52170 GOSUB 52080:NEXT X
52180 '
52190 PRINT "XSORT ML NOW ADDED
TO END OF     BASIC PROGRAM": PRI
NT "AND EXTRA BASIC CODE DELETED
"
52200 '
52210 DEL 52000-52550
52500 DATA 9E,52,17,FE,D1,1F,1,9
F,18,DE,66,33,42,DF,12,17,FE,F9,
A6,C0,97,9,DF,E,DE,C36
52510 DATA 12,9E,18,9F,1A,DF,14,
17,FE,E8,A6,C0,97,B,DF,10,9E,E,9
6,9,97,A,D,B,26,992
52520 DATA 6,D,A,27,1E,20,10,D,A
,27,18,A,B,A,A,A6,80,A1,C0,27,E8
,25,C,DC,14,5C8
52530 DATA DD,12,DC,10,DD,E,96,B
,97,9,DE,14,EC,5E,33,CB,9E,1A,30
,1F,26,BB,DE,12,33,A4C
52540 DATA 5E,10,AE,C4,9E,6A,A6,
C0,A7,80,31,3F,26,F8,9F,6A,DE,12
,33,5E,1F,31,EC,C4,30,BBD
52550 DATA 8B,A6,80,A7,C0,9C,6A,
25,F8,DF,6A,9E,18,30,1F,10,26,FF
,77,39,96E
```

Utilising the buffer

Using the whole buffer in memory for the total file.

32K DECB ARTICLE

by George McLintock

WITH A DISK system, it is possible to extend the row and column concept to a larger array on disk. In effect you can use the whole buffer in memory as a sort/select column for the total file on disk.

For example, if you have say 2000 records on disk with a similar structure as assumed for INDXCARD, then they won't all fit in memory at the same time, but you can still perform the same effective operations as done by INDXCARD with the file.

For this operation you require a separate array for reading each record from disk, ie assume an array, B$(NF), is needed for reading a record from disk, and an array, A$(T), is needed to move records to and from the buffer.

With this set up, the following outlines a procedure which would allow a three level multi level sort of the file using fields 3, 5, and 7.

Searching the file is a simple subset of the sort, and is not described separately, ie for repeated searches of a column, load that column, search it to find the record number required, and then GET the record.

The easy solution is when the three fields required will fit in the buffer.

In this case simply set up the buffer for a maximum of 2000 records with three fields per record, and read in the file with code like:

```
FOR X=1 TO N
GET #1,X
'input the record into B$(NF)
A$(0)=B$(3):A$(1)=B$(5)
A$(2)=B$(7)
'move the array A$(T) to the
 buffer
NEXT X
```

You can then operate on these three fields in the buffer normally. However, when you want to get the full record, you have to use code like:

```
GET #1,Q
'and input the record into
 B$(NF)
```

... where Q is the record number required.

To access the file in sorted sequence, use code like:

```
FOR X=1 TO N
 GET #1,N(X)
 'input the record into B$(NF)
 'print etc record from B$(NF)
NEXT X
```

The worst case situation is where only one field wil fit in the buffer at a time. In this case you have to load each field in sequence, and use the pointer array to get the correct sequence, ie you set up the buffer for a record with a single field only.

For the first sort, you load B$(3) only into the array and set N(N) to a natural sequence, eg ...

```
FOR X=1 TO N
 N(X)=X
NEXT X
```

Then sort the records in the buffer and set the switches for a multi-level sort.

To do the next sort on B$(5), you have to reset the buffer to empty, and read in the file again using code like:

```
FOR X=1 TO N
 GET #1,N(X)
 'input the record into B$(NF)
 A$(0)=B$(5)
 'move the A$(T) array to the
  buffer
NEXT X
```

Then sort the records in the buffer as for a normal second stage muti-level sort, and set the switches for a multi-level sort.

To do the final sort on B$(7), you repeat the operations above, but setting A$(0)=B$(7).

At the end of this sequence, N(N) is in the sequence required to access the file in sorted sequence.

While this sequence may seem a little tedeous, it is in fact a faster operation than a normal disk sort, and it does not require any extra space on the disk.

A normal disk sort requires a minimum of at least the same amount of working storage on disk as is occupied by the file itself.

If you want to change the order of the records on disk then you can copy it to a new file in sorted sequence, ie open two files, where file #1 is the existing file and file #2 is the new one, and copy the file with code like:

```
OPEN "D",#1,"name",Len
OPEN "D",#2,"name1",Len
FIELD #1, Len AS A$
FIELD #2, Len AS B$
FOR X = 1 TO N
 GET #1,N(X)
 LSET B$=A$
 PUT #2,X
NEXT X
```

This code will work for all direct access files, even if they are organised as set up for "INDXCARD".

For files of this size (around 2000 records), you will have a max of around 8K of memory for the program itself, so it may be limited in other functions that can be performed as part of the same program.

AN ALTERNATIVE SEARCH FOR "ONEARRAY"

For some applications with "ONEARRAY", you might not require a column array except as a means of searching the file for a particular record. And at other times, you might not have memory available in the program area to hold such an array.

"XSEARCH" is a ML routine which has been designed specifically to find and return a record from the "ONEARRAY" buffer to the program area.

It cannot be used as an independent utility and will only work in association with the "ONEARRAY" utility.

It is submitted as a separate utility because it will not be required for all applications.

SET UP OF THE ROUTINE

"XSEARCH" is designed to be installed at the end of a Basic program, and must be installed immediately before "ONEARRAY".

It calls some of the routines contained in "ONEARRAY", and uses the parameters which are set up during the initialisation of "ONEARRAY". Hence it must be held in memory at the correct relative position with respect to that other utility.

The routine has a single entry point for execution, at M1+51 where M1=M-144 and M is the starting location for "ONEARRAY".

SEARCH OPTIONS AVAILABLE

"XSEARCH" provides only two options for searching the buffer:

1. Search for an exact match, ie that A$=A$(X), where A$ is the search key and A$(X) is the field in the record being searched, and
2. Search each full string for the search key.

Compared with "ASEARCH", this is the equivalent of using parameters P(4)=1 and P(5)=255, ie it searches the full string.

A switch in the working storage area for "ONEARRAY" is used to select the option required. This switch is at DP+9, where DP is as described for working storage for "ONEARRAY".

If DP+9=0 then search option 1 is performed. If DP+9 is not equal to zero, then option 2 is done.

Before calling the routine, you should poke the switch value required into DP+9. This location is used by "ONEARRAY" as an error switch, and would normally be zero.

The parameter used by "ONEARRAY" for 'this record number' (ie P(0) in the parameter array or Q) is used to specify the record number where the search will start, ie it performs the same function as P(1) for "ASORT".

The search then operates from the starting record number through to the end of the buffer, ie the equivalent parameter for P(2) with ASORT is always the last record in the buffer.

When a match is found, the record containing the field matched is returned to the program area, and the record number is returned in the variable specified in P(0) (eg, Q).

Compared with "ASEARCH", when a match is found, "XSEARCH" performs the equivalent of:

```
Q=P(7):EXEC M+205
```

To search for the next occurance of the search key, set Q=Q+1 and execute the search routine again.

If a match is not found, Q will equal zero, and the first record in the buffer (record #1) will be returned to the program area.

PARAMETERS USED

The parameters required for "XSEARCH" are set up as part of the initialisation routine for "ONEARRAY". These are:

P(0)= variable for this record number, which is used to specify the record number where the search is to start.

After the search, this variable contains the record number containing the match.

If a match is not found, this variable will be zero.

P(2)= variable for this field number. This contains the field number to be searched, eg to search on field #2 set B=2.

P(9)= VARPTR of the search string to be used, eg VARPTR(Q$) where Q$ will contain the string to be searched for.

This parameter is normally used with "ONEARRAY" to specify the start of the pointer array, but as "XSEARCH" is used to avoid the requirement for a pointer array, this parameter is not required for "ONEARRAY", so it is used to specify the search key instead.

If for some reason you want to use "XSEARCH" with a pointer array as well, then you can alter the parameter used to specify the variable for the search key by poking a different value into location M1+54. The value in this location specifies the offset from the start of working storage that contains this parameter.

As submitted, "XSEARCH" contains 98 (decimal). To alter this to provide the parameter in P(8), POKE M1+54,96

AN ALTERNATIVE SORT FOR ONEARRAY

For some applications, you might not have enough memory in the program area to provide the arrays required for "ASORT", but still want to sort the records in the buffer.

"XSORT" is a ML sort routine which has been designed specifically to sort the records on the "ONEARRAY" buffer. It cannot be used as an independent utility and will only work in association with the "ONEARRAY" utility.

It is submitted as a separate utility because it will not be required for all applications.

SET UP OF ROUTINE

"XSORT" is designed to be installed at the end of a Basic program, and it must be installed immediately after "ONEARRAY".

It calls some of the routines contained in "ONEARRAY" and uses the parameters which are set up during the initialisation of "ONEARRAY". Hence it must be held in memory at the correct relative position with "ONEARRAY".

The routine has a single entry point for execution which is the same as described for using "ASORT" with "ONEARRAY", ie at M+521, where M is the starting position for "ONEARRAY".

"ONEARRAY" is set up to be used with "ASORT", and to modify it to work with "XSORT", you have to POKE M+526,14, to change to offset required in a LBSR instruction.

To perform a sort, set the field number required and EXEC M+521.

PARAMETERS REQUIRED

The parameters required for "XSORT" are set up as part of the initialisation routine for "ONEARRAY". These are:

P(1)=number of records in the buffer

P(2)=variable for this field number, which contains the field number to be used for the sort, eg to sort of field #2, set B=2.

SORT ALGORITHM

The sort algorithm used for "XSORT" is a replacement sort, and the records are actually moved in the buffer so that they are in sorted sequence.

1. The sort starts with a counter N, which equals the number of records in the buffer.
2. It searches all of these N records to find the lowest one.
3. At the end of the loop, it copies the selected record to the end of the buffer, and deletes the original copy from its original position in the buffer.
4. It then reduces N to N=N-1 and repeats the operations from 2.
5. when N=0 it has completed the sort, and the records are in sorted sequence in the buffer.

For the sort to work there must be sufficient free space in the buffer to hold a maximum length record.

SORT OPTIONS

The only sort options directly available is to sort in either ascending or descending sequence, and this option is selected by a series of POKEs to the ML routine.

These POKEs alter the condition tests performed. The routine submitted is set up for an ascending sort.

The addresses and values to be poked to select each option are as follows (all numbers are decimal):

Address	Ascending	Descending
M2+48	11	10
M2+52	10	11
M2+58	10	11
M2+71	37	34

... where M2 is the starting address for "XSORT", which is the end address (as described for installing multiple ML routines at the end of a Basic program) minus 148, eg M2 = PEEK(27)*256+PEEK(28)- 148, if "XSORT" is the last ML routine installed.

OTHER SORT OPTIONS

There are a number of procedures available which allow you to do an indirect and multi level sort of a file using "XSORT", and these can be extended to sorting a larger file on disk as well.

For example, the following outlines a procedure for a multi level sort of a file which will fit in memory, where the order of the records in the disk file are to be altered. The file is to be sorted as field 4 within field 2.

Load the file and sort it on field 2 as described, and then write the file out to disk again in sorted sequence.

To do the next level sort, reset the buffer to empty.

Start from the begining of the file and load the records in sequence that have the same value in field 2.

When field 2 changes, sort the records in the buffer on field 4, and then write them back to disk again, into their original positions.

Continue through the file, repeating this operation. At the end of it, the file will be sorted into the sequence required.

Sample code for the second level sort is as follows, ie after the first sorted file is written back to the disk.

Reset the buffer to empty:

```
A=1:B=1:GET #1,1
 'input record into B$(NF)
A$=B$(2)
FOR X = 2 TO N
GET #1,X
 'input file into B$(NF)
IF A$=B$(2)THEN B=B+1
:EXEC M+205:GOTO 200
IF B=1 THEN 100
EXEC M9 'sort records in buffer
FOR Y=1 TO B
Q=Y:EXEC M+105
 'get record
 'print B$(NF) to disk file
PUT #1,A+Y-1
NEXT Y

   line 100 follows ..

'reset buffer to empty
A=X:B=1:A$=B$(2)

   line 200 follows ...

NEXT X
```

Other sorts are possible, but a description of the procedures required become rather involved and wont be attempted here.

The main requirement is to add a record number to each record as it is added to the buffer, and then use this record number after the sort to track the sort operations.

BASIC CODE REQUIRED FOR XSORT AND XSEARCH

"INDXCARD" is a Basic program which demonstrates the coding required to use the "ONEARRAY" utility in association with "ASORT" and "ASEARCH".

"XCARD" is another Basic program which performs similar functions using "XSORT" and "XSEARCH". Most of the code for these two programs is the same, and LISTING 2 contains the modules for "XCARD" which are different to "INDXCARD".

These programs show the set up and calling sequences required to use the ML routines from a Basic program.

A GENERAL COMMENT

As noted, the "ONEARRAY" utility can be used for any application which accesses data in a two dimensional array, and this includes a wide range of different applications.

The buffer size can vary from 32K up to as high as 60K, depending on the size of the Basic program being used. In effect, the buffer size is simply whats left after the program requirements have been satisified.

Applications of this nature frequently have a requirement to sort and or search the data as well, and alternative ML routines for these functions have been provided.

The selection of which sort/search routines to use depends mainly on the application and the number of records involved.

The "ASORT/ASEARCH" routines provide a full range of options for these functions, but can become limiting when the number of records exceed 500-1500. This is because they require 10 bytes

Continued on P 65

Index Card

The framework for an index card system for disk and cassette.

by George McLintock

64K CoCo/CoCo3
UTILITY

INDXCARD IS A Basic program which provides the framework for an index card type of system for a 64K CoCo or CoCo 3. It is designed for a disk system, but can also be used with a cassette system as well.

It includes a facility to save and load the data file to tape as this is the safest way to back up such data. As a result it can be used directly for a cassette system without further modification.

As submitted, the program is set up to handle a simple list of names and addresses, with 11 fields per record, and a maximum number of 500 records in the file. It is also set up for a TP-10 printer, which limits the printed reports produced to 32 characters per line.

These aspects are very easy to change. The program has been designed so that it can be easily modified by anyone with a basic level of knowledge of Basic programming, ie it requires a knowledge of how to edit a program and to layout a printed report.

Various other changes can also be made fairly easily, but the ones listed above are sufficient to obtain a working index card type of system, with a good operational capability for sorting and searching the file etc.

It has also been designed to handle a large file and this is the main reason for not attempting any form of menu setup for file structure or report layout etc. While this could be done, the program code required to do it would occupy memory that could otherwise be used for data storage.

The program include a number of comments, and these are provided to assist with understanding the way the program operates, so that you can work out how to change it. If you remove these comments, you can obtain an additional .5K+ of memory for data storage.

It also includes 1.5K of free space to provide room for program modification without changing the buffer area. This can also be used for extra data storage by changing the buffer start address. Details on how to calculate space requirements are provided later.

PROGRAM CONTROL LOGIC

The program line numbers have been set to group the various functions. This outline of the logic describes the functions performed by each segment:

Lines 20-30: set machine parameters and performs a PCLEAR0.

Lines 40-70: set the parameters associated with file structure.

Lines 110-260: displays the main menu and branches to the code required to perform each function.

Lines 310-320: ensures valid field numbers.

Line 330-340: ensures valid record numbers.

Line 360-400: provides alternative ways of getting a character from the keyboard.

Lines 510-580: displays the contents of a record on the screen.

Lines 710-750: displays the titles given to each field in the record

Lines 900-930: inputs a record from disk.

Lines 1010-1030: outputs a record to disk

Lines 1110-1250: allows you to select a record from the buffer and display it on the screen ...
- also allows use of the arrow keys to step through the file in sorted or unsorted sequence.

Lines 1510-1680: accepts data input to create a new record and adds it to the buffer.

Lines 2010-2060: deletes a record from the buffer
- displays the record first and requires confirmation that it is the correct record to delete.

Lines 2110-2310: allows changes to an existing record.

Lines 2510-2680: allows you to search the file for a particular record.

Lines 2810-3080: sorts the records into any sequence required.

Lines 2810-2920: performs first level sort.

Lines 2940-3080: extends this to any number of multi level sorts.

Lines 3210-3430: prints the file
- is coded for three options with the TP-10 printer,
- contains its own separate menu for options available.

Lines 4010-4110: loads the file from disk and puts it in the buffer.

Lines 4210-4230: saves buffer parameters so that they can be restored at a later stage.

Lines 4310-4340: loads the file from tape.

Lines 4410-4430: saves buffer to file on tape.

Lines 4610-4700: saves buffer to file on disk.

Lines 4810-4860: provides a separate menu for input/output operations.

Lines 4880-4900: restores buffer pointers.

Lines 5010-5520: initialises various parameters required for the paogram

Line 5080: tests for a CoCo3 and sets the ML routines to suit the machine used.

All variables used in the program must be defined at the start of this section, preferably in the DIM commands.

The main functions performed here is to set up for and to initialise the machine language routines used in association with the Basic program.

Apart from assigning titles to the C$(NF) array (Lines 5100-5200) and adding variables to the DIM statements, the rest of this section would not normally be altered.

FILE STRUCTURE

The file structure is established by the values of parameters set at the begining of the program. These variables are then used by the rest of the program as required for various functions.

Parameters set are N$=name of file on disk and tape MR=maximum number of records in the file NF=number of fields in each record ML=maximum size of any record.

- ie number of characters in largest record MD=is calculated as ML+NF and is the maximum record size for the disk file.

FILE ON DISK

The file on disk is set up as a direct access file, but data input/output for each record is performed as for a serial file.

While this may seem a funny way of organising a file, it is described in the manual and it does work. In fact for files of this nature it has certain advantages over both the normal serial file and what I consider to be a normal direct access file.

It has the normal advantages of a direct access file in that each record can be accessed directly for both read and write operations, and records can be appended at the end of the file.

The main advantage with this file structure over a normal direct access file using the FIELD statement, is that the total record length can be set to a lower value.

If you use the FIELD command, then each field in the record has to be set to allow for the largest value for each field.

In general, the sum of the maximum size for each field will be greater than the maximum size for any record, eg it is most unlikely that the longest surname will occur with the longest street name and the longest suburb name etc all together in the same record.

And even if you get it wrong, it is quite easy to increase the maximum size of a record at a later stage.

The approach does require a specific delimeter in the file to separate each field. This program uses the carriage return as a delimeter for this purpose rather than use the WRITE command.

The WRITE command puts quotation marks around each string which requires two extra bytes on the disk for each field. This program uses the PRINT command for output, with a carriage return after each field, which only requires one extra byte on disk for each field.

To keep each field separate, the program uses the LINE INPUT command to read each one back again.

An incidental advantage of doing it this way is that the double quote (CHR$(34)) can be used as data without restriction.

OVERVIEW OF DATA STORAGE

The description with ONEARRAY provides an overview of how data is handled with reference to the ML routines and the buffer.

The Basic program can only access a single record at any one time, and this is contained in the array A$(NF). This is a normal Basic array and can be used by the Basic program normally.

The only restriction here is that if you want to set another string variable equal to a value in the A$(NF) array, you must use code like A1$=A$(X)+"".

If you use code like A1$=A$(X) then the string data in A$(X) is not copied to normal string space. All that happens is that the VARPTR of A1$ is set to the same value as A$(X). If you later change A$(X), then the contents of A1$ will also change to the same value.

This is because the actual string data for the A$(NF) array is not held in normal string space.

To get a record from the buffer, the Basic program sets Q equal to the record number required and EXEC's M+205. This puts the record in the array A$(NF).

The numeric array N(MR) contains the record numbers for the records after they have been sorted into a specific sequence, ie if the records are sorted by surname, then N(MR) will contain the sequence in which the records should be returned from the buffer to print the file in sequence by surname.

So that the code to do this would be along the lines of:

```
FOR X=1 TO N
Q=N(X):EXEC M+205
' print the array here
' using A$(0) to A$(NF) to
' refer to the array
NEXT X
```

If you want to get the records in the same sequence as they exist in the buffer, replace the statement Q=N(X) with Q=X.

As set up, INDXCARD does not allow the Basic program to access a full column from the buffer, although the ML routine used does allow this option.

You can, in fact, access the columns for the records which are stored in the buffer area which is located in the lower 32K of memory, but records stored in the upper 32K of memory will print as garbage.

CHANGING AN EXISTING RECORD

The procedure used to change an existing record may seem a little unusual, but is quite reasonable.

The general principle is that while the records are accessed by their position in the file or buffer, that position, by itself, need not have any particular significance.

The records can be identified by the data contained within them, rather than by their position. Hence it should not be essential that each record should always maintain the same unique position at all times.

Two alternative procedures are provided for ammending a record, and these are intended to satisify alternative approaches to file maintenance.

If you want each record to maintain a fixed position in the file, then you change the record on disk, and not in the buffer.

For this operation, the record is read from disk, altered, and written back to its original position on the disk again. This does not alter what might be in the buffer at that time.

Hence, if you then want to print the the file with those changes included, you have to re-load the file from disk before printing it.

If you identify each record by

its contents, then you change the record in the buffer, and write the complete file back to disk again when you have finished.

For this operation, the old copy of the record is deleted from the buffer, and the new ammended copy is added to the end of the buffer.

With this approach you search the file in memory to find the record required, according to its contents, and this then sets the position number required to return the whole record from the buffer.

You also tend to re-sort the file after altering it and write it out again. This will maintain the file on disk in a sorted sequence.

The particular procedure used depends mainly on the application and personal preference.

As submitted, INDXCARD is set up for the positional approach. This is because when writting the file back to disk, records are extracted from the buffer in a natural sequence.

It can be easily modified to suit the second approach by changing it so that records are extracted in a sorted sequence when writting the file back to tape of disk, ie change the Q=X command to Q=N(X) in lines 4420 and 4630.

NUMBER OF FIELDS

The number of fields in each record can be varied from one to 255.

The record display and column select module can display up to 12-14 fields without modification. But if you have more fields than this, then these modules will require some modification.

For a large number of fields, you can add lines 545 and 735 which contain code like ...

```
IFINT(X/12)*12=X GOSUB 370
```

... which will display 12 lines at a time and wait for a key press before continuing.

ASSIGNING FIELD TITLES

It is normal with programs of this nature to assign a title or name to each field. eg surname, suburb, etc.

These titles are then used for displaying a record and for selecting a field for a particular operation. The title can also be used when printing the lists, although this last use is not applied with this program.

The array C$(NF) is used for this purpose, and the titles are set in lines 5100-5200 of the program.

PRINTING A REPORT

The program is set up to print three different reports on a TP-10 printer (line width of 32 characters).

Printed reports tend to be quite specific to a particular application, so this module has to be modified to suit requirements.

SORTING AND SEARCHING THE FILE

These operations tend to be fairly standard across all applications, and should not require modification.

If you do want to change them, all the features of the ASORT and ASEARCH utilities are available and can be used.

BUFFER SIZE

The size of the buffer available for data storage is effectively the residual of what memory is left after providing for all other requirements.

The starting amount is 64K (ie, all addressable memory), which is then reduced by the following:

- Low memory used by Basic:
  3.5K for disk,
  1.5K for tape.
- Graphic screen memory.
- Memory occupied by the Basic program itself (INDXCARD uses 10.5K).
- Memory required for the ML routines (as set up uses 1.6K).
- Memory required for Basic variables and arrays, which are of two types.
- Variables and arrays used for the program in general, ie not specific to the buffer operations (for INDXCARD this is 0.3K).
- Arrays required for the interface between the buffer and the Basic program, which are A$(NF) to hold the VARPTRS of a single record (is NF*5 bytes).
- space required to hold the data for these strings is the same as maximum number of characters in the record (ML).

If accessing a column of the main array (in the buffer),
- T$(MR) to hold the VARPTR's (is MR*5 bytes)
- N(MR) to hold the numeric variables for pointers (is MR*5 bytes).

If doing multi-level sorts:
- space to hold the sort switches (is MR bytes).

If bringing array column data back to be accessed by the Basic program:
- space to hold the string data for the largest column.
- memory used for normal string space.

This must be large enough to hold the maximum length of a record (ML bytes) plus string space required for other variables in the program, plus a bit for other operations.

The balance of 64K can be used for the buffer. The mimimum size of the buffer is 32K, as the Basic program area cannot extend past the lower 32K of memory.

CHANGING THE SIZE OF THE BUFFER

The size of the buffer is set by specifying the start and end address addresses to be used.

The start address is set by a two byte integer value stored at M+51 and M+55, while the end address is at M+53.

To change either of these simply POKE the new address required into these locations.

If changing the start address of the buffer, the same value must be POKE'd into both M+51 and M+55.

MAXIMUM NUMBER OF RECORDS IN THE BUFFER

The maximum number of records that can be held in the buffer depends on the average size of all records. In general, the average size of a record will be considerably less than the maximum size of a record.

Memory required for each record is average length of record plus one byte for each field in the record plus two bytes.

WORKING STORAGE

The ML routines use the cassette buffer for working storage. However, when tape operations are required, the working storage area for

ONEARRAY is moved to the text screen for those operations, and then reset to the cassette buffer again.

This operation will destroy the parameters set by the initialisation routines for all the other ML's.

OTHER ASPECTS

When using this ML routine, you can sometimes gain advantage from the fact that the data in the buffer is completely separate from and independent of the Basic program which accesses it.

The data will remain in memory until overwritten or the machine is switched off. The two critical parameters required to maintain this independence are the number of records in the buffer, and the address of the last data element in the buffer.

To allow this data to be accessed by different programs, these parameters are POKE'd into memory locations Hex 3FC to Hex 3FF by INDXCARD when the file is loaded, so that the buffer pointers can be restored by another program which can then access the data normally.

In this context, if you have loaded the data file, and press break or reset, and then RUN the program again, you are effectively running a different program because all variables are reset by the RUN command.

The options provided by the input/output menu to reset and restore the buffer pointers are designed to make use of this independence of the data in the buffer.

As an example, it is quite easy to change the maximum size of the records at any time, eg if you are entering a new record which is too large, and you decide to increase the maximum record size to accommodate it.

To do this:

1. Set the buffer pointers
2. Press break or reset
3. Edit line 70 to new size
4. Run the program again.
5. Restore the buffer pointers
6. Continue data entry
7. The next time you save the buffer file to disk it will be the new record length.

If you do this sort of thing then you have to be a little more careful about how you use your backups on disk. The older copies of your file will then have a different record length to the new one.

It makes no difference to tape files because they are normal serial files and are not affected.

The operations performed by these options are:

Set buffer pointers - pokes values of the two parameters into memory.

Restore buffer pointers - restores the value of N and the contents of DP+10 & 11 from the values previously poked by the set buffer option.

Reset buffer - re-initialises the buffer, ie calls the initialisation part of the ONEARRAY ML. Buffer is reset to initial value ie empty.

Use the reset buffer option before re-loading the file. If you dont a second set of records will be added to the end of the existing data in the buffer.

SETTING UP THE PROGRAM

The Basic program requires four ML routines to be included at the end of the program.

The ML routines are set up in the following sequence:

* ONEARRAY
* ASORT
* ASEARCH
* PCLEAR0

Instructions for adding these ML routines to the end of the Basic program are included with the description of those routines.

CONVERTING OTHER DATA FILES TO SUIT

The main advantage with this approach to an index card type of system is that you can hold a lot more records in memory.

If you have an existing system which has grown to fill normal memory, then it is quite easy to change it over to suit this approach without rekeying your data.

Either change the input/output routines to suit your existing file, or write a separate Basic program to read each record from the existing file and write it out to a new file to suit the code here.

USING THE PROGRAM WITH A COCO 3

INDXCARD is set up to run on either a CoCo 3 or an old CoCo. Line 5080 makes the necessary changes to the ML routines to suit the particular machine.

The Listing:

```
1 '**  ONEARRAY  (FOR INDXCARD)
       BY GEORGE MCLINTOCK
       NOV 87
2 GOTO 50000
3 SAVE"116:3":END'7
4 'A GENERAL PURPOSE ML ROUTINE
TO USE THE UPPER 32 K OF COCO
5 'SET UP TO ADD ML AT END OF AN
OTHER BASIC PROGRAM - TO USE  ME
RGE AT END ANOTHER PROGRAM AND
RUN 53000
6 ' ENTRY POINTS - SORT M+219
 MOVE M+0  NULL M+119
7 'INITIAISE ONLY M+219  EXEC MO
VE M+6  EXEC SORT M+223
8 'TO PUT IN FIXED AREA OF MEMOR
Y - DELETE LINES 50110-50150  -
REPACE WITH CLEAR 200,32000:A=32
000
9 '
50000 LN=50500:FOR X=0 TO 538 ST
EP 25:IF X<524 THEN N=25 ELSE N=
13
50010 GOSUB 50030:NEXT X
50020 RESTORE:GOTO 50110
50030 PRINT LN;:A=0:FOR Y=0 TO N
-1
50040 READ C$:B=VAL("&H"+C$):A=A
+B
50050 NEXT Y:READ C$:IF A<> VAL(
"&H"+C$) THEN PRINT "ERROR IN LI
NE NO";LN:STOP
50060 LN=LN+10:RETURN
50070 '
50080 FOR Y= 0 TO N-1:READ C$:PO
KE A,VAL("&H"+C$)
50090 A=A+1:NEXT Y:READ C$:RETUR
N
50100 '
50110 M$="9E1B3089021A6F806F806F
809F1B39":Y=&H01DA
50120 B=0:FOR X=1 TO 30 STEP 2:N
=VAL("&H"+MID$(M$,X,2)):B=B+N:PO
KE Y,N:Y=Y+1:NEXT X
50130 IF B <> &H54E THEN PRINT "
ERROR IN LINE NO 50110":STOP
50140 EXEC &H1DA:CLEAR
50150 A=PEEK(27)*256+PEEK(28)-54
1:LN=50500
50160 FOR X=0 TO 538 STEP 25:IF
X<524 THEN N=25 ELSE N=13
50170 GOSUB 50080:NEXT X
50180 '
50190 PRINT "ONEARRAY ML NOW ADD
ED TO END OF BASIC PROGRAM": PRI
NT "AND EXTRA BASIC CODE DELETED
```

Continued on P 65

ROM packs from disk

Say goodbye to that multi pack!

by George McLintock

CoCo3
UTILITY

ROM PACKS AND disk drives don't mix very well unless you have a multi pack interface permanently attached.

This problem was solved for the old 64K CoCo by an article in Hot CoCo, Sept 1983, by Richard Esposito and Ralph Ramhoff which allows you to save your ROM packs to disk and then load and execute them from disk there after.

Unfortunately, the procedure used for the old CoCo won't work with the CoCo 3. When I got the CoCo 3 I kept the old CoCo and hence have been able to continue to use my ROM packs from disk on that machine.

At different times I have tried various procedures to get the ROM packs to execute from disk on the CoCo 3, but without success until recently when I tried what now seems to be the obvious solution.

The procedure which will work is to copy the old Color Basic ROM to the CoCo 3 as well as the ROM pack, and execute them both together.

The CoCo 3 effectively operates in 'all RAM' mode anyway, so that if you load the old CoCo ROM into the upper 32K of the CoCo 3, it will operate as if it were an old CoCo, with whatever old CoCo ROM you are using.

This submission includes a small ML routine to replace the ROMFIX routine from Hot CoCo, and a description of how to convert the ROMPACK disk files for the old CoCo to suit the CoCo 3.

The ML routine is set up to suit the old CoCo as well, so that the same programs can be executed in both the CoCo 3 and the old CoCo as well. This avoids the requirement to have separate copies of the same program for the different machines.

The ML routine is submitted as a Basic program, called 3ROMFIX, to be poked into memory.

The assembler source code is also provided.

To convert existing ROMPACK programs on disk for the old CoCo, you require two other ML routines on disk.

The first program required is a copy of the old Color Basic ROM. To get this you need an old CoCo and do the following:

```
SAVEM"DOS",&HA000,&HBFFF,&HA000
LOADM"DOS",&H9000
SAVEM"DOS",&H3000,&H4FFF,&H3000
```

The second program required is the one submitted. To get this as a ML routine on disk ...

```
RUN"3ROMFIX"
SAVEM"3RFIX",&H2FA8,&H2FFF,&H2FA8
```

To convert an existing ROMPACK program on disk, do the following:

```
LOADM"ROMPACK",&H1000
LOADM"DOS"
LOADM"3RFIX"
SAVEM"RPACK",&H2FA8,&H6FFF,&H2FA8
```

The routine, RPACK can now be executed on a CoCo 3 (or an old CoCo) by typing:

```
LOADM"RPACK"
EXEC
```

If you have the ROMPASCK code on tape, produced by:

```
CSAVEM"RPACK",&HC000,&HDFFF,&HC000
```

... then you could build the total program required on disk with ...

```
CLOADM"RPACK",&H9000
```

... to have it load into the correct memory location. Follow this with ...

```
LOADM"DOS"
LOADM"3RFIX"
```

... and then SAVEM as before.

Apart from producing a copy of the Colour Basic ROM, the remaining procedures can be done on a CoCo 3.

If you want to start with the ROMPACK, then you need an old CoCo to be able to save the ML code on tape (or copy both Coliur and Extended Basic from an old CoCo to the CoCo 3 first).

To prevent the ROMPACK auto start, cover the cartridge select pin (pin eight) with a piece of sticky tape before plugging it into the CoCo.

PROGRAM SIZE

The instructions given here assume that the ROMPACK program is less than 8K. All of mine are, and I don't know of any that exceed this (except EDTASM+). Some are as small as 2K, but it is normally not worth the trouble to find the end of the program.

Each program on disk will occupy 9 extends (with an 8K ROM), and you can fit 8 programs onto a 35 track disk.

You might fit one or two more on a disk if you make the effort to find the end of each program, and adjust the end address for the SAVEM"RPACK" instruction to suit.

If you simply reduce the end address, then it is not necessary to alter the ML code at all - it will still move the full 8K of memory, but this doesn't matter.

The test for the end of the ROMPACK program, included in the article in Hot CoCo, did not test for the 'no memory' pattern of 0's and 255's. So it would not find an end for the

ROMPACK's.

If you saved these as the full 16K (as I did), then you can reduce them to 8K by typing ...

```
LOADM"RPACK"
SAVEM"RPACK",&H4000,&H5FFF,&H4000
```

... and then proceed as described above.

The instructions provided here are designed to have the ROMPACK program load into 'unused' memory with the default (power up) settings of PCLEAR and CLEAR.

The ML routine, 3RFIX, occupies locations Hex 2FA8 to Hex 4FFF, and the RPACK program occupies Hex 5000 to Hex 6FFF.

The ML routine operates by first moving the Colour Basic ROM to locations Hex A000 to Hex BFFF. It then patches the ROM as per the article in Hot CoCo, and then initializes the Colour Basic ROM. After the initialization, it moves the ROMPACK code to Hex C000 upwards and then executes Hex C000.

You can use the same procedure for a ROMPACK of 12K, but you'll have to CLEAR 200,&H2FA7 before loading it, and ...

```
POKE &H2FF7,&H60
```

This allows the ROMPACK program to occupy memory from hex 5000 to hex 7FFF, and to be moved to hex C000 to hex EFFF.

To move larger amounts of code to the upper 32K of the CoCo 3, you have to change the starting location to a lower address in the lower 32K, and alter the memory locations moved. If you want to use the old CoCo ROM's in the CoCo 3, you also have to set the memory size in the initizlization routines, as done in the patches used here.

The EDTASM+ ROMPACK is larger than 8K, but the procedure described by Roger Schrag in Australian Rainbow, December 1982, will work on the CoCo 3 without any modification, and provides a better way of EDTASM from disk.

The enhanced version by Roger Schrag in Australian Rainbow December 1983, is not likely to work on the CoCo 3. I haven't actually tried it. To get this one to work, you would probably have to alter the bits that change the memory map.

I converted my EDTASM+ to disk using the December 1982 version, and never got round to trying the enhanced version.

The Listing:

```
1 '  ***  3ROMFIX  ***
     BY GEORGE MCLINTOCK
2 GOTO 10
3 SAVE"116B:3":END'7
4 ' ML ROUTINE TO USE ROMPACKS F
ROM DISK WITH COCO 3
5 ' FOLLOWS PROCEDURE DESCRIBED
IN HOT COCO SEPT 83 FOR OLD 64K
COCO
6 ' REQUIRES A COPY OF OLD COCO
COLOR BASIC ROM AS WELL
7 '
10 FOR X=&H2FA8 TO &H2FFF
20 READ A$: B=VAL("&H"+A$)
30 POKE X,B: A=A+B
40 NEXT X
50 IF A <> 12328 THEN PRINT "ERR
OR IN DATA STATEMENTS":STOP
60 PRINT "3ROMFIX NOW IN MEMORY
FROM        HEX 2FA8 TO HEX 2FFF"
100 DATA 1A,50,B7,FF,DF,8E,30,00
,CE,A0,00,A6,80,A7,C0,8C,50,00,2
6,F7,86,7E,B7,A0,51,CC,A0,72,FD,
A0,52,CC,8E,9F,FD,A0,84,CC,FE,7E
,FD,A0,86,CC,A0,93,FD,A0,88,7F,8
0,00,0F,71,86,7E,B7,A0,CB,CC,2F,
EC,FD,A0,CC,7E,A0,27
110 DATA CE,C0,00,8E,50,00,A6,80
,A7,C0,8C,70,00,26,F7,1C,AF,7E,C
0,00
```

The Listing:

```
                 00100 *CALLED 3ROMFIX
2FA8             00110          ORG     $2FA8
2FA8 1A    50    00120 START    ORCC    #$50      MASK INTERRUPTS
2FAA B7    FFDF  00130          STA     $FFDF     FOR OLD COCO AS WELL
2FAD 8E    3000  00140          LDX     #$3000    MOVE COLOR BASIC
2FB0 CE    A000  00150          LDU     #$A000    TO CORRECT LOCATION
2FB3 A6    80    00160 S1       LDA     ,X+
2FB5 A7    C0    00170          STA     ,U+
2FB7 8C    5000  00180          CMPX    #$5000
2FBA 26    F7    00190          BNE     S1
2FBC 86    7E    00200 FIX      LDA     #$7E      PATCH COLOR BASIC
2FBE B7    A051  00210          STA     $A051     ROM AS PER ARTICLE
2FC1 CC    A072  00220          LDD     #$A072    IN HOT COCO
2FC4 FD    A052  00230          STD     $A052
2FC7 CC    8E9F  00240          LDD     #$8E9F    SET MEMORY SIZE
2FCA FD    A084  00250          STD     $A084     WITH LDX INSTRUCTION
2FCD CC    FE7E  00260          LDD     #$FE7E
2FD0 FD    A086  00270          STD     $A086
2FD3 CC    A093  00280          LDD     #$A093
2FD6 FD    A088  00290          STD     $A088
2FD9 7F    8000  00300          CLR     $8000
2FDC 0F    71    00310          CLR     $71
2FDE 86    7E    00320          LDA     #$7E
2FE0 B7    A0CB  00330          STA     $A0CB
```

Continued overleaf...

Graph Maker

Here's a neat way to save your, or your parents, hard earned cash.

32K ECB CoCo
APPLICATION

by Roger Gosney

GRAPH-MAKER PRINTS out a graph onto the DMP110 sideways equal to nine passes of the print head in the finished picture. This will give a picture about 22mm high.

The graph paper is marked at each seventh row, making it simpler to see each pass when planning the picture.

Your picture is then plotted onto the graph marking which print pins are required. After the picture is drawn onto the graph, each of the combinations of print pins are worked out.

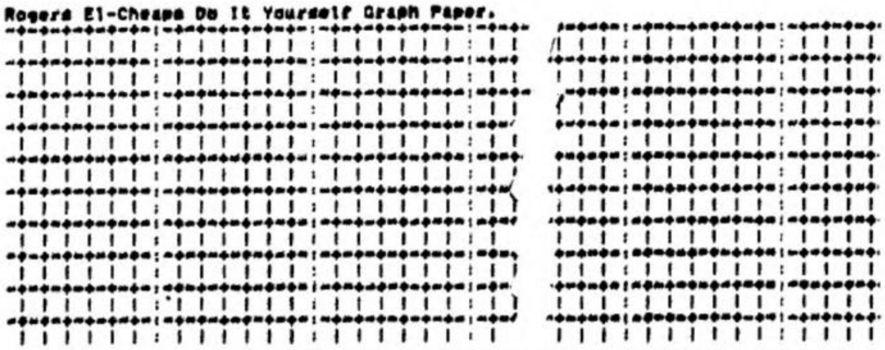
For example, if you needed a solid block such as a building, with a few windows lit you would use the combinations 127, 85, 85, 127, 87, 95, 95, 87.

This tells the DMP110 to print a solid line (127), then each second dot from the first one (85) etc. Each pin can be considered to have a number ranging from 1 at the top to 64 at the bottom. This is not strictly correct, but the program adds the other number required to get the correct value.

See "City Skyline" for an example of this application.

The Listing:

```
0 GOTO10
1 '+---------------------------+
2 '!SMALL GRAPH PAPER PRINTER  !
3 '!SUITABLE FOR THE DMP110 AND!
4 '!PERHAPS OTHER PRINTERS     !
5 '!**BY ROGER GOSNEY**        !
6 '+---------------------------+
7 SAVE"120:3":END'8
10 CLS
20 INPUT"HOW MANY LINES DO YOU R
EQUIRE  THE DMP110 PRINTS 960 A
CROSS A  PAGE. ";A
25 CLS
90 PRINT#-2,CHR$(27)CHR$(28)
100 PRINT#-2,CHR$(27)CHR$(77)
102 PRINT#-2,"Rogers El-Cheapo D
o It Yourself Graph Paper."
105 FORX=1TOA
106 CLS
107 PRINT@ 325,"NOW PRINTING LIN
E "X
110 PRINT#-2,"-+-+-+-+-+-+-:-+-+
-+-+-+-+-:-+-+-+-+-+-+-:-+-+-+-+
-+-+-:-+-+-+-+-+-+-:-+-+-+-+-+-+
-:-+-+-+-+-+-+-:-+-+-+-+-+-+-:-+
-+-+-+-+-+-:"
120 PRINT#-2," ! ! ! ! ! ! : ! !
 ! ! ! ! : ! ! ! ! ! ! : ! ! ! !
 ! ! : ! ! ! ! ! ! : ! ! ! ! ! !
 : ! ! ! ! ! ! : ! ! ! ! ! ! : !
 ! ! ! ! ! :
125 NEXTX
```



Rogers El-Cheapo Do It Yourself Graph Paper.

Continued from previous page

```
        2FE3 CC    2FEC     00340              LDD      #ROMBAK
        2FE6 FD    A0CC     00350              STD      $A0CC
        2FE9 7E    A027     00360              JMP      $A027
                            00370 *AFTER INITIALISATION
        2FEC CE    C000     00380 ROMBAK       LDU      #$C000    MOVE ROMPACK CODE
        2FEF 8E    5000     00390              LDX      #$5000
        2FF2 A6    80       00400 S2           LDA      ,X+
        2FF4 A7    C0       00410              STA      ,U+
        2FF6 8C    7000     00420              CMPX     #$7000
        2FF9 26    F7       00430              BNE      S2
        2FFB 1C    AF       00440              ANDCC    #$AF
        2FFD 7E    C000     00450              JMP      $C000
                   7000     00460 ZZEND        EQU      *
                   2FA8     00470              END      START

        00000 TOTAL ERRORS
```

City Skyline

New York! New York, in silhouette.

32K CoCo + DMP 110
APPLICATION

by Roger Gosney

SKYLINE IS A program to draw the New York skyline in silhouette. It was developed using a program called "Graph". The numbers obtained were put into data statements to be read by the body of the program.

The only drawback to this type program is the large number of points which must be coded.

For a full width picture on the DMP-110 with a height of seven passes requires the plotting of 60480 (give or take a few) separate points. However, don't let this discourage you - a lot of the lines required are solid lines, ie CHR$(127)'s, and the picture can be developed in sections, simply adding data lines at a later date.

The city skyline uses one section several times, to increase the building height.

The program is structured to read negative numbers and use them for multiple printing of pin combinations.

This is my first attempt at a project of this type, using the bit image capabilities of the DMP110. The program would probably run on other Tandy DMP's with little or no modification.

Once you get the hang of this type of program, the results are limited only by your imagination - it can be used for logos, letter heads, signatures or even a program to print "play money" for the kids, so they won't keep pinching the monopoly money. (The play money program may be the subject of a further article.)

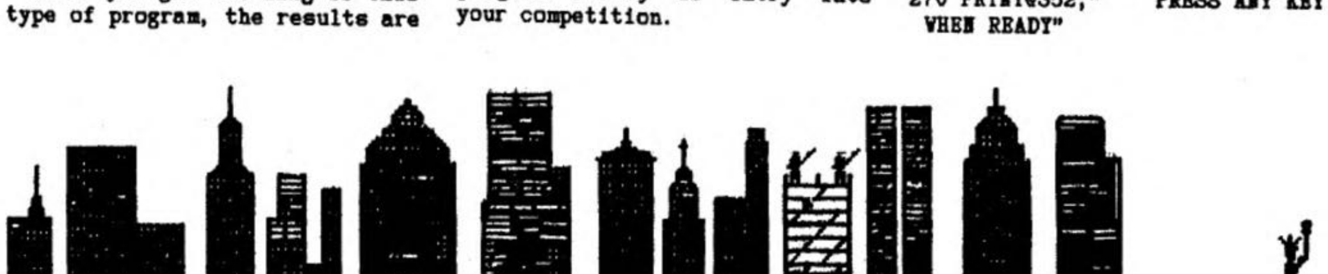
Details of program:

Line 8 - 14 - plays theme song
Line 15 - 31 - credits
Line 32 - 34 - printer on prompt
Line 36 - 38 - subroutine for credits
Line 41 - puts DMP110 into graphics mode
Line 42 - locates print head to left hand margin
Line 43 - reads data, if data =999 then goes to next pass of the print head.
Line 44 - prints data if >= to 0
Line 45 - 46 - checks for negative numbers and if found, prints negative number shown times the next positive data item.
Line 48 - 49 - data for first pass of print head
Line 50 - start of second pass
Line 208 - returns printer to last print mode before running program
Line 209 - 344 - fancy finish

I hope that you consider this program worthy of entry into your competition.

The Listing:

```
0 GOTO10
1 '+---------------------------+
2 '! BIG CITY SKYLINE          !
3 '! A PROGRAM FOR THE DMP110 !
4 '! PRINTER ....              !
5 '! BY ROGER GOSNEY MARCH 87 !
6 '+---------------------------+
7 SAVE"120:3":END'8
10 CLS(0)
20 V$(1)="T3O3L4GP255L4.GL5EP255
L4EL4GP255L4.GL5DL4D"
30 V$(2)="L4O3GL4.O4EP255L5EL4DC
L4.CO3L4BBO4CDO3BAGO4L2.CP6
40 V$(3)="T3L4CL4.CO3L5AL4AO4CL4
.CL5O3GL4GGAO4CO3GO4DL2.C"
50 PLAY V$(1)+"EFGABL2.GP6"
60 PLAY V$(1)+"O4L4DC#DEO3AL2.O4
D"
70 PLAY V$(2)
80 PLAY V$(3)
90 CLS(0)
100 T$=CHR$(143):G$=""
110 FOR J=1TO22:G$=G$+T$:NEXT
120 FOR J=69TO389 STEP32
130 PRINT@J,G$;:NEXT
140 PRINT@202,"CITY SKYLINE";
150 PRINT@239,"BY";
160 PRINT@266,"ROGER GOSNEY";
170 FORK=1 TO8:FORJ=36TO59
180 GOSUB300:NEXT
190 FORJ=91TO443STEP32
200 GOSUB300:NEXT
210 FORJ=442TO420STEP-1
220 GOSUB300:NEXT
230 FORJ=388TO6STEP-32
240 GOSUB300:NEXT:NEXT
250 FORJ=1TO2000:NEXT
260 CLS(7):PRINT@320,"PLEASE TUR
N YOUR PRINTER ON NOW"
270 PRINT@352,"     PRESS ANY KEY
 WHEN READY"
```



```
280 EXEC44539
290 GOTO330
300 C=C+1:IF C=8 THEN C=0
310 PRINT@J,CHR$(143+16*C);
320 RETURN
330 CLS(8):PRINT@96,"    PRINTING
 NOW IN PROGRESS"
340 ' first line
350 PRINT#-2,CHR$(18);
360 PRINT#-2,CHR$(27);CHR$(16);C
HR$(0);CHR$(1);
370 READ N: IF N=999 THEN 440
380 IF N>=0 THEN PRINT#-2,CHR$(1
28+N);:PRINT#-2,CHR$(128+N);:GOT
O370
390 READ M
400 PRINT#-2,CHR$(28);CHR$(-N);C
HR$(128+M);:PRINT#-2,CHR$(28);CH
R$(-N);CHR$(128+M);
410 GOTO 370
420 DATA -70,0,127,-54,0,-2,112,
-24,0,124,-7,84,124,124,-9,84,12
4
430 DATA -99,0,-9,64,-2,0,-9,64,
-18,0,-2,64,127,-2,64,999
440 PRINT#-2
450 PRINT#-2,CHR$(27);CHR$(16);C
HR$(0);CHR$(1);
460 READN: IF N=999 THEN 530
470 IF N>=0 THEN PRINT#-2,CHR$(N
+128);:PRINT#-2,CHR$(N+128);:GOT
O460
480 READ M
490 PRINT#-2,CHR$(28);CHR$(-N);C
HR$(128+M);:PRINT#-2,CHR$(28);CH
R$(-N);CHR$(128+M);:GOTO460
500 DATA -67,0,64,96,112,127,112
,96,64,-46,0,64,120,124,118,95,1
17,85,127,86,124,56,64,-19,0,127
,-2,42,58,58
510 DATA -3,62,126,127,110,111,1
10,110,-5,126,127
520 DATA -99,0,127,42,106,62,42,
-2,43,42,127,-2,0,127,42,-2,46,1
11,46,62,46,127,-14,0,64,96,112,
88,127,119,125,87,127,88,112,96,
64,-12,0,112,-5,40,-8,104,112,99
9
530 '3rd line
540 PRINT#-2
550 PRINT#-2,CHR$(27);CHR$(16);C
HR$(0);CHR$(1);
560 READN:IF N=999 THEN 640
570 IF N>=0 THEN PRINT#-2,CHR$(N
+128);:PRINT#-2,CHR$(128+N);:GOT
O560
580 READ M
590 PRINT#-2,CHR$(28);CHR$(-N);C
HR$(128+M);:PRINT#-2,CHR$(28);CH
R$(-N);CHR$(128+M);:GOTO560
600 DATA -19,0,-22,64,-26,0,127,
47,127,123,127,42,127,-39,0,64,9
6,112,120,56,126,59,127,58,-3,12
7,63,127,63,127
610 DATA 63,-2,127,123,126,124,5
6,112,96,64,-12,0,127,-3,125,-5,
117,-2,119,-2,127,117,125,117,-2
,125,-2,127
620 DATA-20,0,-2,64,112,127,112,
-2,64,-13,0,32,112,32,-18,0,112,
95,117,95,117,95,112,-14,0,64,-1
3,0,64,-2,0,127,93,93,125,85,87,
-2,95
630 DATA127,-2,0,127,93,95,85,11
1,85,-2,87,127,-12,0,-2,96,-2,12
7,42,127,106,-2,127,111,-3,127,1
22,127,-2,96,-10,0,-15,127,999
640 ' 4th line
650 PRINT#-2
660 PRINT#-2,CHR$(27);CHR$(16);C
HR$(0);CHR$(1);
670 READN: IF N=999 THEN 760
680 IF N>=0 THEN PRINT#-2,CHR$(N
+128);:PRINT#-2,CHR$(N+128);:GOT
O670
690 READ M
700 PRINT#-2,CHR$(28);CHR$(-N);C
HR$(128+M);:PRINT#-2,CHR$(28);CH
R$(-N);CHR$(128+M);:GOTO670
710 DATA -9,0,96,-9,0,127,126,12
7,126,127,46,127,63,127,63,127,6
3,-4,127,123,127,123,-3,127,-24,
0,64,96
720 DATA 63,-4,127,117,127,96,64
,-35,0,-2,112,-4,127,63,-5,127,1
19,-15,127,-2,112,-10,0,127,-3,1
19,-5,87,95,-3,87,119,-5,87,127,
32,96,32,96,32,96
730 DATA-8,0,4,124,46,126,47,127
,126,127,111,127,47,127,47,127,1
11,126,110,124,4,-6,0,64,96,127,
96,64,-17,0,127,62,127,110,127,1
23
740 DATA127,-6,0,16,92,124,92,8,
4,2,1,-6,0,16,92,124,92,8,4,2,1,
-3,0,127,46,47,62,46,47,46,46,12
7,-2,0,127,106,122,110,123,106,1
10,106
750 DATA127,-10,0,-2,120,95,-2,1
25,127,125,127,93,127,93,127,117
,127,119,127,119,-2,127,-2,120,-
8,0,127,-11,119,-3,127,-3,124,4,
124,999
760 '5th line
770 PRINT#-2
780 PRINT#-2,CHR$(27);CHR$(16);C
HR$(0);CHR$(1);
790 READN: IF N=999 THEN 890
800 IF N>=0 THEN PRINT#-2,CHR$(N
+128);:PRINT#-2,CHR$(N+128);:GOT
O790
810 READ M
820 PRINT#-2,CHR$(28);CHR$(-N);C
HR$(128+M);:PRINT#-2,CHR$(28);CH
R$(-N);CHR$(128+M);:GOTO790
830 DATA-9,0,127,-9,0,127,125,12
7,125,127,119,127,95,-14,127,-22
,0,126,43,127,107,127,123,-3,127
,123,127,59,127,111
840 DATA126,-7,0,126,-7,42,126,-
5,0,96,32,-4,96,-6,0,-3,127,95,-
9,127,125,-2,127,125,127,-3,125
850 DATA127,125,-7,127,-10,0,127
,107,107,59,-2,43,47,59,54,107,1
27,-3,43,59,-2,43,123,43,111,125
,127,125,127,125,127
860 DATA-9,0,127,95,127,87,-3,12
7,119,-3,127,119,127,119,127,125
,127,-5,0,112,120,47,90,58,42,63
,56,112,-15,0,127,125,127,117,-3
,127,-5,0
870 DATA127,8,15,9,15,-5,8,79,40
,24,-3,8,15,9,15,8,127,-5,0,127,
117,119,117,119,93,103,83,127,-2
,0,127,-2,83,117,103,83,-2,117,1
27,-10,0
880 DATA-2,127,106,127,59,127,12
3,127,63,127,123,127,63,127,123,
-3,127,123,-2,127,-8,0,127,-3,87
,-5,119,127,-3,87,-3,95,-2,127,0
,127,999
890 '6th line
900 PRINT#-2
910 PRINT#-2,CHR$(27);CHR$(16);C
HR$(0);CHR$(1);
920 READN: IF N=999 THEN 1030
930 IF N>=0 THEN PRINT#-2,CHR$(N
+128);:PRINT#-2,CHR$(N+128);:GOT
O920
940 READ M
950 PRINT#-2,CHR$(28);CHR$(-N);C
HR$(128+M);:PRINT#-2,CHR$(28);CH
R$(-N);CHR$(128+M);:GOTO920
960 DATA -7,0,112,126,123,126,11
2,-7,0,127,107,123,47,127,62,127
,62,127,62,127,63,-2,127,123,127
,-3,126,-3,127
970 DATA-22,0,127,93,127,93,127,
-3,125,127,125,-5,127,-7,0,127,9
3,125,95,117,-2,119,87,127,-5,0,
-2,127,85,127,95,127
980 DATA-6,0,127,111,127,111,127
,110,63,123,-8,127,126,-5,127,12
6,-7,127,-8,0,-2,124,-7,55,127,1
19,-5,55,63,55,-6,119,127,-2,119
,127990 DATA-9,0,127,62,127,110,
127,126,127,110,127,111,127,111,
127,122,127,111,127,-5,0,127,117
,127,95,-3,127,95,127,-5,0
1000 DATA126,122,126,127,126,126
,106,-3,126,123,-2,127,58,127,62
,127,-5,0,127,-2,17,81,49,17,30,
22,20,-6,17,81,49,-3,17,127,-5,0
1010 DATA127,106,110,107,58,110,
-2,106,127,-2,0,127,-2,42,43,42,
43,42,106,127,-10,0,-4,127,95,12
7,95,127,119,127,125,127,125
1020 DATA127,125,127,125,127,125
,127,127,-8,0,127,-4,107,-4,123,
-9,127,0,127,999
1030 '7th line
1040 PRINT#-2
1050 PRINT#-2,CHR$(27);CHR$(16);
CHR$(0);CHR$(1);
1060 READN: IF N=999 THEN 1180
1070 IF N>=0 THEN PRINT#-2,CHR$(
N+128);:PRINT#-2,CHR$(N+128);:GO
TO1060
1080 READ M
1090 PRINT#-2,CHR$(28);CHR$(-N);
CHR$(128+M);:PRINT#-2,CHR$(28);C
HR$(-N);CHR$(128+M);:GOTO1060
1100 DATA127,107,-2,123,63,127,4
7,127,47,127,111,127,47,127,-5,0
,127,87,127,119,127,119,127,95,1
27,95,-4,127
1110 DATA125,127,127,125,127,125
,-2,127,124,84,124,116,124,116,-
3,124,60,124,60,124,52,124,-7,0,
127,126,127,63,127,62,127,110,12
7,111,127,122
1120 DATA127,111,127,-4,0,127,-2
,85,87,85,127,-3,85,87,85,127,-5
,0,-2,127,46,127,123,127,-6,0,-6
,127,125,-15,127,125,127,125,127
,95,127,95,127
```

Tandy COMPUTERS 1000 EX Offer!

Choose the 1000 EX, Pick a Monitor & Take Home the Savings!
SAVE UP TO $699
Low As $999
Reg Separate Items 1648.95
SUPER SAVER
TANDY
CLEARANCE!

Give your family a great surprise when you take advantage of this terrific offer and save up to $699 on the 1000 EX and monochrome or color monitor! The EX is MS-DOS compatible so it's ready to run the programs you bring home from the office or school. 256K of memory and built-in 13.3cm disk drive gives you plenty of power and storage. The EX will run your programs up to 50% faster than the IBM® PC! Built-in adapters let you expand quickly and easily. Includes Multi-function Personal DeskMate software so you can begin using your computer system the minute you get home.

Tandy 1000 Computer & VM-2 Mono Monitor. 25-1050/26-3211
Reg 1648.95 .... **Save $649.95** .... **Sale! 999.00**

Tandy 1000 EX Computer and the CM-5 RGBI Color Monitor. 33cm screen display. 25-1050/25-1023
Reg 1898.00 .... **Save $699** .... **Sale! 1199.00**

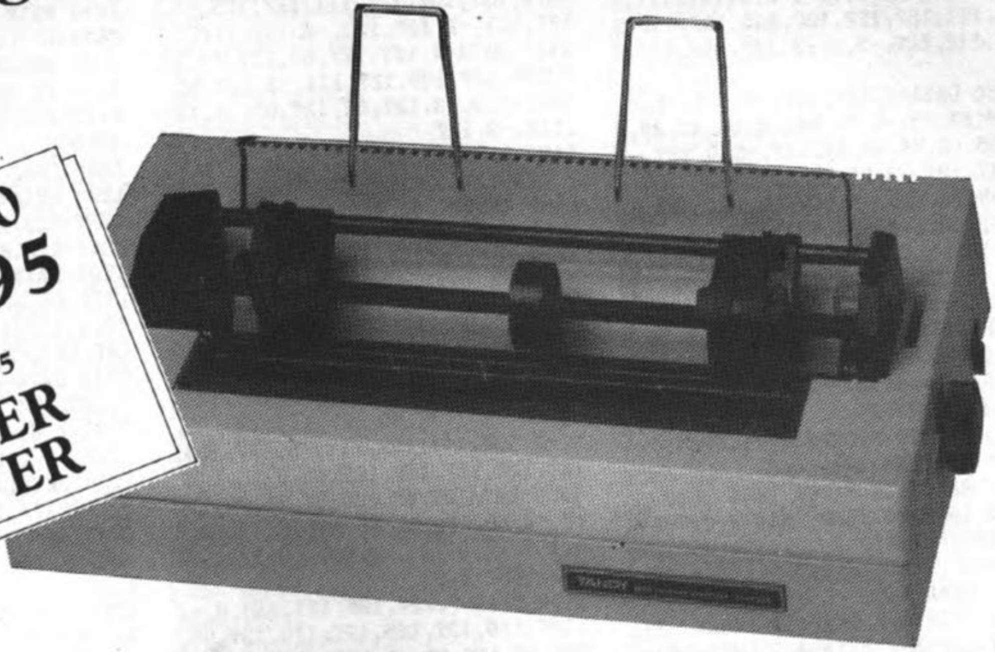
IBM® Registered Trademark of International Business Machines Corp.

Sorry no rainchecks.

Entertainment Software T1000/3000

F15 Strike Eagle. Challenging jet fighter program. 25-1125	79.95
Gunship. Helicopter, gunship simulation program. 25-1305	89.95
Winter Games. A selection of winter sports. 25-1195	69.95
Printshop. Graphics and designing. 25-1304	109.95
Black Cauldron. Entertaining program. 25-1133	79.95
Maestro. Music program. 25-1179	79.95
World Tour Golf. Play golf all around the world. 25-1191	79.95

High Performance Printer!



SAVE $50

349⁹⁵

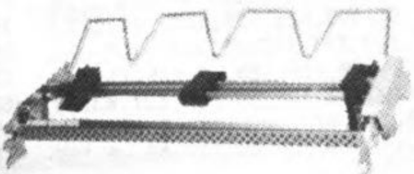
Reg 399.95

SUPER SAVER

Printer For All Your Needs!

- * Budget-Priced
- * IBM PC Compatible

If you've been searching for an efficient printer to match your computer system, then the Tandy DMP-106 is the ideal printer. Ideal for data processing and general purpose use. It's an efficient printer which prints 80 upper and lower case characters per line, at 43 lines per minute. Supports elongated, condensed modes and features bit-image graphics. Give your computer a cost efficient printer that produces professional results. Parallel and color computer compatible serial interfaces. 26-2802



DWP-520 Bi-Directional Tractor. This 'tractor feed' permits paper to feed into the printer from both directions. Designed for easy installation. 26-2801 .... 269.95

Clearance! Color Computer Software

Taxi. You drive through 6 cities. 26-2509 Reg 34.95 .. Sale! 7.95
Star Trap. Trap a star in the maze. 26-2510 Reg 34.95 Sale! 7.95
Grover Rover. Number game. 26-2522 Reg 39.95..... Sale! 7.95
Peanut Butter Panic. Make sandwiches. 26-2523
Reg 39.95.. Sale! 7.95
Ernie's Shapes. Match shapes and colors. 26-2524
Reg 39.95.. Sale! 7.95
Big Bird's Delivery. Picture matching. 26-2525
Reg 39.95.. Sale! 7.95
Cookie's Letter Crunch. Match letters and words. 26-2526
Reg 39.95.. Sale! 7.95
Grobot. Creative exploration game. 26-2527
Reg 34.95.. Sale! 7.95
Timebound. Historical exploration game. 26-2528
Reg 34.95.. Sale! 7.95
Vocabulary I. 26-2568 Reg 14.95........................ Sale! 5.95

Vocabulary II. 26-2569 Reg 14.95 .... Sale! 5.95
Color Logo Rom. 26-2722 Reg 89.95 .... Sale! 7.95
Number Patterns. 26-2752 Reg 2.95 .... Sale! 1.95
Color Cubes. Cube puzzle on your computer. 26-3075
Reg 39.95.. Sale! 7.95
Micropainter. Electronic colouring book. 26-3077
Reg 59.95.. Sale! 7.95
Reactoids. Nuclear game. 26-3092 Reg 29.95.......... Sale! 7.95
Color Stat. 26-3107 Reg 39.95.......................... Sale! 19.95
Hi-Res Screen Print. Graphics program. 26-3121
Reg 17.95.. Sale! 7.95
CC Edit Assembler. Machine language. 26-3250
Reg 79.95 .... Sale! 39.95
Investograph CoCo. Charts, stocks and trends. 26-3258
Reg 99.95 .... Sale! 49.95

Tandy ELECTRONICS

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

Over 360 Stores Australia-Wide

WE SERVICE WHAT WE SELL!

Available From 350 Stores Australiawide Including Tandy Computer Centres or Order On VIATEL *642614#

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

```
1130 DATA-8,0,127,-8,85,-2,95,-2
,127,-5,119,95,-8,93,127
1140 DATA-9,0,127,125,-3,127,125
,-3,127,95,-7,127,-3,0,127,111,1
27,111,127,122,127,126,127,110,1
27,111,127,-3,0,-3,127,117,127,1
26
1150 DATA-3,127,117,-7,127,-5,0,
127,34,35,-5,34,98,34,50,42,38,3
4,35,-3,34,98,34,127,-5,0,127,-3
,117,127,-2,95,93,127,-2,0,127,9
3,95,93,125,-2,119,87,127,-10,0
1160 DATA-6,127,63,127,110,127,1
23,127,111,127,111,127,111,127,1
11,-2,127,-8,0,127,-8,119,-2,127
,-8,87,127
1170 DATA-49,0,32,64,0,96,0,64,3
2,10,123,10,999
1180 '8th line
1190 PRINT#-2
1200 PRINT#-2,CHR$(27);CHR$(16);
CHR$(0);CHR$(1);
1210 READN:IF N=999 THEN 1330
1220 IF N>=0 THEN PRINT#-2,CHR$(
N+128);:PRINT#-2,CHR$(N+128);:GO
TO1210
1230 READ M
1240 PRINT#-2,CHR$(28);CHR$(-N);
CHR$(128+M);:PRINT#-2,CHR$(28);C
HR$(-N);CHR$(128+M);:GOTO1210
1250 DATA127,111,127,111,127,111
,127,63,125,127,125,127,111,127,
-5,0,127,107,127,111,127,123,-3,
127,111,-2,127,123,-4,127,111
1260 DATA-4,127,127,63,127,63,12
7,125,127,125,127,111,-3,127,63,
127,-7,0,-3,127,63,127,63,-3,127
,119,-3,127
1270 DATA125,127,-4,0,127,106,12
7,122,127,122,-2,127,123,127,122
,127,-5,0,63,127,85,-3,127,-6,0,
-9,127,111,127,111,127,-5,111
1280 DATA-6,127,110,127,106,-3,1
27,-8,0,127,42,106,58,46,-2,47,1
11,127,123,127,127,-2,126,-3,127
,-4,63,43,127,107,127,111,-2,127
1290 DATA-9,0,127,123,127,123,-3
,127,123,-9,127,-3,0,-3,127,117,
-3,127,95,127,95,127,117,127,-3,
0,-3,127,111,127,111,-5,127,126,
127,123,127,126,127,-5,0
1300 DATA127,68,100,84,76,68,70,
69,-4,68,100,84,76,68,70,69,-2,6
8,127,-5,0,127,-2,122,-2,123,-3,
111,127,-2,0
1310 DATA-7,127,126,127,-10,0,-2
,127,119,127,125,127,119,127,95,
127,95,127,95,-8,127,-8,0,-11,12
7,-8,126,127
1320 DATA-51,0,79,127,95,66,112,
48,31,999
1330 '9th line
1340 PRINT#-2
1350 PRINT#-2,CHR$(27);CHR$(16);
CHR$(0);CHR$(1);
1360 READN:IF  N=999 THEN 1480
1370 IF N>=0 THEN PRINT#-2,CHR$(
N+128);:PRINT#-2,CHR$(N+128);:GO
TO1360
1380 READ M
1390 PRINT#-2,CHR$(28);CHR$(-N);
CHR$(128+M);:PRINT#-2,CHR$(28);C
HR$(-N);CHR$(128+M);:GOTO1360
1400 DATA127,111,127,111,127,111
,127,63,125,127,125,127,111,127,
-5,0,127,107,127,111,127,123,-3,
127,111,-2,127,123,-4,127,111
1410 DATA-4,127,127,63,127,63,12
7,125,127,125,127,111,-3,127,63,
127,-7,0,-3,127,63,127,63,-3,127
,119,-3,127
1420 DATA125,127,-4,0,127,106,12
7,122,127,122,-2,127,123,127,122
,127,-5,127,63,127,85,-3,127,-6,
0,-9,127,111,127,111,127,-5,111
1430 DATA-6,127,110,127,106,-3,1
27,-8,0,127,42,106,58,46,-2,47,1
11,127,123,127,127,-2,126,-3,127
,-4,63,43,127,107,127,111,-2,127
1440 DATA-9,0,127,123,127,123,-3
,127,123,-9,127,-3,0,-3,127,117,
-3,127,95,127,95,127,117,127,-3,
0,-3,127,111,127,111,-5,127,126,
127,123,127,126,127,-5,0
1450 DATA127,68,100,84,76,68,70,
69,-4,68,100,84,76,68,70,69,-2,6
8,127,-5,0,127,-2,122,-2,123,-3,
111,127,-2,0
1460 DATA-7,127,126,127,-10,0,-2
,127,119,127,125,127,119,127,95,
127,95,127,95,-8,127,-8,0,-11,12
7,-8,126,127
1470 DATA-50,0,96,-3,127,99,999
1480 '10th line
1490 PRINT#-2
1500 PRINT#-2,CHR$(27);CHR$(16);
CHR$(0);CHR$(1);
1510 READN:IF  N=999 THEN 1630
1520 IF N>=0 THEN PRINT#-2,CHR$(
N+128);:PRINT#-2,CHR$(N+128);:GO
TO1510
1530 READ M
1540 PRINT#-2,CHR$(28);CHR$(-N);
CHR$(128+M);:PRINT#-2,CHR$(28);C
HR$(-N);CHR$(128+M);:GOTO1510
1550 DATA127,111,127,111,127,111
,127,63,125,127,125,127,111,127,
-5,0,127,107,127,111,127,123,-3,
127,111,-2,127,123,-4,127,111
1560 DATA-4,127,127,63,127,63,12
7,125,127,125,127,111,-3,127,63,
127,-7,0,-3,127,63,127,63,-3,127
,119,-3,127
1570 DATA125,127,-4,0,127,106,12
7,122,127,122,-2,127,123,127,122
,127,-5,127,63,127,85,-3,127,-6,
0,-9,127,111,127,111,127,-5,111
1580 DATA-6,127,110,127,106,-3,1
27,-8,0,127,42,106,58,46,-2,47,1
11,127,123,127,127,-2,126,-3,127
,-4,63,43,127,107,127,111,-2,127
1590 DATA-9,0,127,123,127,123,-3
,127,123,-9,127,-3,0,-3,127,117,
-3,127,95,127,95,127,117,127,-3,
127,-3,127,111,127,111,-5,127,12
6,127,123,127,126,127,-5,0
1600 DATA127,68,100,84,76,68,70,
69,-4,68,100,84,76,68,70,69,-2,6
8,127,-5,0,127,-2,122,-2,123,-3,
111,127,-2,0
1610 DATA-7,127,126,127,-10,0,-2
,127,119,127,125,127,119,127,95,
127,95,127,95,-8,127,-8,0,-11,12
7,-8,126,127
1620 DATA-50,0,-5,127,999
1630 '11th line
1640 PRINT#-2
1650 PRINT#-2,CHR$(27);CHR$(16);
CHR$(0);CHR$(1);
1660 READN:IF  N=999 THEN 1780
1670 IF N>=0 THEN PRINT#-2,CHR$(
N+128);:PRINT#-2,CHR$(N+128);:GO
TO1660
1680 READ M
1690 PRINT#-2,CHR$(28);CHR$(-N);
CHR$(128+M);:PRINT#-2,CHR$(28);C
HR$(-N);CHR$(128+M);:GOTO1660
1700 DATA127,111,127,111,127,111
,127,63,125,127,125,127,111,127,
-5,0,127,107,127,111,127,123,-3,
127,111,-2,127,123,-4,127,111
1710 DATA-4,127,127,63,127,63,12
7,125,127,125,127,111,-3,127,63,
127,-7,0,-3,127,63,127,63,-3,127
,119,-3,127
1720 DATA125,127,-4,0,127,106,12
7,122,127,122,-2,127,123,127,122
,127,-5,127,63,127,85,-3,127,-6,
127,-9,127,111,127,111,127,-5,11
1
1730 DATA-6,127,110,127,106,-3,1
27,-8,0,127,42,106,58,46,-2,47,1
11,127,123,127,127,-2,126,-3,127
,-4,63,43,127,107,127,111,-2,127
1740 DATA-9,0,127,123,127,123,-3
,127,123,-9,127,-3,0,-3,127,117,
-3,127,95,127,95,127,117,127,-3,
127,-3,127,111,127,111,-5,127,12
6,127,123,127,126,127,-5,0
1750 DATA127,68,100,84,76,68,70,
69,-4,68,100,84,76,68,70,69,-2,6
8,127,-5,0,127,-2,122,-2,123,-3,
111,127,-2,0
1760 DATA-7,127,126,127,-10,120,
-2,127,119,127,125,127,119,127,9
5,127,95,127,95,-8,127,-8,64,-11
,127,-8,126,127
1770 DATA-49,0,124,-5,127,124,99
9
1780 '12th line
1790 PRINT#-2
1800 PRINT#-2,CHR$(27);CHR$(16);
CHR$(0);CHR$(1);
1810 READN:IF N=999 THEN 1910
1820 IF N>=0 THEN PRINT#-2,CHR$(
N+128);:PRINT#-2,CHR$(N+128);:GO
TO1810
1830 READ M
1840 PRINT#-2,CHR$(28);CHR$(-N);
CHR$(128+M);:PRINT#-2,CHR$(28);C
HR$(-N);CHR$(128+M);:GOTO1810
1850 DATA127,111,-3,127,63,-4,12
7,109,-3,127,-5,124,-9,127,119,1
27,119,-17,127,119,127,119,127,1
19,-3,127,-3,112,80
1860 DATA112,80,112,127,125,127,
125,127,125,-3,127,95,127,95,-3,
127,-4,0,-3,127,-11,126,-5,127,1
22,127,43,127
1870 DATA-6,127,-30,127,-6,0,126
,86,87,-3,85,-2,93,95,-21,127
1880 DATA-9,0,-20,127,127,59,127
,39,-3,127,62,-3,127,111,-7,127,
125,127,125,-11,127,-5,0
1890 DATA127,-7,120,124,122,121,
-9,120,127,-5,0,-23,127,-7,127,-
```

```
10,127,126,127,123,127,123,127,1
23,127,123,-2,127,-8,127,-20,127
1900 DATA-7,0,-11,64,-29,0,120,-
9,127,120,999
1910 '13th line
1920 PRINT#-2
1930 PRINT#-2,CHR$(27);CHR$(16);
CHR$(0);CHR$(1);
1940 READN: IF N=999 THEN 2010
1950 IF N>=0 THEN PRINT#-2,CHR$(
N+128);:PRINT#-2,CHR$(128+N);:GO
TO1940
1960 READ M
1970 PRINT#-2,CHR$(28);CHR$(-N);
CHR$(128+M);:PRINT#-2,CHR$(28);C
HR$(-N);CHR$(128+M);:GOTO1940
1980 DATA127,65,117,101,91,127,9
9,93,85,101,-10,127,126,-31,127,
119,127,119,-2,127,126,127,126,1
27,126,127,-5,127,-4,124,-127,12
7,125,125,127,125,-148,127
1990 DATA-3,96,98,-13,127,98,96,
32,96,32,96,32,96,32,96,-2,64,96
,64,96,64,96,64,96,64,96,64,96,6
4,96
2000 DATA124,126,-13,127,126,124
,96,64,96,64,96,64,96,64,96,64,9
6,64,96,64,999
2010 'return printer from graphi
cs mode
2020 PRINT#-2,CHR$(30)
2030 CLS(8)
2040 FOR H=0TO63
2050 SET(H,0,3)
2060 NEXTH
2070 FORV=0TO31
2080 SET(63,V,3)
2090 NEXT V
2100 FOR H=63TO0 STEP-1
2110 SET(H,31,3)
2120 NEXTH
2130 FORV=31 TO 0 STEP-1
2140 SET(0,V,3)
2150 NEXT V
2160 FOR H=3 TO 60
2170 SET(H,2,1)
2180 NEXT H
2190 FOR V=3 TO 29
2200 SET(60,V,1)
2210 NEXT V
2220 FOR H=60 TO 3 STEP-1
2230 SET(H,29,1)
2240 NEXT H
2250 FOR V=29 TO 3 STEP-1
2260 SET(3,V,1)
2270 NEXT V
2280 FOR H=7TO56STEP2
2290 RESET(H,6)
2300 NEXTH
2310 FORV=8 TO 24 STEP2
2320 RESET(57,V)
2330 NEXTV
2340 FOR H=56 TO 6 STEP-2
2350 RESET(H,26)
2360 NEXT H
2370 FOR V=24 TO 7 STEP-2
2380 RESET(6,V)
2390 NEXT V
2400 FOR H=9TO13
2410 RESET(H,12)
2420 NEXTH
2430 FOR V=13TO18
2440 RESET(11,V)
2450 NEXTV
2460 FOR H=17TO20
2470 RESET(H,15)
2480 NEXTH
2490 FORV=12TO18
2500 RESET(16,V)
2510 NEXTV
2520 FORV=12TO18
2530 RESET(21,V)
2540 NEXTV
2550 FORH=25TO27
2560 RESET(H,12)
2570 NEXTH
2580 FORH=25TO27
2590 RESET(H,15)
2600 NEXTH
2610 FORH=25TO27
2620 RESET(H,18)
2630 NEXTH
2640 FORV=12TO18
2650 RESET(24,V)
2660 NEXTV
2670 FORH=34TO37
2680 RESET(H,12)
2690 NEXTH
2700 FORH=34TO37
2710 RESET(H,15)
2720 NEXTH
2730 FORH=34TO37
2740 RESET(H,18)
2750 NEXTH
2760 FORV=12TO18
2770 RESET(34,V)
2780 NEXTV
2790 FORV=12TO18
2800 RESET(40,V)
2810 NEXTV
2820 FORV=12TO18
2830 RESET(46,V)
2840 NEXTV
2850 H=41
2860 RESET(H,12)
2870 H=41
2880 RESET(H,13)
2890 H=42
2900 RESET(H,14)
2910 H=43
2920 RESET(H,15)
2930 H=44
2940 RESET(H,16)
2950 H=45
2960 RESET(H,17)
2970 H=45
2980 RESET(H,18)
2990 FORH=49TO54
3000 RESET(H,12)
3010 NEXTH
3020 FORH=49TO54
3030 RESET(H,18)
3040 NEXTH
3050 FORV=13TO17
3060 RESET(50,V)
3070 NEXTV
3080 FORV=13TO17
```

Continued on P 58

Lotto Checker

Ah! Just what we needed.

by Barry Sidebottom

CoCo3
APPLICATION

A COUPLE OF years ago I sent in a program that automatically checks four games of Lotto. This is the same game, except rewritten for the CoCo 3. It's improved graphics allows a better looking screen and visual presentation of the finished card.

The program is straight forward once running and so no instructions are necessary. A couple of notes however.

As supplied the program checks 4 games I take out: Barry (mine), Linda (my wife's), Barlin (ours) and Bank (Linda's work).

To use for yourself, you will have to renumber the variables in line 279 to your own game's names and then change the data statements in lines 285-334 to your particular numbers.

One final note - when entering the weeks numbers at the start, they need not be in order, the program will sort them for you.

Hope you can use it. If any problems give me a ring on ...

(03) 744-6281.

The Listing:

```
0 GOTO10
1 ' ***********************
2 ' * TATTSLOTTO CHECKER *
3 ' *         by         *
4 ' *  BARRY SIDEBOTTOM  *
5 ' ***********************
6 ' * Ver 2.6 (3) - 1987 *
7 ' ***********************
8 SAVE"107:3":END'8
9 '
10 ON BRK GOTO555:ON ERR GOTO555
11 WIDTH40:ATTR3,2:CLS:PRINT"STA
NDBY ..":DIM L(48,2):POKE65497,0
:PALETTE0,2:PALETTE7,56:PALETTE1
5,60:GOSUB441:GOSUB 261
12 ' **INPUT ROUTINE
13 ATTR3,2:CLS:PRINT"INPUT THIS
WEEK'S DRAW:"
14 PRINT:FOR X=1TO6:SOUND220,1:P
RINT"NUMBER: (";X;")";:INPUTK(X)
:NEXT
15 SOUND220,1:GOSUB410
16 GOSUB391
17 ATTR5,2:PRINT:PRINT"ARE THEY
CORRECT? (Y/N)":SOUND220,1
18 PRINT:ATTR3,2:FOR X=1TO6:PRIN
TD(X);:NEXT
19 Z$=INKEY$:IF Z$="Y"THEN 25 EL
SE IF Z$="N"THEN 13 ELSE 19
20 '
21 '
22 '
23 '
24 ' ** SUPPLEMENTARY
25 PRINT:PRINT:PRINT"INPUT THIS
WEEK'S SUPP.S:"
26 PRINT:SOUND220,1:INPUT"1 ";S(
1):SOUND220,1:INPUT"2 ";S(2)
27 SOUND220,1:GOSUB422
28 GOSUB375
29 GOSUB433
30 ATTR5,2:PRINT:PRINT"ARE THE S
UPP's CORRECT? (Y/N)":PRINT:ATTR
3,2:PRINTS(1):PRINTS(2):SOUND220
,1
31 Z$=INKEY$:IF Z$="Y"THEN 33 EL
SE IF Z$="N"THEN25ELSE 31
32 ' ** CHECK MENU
33 GOSUB443:GOSUB454:GOSUB483:GO
SUB551:HCOLOR3:HPRINT(0,0),"CHOO
SE":HPRINT(0,2),"(M)anual or":HP
RINT(0,3),"(A)utomatic":HPRINT(5
,4),"check.":SOUND220,1
34 TT$=INKEY$:IF TT$=""THEN 34
35 IF TT$="M"OR TT$="A"THEN36 EL
SE 33
36 GOSUB494:IF TT$="M"THEN37ELSE
 Z=Z+1:GOTO38
37 HCOLOR0:HPRINT(2,4),"WHICH GA
ME?"
38 HCOLOR0:HPRINT(3,6),"1 - BARR
Y":HPRINT(3,7),"2 - BARLIN"
39 HPRINT(3,8),"3 - LINDA":HPRIN
T(3,9),"4 - BANK":IF TT$="A"THEN
 HLINE(3,43)-(115,83),PSET,B:SOU
ND220,1:GOTO42 ELSE HLINE(3,27)-
(115,83),PSET,B:SOUND220,1
40 ZZ$=INKEY$:IF ZZ$=""THEN40
41 Z=VAL(ZZ$):IF Z<1THEN40
42 IF Z>4THEN339
43 HCOLOR3:HPRINT(1,Z+5),"*":ON
Z GOTO49,80,111,142
44 GOTO40
45 '
46 '
47 '
48 '
49 ' ** BARRY
50 IF TT$="A"THEN M$="A"ELSE GOS
UB182
51 F=1:G=6:GOSUB508
52 HCOLOR13:HPRINT(1,15),E$:HLIN
E(8,128)-(46,128),PSET
53 GOSUB173
54 Y=9:FOR X=F TO G:AA=A(X):GOSU
B547:NEXT
55 I=0:J=0
56 FOR E=F TO G
57 FOR X=1 TO 6
58 IF D(X)=A(E)THEN I=I+1
59 NEXT X
60 NEXT E
61 FOR X=F TO G
62 IF S(1)=A(X)THEN J=J+1
63 IF S(2)=A(X)THEN J=J+1
64 NEXT
65 GOSUB191
66 GOSUB252:GOSUB539
67 F=F+6:G=G+6:I=0:J=0:IF G>60TH
EN68ELSE52
68 IF TT$="A"THEN Z=Z+1:GOSUB535
69 GOSUB218
70 IF Z$="Y"THEN523ELSE IF Z$="N
"THEN GOSUB225ELSE69
71 GOSUB519:HCOLOR13:Y$=K$:GOSUB
176
72 Y=9:FOR X=F TO G:AA=A(X):GOSU
B547:NEXT
73 GOSUB 248
74 IF Z$="Y"THEN GOSUB539:GOSUB
235 ELSE IF Z$="N"THEN 233
75 GOTO71
76 '
77 '
78 '
79 '
80 ' ** BARLIN
81 IF TT$="A"THEN M$="A"ELSE GOS
UB182
82 F=1:G=6:GOSUB508
83 HCOLOR13:HPRINT(1,15),F$:HLIN
E(8,128)-(54,128),PSET
84 GOSUB173
85 Y=9:FOR X=F TO G:AA=B(X):GOSU
B547:NEXT
86 I=0:J=0
87 FOR E=F TO G
88 FOR X=1 TO 6
89 IF D(X)=B(E)THEN I=I+1
90 NEXT X
```

```
91 NEXT E
92 FOR X=F TO G
93 IF S(1)=B(X)THEN J=J+1
94 IF S(2)=B(X)THEN J=J+1
95 NEXT
96 GOSUB191
97 GOSUB252:GOSUB539
98 F=F+6:G=G+6:I=0:J=0:IF G>60TH
EN99ELSE83
99 IF TT$="A"THEN Z=Z+1:GOSUB535
100 GOSUB218
101 IF Z$="Y"THEN523ELSE IF Z$="
N"THEN GOSUB225ELSE100
102 GOSUB519:HCOLOR13:Y$=K$:GOSU
B176
103 Y=9:FOR X=F TO G:AA=B(X):GOS
UB547:NEXT
104 GOSUB 248
105 IF Z$="Y"THEN GOSUB539:GOSUB
 235 ELSE IF Z$="N"THEN 233
106 GOTO102
107 '
108 '
109 '
110 '
111 ' ** LINDA
112 IF TT$="A"THEN M$="A"ELSE GO
SUB182
113 F=1:G=6:GOSUB508
114 HCOLOR13:HPRINT(1,15),G$:HLI
NE(8,128)-(46,128),PSET
115 GOSUB173
116 Y=9:FOR X=F TO G:AA=C(X):GOS
UB547:NEXT
117 I=0:J=0
118 FOR E=F TO G
119 FOR X=1 TO 6
120 IF D(X)=C(E)THEN I=I+1
121 NEXT X
122 NEXT E
123 FOR X=F TO G
124 IF S(1)=C(X)THEN J=J+1
125 IF S(2)=C(X)THEN J=J+1
126 NEXT
127 GOSUB191
128 GOSUB252:GOSUB539
129 F=F+6:G=G+6:I=0:J=0:IF G>60T
HEN130ELSE114
130 IF TT$="A"THEN Z=Z+1:GOSUB53
5
131 GOSUB218
132 IF Z$="Y"THEN523ELSE IF Z$="
N"THEN GOSUB225ELSE131
133 GOSUB519:HCOLOR13:Y$=K$:GOSU
B176
134 Y=9:FOR X=F TO G:AA=C(X):GOS
UB547:NEXT
135 GOSUB 248
136 IF Z$="Y"THEN GOSUB539:GOSUB
 235 ELSE IF Z$="N"THEN 233
137 GOTO133
138 '
139 '
140 '
141 '
142 ' ** BANK
143 IF TT$="A"THEN M$="A"ELSE GO
SUB182
144 F=1:G=6:GOSUB508
145 HCOLOR13:HPRINT(1,15),R$:HLI
NE(8,128)-(38,128),PSET
146 GOSUB173
147 Y=9:FOR X=F TO G:AA=V(X):GOS
UB547:NEXT
148 I=0:J=0
149 FOR E=F TO G
150 FOR X=1TO6
151 IF D(X)=V(E)THEN I=I+1
152 NEXT X
153 NEXT E
154 FOR X=F TO G
155 IF S(1)=V(X)THEN J=J+1
156 IF S(2)=V(X)THEN J=J+1
157 NEXT
158 GOSUB191
159 GOSUB252:GOSUB539
160 F=F+6:G=G+6:I=0:J=0:IF G>60T
HEN161ELSE145
161 IF TT$="A"THEN Z=Z+1:GOSUB53
5
162 GOSUB218
163 IF Z$="Y"THEN523ELSE IF Z$="
N"THEN GOSUB225ELSE162
164 GOSUB519:HCOLOR13:Y$=K$:GOSU
B176
165 Y=9:FOR X=F TO G:AA=V(X):GOS
UB547:NEXT
166 GOSUB248
167 IF Z$="Y"THEN GOSUB539:GOSUB
235ELSE IF Z$="N"THEN233
168 GOTO165
169 '
170 '
171 '
172 '
173 ' ** DETERMINE GAME
174 IF F=1THEN Y$="A"ELSE IF F=7
THEN Y$="B"ELSE IF F=13THEN Y$="
C"ELSE IF  F=19THEN Y$="D"ELSE I
F F=25THEN Y$="E"
175 IF F=31THEN Y$="F"ELSE IF F=
37THEN Y$="G"ELSE IF F=43THEN Y$
="H"ELSE IF F=49THEN Y$="I"ELSE
IF F=55THEN Y$="J"
176 HCOLOR3:HPRINT(1,17),D$:HPRI
NT(6,17),Y$:HCOLOR13
177 RETURN
178 '
179 '
180 '
181 '
182 ' ** AUTO OR MANUAL CHECK
183 HCOLOR3:HPRINT(1,11),"(M)anu
al or":HPRINT(1,12),"(Auto check
?":SOUND220,1
184 M$=INKEY$:IF M$=""THEN184
185 RETURN
186 '
187 '
188 '
189 '
190 ' ** DETERMINE DIVISION
191 HCOLOR6:IF I=6 THEN 193 ELSE
 IF I=5 AND J>0 THEN 196 ELSE IF
 I=5 THEN 198 ELSE IF I=4 THEN 2
01 ELSE IF I=3 AND J>0 THEN 204
192 GOTO207
193 HPRINT(1,19),I$:HPRINT(28,19
),"1"
194 GOSUB214
195 GOTO209
196 HPRINT(1,19),I$:HPRINT(28,19
),"2"
197 GOTO214:GOTO209
198 HPRINT(1,19),I$:HPRINT(28,19
),"3"
199 GOSUB214
200 GOTO209
201 HPRINT(1,19),I$:HPRINT(28,19
),"4"
202 GOSUB214
203 GOTO209
204 HPRINT(1,19),I$:HPRINT(28,19
),"5"
205 GOSUB214
206 GOTO209
207 QQ=0:HCOLOR13:HPRINT(1,19),"
SORRY, ONLY":HPRINT(12,19),I:HPR
INT(15,19),"NUMBER(S)":IF J=1 TH
EN HPRINT(25,19),"& ONE SUPP."EL
SE IF J=2THEN HPRINT(25,19),"& B
OTH SUPP's"ELSE208
208 SOUND 100,1:SOUND 100,3
209 RETURN
210 '
211 '
212 '
213 '
214 ' ** HOLD FOR WIN
215 QQ=1
216 FOR X=1 TO 5:PALETTE6,RND(64
)-1:PLAY P$:NEXT:PALETTE6,9
217 Z$=INKEY$:IF Z$=""THEN 217 E
LSE RETURN
218 HCOLOR13:HPRINT(1,19),"DO YO
U WISH TO VIEW AGAIN ? (Y/N)":SO
UND220,1
219 Z$=INKEY$:IF Z$=""THEN 219
220 RETURN
221 '
222 '
223 '
224 '
225 ' ** REVIEW GAME
226 GOSUB519:HCOLOR13:HPRINT(1,1
9),"DO YOU WISH TO REVIEW A GAME
 ? (Y/N)":SOUND220,1
227 Z$=INKEY$:IF Z$="Y"THEN235EL
SE IF Z$="N"THEN232ELSE227
228 '
229 '
230 '
231 '
232 ' ** RETURN TO MAIN MENU
233 GOSUB519:HCOLOR13:HPRINT(1,1
9),"RETURN TO MAIN MENU ? (Y/N)"
:SOUND220,1
234 Z$=INKEY$:IF Z$="Y"THEN531EL
SE IF Z$="N"THEN543ELSE234
235 GOSUB519:HCOLOR13:HPRINT(1,1
9),"WHICH GAME ? (A - J)":SOUND2
20,1
236 K$=INKEY$:IF K$=""THEN236
237 IF K$="A"THEN F=1:G=6:RETURN
 ELSE 238
238 IF K$="B"THEN F=7:G=12:RETUR
N ELSE 239
239 IF K$="C"THEN F=13:G=18:RETU
RN ELSE 240
240 IF K$="D"THEN F=19:G=24:RETU
RN ELSE 241
241 IF K$="E"THEN F=25:G=30:RETU
RN ELSE 242
```

```
242 IF K$="F"THEN F=31:G=36:RETU
RN ELSE 243
243 IF K$="G"THEN F=37:G=42:RETU
RN ELSE 244
244 IF K$="H"THEN F=43:G=48:RETU
RN ELSE 245
245 IF K$="I"THEN F=49:G=54:RETU
RN ELSE 246
246 IF K$="J"THEN F=55:G=60:RETU
RN ELSE 247
247 GOTO 236
248 HPRINT(1,19),"ANOTHER ?  (Y/
N)":SOUND220,1
249 Z$=INKEY$:IF Z$=""THEN 249
250 IF Z$<>"Y"AND Z$<>"N"THEN249
ELSE251
251 RETURN
252 IF M$="A"THEN253ELSE255
253 RETURN
254 FOR Q=1TO1000:NEXT:RETURN
255 IF QQ=0THEN256ELSE RETURN
256 Z$=INKEY$:IF Z$=""THEN 256 E
LSE RETURN
257 '
258 '
259 '
260 '
261 ' ** TITLES
262 HCOLOR14:HPRINT(1,18),"TATTS
LOTTO CHECKER.":HPRINT(1,19),"by
":HPRINT(1,20),"BARRY SIDEBOTTOM
.":HPRINT(1,21),"6 FROM 45"
263 HCOLOR3:HPRINT(1,23),"versio
n 2.6(3) 1987":SOUND180,2
264 FOR X=1TO3000:NEXT
265 '
266 '
267 '
268 '
269 ' ** DIM ARRAYS
270 DIM A(60),B(60),C(60),V(60)
271 FOR X=1TO60:READ A(X):NEXT
272 FOR X=1TO60:READ B(X):NEXT
273 FOR X=1TO60:READ C(X):NEXT
274 '
275 '
276 FOR X=1 TO 60:READ V(X):NEXT
277 '
278 '
279 D$="Game":E$="BARRY":F$="BAR
LIN":G$="LINDA":H$="THIS WEEK'S
NO.S:":I$="CONGRATULATIONS!!-DIV
ISION":P$="T255;O3;ABABABABABABA
BAB":R$="BANK":HSCREEN0:RETURN
280 DATA 125,149,173,197,221,245
,269,293
281 '
282 '
283 '
284 '
285 ' ** BARRY'S GAMES
286 DATA 1,7,17,20,34,40
287 DATA 2,18,20,26,30,35
288 DATA 5,11,13,14,15,32
289 DATA 2,3,16,26,31,40
290 DATA 2,19,24,26,31,34
291 DATA 5,11,21,24,36,40
292 DATA 4,6,9,11,17,38
293 DATA 17,21,22,27,35,37
294 DATA 1,7,8,23,29,39
295 DATA 6,10,14,17,24,36
```

```
296 '
297 '
298 ' ** BARLIN'S GAMES
299 DATA 2,7,17,30,34,39
300 DATA 17,19,25,34,35,38
301 DATA 5,8,11,21,24,27
302 DATA 2,11,29,33,35,40
303 DATA 3,14,15,26,34,36
304 DATA 8,12,18,20,22,31
305 DATA 4,9,19,21,23,38
306 DATA 1,6,10,26,27,32
307 DATA 4,7,19,25,28,36
308 DATA 7,13,16,29,33,39
309 '
310 '
311 ' ** LINDA'S GAMES
312 DATA 5,16,20,25,30,39
313 DATA 1,7,10,28,30,36
314 DATA 2,8,13,17,24,37
315 DATA 2,5,14,16,33,38
316 DATA 4,9,14,19,25,37
317 DATA 5,11,17,21,23,32
318 DATA 9,12,22,27,36,39
319 DATA 7,10,18,28,30,31
320 DATA 1,10,22,28,33,39
321 DATA 6,12,21,22,32,39
322 '
323 '
324 ' ** LINDA'S BANK
325 DATA 8,12,13,22,23,33
326 DATA 4,15,23,35,42,44
327 DATA 5,15,27,32,42,45
328 DATA 9,10,17,18,30,31
329 DATA 2,7,14,28,29,35
330 DATA 3,16,21,22,43,45
331 DATA 1,25,26,30,34,40
332 DATA 1,5,19,20,28,29
333 DATA 4,6,10,21,24,43
334 DATA 2,11,37,38,39,41
335 '
336 '
337 '
338 '
339 ' ** REPEAT?
340 HCOLOR13:HPRINT(1,19),"DO YO
U WISH TO GO THRU' AGAIN? (Y/N)"
:SOUND220,1
341 Z$=INKEY$:IF Z$=""THEN341
342 IF Z$="Y"THEN TT$="":Z=0:GOT
O33ELSE348
343 '
344 '
345 '
346 '
347 ' ** CHECK A GAME?
348 GOSUB519:HCOLOR13:HPRINT(1,1
9),"DO YOU WISH TO CHECK ANY GAM
E? (Y/N)":SOUND220,1
349 Z$=INKEY$:IF Z$=""THEN349
350 IF Z$="Y"THEN351ELSE543
351 GOSUB519:HCOLOR13:HPRINT(1,1
9),"? : 1.Barry  2.Barlin  3.Lin
da  4.Bank":SOUND220,1
352 Z$=INKEY$:IF Z$=""THEN352
353 Z=VAL(Z$):IF Z<0OR Z>4THEN35
2ELSE GOSUB235
354 ON Z GOTO 355,359,363,367
355 GOSUB519:HCOLOR13:Y$=K$:GOSU
B176
356 Y=9:FOR X=F TO G:AA=A(X):GOS
UB547:NEXT
```

```
357 GOSUB 248
358 IF Z$="Y"THEN GOSUB539:GOTO3
53 ELSE IF Z$="N"THEN GOSUB527:G
OTO340
359 GOSUB519:HCOLOR13:Y$=K$:GOSU
B176
360 Y=9:FOR X=F TO G:AA=B(X):GOS
UB547:NEXT
361 GOSUB248
362 IF Z$="Y"THEN GOSUB539:GOTO3
53 ELSE IF Z$="N"THEN GOSUB527:G
OTO340
363 GOSUB519:HCOLOR13:Y$=K$:GOSU
B176
364 Y=9:FOR X=F TO G:AA=C(X):GOS
UB547:NEXT
365 GOSUB248
366 IF Z$="Y"THEN GOSUB539:GOTO3
53 ELSE IF Z$="N"THEN GOSUB527:G
OTO340
367 GOSUB519:HCOLOR13:Y$=K$:GOSU
B176
368 Y=9:FOR X=F TO G:AA=V(X):GOS
UB547:NEXT
369 GOSUB248
370 IF Z$="Y"THEN GOSUB539:GOTO3
53 ELSEELSE IF Z$$="N"THEN GOSUB
527:GOTO340
371 '
372 '
373 '
374 '
375 ' ** COMPARE SUPP WITH NUMBE
RS
376 Y=1
377 FOR X=1TO 6:IF S(Y)=D(X)THEN
 380 ELSE NEXT
378 Y=Y+1:IF Y=3THEN379ELSE377
379 RETURN
380 SOUND 100,2:SOUND 100,2:CLS:
PRINT"YOU HAVE ALREADY ENTERED"S
(Y)"AS ONE OF THE SIX DRAWN NUMB
ERS"
381 PRINT:PRINTTHIS EEK'S NUMBER
S:":FOR X=1 TO 6:PRINT D(X);:NEX
T
382 PRINT:PRINT:PRINT"THE SUPP's
: ";S(1);"  "S(2)
383 PRINT:PRINT:PRINT:PRINT"CHAN
GE 1> THE WEEKLY NUMBERS
    OR      2> THE SUPP."
384 Z$=INKEY$:IF Z$=""THEN384
385 Z=VAL(Z$):IF Z<0OR Z>2THEN38
4
386 ON Z GOTO 13,25
387 '
388 '
389 '
390 '
391 ' ** SORT ROUTINE
392 G=0:J=0
393 F=0
394 F=F+1
395 G=G+1
396 IF G>6 THEN RETURN
397 FOR H=1 TO 6
398 IF K(H)<K(F) THEN F=H
399 NEXT H
400 GOSUB403
401 K(F)=100
402 GOTO393
```

```
403 J=J+1
404 D(J)=K(F)
405 RETURN
406 '
407 '
408 '
409 '
410 ' ** COMPARE FOR OVER 45
411 FOR X=1TO6
412 IF K(X)>45 THEN 414 ELSE NEX
T
413 RETURN
414 CLS:SOUND 100,2:SOUND 100,2:
PRINT:PRINT:PRINT"SORRY ENTRY "X
" IS TOO HIGH"
415 PRINT:FOR A=1TO6:PRINT"(";A;
")";K(A):NEXT:PRINT:PRINT
416 INPUT"PLEASE RE-ENTER: ";K(X
)
417 GOTO410
418 '
419 '
420 '
421 '
422 ' ** COMPARE SUPP FOR >45
423 X=1
424 IF S(X)>45 THEN 426 ELSE 425
425 X=X+1:IF X>3THEN RETURN ELSE
 424
426 PRINT:PRINTS(1);S(2):SOUND 1
00,2:SOUND 100,2:PRINT:PRINT:PRI
NT"SORRY"S(X)"TOO HIGH"
427 PRINT:INPUT"PLEASE RE-ENTER:
 ";S(X)
428 GOTO 424
429 '
430 '
431 '
432 '
433 ' *** SORT SUPP'S ROUTINE
434 S(3)=S(1):S(4)=S(2)
435 IF S(3)<S(4)THEN RETURN ELSE
 S(1)=S(4):S(2)=S(3)
436 RETURN
437 '
438 '
439 '
440 '
441 '** DRAW SCREEN FOR TITLES
442 GOSUB477
443 HSCREEN2:HCOLOR7,7:HCLS
444 HCOLOR15:HLINE(125,4)-(315,1
00),PSET,BF
445 HCOLOR14:HPRINT(17,1),"1  2
 3  4  5  6  7  8":HPRINT(17,3),
"9 10 11 12 13 14 15 16":HPRINT(
16,5),"17 18 19 20 21 22 23 24":
446 HPRINT(16,7),"25 26 27 28 29
 30 31 32":HPRINT(16,9),"33 34 3
5 36 37 38 39 40":HPRINT(16,11),
"41 42 43 44 45"
447 HCOLOR2:HLINE(124,3)-(316,10
1),PSET,B:FOR X=1TO48:HLINE(L(X,
1),L(X,2))-(L(X,1)+24,L(X,2)+16)
,PSET,B:NEXT
448 HCOLOR7:HLINE(246,85)-(319,1
25),PSET,BF
449 RETURN
450 '
451 '
452 '
453 '
454 ' ** SET SQUARES VARIABLES
455 D(7)=S(1):D(8)=S(2)
456 FOR X=1TO8
457 R(X)=1:IF D(X)>8THEN458ELSE4
62
458 R(X)=2:IF D(X)>16THEN459ELSE
462
459 R(X)=3:IF D(X)>24THEN460ELSE
462
460 R(X)=4:IF D(X)>32THEN461ELSE
462
461 R(X)=5:IF D(X)>40THEN R(X)=6
462 NEXT
463 FOR X=1TO8
464 IF R(X)=1THEN CL(X)=D(X):GOT
O466
465 CL(X)=D(X)-((R(X)-1)*8)
466 NEXT
467 FOR X=1TO6
468 HCOLOR3:A=L(D(X),1):B=L(D(X)
,2):HLINE(A,B)-(A+24,B+16),PSET,
BF:HCOLOR2:HLINE(A,B)-(A+24,B+16
),PSET,B:NEXT
469 FOR X=7TO8:HCOLOR4:A=L(D(X),
1):B=L(D(X),2):HLINE(A,B)-(A+24,
B+16),PSET,BF:HCOLOR2:HLINE(A,B)
-(A+24,B+16),PSET,B:NEXT
470 HCOLOR4:FOR X=1TO6:GOSUB471:
NEXT:GOTO475
471 IF D(X)<9THEN Q=1:GOTO473
472 Q=R(X)*2-1
473 P=(CL(X)*3)+12:IF D(X)<10THE
N P=P+1
474 HPRINT(P,Q),D(X):RETURN
475 HCOLOR3:FOR X=7TO8:GOSUB471:
NEXT
476 RETURN
477 C=4:FOR X=1TO48:READ L(X,1):
L(X,2)=C:IF X/8=INT(X/8)THEN478E
LSE479
478 C=C+16:IF X>47THEN479ELSE RE
STORE
479 NEXT X
480 RETURN
481 '
482 '
483 ' ** PRINT CURRENT DRAW AT B
OTTOM
484 HCOLOR1:HLINE(0,183)-(319,19
1),PSET,BF:HCOLOR2:HPRINT(0,23),
"THIS DRAW:"
485 Y=10:FOR X=1TO6
486 HPRINT(Y,23),D(X)
487 IF D(X)<10THEN Y=Y+2ELSE Y=Y
+3
488 NEXT
489 HCOLOR3:HPRINT(Y+1,23),CHR$(
123):Y=Y+2
490 HPRINT(Y,23),S(1):Y=Y+LEN(ST
R$(S(1)))+1:HPRINT(Y,23),S(2):Y=
Y+LEN(STR$(S(2)))+1:HPRINT(Y,23)
,CHR$(125):RETURN
491 RETURN
492 '
493 '
494 ' ** ERASE MANUAL/AUTO & FLA
SH                  ANSWER
495 HCOLOR7:HLINE(0,0)-(100,40),
PSET,BF
496 PALETTE13,5:HCOLOR13:IF TT$=
"A"THEN497ELSE499
497 HPRINT(3,1),"AUTOMATIC"
498 HLINE(19,3)-(99,19),PSET,B:G
OTO501
499 HPRINT(4,1),"MANUAL"
500 HLINE(27,3)-(83,19),PSET,B
501 FOR X=1TO5:PALETTE13,63:PALE
TTE13,5:SOUND220,1:NEXT:RETURN
502 '
503 '
504 '** ERASE LINES 11,12
505 HCOLOR7:HLINE(4,84)-(113,104
),PSET,BF:RETURN
506 '
507 '
508 '** PRINT AUTO/MANUAL
509 GOSUB504
510 HCOLOR3:IF M$="M"THEN HPRINT
(4,11),"Manual":HPRINT(2,12),"pr
ess a key"ELSE HPRINT(5,11),"Aut
o"
511 RETURN
512 '
513 '
514 '** SET GAP BETWEEN PRINTING
 OF                 NUMBERS
515 IF AA<10THEN Y=Y+2ELSE Y=Y+3
516 RETURN
517 '
518 '
519 '** ERASE LINE 19
520 HCOLOR7:HLINE(5,149)-(314,16
0),PSET,BF:RETURN
521 '
522 '
523 '** ERASE LINES 11,12,19 & R
ETURN               TO 'ON Z GOTO' (
auto).
524 GOSUB519:GOSUB504:GOTO43
525 '
526 '
527 '** ERASE 15 - 19 & '*'
528 HCOLOR7:HLINE(5,118)-(314,16
2),PSET,BF:HLINE(6,47)-(15,80),P
SET,BF:RETURN
529 '
530 '
531 '** ERASE 11,12 & 15 - 19 &
RETURN              TO 'WHICH GAME'
(manual).
532 GOSUB504:GOSUB527:GOTO37
533 '
534 '
535 '** ERASE 15 - 19 & "*"THEN
RETURN              TO SORT NEXT GAM
E (auto).
536 GOSUB527:GOTO42
537 '
538 '
539 '** ERASE LINES 17 - 19
540 HCOLOR7:HLINE(5,132)-(314,16
2),PSET,BF:RETURN
541 '
542 '
543 '** COLD START
544 PALETTE RGB:POKE113,0:EXEC40
999
545 '
546 '
547 '** PRINT NUMBERS ON SCREEN
```

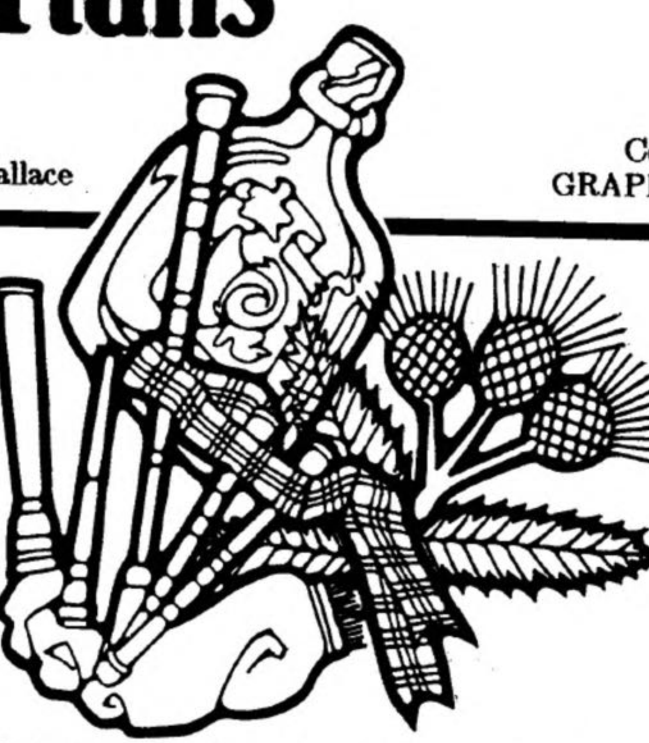
Continued on P 67

Scottish Tartans

Six of the best from one of the best.

by Joy Wallace

CoCo3
GRAPHICS



MAL MCLAUCHLAN SAID he would like to see some Scottish Tartans on the CoCo 3 screen, so with a name like Wallace and also a member of the Sottish Dance Band, I decided to jump right in.

There are six tartans altogether, the first of which is the Wallace. I don't know what the sixth one is, so I called it "Tartan No 6".

Press the break key after each screen and there you are.

The Listing:

```
0 GOTO5
1 '******* 6 TARTAN SCREENS*****
*  ****************************
*
3 SAVE"85:3":END'5
5 HSCREEN2:HCLS11
10 POKE65497,0
15 FORX=0TO290STEP10
20 FORP=0TO10:C=RND(15):PALETTEP
,C:NEXTP:PALETTE11,35:PALETTE12,
0
25 HCOLOR RND(10)
30 HLINE(10,0)-(0,10),PSET:HLINE
(20,0)-(0,20),PSET:HLINE(300,0)-
(320,20),PSET:HLINE(310,0)-(320,
10),PSET:HLINE(X,0)-(X+30,30),PS
ET:HLINE(X+30,0)-(X,30),PSET:HLI
NE(X,161)-(X+30,191),PSET:HLINE(
X+30,161)-(X,191),PSET:NEXTX
35 FORY=10TO160STEP10
40 HCOLOR RND(10):HLINE(0,170)-(
20,190),PSET:HLINE(0,180)-(10,19
0),PSET:HLINE(300,190)-(320,170)
,PSET:HLINE(310,190)-(320,180),P
SET:HLINE(0,Y)-(30,Y+30),PSET:HL
INE(0,Y+30)-(30,Y),PSET:HLINE(29
0,Y)-(320,Y+30),PSET:HLINE(290,Y
+30)-(320,Y),PSET:NEXTY
45 HCOLOR12:HLINE(30,30)-(290,16
0),PSET,B
50 HPAINT(160,96),11,12
55 HPRINT(11,5),"***************
***":HPRINT(11,7),"6 SCOTTISH TA
RTANS":HPRINT(11,9),"***********
*******":HPRINT(18,11),"by":HPRI
NT(14,13),"JOY WALLACE.":HPRINT(
5,18),"Press BREAK after each sc
reen."
60 FORP=0TO10:C=RND(64):PALETTEP
,C:NEXTP:FORP=0TO10:C=RND(32):PA
LETTEP,C:NEXTP
65 ON BRK GOTO 80
70 GOTO60
75 '*******WALLACE TARTAN*******
*   ***************************
*
80 PALETTE0,0:PALETTE1,54:PALETT
E2,4:PALETTE3,36
85 CLS2
90 WIDTH40:HSCREEN2:HCLS2:HCOLOR
0
95 FORX=24TO320 STEP48:HLINE(X,0
)-(X,191),PSET:NEXTX:FORY=24TO19
1STEP48:HLINE(0,Y)-(320,Y),PSET:
NEXTY
100 FORX=20TO320STEP96:HPAINT(X,
15),3,0:HPAINT(X,99),3,0:HPAINT(
X,175),3,0:NEXTX
105 FORX=30TO320STEP96:HPAINT(X,
40),0,0:HPAINT(X,128),0,0:HPAINT
(X,136),0,0:NEXTX
110 FORX=20TO320STEP96:HPAINT(X,
90),3,0:NEXTX
115 HCOLOR1
120 HLINE(48,0)-(48,191),PSET:HL
INE(144,0)-(144,191),PSET:HLINE(
240,0)-(240,191),PSET:HLINE(0,48
)-(320,48),PSET:HLINE(0,144)-(32
0,144),PSET
125 HCOLOR0
130 HLINE(95,0)-(96,191),PSET,B:
HLINE(191,0)-(192,191),PSET,B:HL
INE(287,0)-(288,191),PSET,B:HLIN
E(0,95)-(320,96),PSET,B
135 HCOLOR2:HLINE(75,0)-(212,10)
,PSET,BF
140 HCOLOR1:HPRINT(11,0),"WALLAC
E TARTAN"
145 ON BRK GOTO 160
150 GOTO 150
155 '***WALLACE HUNTING***
      ***********************
160 PALETTE0,0:PALETTE1,54:PALET
TE2,6:PALETTE3,49:PALETTE4,4
165 HCOLOR2:HLINE(75,0)-(212,10)
,PSET,BF:HCOLOR1:HPRINT(11,0),"H
UNTING WALLACE"
170 ON BRK GOTO 185
175 GOTO 175
180 '*****MCQUEEN TARTAN*********
      ***************************
185 HSCREEN2:HCLS0
190 HCLS4:HCOLOR0
195 PALETTE0,0:PALETTE1,54:PALET
TE2,4:PALETTE3,36:PALETTE5,14
200 ON BRK GOTO395
205 FORX=48TO320STEP48:HLINE(X,0
)-(X,191),PSET:NEXTX:FORY=24TO19
1STEP48:HLINE(0,Y)-(320,Y),PSET:
NEXTY
210 FORX=50TO320STEP96:HPAINT(X,
15),3,0:HPAINT(X,99),3,0:HPAINT(
X,175),3,0:NEXTX
215 FORX=24TO320STEP96:HPAINT(X,
40),0,0:HPAINT(X,128),0,0:HPAINT
(X,136),0,0:NEXTX
220 HCOLOR1
225 FORX=24TO320STEP96:HLINE(X,0
)-(X,191),PSET:NEXTX:FORY=48TO14
4STEP96:HLINE(0,Y)-(320,Y),PSET:
NEXTY
```

```
230 HCOLOR0
235 HLINE(60,0)-(156,191),PSET,B
:HLINE(61,0)-(157,191),PSET,B:HL
INE(72,0)-(168,191),PSET,B:HLINE
(73,0)-(169,191),PSET,B:HLINE(84
,0)-(180,191),PSET,B:HLINE(85,0)
-(181,191),PSET,B:HLINE(276,0)-(
277,191),PSET,B:HLINE(264,0)-(26
5,191),PSET,B
240 HLINE(252,0)-(253,191),PSET,
B
245 HLINE(0,0)-(320,96),PSET,B:H
LINE(0,12)-(320,108),PSET,B:HLIN
E(0,1)-(320,97),PSET,B:HLINE(0,1
3)-(320,109),PSET,B:HLINE(0,84)-
(320,180),PSET,B:HLINE(0,85)-(32
0,181),PSET,B:HLINE(0,191)-(320,
191),PSET;B
250 HCOLOR2:HLINE(100,0)-(237,10
),PSET,BF:HCOLOR1:HPRINT(14,0),"
McQueen Tartan"
255 ON BRK GOTO 270
260 GOTO 260
265 '*****ROYAL STEWART********
    ***************************
270 HSCREEN2:HCLS0
275 PALETTE0,0:PALETTE1,54:PALET
TE2,6:PALETTE3,36:PALETTE4,24:PA
LETTE5,36:PALETTE6,63:PALETTE7,2
:PALETTE8,0:PALETTE9,0
280 HCOLOR5:HLINE(72,0)-(96,191)
,PSET,BF:HLINE(216,0)-(240,191),
PSET,BF:HLINE(0,72)-(320,96),PSE
T,BF
285 HCOLOR2:FORX=60TO96STEP36:HL
INE(X,0)-(X+12,191),PSET,BF:NEXT
X:FORX=204TO240STEP36:HLINE(X,0)
-(X+12,191),PSET,BF:NEXTX:FORY=6
0TO96STEP36:HLINE(0,Y)-(320,Y+12
),PSET,BF:NEXTY
290 HCOLOR7:FORX=60TO96STEP36:HL
INE(X,60)-(X+12,108),PSET,B:NEXT
X:FORX=204TO240STEP36:HLINE(X,60
)-(X+12,108),PSET,B:NEXTX:FORY=6
0TO96STEP36:HLINE(60,Y)-(108,Y+1
2),PSET,B:HLINE(204,Y)-(252,Y+12
),PSET,B:NEXTY
295 FORX=65TO101STEP36:HPAINT(X,
65),7,7:HPAINT(X,100),7,7:NEXTX:
FORX=210TO246STEP36:HPAINT(X,65)
,7,7:HPAINT(X,100),7,7:NEXTX
300 HCOLOR8:FORX=44TO192STEP144:
HLINE(X,0)-(X+10,191),PSET,BF:NE
XTX:FORX=114TO258STEP144:HLINE(X
,0)-(X+10,191),PSET,BF:NEXTX:FOR
Y=44TO114STEP70:HLINE(0,Y)-(320,
Y+10),PSET,BF:NEXTY
305 HLINE(0,186)-(320,191),PSET,
BF
310 HPAINT(84,74),3,7:HPAINT(228
,74),3,7:FORX=20TO300STEP140:HPA
INT(X,20),3,8:HPAINT(X,170),3,8:
NEXTX
315 HCOLOR4:HLINE(42,0)-(44,191)
,PSET,BF:HLINE(124,0)-(126,191),
PSET,BF:HLINE(186,0)-(188,191),P
SET,BF:HLINE(268,0)-(270,191),PS
ET,BF:HLINE(0,42)-(320,44),PSET,
BF:HLINE(0,124)-(320,126),PSET,B
F:HLINE(0,186)-(320,188),PSET,BF
320 HCOLOR6:HLINE(84,0)-(228,191
),PSET,B:HLINE(0,84)-(320,84),PS
ET:HLINE(57,0)-(111,191),PSET,B:
HLINE(201,0)-(255,191),PSET,B:HL
INE(0,57)-(320,111),PSET,B
325 HCOLOR9:HLINE(79,0)-(89,191)
,PSET,B:HLINE(78,0)-(90,191),PSE
T,B:HLINE(223,0)-(233,191),PSET,
B:HLINE(222,0)-(234,191),PSET,B:
HLINE(0,79)-(320,89),PSET,B:HLIN
E(0,78)-(320,90),PSET,B
330 HCOLOR1:HLINE(54,0)-(114,191
),PSET,B:HLINE(198,0)-(258,191),
PSET;B:HLINE(0,54)-(320,114);PSE
T,B
335 HCOLOR0:HLINE(0,0)-(320,191)
,PSET,B
340 HCOLOR4:HLINE(88,0)-(224,10)
,PSET,BF:HCOLOR0:HPRINT(13,0),"R
OYAL STEWART"
345 ON BRK GOTO 360
350 GOTO 350
355 '******DRESS STEWART*******
    ***************************
360 PALETTE1,54:PALETTE2,24:PALE
TTE3,36:PALETTE4,10:PALETTE5,39:
PALETTE6,63:PALETTE7,3:PALETTE8,
0:PALETTE9,6
365 HCOLOR4:HLINE(0,0)-(320,191)
,PSET,B
370 FORX=20TO300STEP140:HPAINT(X
,170),6,4:HPAINT(X,20),6,4:NEXTX
375 HCOLOR0:HLINE(0,0)-(320,191)
,PSET,B
380 HCOLOR0:HLINE(88,0)-(224,10)
,PSET,BF:HCOLOR4:HPRINT(13,0),"D
RESS STEWART"
385 ON BRK GOTO 400
390 GOTO 390
395 '*****TARTAN NO6**********
    ************************
400 HCLS4:HCOLOR0
405 PALETTE0,0:PALETTE1,2:PALETT
E2,32:PALETTE3,37:PALETTE4,14:PA
LETTE5,6:PALETTE6,63:PALETTE7,55
:PALETTE11,35
410 FORX=24TO320STEP72:HLINE(X,0
)-(X,191),PSET:NEXTX:FORX=48TO32
0STEP72:HLINE(X,0)-(X,191),PSET:
NEXTX:FORY=48TO120STEP72:HLINE(0
,Y)-(320,Y),PSET:NEXTY:FORY=72TO
144STEP72:HLINE(0,Y)-(320,Y),PSE
T:NEXTY
415 FORX=26TO320STEP72:HPAINT(X,
10),2,0:HPAINT(X,82),2,0:HPAINT(
X,154),2,0:NEXTX:FORX=12TO320STE
P72:HPAINT(X,50),2,0:HPAINT(X,12
2),2,0:NEXTX:FORX=26TO320STEP72:
HPAINT(X,50),3,0:HPAINT(X,122),3
,0:NEXTX
420 FORX=12TO320STEP72:HLINE(X,0
)-(X,191),PSET:NEXTX:FORX=60TO32
0STEP72:HLINE(X,0)-(X,191),PSET:
NEXTX:
425 FORY=12TO190STEP72:HLINE(0,Y
)-(320,Y),PSET:NEXTY:FORY=36TO19
0STEP72:HLINE(0,Y)-(320,Y),PSET:
NEXTY
430 FORX=15TO320STEP36:HPAINT(X,
6),1,0:HPAINT(X,42),1,0:HPAINT(X
,78),1,0:HPAINT(X,114),1,0:HPAIN
T(X,150),1,0:HPAINT(X,186),1,0:N
EXTX435 FORX=8TO320STEP72:HPAINT
(X,6),5,0:HPAINT(X,42),5,0:HPAIN
T(X,78),5,0:HPAINT(X,114),5,0:HP
AINT(X,150),5,0:HPAINT(X,186),5,
0:NEXTX
440 FORX=18TO320STEP36:HPAINT(X,
24),5,0:HPAINT(X,96),5,0:HPAINT(
X,168),5,0:NEXTX
445 HCOLOR6
450 HLINE(0,60)-(320,61),PSET,BF
455 HLINE(108,0)-(109,191),PSET,
BF:HLINE(253,0)-(254,191),PSET,B
:HLINE(0,132)-(320,133),PSET,B:H
COLOR0:HLINE(102,0)-(103,191),PS
ET,B:HLINE(114,0)-(115,191),PSET
,B:HLINE(247,0)-(248,191),PSET,B
:HLINE(259,0)-(260,191),PSET,B:H
LINE(0,126)-(320,127),PSET
460 HLINE(0,138)-(320,139),PSET,
B:HLINE(33,0)-(34,191),PSET,B:HL
INE(39,0)-(40,191),PSET,B:HLINE(
177,0)-(178,191),PSET,B:HLINE(18
3,0)-(184,191),PSET,B:HLINE(0,53
)-(320,54),PSET,B:HLINE(0,66)-(3
20,67),PSET,B
465 HCOLOR7:HLINE(0,0)-(1,191),P
SET,B:HLINE(72,0)-(73,191),PSET,
B:HLINE(144,0)-(145,191),PSET,B:
HLINE(216,0)-(217,191),PSET,B:HL
INE(288,0)-(289,191),PSET,B:HLIN
E(0,24)-(320,25),PSET,B:HLINE(0,
96)-(320,97),PSET,B:HLINE(0,168)
-(320,169),PSET,B
470 HCOLOR5:HLINE(80,0)-(217,10)
,PSET,BF:HCOLOR6:HPRINT(13,0),"T
ARTAN NO4"
475 ON BRK GOTO 485
480 GOTO 480
485 HSCREEN2:PALETTE11,35:HCLS11
490 Y=10:Z=0:FORP=0TO10:C=RND(15
):PALETTEP,C:NEXTP:PALETTE11,35:
PALETTE12,0
500 FORX=10TO190STEP10:HCOLOR RN
D(10):HLINE(X,0)-(Z,Y),PSET:Y=Y+
10:NEXTX
505 FORX=200TO320STEP10:HCOLOR R
ND(10):HLINE(X,0)-(X-190,191),PS
ET:NEXTX
510 Y=10:FORX=140TO320STEP10:HCO
LOR RND(10):HLINE(X,191)-(320,Y)
,PSET:Y=Y+10:NEXTX
515 Y=180:Z=0:FORX=10TO190STEP10
:HCOLOR RND(10):HLINE(X,191)-(Z,
Y),PSET:Y=Y-10:NEXTX
520 FORX=200TO320STEP10:HCOLOR R
ND(10):HLINE(X,191)-(X-191,0),PS
ET:NEXTX
525 Y=180:FORX=140TO320STEP10:HC
OLOR RND(10):HLINE(X,0)-(320,Y),
PSET:Y=Y-10:NEXTX
530 HCOLOR11:HLINE(130,90)-(197,
75),PSET,BF:HCOLOR12:HPRINT(17,1
0),"THE END"
535 FORP=0TO15:C=RND(63):PALETTE
P,C:NEXTP:FORP=0TO15:C=RND(32):P
ALETTEP,C:NEXTP
540 ON BRK GOTO 545
543 GOTO 535
545 PALETTECMP:POKE65496,0:END
```

Password

Keeping your treasures from unwanted 'guests'.

by Fred Remin

16K ECB
UTILITY

PASSWORD IS A short program found by accident while playing around with my CoCo 3. I meant to ...

POKE280,PEEK(275)

... but misjudged and accidently POKE'd location 288 instead.

The result was a complete crash of the CoCo, as it would show a syntax error on almost anything I typed in, even when the command was right.

The only way to get out of it was to turn the CoCo off and back on again.

The result of further investigation reveals that you can place a few lines in the beginning of your program (which is a very effective protection), and utilize the 'ON BRK' command of the CoCo 3.

If you know of a POKE or a PEEK that will disable the break key of the CoCo 2, it could be used for the 1 and 2 as well.

I have placed the lines in front of a small program I have to illustrate its created effectiveness.

Remember, the only way out is to turn the CoCo off and on again.

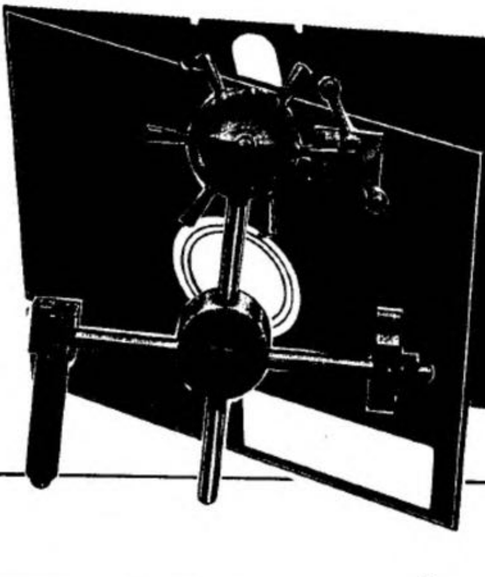
The password I have used in this case is the letter "W", however, you could use any letter on the keyboard including one of the function keys. Also to confuse the would-be program flogger, I have used the screen disable poke (359,255).

The only way the protect would not work is if the person had the habit of LISTing any program first before running it (doubtful).

Anyway here it is and I hope someone finds a use for it.

The Listing:

```
0 GOTO10
1 '***** PROTECT
2 '***** FRED REMIN
3 SAVE"86:3":END'7
10 CLS
20 PRINT"THIS IS A PROTECTED PRO
GRAM"
21 ONBRK GOTO40
25 PRINT"WHAT IS THE PASSWORD???
"
30 PRINT"YOU HAVE THREE CHANCES
ONLY"
35 PRINT"BEFORE I SELF DESTRUCT"
36 FOR X=1TO3000:NEXT
37 CLS:POKE359,255
40 P$=INKEY$:IF P$=""THEN40
45 IF P$<>"W"THEN60ELSE100
60 POKE288,PEEK(275):SOUND150,1:
CLSRND(8):GOTO60
100 POKE359,126:
101 GOTO140
110 '******ALLEYS******
120 '**BY FRED REMIN***
130 SAVE"ALLEYS",A
140 CLS(RND(8))
150 ONBRK GOTO500
160 PRINT"THIS GAME IS THE SAME
AS YOU      USED TO PLAY AT THE CA
RNIVAL      WHEN YOU ROLLED THE BA
LLS         INTO THE CLOWNS MOUTH
AND         CAME UP WITH A SCORE.T
HE ONLY     DIFFERENCE HERE IS THA
T IT        WON'T COST YOU A CENT.
"
170 FOR X=1TO8000:NEXT X
180 CLS
190 INPUT "HOW MANY TRIES DO YOU
 WANT";NT
200 CLS:INPUT"PRESS ENTER TO BEG
IN";RT$
210 CLS:WI=0
220 PRINT"ROUND#","ALLEYS"
230 FOR CO=1TO NT
240 PRINTCO,
250 SU=0
260 FOR B=1TO4
270 A=INT(1*RND(4))
280 PRINTA;" ";
290 SU=SU+A
300 NEXT B:SOUND150,1
310 IF SU<=30AND SU>=10THEN330
320 PRINT"*WIN*";:WI=WI+1:PLAY"T
100;G;F;E;B"
330 PRINT
340 P=WI/CO
350 NEXT CO
360 PRINT"WON";WI;"OUTOF";NT;"TR
IES"
370 PRINT"WINNING PERCENTAGE OF"
P*100;"%":FOR X=1TO1500:NEXT X
380 CLS(RND(8)):PRINT@260,"ANOTH
ER GAME (Y/N)
390 I$=INKEY$:IF I$=""THEN390
400 IF I$="Y"THEN180 ELSE IFI$="
N"THEN PRINT"I HOPE YOU ENJOYED
THE GAME":FOR X=1TO1000:NEXT X:C
LS(RND(5))
410 PRINT@260,"MAYBE YOU WILLHAV
E BETTER"
420 PRINT@292,"LUCK NEXT TIME...
?"
500 EN
```



Headings II

This great program sets new borders for the CoCo!

32K ECB
UTILITY

by Michael Shoobridge

MY PROGRAM MAY BE used on a CoCo 3 ... or 2 or 1. It makes use of strings to put colourful bordars around a menu or other headings.

The print instructions (line 470) are long and detailed to show how this is wrapped around the two lines of print. Be careful typing this out. There are 3 "Q"'s and 1 "O", followed by an "A", "B" & "C" at the end, not 4 "Q"'s.

When designing a different format, it is often easier to give "PRINT@" statements for the two lines of print, rather than wrapping them up in a single string, as here. The "PRINT@" statement may be removed later.

The Listing:

```
0 '****************************
   *  ABBREVIATED VERSION OF   *
   *        'HEADINGS'         *
   *            by             *
   *    MICHAEL SHOOBRIDGE     *
   * (COMPARE LINE 470 IN EACH)*
   ****************************
1 GOTO10
3 SAVE"47A:3":END'7
4 '
5 ' ABBRHEAD/BAS  MULTIPLE MENUS
 WITH ONE OR SEVERAL COLOURED SU
RROUNDS  MPK SHOOBRIDGE  14 SEPT
 1987 23 NAUGHTON GROVE BLACKBUR
N 3130 VICTORIA
10 CLEAR 2000
20 ON BRK GOTO 30: 'FOR COCO 3 O
NLY. DELETE FOR COCO 1 AND 2
30 S1$="COLOURFUL MARGINS DEMO":
   S2$="<P>  TO PRINT DETAIL"
300 P1=LEN(S1$): P2=(24-P1)/2: P
3=INT((24-P1)/2): IF P2>P3 THEN
P2=P3+1
310 P2$=STRING$(P2,143+16*4): P3
$=STRING$(P3,143+16*4)
320 Q1=LEN(S2$): Q2=(24-Q1)/2: Q
3=INT((24-Q1)/2): IF Q2>Q3 THEN
Q2=Q3+1
330 Q2$=STRING$(Q2,143+16*4): Q3
$=STRING$(Q3,143+16*4)
340 X1$=STRING$(24,143+16*4)
350 X=127
370 C1=RND(7):     C2=RND(7):
    C3=RND(7):     C4=RND(7):
    C5=RND(8):'          1=YELLOW
 2=BLUE     3=RED        4=BUFF
 5=CYAN     6=MAGENTA  7=ORANGE
 OR SUBSTITUTE FIXED NUMBER (1-7
)    IN C1/C2/C3/C4/C5
380 E=X+RND(16): F=X+RND(16): G=
X+RND(16): H=X+RND(16): '128-143
 -> VERTICAL     = 133/138
    HORIZONTAL = 131/140
    PLAIN        = 128/143
    CHECKER      = 134/137
    L SHAPES= 129/130/132/136
    REVERSE"= 135/139/141/142
390 CNT=CNT+1: IF CNT>3 THEN E=F
400 IF CNT>5 THEN G=E
410 IF CNT>7 THEN H=E
420 IF CNT>9 THEN CNT=0
430 A$=STRING$(2,E+16*C1):
    B$=STRING$(2,F+16*C2):
    C$=STRING$(2,G+16*C3):
    D$=STRING$(2,H+16*C4)
440 K$=A$+B$:       L$=C$+D$:
    O$=A$+B$+C$+D$
450 Q$=A$+B$+C$+D$+A$+B$+C$+D$
460 CLS(C5)
470 PRINT Q$Q$Q$Q$;B$C$;X1$;D$A$
B$C$;P2$S1$P3$;D$A$C$D$;Q2$S2$Q3
$;A$B$C$D$;X1$;A$B$D$;Q$Q$Q$O$A$
B$C$
480 PRINT@320,"colours    C1="C1
" C2="C2" C3="C3" LINE 370   C4=
"C4" C5="C5" C6="C6:PRINT@384,"c
haracters E="E"F="F"G="G"LINE 38
0   H="H;"       count=";CNT
490 AN$=INKEY$::IF AN$=""THEN490
ELSE IF AN$="P"THEN520
500 MK=7:S1$="COLOURFUL MARGINS
DEMO": S2$="<P>  TO PRINT OUT DA
TA":' MK=7 SENDS PROGRAM TO RND(
7) OR RANDOM COLOURS (LINES 300-
550
510 GOTO300
520 CLS:INPUT"IS THE PRINTER ONL
INE ?            <ENTER> WHEN READY
";R$
530 PRINT:PRINT"PRINTING THE SET
TINGS REQUIRED":PRINT#-2,"THE BO
RDER from your ABBRHEAD/BAS prog
ram was made up as follows:-
540 PRINT#-2,"COLOURS:-      C1="
C1"    C2="C2"    C3="C3"    C4="C4
"    C5="C5"    C6="C6
550 PRINT#-2,"CHARACTERS:-  E="E
"  F="F"  G="G"  H="H"  and COUN
T="CNT:PRINT#-2,"COUNT=1-3 (rand
om)  CNT=4-5 (2 Chars same 2 dif
ferent)  CNT=6-7 (3 chars same)
    CNT=8-9 (all chars same).":P
RINT#-2:PRINT#-2:CLS:GOTO500
```

Hang on Snoopy

Just a few swift 'linus' and....bingo!

by Nick Kostarelas

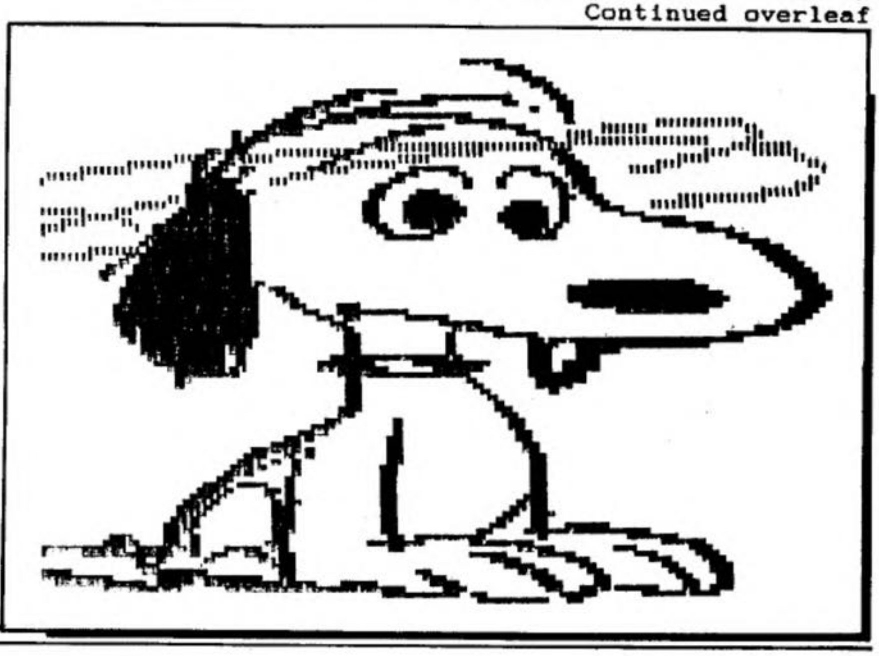
16K ECB GRAPHICS

HERE IS ONE OF my favourite cartoon characters - Snoopy. He seems to be quite dizzy for some reason. After he is drawn, he says something.

The Listing:

```
0 GOTO90
3 SAVE"93A:3":END'5
10 '****************************
20 '*                          *
30 '*          SNOOPY          *
40 '*                          *
50 '*   BY NICK KOSTARELAS     *
60 '*        20/11/83          *
70 '*                          *
80 '****************************
90 POKE65495,0:CLS:PMODE0,1:PCLE
AR4
100 PMODE4,1:PCLS:SCREEN1,1
110 FORY=0TO186STEP2
120 READNUMBER
130 FORI=1TO NUMBER
140 READX,D
150 FORJ=0TO1
160 LINE(X,Y+J)-(X+D*2,Y+J),PSET
170 NEXTJ,I,Y
180 DATA 1,134,6
190 DATA 1,144,4
200 DATA 1,148,3
210 DATA 1,152,3
220 DATA 1,154,3
230 DATA 2,100,7,158,2
240 DATA 3,96,4,122,13,160,2
250 DATA 4,90,5,110,8,146,2,162,
2
260 DATA 4,84,5,102,5,154,1,162,
2
270 DATA 3,82,4,94,21,164,1
280 DATA 4,78,4,92,5,132,7,164,1
290 DATA 3,76,8,118,5,142,5
300 DATA 4,66,1,72,7,112,5,150,3
310 DATA 4,66,1,70,3,108,3,156,2
320 DATA 2,66,3,104,2
330 DATA 4,58,2,64,3,100,2,160,2
340 DATA 4,56,2,64,2,96,2,160,3
350 DATA 4,54,2,62,2,92,2,162,3
360 DATA 7,54,2,60,2,68,2,88,2,1
18,7,148,4,164,3
370 DATA 8,54,1,58,2,68,1,114,3,
130,2,146,1,154,3,164,3
380 DATA 6,54,7,112,2,134,1,144,
1,160,1,166,3
390 DATA 6,52,7,108,2,134,1,144,
1,160,2,166,3
400 DATA 7,52,8,70,1,106,2,120,3
,124,1,162,1,168,2
410 DATA 5,52,9,106,2,118,6,162,
2,170,2
420 DATA 6,52,9,104,2,116,8,148,
4,162,1,172,2
430 DATA 6,50,10,104,2,116,9,146
,6,162,1,174,3
440 DATA 6,48,11,104,2,116,9,144
,7,162,1,180,3
450 DATA 6,46,12,104,2,116,8,144
,7,162,1,186,6
460 DATA 7,42,14,106,2,118,3,126
,2,146,6,160,1,192,7
470 DATA 6,42,14,108,2,124,2,148
,4,158,1,202,5
480 DATA 4,40,15,110,7,150,5,210
,3
490 DATA 3,38,1,42,14,214,4
500 DATA 2,40,15,218,4
510 DATA 3,34,1,38,16,220,4
520 DATA 3,32,1,38,16,222,4
530 DATA 3,28,2,36,17,224,4
540 DATA 3,26,2,36,17,226,4
550 DATA 3,26,1,34,18,228,4
560 DATA 4,34,18,72,1,170,15,230
,3
570 DATA 4,32,20,76,2,164,20,230
,4
580 DATA 4,32,20,78,2,164,22,232
,4
590 DATA 4,32,20,80,2,164,23,232
,4
600 DATA 5,30,21,82,2,96,3,168,2
0,230,4
610 DATA 5,30,21,86,2,96,10,178,
13,228,4
620 DATA 5,30,21,90,2,98,2,116,6
,226,4
630 DATA 4,30,21,94,2,126,6,224,
4
640 DATA 5,32,20,96,2,130,1,134,
5,218,5
650 DATA 6,32,2,38,17,98,2,128,1
,142,6,210,7
660 DATA 6,34,1,38,16,98,2,128,1
,152,2,158,28
670 DATA 5,38,16,98,2,128,1,152,
4,166,19
680 DATA 6,38,12,64,2,98,2,128,1
,152,3,166,6
690 DATA 7,40,1,44,9,66,1,94,17,
130,1,152,3,166,3
700 DATA 6,42,10,66,1,90,8,118,1
1,152,3,162,5
710 DATA 6,46,8,64,2,90,10,124,7
,152,3,162,5
720 DATA 3,48,9,96,19,152,8
730 DATA 4,48,2,98,2,134,1,154,7
```

Continued overleaf



Mini Graphics Editor

It's oh, so easy!

32K ECB

by Shannon Glenn UTILITY

THIS PROGRAM was written to show how easy it is to draw easy pictures and be able to save them to tape (and reload) at any time!!

Further instructions are in the program.

I would like to see some printer routines to produce hardcopies of the screen.

If anybody knows some, write to:

Shannon Glenn,
57 Pacific Haven,
Howard, 4651

There is a demonstration file at the end of the program on this month's CoCoOz.

The Listing:

```
0 GOTO10
1 '***** MINI GRAPHICS EDITOR
2 '***** SHANNON GLEN
3 SAVE"109:3":END'7
10 CLS3:PRINT@13,"ONE MOMENT PLE
ASE..";
20 CLOADM"ZOOM"
30 DEFUSR=&H3F00
40 CLS0
50 PRINT@74,"mini";:PRINT@79,"gr
aphics";:PRINT@109,"editor";:PRI
NT@143,"by";:PRINT@169,"shannon"
;:PRINT@177,"c";:PRINT@179,"glen
";
60 PLAY"L255;ABDCABDCABDCABDCABD
CABDC"
70 PLAY"L12;ABABACDEF"
80 FORX = 1 TO 1000
90 NEXT
100 CLS
110 INPUT"PMODE??";G:CLS
120 PRINT@5,"<<< INSTRUCTIONS >>
>"
130 PRINT:PRINT"U           UP"
140 PRINT      "D           DOWN"
150 PRINT      "L           LEFT"
160 PRINT      "R           RIGHT"
170 PRINT:PRINT"C           COLOR C
HANGE"
180 PRINT      "N           NEW PAG
E"
190 PRINT      "E           ENLARGE
 TOP QUADRANT"
200 PRINT      "S           SAVE"
210 PRINT      "X           LOAD"
220 PRINT:PRINT@453,"<< ANY KEY
TO CONT >>"
230 I$=INKEY$
240 IF I$=""THEN 230
250 PMODEG:PCLS:SCREEN1,1
260 RESTORE
270 X=128:Y=96
280 I$=INKEY$
290 IF I$="U" THEN Y=Y-1
300 IF I$="D" THEN Y=Y+1
310 IF I$="L" THEN X=X-1
320 IF I$="R" THEN X=X+1
330 IF I$="N"THEN 250
340 IF I$="C" THEN C=C+1
350 IF I$="E"THEN A=USR(0)
360 IF C>8 THEN C=0
370 IF I$="S"THEN 410
380 IF I$="X"THEN 430
390 PSET(X,Y,C)
400 GOTO 280
410 CSAVEM"PICFILE",1536,7679,0
420 GOTO 250
430 CLOADM"PICFILE":GOTO 280
```

Continued from previous page

```
740 DATA 4,48,1,98,2,136,1,154,5
750 DATA 3,98,2,138,1,158,1
760 DATA 2,98,2,140,1
770 DATA 2,98,2,142,1
780 DATA 2,96,3,142,2
790 DATA 2,96,2,144,2
800 DATA 3,92,2,112,1,146,2
810 DATA 3,88,3,112,1,146,2
820 DATA 3,84,4,112,1,148,2
830 DATA 3,80,4,110,2,148,2
840 DATA 3,76,4,110,2,150,2
850 DATA 5,70,5,82,1,86,1,110,2,
150,2
860 DATA 5,68,3,80,1,86,1,110,2,
152,2
870 DATA 5,66,3,76,1,82,1,110,2,
152,2
880 DATA 5,64,3,72,1,82,1,108,2,
152,2
890 DATA 5,62,2,68,1,82,1,108,2,
152,2
900 DATA 4,60,2,80,1,108,2,152,2
910 DATA 5,56,3,66,4,80,1,108,2,
152,2
920 DATA 6,54,3,62,1,74,2,80,1,1
08,2,152,2
930 DATA 6,52,3,60,1,74,1,80,1,1
08,2,150,3
940 DATA 6,50,3,58,1,76,1,80,2,1
08,2,148,4
950 DATA 6,48,3,56,1,76,3,108,2,
146,2,152,2
960 DATA 6,48,2,54,1,76,3,108,2,
144,2,152,3
970 DATA 6,46,2,52,1,76,3,108,2,
142,2,152,2
980 DATA 7,44,3,52,1,76,3,108,2,
140,2,152,2,162,6
990 DATA 6,44,2,50,1,76,3,108,2,
136,2,150,18
1000 DATA 8,28,3,44,2,52,1,76,2,
110,2,118,10,154,1,180,7
1010 DATA 8,26,3,34,1,44,2,52,1,
74,3,104,8,134,9,190,6
1020 DATA 9,4,14,36,7,52,1,62,5,
76,2,110,2,148,5,176,3,198,3
1030 DATA 10,0,4,20,13,52,1,58,2
,68,3,78,2,134,5,154,5,182,3,200
,3
1040 DATA 10,0,1,34,2,42,1,56,1,
72,2,78,2,140,4,162,3,186,3,202,
3
1050 DATA 7,0,2,36,3,76,3,144,4,
166,2,190,2,204,3
1060 DATA 8,2,7,38,9,78,2,118,2,
148,4,168,2,192,2,204,3
1070 DATA 8,6,11,42,15,78,2,120,
3,152,3,170,2,194,2,206,2
1080 DATA 10,12,13,42,5,56,11,80
,4,96,5,122,4,154,3,170,2,196,2,
206,2
1090 DATA 10,26,11,62,11,86,7,10
2,6,126,3,156,2,166,4,176,6,196,
2,204,3
1100 DATA 5,108,11,134,2,160,5,1
84,6,198,4
1110 DATA 2,136,10,162,2
1120 FORY=20TO66STEP2
1130 READNUMBER
1140 FORI=1TONUMBER
1150 READX,D
1160 FORJ=X TO X-2+D*2STEP2
1170 LINE(J,Y)-(J,Y+2),PSET
1180 NEXTJ,I,Y
1190 DATA 1,191,13
1200 DATA 2,175,7,217,2
1210 DATA 2,165,4,219,2
1220 DATA 2,165,21,217,2
1230 DATA 2,109,29,213,9
1240 DATA 4,79,11,125,8,205,4,21
9,7
1250 DATA 5,49,4,69,5,115,5,197,
5,231,2
1260 DATA 4,35,7,101,7,191,4,235
,2
1270 DATA 4,21,8,93,4,183,5,239,
1
1280 DATA 2,11,6,83,4
1290 DATA 3,7,2,77,3,235,2
1300 DATA 2,3,2,231,3
1310 DATA 2,1,2,221,6
1320 DATA 3,1,1,47,2,197,13
1330 DATA 3,1,1,37,5,189,8
1340 DATA 2,3,10,31,3
1350 DATA 1,21,6
1360 DATA 2,15,4,33,2
1370 DATA 2,7,5,37,1
1380 DATA 1,5,3
1390 DATA 1,3,2
1400 DATA 1,1,1
1410 DATA 2,1,1,23,8
1420 DATA 1,3,11
1430 POKE65494,0
1440 FORI=1TO1300:NEXT
1450 PRINT@170,"SNOOPY SAYS:":PR
INT@227,"NEVER TRUST A SMILING C
AT."
1460 FORI=1TO2000:NEXT:CLS:END
65535 'NICK KOSTARELAS 1983
```

Colour Scheme

254 colours on a PMODE 3 or 4 screen.

by Brian Hainworth

**32K ECB
UTILITY**

G'DAY! I'M A year 10 student (15 years of age), who has just completed my first program which I think will be good enough for your magazine.

My program is a utility which shows off 254 colours on a PMODE 3 or 4 screen in either SCREEN 1,1 or SCREEN 1,0.

The idea hit me when I was browsing through the October '87 Softgold edition.

I was reading the instructions for John Baker's Dire Straits program, when I came across the part which explains how to get the different colours with the command - POKE 178,(1 to 255).

I then wrote this program.

NOTE: If CoCo users like a poke they see, they should disregard any leading zeros before the prime number when using it in their program.

For those CoCo's which cannot handle the high speed poke, delete lines 120 and 132 before running/saving.

HINT:

If you accidentally save a program in high speed poke, then when you try to load it, it doesn't work.
A simple solution is -
TYPE :

```
(for coco 2) -  POKE 65495,0:
                (C)LOAD"FILENAME"

(for coco 3) -  POKE 65497,0:
                (C)LOAD"FILENAME"
```

The Listing:

```
0 GOTO8
1 '***** COLOUR SCHEME
2 '***** BRIAN HAINSWORTH
3 SAVE"96:3":END'7
8 GOSUB 5000
9 GOSUB 7000
10 CLS
15 INPUT"DO YOU WISH AUTO OR MAN
UAL       DISPLAY  [A/M]";Z$
16 IF Z$="A" THEN V=1 ELSE IF Z$
="M" THEN V=2
19 CLS
20 INPUT" WHICH PMODE DO YOU WIS
H TO SEE               3 OR 4";Q
25 IF Q<3 OR Q>4 THEN 10
30 IF Q=3 THEN W=3 ELSE IF Q=4 T
HEN W=4
35 IF W=3 THEN U=4 ELSE IF W=4 T
HEN U=3
40 CLS:INPUT"WHICH SCREEN DO YOU
 WISH TO SEE 1 = 1,1 OR 2 = 1,0"
;E
45 IF E<1 OR E>2 THEN 40
50 IF E=1 THEN S=1 ELSE IF E=2 T
HEN S=0
60 CO=0
70 P=0
99 PMODE W,1:PCLS:SCREEN1,S:PMOD
EU
100 PCLS
120 K=65495
130 GOSUB 6000
132 POKE K,0
135 P=P+1
136 PMODE4
140 POKE 178,P
145 CO=CO+1
146 IF CO=>255 OR P=>255 THEN CL
S:PRINT"SORRY, YOU HAVE EXCEDED
THE       MAXIMUM COLOURS VISIBLE.
........RERUN THE PROGRAM FOR AN
OTHER   SELECTION":END ELSE 150
150 LINE(90,20)-(170,100),PSET,B
F
160 CO$=STR$(CO)
170 L1=VAL(RIGHT$(CO$,1))
180 L2=VAL(RIGHT$(CO$,2))
190 L3=VAL(RIGHT$(CO$,3))
200 L3=L3/100:L2=L2/10
205 DRAW"BM82,156C1U12R6F2D4G2L6
BR12BD4U12R8D12NL8BR4U6NU6R2NE6F
6BR4NR8U6NR4U6R8BR12D8ND4R6E2U4H
2L6BR16BD10D2G2"
210 COLOR2:LINE(164,156)-(204,17
2),PSET,BF:COLOR1:DRAW"BM164,156
"+A$(L3)+A$(L2)+A$(L1)
220 IF V=1 THEN FOR T=1 TO 500:N
EXT ELSE 4000
4000 GOTO 6020
5000 CLS0
5005 J=0
5006 J=J+1
5007 IF J=8 THEN RETURN
5008 CLS J
5010 PRINT@165,"the coco colour
scheme";
5020 PRINT@239,"by";:PRINT@296,"
brian hainsworth";
5030 FOR T=1 TO 900:NEXT
5040 GOTO 5006
6000 A$(1)="R2U12G4BF8NL4BR4"
6001 A$(2)="E8U2H2L2G2D2BF8NL8BR
4"
6002 A$(3)="BU4D2F2R4E2U2H2NL2E2
U2H2L4G2D2BD8BR12"
6003 A$(4)="BR6U6NU4NR2L6U6BD12B
R12"
6004 A$(5)="BU2F2R4E2U4H2L2G2L2U
6R8BD12BR4"
6005 A$(6)="BU12BR8BD2H2L4G2D8F2
R4E2U4H2L4G2BR8BD6BR4"
6006 A$(7)="U4E8L8BR8BD12BR4"
6007 A$(8)="BR2R4E2U2H2E2U2H2L4G
2D2F2G2D2F2BR10"
6008 A$(9)="BR8U10H2L4G2D2F2R4E2
BD8BR4"
6009 A$(0)="BR2R4E2U8H2L4G2D8F2B
R12"
6010 RETURN
6020 IF V=1 THEN 100 ELSE IF V=2
 THEN EXEC44539:GOTO 100
7000 CLS:PRINT"THE COCO CAN PROD
UCE MORE THAN  FOUR.COLOURS ON P
MODE'S 3 OR 4. TO SEE THE 254 CO
LOURS , YOU    MUST CHOOSE YOUR
PMODE (3 OR 4) THEN CHOOSE A SCR
EEN (1,1 OR 1,0).THEN THE COCO W
ILL PRODUCE A   GRAPHIC SCREEN I
N THAT            ";
7010 PRINT"COMBINATION AND YOU M
AY PRESS    ANY KEY TO SEE THE NE
XT           VARIATION OF COLOUR."
7020 PRINT:PRINT"ANY KEY TO STAR
T":SCREEN0,1
7030 EXEC 44539:RETURN
```

Disk Label Maker

This is a sticky one!

by Jason Hall

CoCo2 with slashes not dashes!

THIS IS A disk label maker program, designed for stickers measuring 25mm x 70mm. All the instructions are contained within the program.

But just to whet your appetite, the program will eventually ask you to choose from a menu of four options.

Option one, "Main Program Disk", is for those programs such as databases, spreadsheet programs, etc; ie, those programs using the entire disk.

Option two, "Data Disk", basically supports the first option, or the result of the "Main Program Disk".

Option three, "Game Disk", is self-explanitory. All it will ask is the type of game it is, eg Adventure, Graphic, etc.

Option four, "Utilities Disk", is again, self-explanitory.

```
0 GOTO10
1 '***** LABELLER
2 '***** JASON HALL
3 SAVE"74A:3":END'7
10 POKE 359,57:POKE65314,50
20 CLS0:FORI=1TO 256 :READ A:PRI
NTCHR$(A+128);:NEXT
30 DATA,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,
40 DATA16,,,112,,,,,31,28,27,,20
,31,24,,31,28,28,,31,17,30,,,,,,
,,,
50 DATA,,,124,124,124,120,,31,16
,31,,16,31,16,,28,28,31,,31,29,1
8,,124,124,124,120,,,,
60 DATA,,,,,,,,28,28,24,,20,28,2
4,,28,28,28,,28,16,28,,,,,,,,,
70 DATA31,16,16,,23,28,27,,31,28
,27,,31,28,28,,31,16,16,,31,16,1
6,,31,28,28,,31,28,31,
80 DATA31,16,16,,31,28,31,,31,28
,27,,31,28,28,,31,16,16,,31,16,1
6,,31,28,28,,31,29,18,
90 DATA28,28,28,,28,16,28,,28,28
,24,,28,28,28,,28,28,28,,28,28,2
8,,28,28,28,,28,16,28,
100 DATA12,12,12,12,12,12,12,12,
12,12,12,12,12,12,12,12,12,12,12
,12,12,12,12,12,12,12,12,12,12,1
2,12,
110 A=PEEK(65314)
120 IF A=49 THEN GOSUB 630
130 IF A=48 THEN 140
140 PRINT@296,"By   J & J  Hall.
"
150 PRINT@330,"Pakenham Vic"
160 PRINT@417,"Do you want Instr
uctions   Y/N";
170 YN$=INKEY$:IF YN$="" THEN 17
0
180 IF YN$="Y" GOTO 190 ELSE 710
190 CLS
200 PRINT"This disk label maker
programme was designed for use w
ith self  adhesive labels 25mm X
 70mm.":PRINT""
210 PRINT"It is intended to iden
tify your disks so that you can
keep trackof the programmes in y
our libary":PRINT""
220 PRINT"This self explanatory
programme is easy to use and has
 built in pronpts."
230 GOSUB 590
240 CLS
250 PRINT@6,"GENERAL INSTRUCTION
S":PRINT""
260 PRINT"1. Press <ENTER> after
 every          input from the key
board.
270 PRINT"2. The maxium number o
f charact-   ers for the first i
nput on      every screen is (12
).
280 PRINT"3. All other inputs ca
n have up    to (40) characters.
290 PRINT"4. Each label will be
printed as   soon as the last in
put has      been entered.
300 PRINT"5. The programme will
not run if   the printer is not
connected    and turned on."
310 GOSUB 590
320 CLS
330 PRINT@10,"SELECTION  1"
340 PRINT@70,"Main Programme Dis
k":PRINT""
350 PRINT"This is for a Main Pro
gramme, EGDATABASE, CALC etc. Ju
st enter  the Programme name, It
 will thenask you for loading in
structionssuch as LOAD(M)"
360 GOSUB590
370 CLS
380 PRINT@10,"SELECTION  2"
390 PRINT@75,"Data Disk":PRINT""
400 PRINT"The first thing that y
ou will beasked is (Data for whi
ch progra-mme) EG: DATABASE, it
will then ask you for the (Title
)  Co Co  Club, the last questio
n is (Datawhich could be CLUB ME
MBERS"
410 GOSUB 590
420 CLS
430 PRINT@10,"SELECTION  3"
440 PRINT@75,"Games Disk":PRINT"
"
450 PRINT"If you have a lot of G
AMES disksthis will be handy, fi
rst of allit will print (GAMES)
on the topline of the label, it
will thenask you for the Type of
 Games EGAdventure, Educational
etc, it   will then ask you for
instruct-"
460 PRINT"ions which could be Jo
ysticks,  Keyboard, 10 year old
games etc."
470 GOSUB 590
480 CLS
490 PRINT@10,"SELECTION  4"
500 PRINT@73,"Utilities Disk":PR
INT""
510 PRINT"This will ask you what
 Type of  Utilities is on the Di
sk EG:     Disk, Screen, Tape, Ge
neral etc,it will then ask you F
or use     with, this can be anyt
hing you  want to use it with."
520 GOSUB 590
530 CLS
540 PRINT@10,"SELECTION 5"
550 PRINT@78,"Exit":PRINT""
560 PRINT"Entering number 5 will
  coldstartthe computer and retur
n you to  basic."
570 GOSUB 590
580 GOTO 710
590 PRINT@484,"PRESS ENTER TO CO
NTINUE";
600 EXEC44539
610 RETURN
620 END
630 PRINT@486,"PRINTER IS NOT RE
ADY";
```

Continued on P 53

Cassette Labeller

Hey! What do 'blank strings' sound like?

by Barry Sidebottom

CoCo3
UTILITY

CASSETTE LABELLER IS written for the CoCo 3 and is designed to print out labels for your audio cassettes. It produces two columns of 20 characters plus the ID flap.

The program is self explanatory

One note however, when viewing the finished product on the screen, do not be alarmed if odd characters appear on blank lines. For some reason the CoCo 3 does this with blank strings. Why, I don't know. Perhaps one of our more knowledgable programmers could tell us.

Anyway they won't print out, so they're not a real problem.

Hope you can use it, and if you have any problems, give me a call (03) 744-6281

The Listing:

```
0 GOTO20
1 ' ====================
2 '   CASSETTE LABELLER
3 '          by
4 '   BARRY  SIDEBOTTOM
5 ' --------------------
6 '     version 1.1(3)
7 '        (c) 1987
8 ' ====================
9 '
10 SAVE"107A:3":END'7
20 CLEAR5000:DIM E$(2,20):POKE65
497,0
21 ON BRK GOTO144
22 S$(1)="Side 1":S$(2)="side 2"
23 HSCREEN2:A=2:B=4:C=3:D=5:E=8
24 LN=1:X=X+1:IF X>2THEN52ELSE G
OSUB25:GOTO38
25 HCOLOR A,B:HCLS
26 HLINE(5,5)-(180,186),PSET,B:H
LINE(185,5)-(314,186),PSET,B:HLI
NE(185,150)-(314,150),PSET
27 HCOLOR C:HPRINT(1,1),S$(X)
28 HCOLOR D:HPRINT(24,1),"INSTRU
CTIONS:":HCOLOR E:HLINE(194,16)-
(299,16),PSET:HLINE(194,18)-(299
,18),PSET
29 HCOLOR D:HPRINT(24,3),"17 LIN
ES OF 20":HPRINT(24,4),"CHARACTE
RS ARE"
30 HPRINT(24,5),"AVAILABLE FOR":
HPRINT(24,6),"YOUR ENTRIES."
31 HPRINT(24,8),"ENTER AS YOU":H
PRINT(24,9),"NORMALLY WOULD"
32 HPRINT(24,10),"USING ALL TEXT
":HPRINT(24,11),"AVAILABLE."
33 HCOLOR E:HPRINT(24,13),CHR$(9
5):HCOLOR D:HPRINT(25,13),": back
spaces."
34 HPRINT(24,15),"'":HCOLOR E:HP
RINT(25,15),"@@@":HCOLOR D:HPRIN
T(28,15),"':on last":HPRINT(30,1
6),"line to":HPRINT(30,17),"term
inate"
35 HCOLOR A:HPRINT(24,20),"curre
nt status":HPRINT(26,21),"side :
":IF X>2THEN37ELSE HPRINT(26,22)
,"line :"
36 HCOLOR C:HPRINT(33,21),X:HPRI
NT(33,22),1
37 LN=3:RETURN
38 RW=20:CL=1:XX=1
39 FOR Y=1TO17:HCOLOR B:HPRINT(3
3,22),LN-3:HCOLOR C:HPRINT(33,22
),LN-2:GOSUB40:NEXT Y:GOTO48
40 SOUND220,1
41 HCOLOR A:IF LEN(E$(X,Y))=0THE
N HPRINT(XX,LN),CHR$(127)ELSE HP
RINT(XX-1,LN),Z$+CHR$(127)
42 Z$=INKEY$:IF Z$=""THEN42
43 IF Z$=CHR$(13)THEN49ELSE IF Z
$=CHR$(8)THEN50
44 IF XX>RW THEN42ELSE XX=XX+1
45 HCOLOR B:HPRINT(XX-1,LN),CHR$
(127)
46 E$(X,Y)=E$(X,Y)+Z$:IF E$(X,Y)
="@@@"THEN48ELSE41
47 XX=1:LN=LN+1:RETURN
48 IF Q>0THEN RETURN ELSE E$(X,Y
)="":XX=1:A=3:C=2:GOTO24
49 HCOLOR B:HPRINT(XX,LN),CHR$(1
27):GOTO47
50 HCOLOR B:HPRINT(XX,LN),CHR$(1
27):XX=XX-1:IF XX<CL THEN XX=CL:
GOTO51ELSE51
51 IF LEN(E$(X,Y))=0THEN41ELSE L
=LEN(E$(X,Y)):HPRINT(XX,LN),RIGH
T$(E$(X,Y),1):E$(X,Y)=LEFT$(E$(X
,Y),L-1):GOTO41
52 Q=1:PALETTE0,0
53 HSCREEN4:HCOLOR8,1:HCLS
54 HPRINT(1,1),"TITLE FLAP."
55 HLINE(10,30)-(350,50),PSET,B
56 HLINE(320,41)-(347,48),PSET,B
F
57 HPRINT(22,7),CHR$(94):HPRINT(
1,8),"INSTRUCTIONS:"
58 HPRINT(1,10),"Enter the title
 flap. This is done as a":HPRINT
(1,11),"complete entry rather th
an 2 seperate":HPRINT(1,12),"sid
es."59 HPRINT(1,14),"The boxed i
n area is reserved for the":HPRI
NT(1,15),"tape number.Therefore,
 you have 41":HPRINT(1,16),"char
acters available on the first li
ne":HPRINT(1,17),"& 38 on the se
cond."
60 HPRINT(1,19),"The arrow shows
 the centre."
61 CL=2:LN=4:XX=2:X=1:A=8:B=1
62 FOR Y=18TO19:SOUND220,1:IF Y=
18THEN RW=42ELSE RW=39
63 GOSUB41:IF E$(X,Y)="@@@"THEN
E$(X,Y)="":LN=LN+1:GOTO64ELSE64
64 XX=2:NEXT Y:GOTO65
65 HCOLOR A:HLINE(400,0)-(400,19
2),PSET
66 HPRINT(1,23),"ENTER CASSETTE
#:":SOUND220,1
67 LN=23:RW=21:CL=18:XX=19:X=2:Y
=18
68 GOSUB41
69 HCOLOR1:HPRINT(1,23),"ENTER C
ASSETTE #:":HPRINT(19,23),E$(X,Y
):HCOLOR4:HPRINT(1,23),"NUMBER C
ORRECT? (Y/N)  "+E$(X,Y):SOUND22
0,1
70 V$=INKEY$:IF V$=""THEN70
71 IF V$="Y"OR V$="y"THEN73ELSE
IF V$="N"OR V$="n"THEN72ELSE70
72 SE=0:HCOLOR1:HPRINT(1,23),"NU
MBER CORRECT? (Y/N)  "+E$(X,Y):H
COLOR4:E$(X,Y)="":GOTO66
73 HLINE(320,41)-(347,48),PRESET
,BF
74 IF LEN(E$(X,Y))=3THEN CN=40EL
SE IF LEN(E$(X,Y))=2THEN CN=41EL
SE CN=42
75 HPRINT(CN,5),E$(X,Y)
76 HCOLOR1:HPRINT(1,23),"NUMBER
CORRECT? (Y/N)  "+E$(X,Y):HCOLOR
2:HPRINT(1,23),"TITLE FLAP CORRE
CT? (Y/N)":SOUND220,1
77 Z$=INKEY$:IF Z$=""THEN77
78 IF Z$="Y"OR Z$="y"THEN80ELSE
IF Z$="N"OR Z$="n"THEN79ELSE77
79 E$(1,18)="":E$(1,19)="":E$(2,
18)="":GOTO52
```

```
80 ' ** OVERALL VIEW
81 E$(1,20)=""
82 HSCREEN4
83 PALETTE1,63:PALETTE0,27:HCOLO
R0,1:HCLS
84 HLINE(32,8)-(368,176),PSET,B
85 HLINE(32,128)-(368,128),PSET:
HLINE(32,144)-(368,144),PSET
86 HCOLOR3:FOR X=1TO14:HPRINT(1,
X+1),X:HPRINT(46,X+1),X:NEXT X
87 HPRINT(0,16),"F1":HPRINT(49,1
6),"F1":HPRINT(0,17),"F2":HPRINT
(49,17),"F2"
88 FOR X=1TO3:HPRINT(1,X+17),X+1
4:HPRINT(46,X+17),X+14:NEXT
89 HPRINT(4,1),"  side 1":HPRINT
(25,1),"  side 2":HPRINT(11,21),
"B&L SIDEBOTTOM     744 6281"
90 HCOLOR2
91 FOR X=1TO14:HPRINT(4,X+1),E$(
1,X):HPRINT(25,X+1),E$(2,X):NEXT
 X
92 FOR Y=18TO19:X=1:HPRINT(4,Y-2
),E$(X,Y):NEXT
93 L=LEN(E$(2,18)):HPRINT(45-L,1
7),E$(2,18)
94 FOR X=15TO17:HPRINT(4,X+3),E$
(1,X):HPRINT(25,X+3),E$(2,X):NEX
T
95 HCOLOR0:HLINE(410,0)-(410,191
),PSET
96 HCOLOR2:HPRINT(52,1),"CORRECT
? (Y/N)":SOUND220,1
97 Z$=INKEY$:IF Z$=""THEN97
98 IF Z$="Y"OR Z$="y"THEN125ELSE
 IF Z$="N"OR Z$="n"THEN99ELSE97
99 ' ** EDIT OVERALL
100 HCOLOR1:HPRINT(52,1),"CORREC
T? (Y/N)":HCOLOR2
101 HDRAW"BM418,50;U40R10F15E15R
10D40L10U25G10L10H10D25L10"
102 HDRAW"BM472,50;U40R50D8L40D8
R30D8L30D8R40D8L50"
103 HDRAW"BM526,50;U40R10F30U30R
10D40L10H30D30L10"
104 HDRAW"BM580,37;U27R10D27":HC
IRCLE(605,37),16,2,1,0,.5:HDRAW"
BM620,37;U27R10D27":HCIRCLE(605,
37),26,2,1,0,.5
105 Y=419:FOR X=1TO4:HPAINT(Y,11
),3,2:Y=Y+54:NEXT
106 HCOLOR3:HLINE(409,0)-(640,12
0),PSET,B
107 HCOLOR2:HPRINT(52,8),"CHOOSE
:-"
108 HCOLOR3:HPRINT(58,10),"1.EDI
T SIDE 1":HPRINT(58,11),"2.EDIT
SIDE 2":HPRINT(58,12),"3.EDIT FL
AP":SOUND220,1
109 Z$=INKEY$:IF Z$=""THEN109
110 IF VAL(Z$)<1OR VAL(Z$)>3THEN
109
111 ON VAL(Z$)GOTO113,114,124
112 GOTO109
113 D=1:GOTO115
114 D=2
115 HPRINT(58,14),"LINE #:":SOUN
D220,1
116 RV=67:LN=14:XX=66:CL=66:X=1:
Y=20:GOSUB41
```

Continued on P 54

From P 51

```
640 SOUND150,2
650 PRINT@486,"printer is not re
ady";
660 SOUND 159,2
670 A=PEEK(65314)
680 IF A=48 THEN PRINT@480,"
                          ";:RET
URN
690 GOTO 630
700 END
710 CLS
720 'Printer Baud Rate Poke Set
for DMP 200, Baud 1200
730 POKE150,41
740 'Screen & Lower case Poke:
750 POKE359,57:POKE 65314,80
760 'Main Menu Screen:
770 CLS:PRINT@100,"##  DISK LABE
L MAKER  ##"
780 PRINT@171,"Main  Menu"
790 PRINT@203,STRING$(10,131);
800 PRINT@292,"1  Main Programme
 Disk:"
810 PRINT@324,"2  Data  Disk:"
820 PRINT@356,"3  Games  Disk:"
830 PRINT@388,"4  Utilities Disk
:"
840 PRINT@420,"5  Exit"
850 PRINT@485,"Enter Selection 1
 - 5";
860 A$=INKEY$:IF A$="" THEN 860
870 'Selection:
880 IF A$="1" GOTO 1030
890 IF A$="2" GOTO 1180
900 IF A$="3" GOTO 1370
910 IF A$="4" GOTO 1520
920 IF A$="5" GOTO 1670
930 IF A$=CHR$(13) GOTO 720
940 IF A$>"5" GOTO 960
950 'Unidentified selections
960 CLS
970 PRINT@138,"You  Idiot"
980 PRINT@163,"there is no such
selection"
990 SOUND 150,3
1000 PRINT@298,"Try  Again"
1010 FOR T = 1 TO 2000:NEXT T
1020 GOTO 720
1030 CLS
1040 PRINT@68,"Main Programme Di
sk Menu:"
1050 PRINT@100,STRING$(25,131);
1060 PRINT@192,"Programme:"
1070 PRINT@320,"Loading Instruct
ions:"
1080 PRINT@256,"";:INPUT MP$
1090 PRINT@384,"";:INPUT LI$
1100 'Output to Printer Main Pro
gramme Disk
1110 PRINT#-2,CHR$(27)CHR$(18)
1120 PRINT#-2,CHR$(27)CHR$(10);C
HR$(27)CHR$(10)
1130 PRINT#-2,TAB(2)CHR$(27)CHR$
(14);CHR$(15);MP$;CHR$(14);CHR$(
27)CHR$(15)
1140 PRINT#-2,"":PRINT#-2,""
1150 PRINT#-2,TAB(2)LI$
1160 PRINT#-2,"":PRINT#-2,""
1170 GOTO 770
1180 CLS
1190 PRINT@9,"Data Disk Menu:"
1200 PRINT@41,STRING$(15,131);
1210 PRINT@128,"Data For Which P
rogramme:"
1220 PRINT@256,"Titel:"
1230 PRINT@384,"Data:"
1240 PRINT@192,"";:INPUT DP$
1250 PRINT@320,"";:INPUT T$
1260 PRINT@448,"";:INPUT D$
1270 'Output to Printer Date Dis
k Menu
1280 PRINT#-2,CHR$(27)CHR$(18)
1290 PRINT#-2,CHR$(27)CHR$(10);C
HR$(27)CHR$(10)
1300 PRINT#-2,TAB(2)CHR$(27)CHR$
(31)"DATA: "CHR$(27)CHR$(32);T$
1310 PRINT#-2,""
1320 PRINT#-2,TAB(2)CHR$(27)CHR$
(14);CHR$(15);DP$;CHR$(14);CHR$(
27)CHR$(15)
1330 PRINT#-2,""
1340 PRINT#-2,TAB(2)D$
1350 PRINT#-2,""
1360 GOTO 770
1370 CLS
1380 PRINT@40,"Games Disk Menu:"
1390 PRINT@72,STRING$(16,131);
1400 PRINT@160,"Type of Games:"
1410 PRINT@320,"Instructions:"
1420 PRINT@224,"";:INPUT TG$
1430 PRINT@384,"";:INPUT IN$
1440 'Output to Printer Games Di
sk Menu
1450 PRINT#-2,TAB(2)CHR$(27)CHR$
(14)"GAMES:"CHR$(27)CHR$(15)
1460 PRINT#-2,""
1470 PRINT#-2,TAB(2) TG$
1480 PRINT#-2,""
1490 PRINT#-2,TAB(2) IN$
1500 PRINT#-2,""
1510 GOTO 770
1520 CLS
1530 PRINT@6,"Utilities Disk Men
u:"
1540 PRINT@38,STRING$(20,131);
1550 PRINT@128,"Type of Utility:
"
1560 PRINT@256,"For use with:"
1570 PRINT@192,"";:INPUT TU$
1580 PRINT@320,"";:INPUT UU$
1590 'Output to printer Utilitie
s
1600 PRINT#-2,CHR$(27)CHR$(18)
1605 PRINT#-2,CHR$(27)CHR$(10);C
HR$(27)CHR$(10)
1610 PRINT#-2,TAB(2)CHR$(27)CHR$
(14);CHR$(15)"UTILITIES:"CHR$(14
);CHR$(27)CHR$(15)
1620 PRINT#-2,""
1630 PRINT#-2,TAB(2) TU$
1640 PRINT#-2,""
1650 PRINT#-2,TAB(2) UU$
1655 PRINT#-2,""
1660 GOTO 770
1670 CLS
1680 POKE113,0:EXEC40999
```

Print at

Here's a cute 'ute for making title screens.

by Aaron West

16K CB
UTILITY

PRINT AT IS A simple utiltity. The purpose of this program is to find screen co-ordinates.

After (C)LOADing the program and typing RUN, there is a hidden menu.

Some of the commands are:

* SHIFT+UP ARROV will get you to the normal mode.
* SHIFT+DOWN ARROV takes you to text mode.
* ENTER gives you the current co-ordinates.
* CLEAR ends the program.

In the normal mode, use the arrow keys to guide you around the screen. Press enter when you want the co-ordinates. The text mode is exactly the same as the normal mode, except that you can type on the screen to see how it looks.

I find this program very useful when I'm making titles screens for games.

I hope you find it useful.

The Listing:

```
 1 GOTO 10
2 '###############################
3 '#           PRINT AT           #
4 '#        1987 VESTSOFT         #
5 '#      BY AARON VEST(10)       #
6 '# A UTILTY TO LOCATE PRINT@ #
7 '#    SCREEN CO-ORDINATES       #
8 '###############################
9 SAVE"81A:3":END'7
10 'NORMAL MODE
20 CLS
30 A=0
40 PRINT@A,CHR$(128);
50 A$=INKEY$:IF A$="" THEN 50
60 PRINT@A,""
70 IF A$=CHR$(91) THEN CLS:GOSUB
170
80 IF A$=CHR$(13) THEN GOSUB380
90 IF A$=CHR$(12) THEN CLS:END
100 IF A$=CHR$(8) THEN A=A-1
110 IF A$=CHR$(9) THEN A=A+1
120 IF A$=CHR$(10) THEN A=A+32
130 IF A$=CHR$(94) THEN A=A-32
140 IF A<0 THEN A=0
150 IF A>510 THEN A=510
160 GOTO 40
170 'TEXT VRITE MODE
180 IF A<0 THEN A=0
190 IF A>510 THEN A=510
200 'NOTE:!!!!!!!!!!!!!!!!!!!!!!
210 '!   TO GET LOWERCASE "T"    !
220 '! PRESS SHIFT+0 BUT THESE !
184 '!   VILL APPEAR AS REVERSE !
230 '!        CHARACTERS.        !
240 '! PRESS SHIFT+0 TO RETURN !
250 '!          TO NORMAL        !
260 '!!!!!!!!!!!!!!!!!!!!!!!!!!!!
270 PRINT@A,"t";
280 A$=INKEY$:IF A$="" THEN 280
290 PRINT@A,CHR$(32);
300 IF A$=CHR$(95) THEN RETURN
310 IF A$=CHR$(13) THEN GOSUB380
320 IF A$=CHR$(12) THEN CLS:END
330 IF A$=CHR$(8) THEN A=A-1:GOT
O 170
340 IF A$=CHR$(94) THEN A=A-32:G
OTO 170
350 IF A$=CHR$(10) THEN A=A+32:G
OTO 170
360 IF A$=CHR$(9) THEN A=A+1:GOT
O 170
370 PRINT@A,A$:A=A+1:GOTO 270
380 'PRINT MODE
390 CLS:PRINT"PRINT@";A;",";CHR$
(34);"  ";CHR$(34);""
400 A$=INKEY$:IF A$="" THEN 400
410 IF A$=CHR$(12) THEN CLS:END
420 IF A$=CHR$(95) THEN RUN
430 IF A$=CHR$(91) THEN CLS:GOTO
 170
440 GOTO 400
450 '&&&&&&&&&&&&&&&&&&&&&&&&&&&
460 '& ANY QUESTIONS PHONE:     &
470 '&       (066) 544328       &
480 '& OR VRITE TO:             &
490 '&        AARON VEST        &
500 '&  LOT 132 GRAFTON RD      &
510 '&  CORAMBA    N.S.V.       &
520 '&          2450.           &
530 '&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

From P 53

```
117 HCOLOR3:Y=VAL(E$(1,20))
118 HPRINT(52,15),"old entry":H
PRINT(55,16),E$(D,Y)
119 SOUND200,1
120 HPRINT(52,18),"new entry:"
121 SOUND200,1:E$(D,Y)=""
122 RV=74:LN=19:XX=55:CL=55:X=D:
GOSUB41
123 GOTO80
124 PALETTE RGB:GOTO52
125 ' ** PRINTOUT
126 HPRINT(52,2),"Prepare printe
r -":HPRINT(52,3),"Press any key
!":SOUND220,1
127 IF INKEY$=""THEN127
128 POKE65496,0
129 PRINT#-2,STRING$(43,"-");" c
ut"
130 PRINT#-2,"   ";"side 1";TAB(
24)"side 2"
131 FOR X=1TO14:PRINT#-2," ";E$(
1,X);TAB(22)E$(2,X):NEXT
132 PRINT#-2,STRING$(43,"-");" f
old"
133 PRINT#-2," ";E$(1,18):PRINT#
-2," ";E$(1,19);
134 L=LEN(E$(2,18)):TB=41-L:PRIN
T#-2,TAB(TB);E$(2,18)
135 PRINT#-2,STRING$(43,"-");" f
old"
136 FOR X=15TO17:PRINT#-2," ";E$
(1,X);TAB(22)E$(2,X):NEXT
137 PRINT#-2,TAB(8)"B&L SIDEBOTT
OM    744 6281"
138 PRINT#-2,STRING$(43,"-");" c
ut"
139 HPRINT(52,5),"Another (Y/N)?
":SOUND200,1
140 Z$=INKEY$:IF Z$=""THEN140
141 IF Z$="Y"OR Z$="y"THEN143ELS
E IF Z$="N"OR Z$="n"THEN142ELSE1
40
142 POKE 113,0:EXEC40999
143 FOR X=1TO20:E$(1,X)="":E$(2,
X)="":NEXT:Q=0:X=0:POKE65497,0:G
OTO23
144 POKE65496,0:PALETTE RGB:WIDT
H40:ATTR3,2:CLS
```

Karate Chop

This is a case of keeping my foot out of YOUR mouth!

by Nigel Fredericks

CoCo3
GAME

PRESENTED HERE is a program called "Karate", for the CoCo 3.

The object of this game is to defeat 14 karate fighters and to do this you have a variety of punches and kicks at your disposal.

As you progress up the levels, your opponents increase in strength, and their abilities to block your moves thus increase.

The background colour changes also as you progress up levels, so as to make the game more appealing.

Have fun!

The Listing:

```
0 GOTO10
3 SAVE"129:3":END'1
10 'COLOUR COMPUTER KARATE
20 'BY NIGEL FREDERICKS
30 'COPYRIGHT 1988 BY SYVEN SYST
EMS CORP.
40 'FOR THE AUSTRALIAN COCO MAGA
ZINE
50 PALETTE 0,56:PALETTE 1,0:PALE
TTE 2,63:PALETTE 3,46:PALETTE 4,
0:PALETTE 5,36:PALETTE 6,46
60 PALETTE 7,29:PALETTE 8,16:PAL
ETTE 9,54:PALETTE 10,36:PALETTE
11,0:PALETTE 12,17:PALETTE 13,56
70 ON BRK GOTO 2590
80 Y=170
81 X=100
90 LI=10
100 L2=10
110 LV=1
120 PS=195
130 PLAY"T255;O3"
140 POKE 65497,0
150 HBUFF 1,1524
160 HBUFF 2,100
170 HBUFF 3,300
180 HBUFF 4,1500:HBUFF 5,100:HBU
FF 6,300
190 HBUFF 7,1000
200 HSCREEN 2
210 HCLS 0
220 HDRAW"BM12,0;C1;R6;C2;D1L6D1
L2R1D2"
230 HDRAW"BM18,2;C1;L6D1R1L2D1R2
D1L2R1D1U1R2F2"
240 HDRAW"BM18,3;C3L4D1R5D1L4F1R
2G1F1L5H1R3H1L1"'FACE COLOR"
250 HDRAW"BM16,4;C1;R1"
260 HDRAW"BM13,9;C3R3F1L3F1R2F1L
3F1R2G1L1"
270 HDRAW"BM12,9;C2;L1G1R3F1L5D1
R5F1L7D1R7F1R1L9D1R8G1L6F1R5D1L5
D1R8U1"
280 HDRAW"BM12,13;C1;D2F1R1F1D2L
4H2"
290 HDRAW"BM19,10;C2;L1D1R2F1L2D
1R2F1L4R1F1R5D1L4F1R3"
300 HDRAW"BM17,19;C3;L1U1R3E1L4E
1R3H1L1"
310 HDRAW"BM27,17;L1U1R3E1L4E1R3
H1L1"
320 HDRAW"BM10,21;C2;R8D1L8G1R10
D1L10G1R12F1L6F1R5F1L5F1R4D1L4F1
R4D1L4D1R4D1L4"
330 HDRAW"BM7,26;R6G1L5G1R6G1L5D
1R4G1L4D1R4D1L4D1R3D1L3"
340 HDRAW"BM10,21;C1;R2D1R2F3H2G
1D1U1E2R1U1R1"
350 HDRAW"BM8,36;C3;L4G1R5G1L4"'
FOOT
360 HDRAW"BM21,35;R4F1L5D1R5"
370 HDRAW"BM13,27;C1;G1D1G3D3"
380 HDRAW"BM40,11;C2;R9D1L9D1R9G
1;C1;L6G1E1;C2;L2D1;C3;D2"
390 HDRAW"BM50,11;C3;R2F1L3D1R4D
1L2D1L1R2"
400 HDRAW"BM40,21;C2;D1R15D1L15D
1R15D1L15;BM56,22;C3;R3F1L4D1R3L
3;C1;D1"
410 HDRAW"BM90,0;C4;L6;C5;D1R6D1
R2L1D2"
420 HDRAW"BM83,2;C4;R6D4E1U2L2D2
L1G2"
430 HDRAW"BM83,3;C6;R4D1L5D1R4G1
L2F1G1R5E1L3E1R1D3L3G1R3G1L2G1R3
G1L2F1R1"
440 HDRAW"BM84,4;C4;R1"
450 HDRAW"BM89,9;C5;R1F1L3G1R5D1
L5G1R7D1L7G1L1R9D1L8F1R6G1L5D1R5
D1L8U1"
460 HDRAW"BM83,10;L1G1R2G1L2D1R2
F1L4G1L2R5G1L4D1R3"
470 HDRAW"BM89,13;C4;D2G1L2G1D2R
4E2"
480 HDRAW"BM72,13;C6;R1F1L3D1R4D
1L3R1F1R1"'HAND
490 HDRAW"BM82,15;R1F1L3D1R4D1L3
R1F1R1"
500 HDRAW"BM83,21;C5;R8D1L8G1R10
D1L10G1R12F1L6D1R6F1L6F1R5D1L4F1
R4D1L4D1R4D1L4F1R3"
510 HDRAW"BM85,26;L4G1R5G1L5G1R5
G1L4D1R4G1L4D1R4D1L4"
520 HDRAW"BM83,21;C4;R2D1R1F2H1G
2E3R2U1R2"
530 HDRAW"BM88,27;F1D1F3D3"
540 HDRAW"BM93,35;C6;R4F1L5D1R5"
550 HDRAW"BM80,35;L4G1R5D1L5"
560 HDRAW"BM114,11;C5;R9D1L9D1R9
D2H1;C4L8;R7F1;C6;F1D1"
570 HDRAW"BM113,11;L2G1R3D1L4D1R
2G1R2"
580 HDRAW"BM129,21;C5;D1L15D1R15
D1L15D1R15L15;C4;L1;C6;U1L3H1R4U
1L3"
590 HDRAW"BM143,0;C1;R5;C2;D1L6D
1L2R1D2;BM149,2;C1;L6G1R2D1L2D1F
1U1R2F2"'JUMP
600 HDRAW"BM149,3;C3;L4D1R5D1L4F
1R2G1F1L5H1R3H1L1D3R3F1L2F1R1F1L
2F1R1D1L1"
610 HDRAW"BM148,4;C1;L1"
620 HDRAW"BM145,10;C2;L2G1R4D1L5
D1R6D1L7D1R9G1L8D1R8D2L6"
630 HDRAW"BM143,21;R8F1L9F1R8F1L
9R1F1R7D1L12D1R11D1L11D1R10"
640 HDRAW"BM150,25;C1;D2G3L6"
650 HDRAW"BM142,20;C1;R4D1R2F2H1
G1U2E1R2;C2;H1L3"
660 HDRAW"BM149,10;C2;D1R1D3R1D2
"
670 HDRAW"BM154,14;C3;D2L1U3G1D2
;BM150,15;D3L1U2"
680 HDRAW"BM144,13;C1D2R3F1D2L7H
1"
690 HDRAW"BM141,26;C3;D3G1U4L1D5
L1U5"
700 HGET(0,0)-(32,38),1
710 HGET(40,11)-(54,17),2
720 HGET(40,21)-(60,25),3
730 HGET(68,0)-(100,38),4
740 HGET(109,11)-(123,17),5
750 HGET(109,21)-(129,25),6
760 HGET(138,0)-(158,36),7
770 GOSUB 2450
780 GOSUB 2210
790 GOSUB 1580
800 HCOLOR 0,0
810 'MAIN BOARD
820 GOSUB 860
830 TI=TI+1
840 FOR V=0 TO DIF:GOSUB 1270:NE
XT V
850 GOTO 810
```

Welcome to the CoCo3 World of Karate

```
860 I$=INKEY$
870 IF I$="" THEN HPUT(X,100)-(X
+32,138),1,PSET: RETURN
880 IF I$="Z" THEN X=X-2 ELSE IF
 I$="X" THEN X=X+2
890 HPUT(X,100)-(X+32,138),1,PSE
T
900 IF I$<>CHR$(103) THEN 930 EL
SE HPUT(X+20,111)-(X+34,117),2,P
SET
910 IF HPOINT(X+36,113)=4 OR HPO
INT(X+36,114)=5 OR HPOINT(X+36,1
14)=6 THEN PLAY"AACCDFE":GOSUB 1
480
920 HLINE(X+30,111)-(X+34,117),P
SET,BF:HPUT(X,100)-(X+32,138),1,
PSET
930 IF I$<>CHR$(4) THEN 960 ELSE
 HPUT(X+20,121)-(X+41,125),3,PSE
T:HLINE(X+15,126)-(X+26,138),PSE
T,BF
940 IF HPOINT(X+42,123)=4 OR HPO
INT(X+42,123)=5 OR HPOINT(X+42,1
23)=6 THEN PLAY"AACCDFE":GOSUB 1
480
950 HPUT(X,100)-(X+32,138),1,PSE
T:HLINE(X+34,121)-(X+41,125),PSE
T,BF
960 IF I$="Q" OR I$="E"THEN 1240
970 IF I$="A" OR I$="D" THEN 118
0
980 IF I$<>"W"THEN 1000 ELSEX1=1
00:HLINE(X,100)-(X+32,138),PSET,
BF:FORT=1 TO 16:HPUT(X,X1)-(X+20
,X1+36),7,PSET:X1=X1-2:FORV=0TOD
IF:GOSUB1270:NEXT V:GOSUB1020:NE
XTT:FORT=1TO17:HLINE(X,X1-2)-(X+
20,X1),PSET,BF:HPUT(X,X1)-(X+20,
X1+36),7,PSET:X1=X1+2
990 FOR V=1TO DIF:GOSUB1270:NEXT
V:GOSUB1100:NEXT: HPUT(X,100)-(X
+32,138),1,PSET:RETURN
1000 IF I$<>"S" THENRETURN ELSEX
1=100:HLINE(X,100)-(X+32,138),PS
ET,BF:FORT=1 TO8:TI=TI+1:HPUT(X,
X1)-(X+20,X1+36),7,PSET:X1=X1-2:
FORV=0TODIF:GOSUB1270:NEXT V:GOS
UB1020:NEXTT:FORT=1TO10:TI=TI+1:
HLINE(X,X1-2)-(X+20,X1),PSET,BF:
HPUT(X,X1)-(X+20,X1+36),7,PSET
1010 X1=X1+2:FORV=0 TO DIF:GOSUB
1270:NEXTV:GOSUB 1100:NEXTT:HPUT
(X,100)-(X+32,138),1,PSET:RETURN
1020 'JUMP MOVES
1030 I$=INKEY$:IF I$="" THEN RET
URN
1040 IF I$<>CHR$(103) THEN 1070
ELSE HPUT(X+13,X1+12)-(X+27,X1+1
8),2,PSET:'SOUND100,1
1050 IF HPOINT(X+28,X1+14)=4 OR
HPOINT(X+28,X1+14)=5 OR HPOINT(X
+28,X1+14)=6 THEN PLAY"AACCDFE":
GOSUB 1480
1060 HLINE(X+14,X1+12)-(X+27,X1+
19),PSET,BF:RETURN
1070 IF I$=CHR$(4) THEN HPUT(X+1
2,X1+21)-(X+32,X1+25),3,PSET:HLI
NE(X+11,X1+26)-(X+17,X1+36),PSET
,BF':SOUND 100,1
1080 IF HPOINT(X+33,X1+27)=4 OR
HPOINT(X+33,X1+27)=5 OR HPOINT(X
+33,X1+27)=6 THEN PLAY"AACCDFE":
GOSUB1480
1090 HLINE(X+14,X1+21)-(X+32,X1+
26),PSET,BF:RETURN
1100 I$=INKEY$:IF I$="" THEN RET
URN
1110 IF I$<>CHR$(103) THEN 1140
ELSE HPUT(X+13,X1+8)-(X+27,X1+14
),2,PSET':SOUND 100,1
1120 IF HPOINT(X+28,X1+10)=4 OR
HPOINT(X+28,X1+10)=5 OR HPOINT(X
+28,X1+10)=6 THEN PLAY"AACCDFE":
GOSUB 1480
1130 HLINE(X+14,X1+8)-(X+27,X1+1
5),PSET,BF:RETURN
1140 IF I$<>CHR$(4) THEN RETURN
ELSE HPUT(X+12,X1+17)-(X+32,X1+2
1),3,PSET:HLINE(X+11,X1+22)-(X+1
7,X1+32),PSET,BF':SOUND 100,1
1150 IF HPOINT(X+34,X1+19)=4 OR
HPOINT(X+34,X1+19)=5 OR HPOINT(X
+34,X1+19)=6 THEN PLAY"AACCDFE":
GOSUB 1480
1160 HLINE(X+14,X1+17)-(X+32,X1+
22),PSET,BF:RETURN
1170 RETURN
1180 IFI$<>"D"THEN1210ELSEX1=100
:HLINE(X,100)-(X+32,138),PSET,BF
:FORT=1TO16:TI=TI+1:HPUT(X,X1)-(
X+20,X1+36),7,PSET:HLINE(X-2,X1+
26)-(X-1,X1+36),PSET,BF:X=X+2:X1
=X1-2:GOSUB1270:NEXTT:FORT=1TO8:
TI=TI+1:HPUT(X,X1)-(X+20,X1+36),
7,PSET
1190 HLINE(X-2,X1+26)-(X-1,X1+36
),PSET,BF:X=X+2:GOSUB1270:NEXTT:
FORT=1TO17:TI=TI+1:HPUT(X,X1)-(X
+20,X1+36),7,PSET:HLINE(X-2,X1+2
4)-(X-1,X1+36),PSET,BF
1200 HLINE(X+2,X1-2)-(X+8,X1-1),
PSET,BF::X=X+2:X1=X1+2:GOSUB 127
0:NEXTT:HLINE(X-2,X1+24)-(X-1,X1
+36),PSET,BF:HPUT(X,100)-(X+32,1
38),1,PSET:RETURN
1210 IFI$="A"THENX1=100:HLINE(X,
100)-(X+32,138),PSET,BF:FORT=1TO
16:TI=TI+1:HPUT(X,X1)-(X+20,X1+3
6),7,PSET:X=X-2:X1=X1-2:GOSUB127
0:NEXTT
1220 FORT=1TO8:TI=TI+1:HPUT(X,X1
)-(X+20,X1+36),7,PSET:X=X-2:GOSU
B 1270:NEXTT
1230 FORT=1TO17:TI=TI+1:HPUT(X,X
1)-(X+20,X1+36),7,PSET:HLINE(X+6
,X1-2)-(X+12,X1-1),PSET,BF:X=X-2
:X1=X1+2:GOSUB 1270:NEXTT:HPUT(X
,100)-(X+32,138),1,PSET:RETURN
1240 IF I$<>"E"THEN 1260ELSEX1=9
8:HLINE(X,100)-(X+32,138),PSET,B
F:X=X+8:FORI=1TO14:TI=TI+1:HPUT(
X,X1)-(X+20,X1+36),7,PSET:GOSUB1
020:FORV=0TODIF:GOSUB1270:NEXTV:
X=X+2:HLINE(X-2,X1+24)-(X-1,X1+3
6),PSET,BF
1250 NEXTI:HLINE(X,X1)-(X+20,X1+
36),PSET,BF:HPUT(X,100)-(X+32,13
8),1,PSET:RETURN
1260 IF I$<>"Q" THEN RETURN ELSE
X1=98:HLINE(X,100)-(X+32,138),PS
ET,BF:FORI=1TO18:TI=TI+1:HPUT(X,
X1)-(X+20,X1+36),7,PSET:GOSUB102
0:FORV=0TODIF:GOSUB1270:NEXTV:X=
X-2:NEXTI:HLINE(X,X1)-(X+20,X1+3
6),PSET,BF:HPUT(X,100)-(X+32,138
),1,PSET:RETURN
1270 'MONSTER MOVE
1280 HPUT(Y,100)-(Y+32,138),4,PS
ET
1290 DES=RND(100)
1300 IF DES<=20 THEN Y=Y+2
1310 IF DES>=21 AND DES<=45 THEN
 Y=Y-2
1320 IF DES>=46 AND DES<=65 THEN
 GOTO 1380
1330 IF DES>=66 AND DES<=85 THEN
 HPUT(Y,100)-(Y+32,138),4,PSET
1340 IF DES>=86 AND DES<=100 THE
N GOTO 1440
1350 HPUT(Y,100)-(Y+32,138),4,PS
ET
1360 RETURN
1370 RETURN
1380 'PUNCH
1390 HPUT(Y-1,111)-(Y+13,117),5,
PSET
1400 IF HPOINT(Y-3,114)=1 OR HPO
INT(Y-3,114)=2 OR HPOINT(Y-3,114
)=3 THEN PLAY"DDFEACCB":GOSUB155
0
1410 HLINE(Y-1,111)-(Y,117),PSET
,BF:HPUT(Y,100)-(Y+32,138),4,PSE
T
1420 RETURN
1430 RETURN
1440 HPUT(Y-5,121)-(Y+14,125),6,
PSET:HLINE(Y+5,126)-(Y+17,138),P
SET,BF':SOUND200,1
1450 IF HPOINT(Y-7,123)=1 OR HPO
INT(Y-7,123)=2 OR HPOINT(Y-7,123
)=3 THEN PLAY"DDFEACCB":GOSUB 15
50
1460 HPUT(Y,100)-(Y+32,138),4,PS
ET:HLINE(Y-5,121)-(Y,125),PSET,B
F
1470 RETURN
1480 'CHECK IF BLOCK
1490 B=RND(100):IF B<LV*3 THEN H
COLOR1:HPRINT(15,19),"BLOCKED!":
PLAY"O1;CBCBCBFAD;O3":HCOLOR0,0:
HLINE(120,150)-(180,160),PSET,BF
:RETURN
1500 L2=L2-1
1510 IF L2>0THEN1530 ELSE PLAY"T
255;ACBFGDEEEACAEEDFFGB":HLINE(Y
,100)-(Y+32,138),PSET,BF:HLINE(X
-2,X1-2)-(X+32,X1+38),PSET,BF:HP
UT(140,100)-(172,138),1,PSET:HCO
LOR1,0:HPRINT(10,10),"White Wins
 Stage ":HPRINT(26,10),LV:HPRINT
(8,11),"PREPARE FOR NEXT STAGE
1520 LV=LV+1:FOR I=1 TO 20:PLAY"
05;CGF;01;ABEBO3":NEXT I:X=100:T
T=TT+(300-TI):T1=0:Y=170:LI=10:L
2=10+LV:PS=195:P2=0:GOSUB 2330:G
OSUB 1580:HCOLOR0,0:GOSUB 2420:G
OTO 810
1530 PS=PS+5:HCOLOR5:HLINE(PS,15
0)-(PS+3,154),PSET,BF:HCOLOR0,0
1540 RETURN
1550 LI=LI-1:IF LI<=0 THEN PLAY"
T10;O4;ACBFGGEADCBFFF":GOTO 2350

1560 P2=P2+5:HCOLOR 2:HLINE(P2,1
50)-(P2+3,154),PSET,BF:HCOLOR0,0
1570 RETURN
1580 HSCREEN2:HCLS0:FORI=4 TO LI
*5 STEP 5:HCOLOR1:HLINE(I,149)-(
I+4,155),PSET,B:NEXT I
1590 FOR I=199 TO L2*5+195 STEP
5:HLINE(I,149)-(I+4,155),PSET,B:
NEXT I
1600 HLINE(0,65)-(320,65),PSET
1610 HPAINT(0,0),7,1
1611 HCIRCLE(0,0),22,9
1612 HPAINT(10,10),9,9
1613 HCOLOR 1
1620 HLINE(0,0)-(320,192),PSET,B
1630 IF LV<4 THEN GOSUB 1800:GOT
O 1850
1640 IF LV>3 AND LV<7 THEN PALET
TE 8,32:GOSUB 1800:GOTO 1850
1650 IF LV>6 AND LV<9 THEN GOSUB
 1800:GOSUB 1680:GOTO 2100
1660 IF LV>8 AND LV<12 THEN PALE
TTE 8,16:GOSUB 1800:GOSUB 1680:G
OTO 2100
1670 IF LV>11 THEN GOSUB 1900:GO
TO 2100
1680 '
1690 HDRAW'BM73,64;C11;R4E1R1E1R
2E1R29L29E1R1E1R1E2RE2R25L25U1E2
R1E2R24L24E5R21L21E4U1R19L19E5R1
5L15E2U1E1R11L11E3U1E1R9L9E2U1E1
R6L6E1U1E1U1E1U2"
1700 HDRAW"BM125,15;R1F3R2E2R3E1
R4D1R8E4R1D1F2L1E2F2R4E3R1F4R3E3
F1D3F2D1F2D1F1D1F2D1F1D1F1D2F1D1
F2D1F1D1F1D1F2R1F1D1F2D1F6D1F5R1
F3R3"
1710 HDRAW"BM131,18;D2F1R1F1D3G2
L1D1L1D1G1D4G1D2F2R1F1R1F1D2F1D1
G1D1F1E3U1E1U2E1U3R1E1R1E1U3E1R1
E1U6H1L2H1U2E1R2E2"
1720 HDRAW"BM157,15;D1G1D1F2R2F1
D5G1D1R1F2R1U1L1H1U1E1R2E1L1E1U1
E1U4H1U1H1"
1730 HLINE(75,65)-(215,65),PSET
1740 HPAINT(155,20),12,11
1750 HPAINT(140,17),10,11
1760 HPAINT(164,16),10,11
1770 FOR I=1 TO 50:HH=RND(60)+12
0:VV=RND(12):HSET(HHH,VV,5):NEXT
 I
1780 FOR I=1TO35:HH=RND(60)+120:
VV=RND(12):HSET(HH,VV,9):NEXT I
1790 RETURN
1800 A$=";C1;R1E5R1U1R1E5R1E1U1E
3U1E2R1E1R1E1R1E2R1E1R1E2R1D1F1D
1F1D1F2R2E4U1E1R2F1R2F2R1F1R4E1U
2E1R1F1D1F17R1F4"
1810 HDRAW"BM0,50"+A$
1820 HDRAW"BM80,50"+A$:HDRAW"BM1
60,50"+A$:HDRAW"BM240,50"+A$
1830 HDRAW"BM0,50;C1;D15R319U16"
:HPAINT(5,64),8,1
1840 RETURN
1850 B$=";C2;E2U1E7U1E8R6F1R1F6R
1F6G2D2L4H1L1H2L2H1L2G2D1G3L1H1U
1H1U1H1U1H1L1G2L1G1L1G2L2G1"
1860 HDRAW"BM20,30"+B$
1870 HDRAW"BM100,30"+B$:HDRAW"BM
180,30"+B$:HDRAW"BM260,30"+B$
1880 FOR I=40 TO 280 STEP 80:HPA
INT(I,25),2,2:NEXT I
1890 GOTO 2100
1900 HDRAW"BM57,64;C1;R3E1R2E2R1
E1R1E1R1E1R1E2R1E1R1E1R1E3R1E1R1
E2R1E3R1E2R1E2R1E9U1E4U1E1U1E1U3
E1U2E1R1F1R1F1D5G3D3F1D4G1D1G3L1
G3D2F1R1E1R1E1R1E1R1E1R1E2R1U1R1
U1H1U1E1F1R2E1U1E1U4E3U1H1U1H4E1
R5"
```

Fight 14 enemy Karate fighters and become the KARATE MASTER

```
1910 HDRAW"BM133,17;F2D2F1D1G2D3
F1E3U1E1R2E1U1E1U1H3U1E1R1F1R1F1
R3E1D1F1D1D2F1D3G3L1G1D4G1D6G1D1F1
R1E2U1E1U1E1R2E1U2E1U1E1R3F3R3E1
U1H1L1H4U2H1U1H2E1U2E1R1E1R2"
1920 HDRAW"BM163,15;R2F1R1F6D1F1
D1F2R1F1R2F3D1F4R2F1R1E1H1U2H4U1
H3L1H1U1H6L1H2L1H1L2H1L18H1L3G1L
5G1L9G1L10G1L3"
1930 HDRAW"BM193,36;F12R1F1R1F1R
1F1R1F2R2F1R1F1R1F1R2F1R1F3R1F1R
2F1R2F1R3F2"
1940 HPAINT(150,60),8,1:HPAINT(1
25,20),10,1
1950 FORI=1TO60:HH=RND(55)+115:V
V=RND(12):HSET(HH,VV,10):NEXT I
1960 FOR I=1TO30:HH=RND(55)+115:
VV=RND(12):HSET(HH,VV,9):NEXT I
1970 A$="G2L9G1L1G3L3D1L2G1D2F2R
2E1R4F1R1F1R4E1R4E1R3F2R5E1R2E1R
1F1R3E1U1E1H1U1H2L1H4L4H1L8"
1980 HDRAW"BM40,9;C2"+A$:HDRAW"B
M210,12"+A$:HDRAW"BM280,4"+A$
1990 HPAINT(40,15),2,2:HPAINT(21
0,16),2,2:HPAINT(280,8),2,2
2000 RETURN
2010 A$=";C1;U9E1U1H1U1E1U1H1U2R
4D2G1D1F1D1G1D1F1D9L3;C13;U9R1D9
R1U9H1U7L1D1R2U1G1D2G1D1R2U1;":A
A$=";F1R1F1R1F1R1F1R4F1R4F1R13E1
R4E1R4E1R1E1R1E1R1E1"
2020 HDRAW"BM0,65"+A$:HDRAW"BM5,
47"+AA$
2030 HDRAW"BM53,65"+A$:HDRAW"BM5
8,47"+AA$
2040 HDRAW"BM106,65"+A$:HDRAW"BM
111,47"+AA$
2050 HDRAW"BM160,65"+A$:HDRAW"BM
165,47"+AA$
2060 HDRAW"BM213,65"+A$:HDRAW"BM
218,47"+AA$
2070 HDRAW"BM266,65"+A$:HDRAW"BM
271,47"+AA$
2080 HDRAW"BM315,65"+A$
2090 RETURN
2100 GOSUB 2010
2110 HCOLOR1,0:HPRINT(1,20),"SCO
RE:":HPRINT(6,20),TT
2120 HPRINT(1,21),"STAGE:":HPRIN
T(6,21),LV
2130 COL=COL+RND(10):IF COL>=63T
HEN COL=0:GOTO 2130
2140 PALETTE 5,COL
2150 IF LV=14 THEN PALETTE 5,0
2160 HPRINT(1,23),"NAME:":HPRINT
(6,23),N$
2170 HPRINT(25,21),"DIFFICULTY:"
:HPRINT(35,21),DIF
2180 HCOLOR 9:HPRINT(5,22),"A SY
VEN SOFTWARE PRESENTATION"
2190 HCOLOR2:IF LV=14 THEN HPRIN
T(12,21),"FINAL STAGE"
2200 RETURN
2210 WIDTH 40
2220 CLS 2
2230 CLS3:LOCATE11,0:ATTR 3,2,U:
PRINT"COCO-3 KARATE";:ATTR3,2
2240 LOCATE0,1:Q$="Welecome play
er to the CoCo-3 world of  Karat
e.The object of this game is to
   fight 14 enemy Karate fighter
s so as to gain the title of KAR
ATE MASTER."
2250 PRINT Q$;
2260 Q$="As you  complete each l
evel your opponants      increas
e in strength and ability so be
 warned! GOOD LUCK (you'll need
it)"2270 PRINTQ$
2280 LOCATE13,9:ATTR3,2,U:PRINT"
CONTROLS";:ATTR 3,2
2290 LOCATE0,10:PRINT"Z=Left Mov
ement":PRINT"X=Right Movement":P
RINT"f1=Punch","f2=Kick":PRINT"A
=Diagonal Left High Flip":PRINT"
S=Low Jump":PRINT"D=Diagonal Rig
ht High Flip":PRINT"Q=Left Low F
lying Move":PRINT"W=High Jump":P
RINT"E=Right Low Flying Move"
2300 PRINT:PRINT"You control the
 WHITE player on the lefthand si
de;":LINEINPUT"What is your name
-";N$:INPUT"What difficulty leve
l";DIF
2310 IF DIF >3 THEN 2210
2320 RETURN
2330 '
2340 IF LV>14 THEN 2430 ELSE RET
URN
```

Oooo.... that's not very nice!

```
2350 HLINE(X,100)-(X+32,138),PSE
T,BF:HLINE(Y,100)-(Y+32,138),PSE
T,BF:HPUT(140,100)-(172,136),4,P
SET
2360 HCOLOR1:HPRINT(8,10),"YOU H
AVE BEEN DEFEATED!!":PLAY"O3;AAC
DEFFFDGFEFEDACCBAEDBCFGGG"
2361 FOR I=0 TO 320:HCOLOR RND(1
5),0:HLINE(I,0)-(I,192),PSET:NEX
T I
2370 WIDTH 40:ATTR3,2:PRINT"You
have lost your battle against yo
ur  opponant.The grand master ha
s ordered   your destruction bec
ause of your failureBetter luck
next time."
2380 LOCATE 0,5:PRINT"YOUR SCORE
-";TT
2390 LOCATE 0,7:PRINT"LAST STAGE
 ATTEMPTED-";LV
2400 PRINT:LOCATE 12,15:ATTR3,6,
B:PRINT"PRESS ANY KEY";:EXEC4453
9:GOTO 2590
2410 GOTO 2410
2420 PLAY"T10;O2;FFFAAGG;P1;FFFA
AGG;P2;CCCDDEEEGGAAGGAAGGAAGGAA;
P2":PLAY"T255;O3":RETURN
2430 FOR I=0TO192:HCOLOR RND(15)
,0:HLINE(0,I)-(320,I),PSET:NEXT:
WIDTH 40:CLS:LOCATE 13,9:PRINT"Y
OU HAVE WON!!!":GOSUB 2420:PRINT
:PRINT"The grand master congradu
lates you on   your fine win and
 has awarded you the   title of
MASTER.Good work!!":EXEC44539
2440 GOTO 2590
2450 HCLS0
2460 HCOLOR 1,0:HLINE(0,65)-(320
,65),PSET
2470 HPAINT(1,1),7,1
2480 GOSUB 1900
2490 HCIRCLE(160,140),40,1:HCIRC
LE(160,120),14,1,1.5,.75,.25:HCI
RCLE(160,160),14,1,1.5,.25,.75
2500 HCIRCLE(155,120),7,1:HCIRCL
E(165,160),7,1
2510 HPAINT(150,130),5,1:HPAINT(
170,150),2,1
2520 GOSUB 2010
2530 HPUT(50,120)-(82,158),1,PSE
T:HPUT(238,120)-(270,158),4,PSET
2540 HCOLOR1,0:HPRINT(8,9),"COLO
UR COMPUTER 3 KARATE"
2550 HPRINT(8,11),"BY NIGEL FRED
ERICKS 1987"
2560 HPRINT(1,24),"COPYRIGHT 198
8-SYVEN SOFTWARE SYSTEMS"
2570 GOSUB 2420:GOSUB 2420
2580 EXEC44539:RETURN
2590 RGB
2600 POKE65496,0
2610 ATTR 0,0
2620 CLS 9
```

⌽

Max on the 3

by Ken Wagnitz

I SAW YET another article on running Cocomax on a CoCo 3, in the NCTCUG newsletter from the USA. This meant that there must be something in it - the authors actually said they had it working!

Previously all I had seen were reports of 'others' who had done it. Well, it didn't work on a Y-cable, and it didn't work on my MultiPak which has been modified for the CoCo3.

But they said it worked on an unmodified MultiPak, so I set to and modified my modification, to make it reversible.

The details for doing that were given in the January 87 US Rainbow. And it DOES work. Just do the following:

Place a copy of the Cocomax disk in drive 0. Type:

```
LOADM"COCOMAX/SYS"
SAVEM"COCOMAX/SYS",&HE00,&H18F0,
0
```

Thats it! That disk should now work on a CoCo 3, using an unmodified MultiPak. I tested CoComax 2, but version 1 should also work.

As for the un-mod mod I added, here is what to do if your MultiPak has been modified by Tandy for the CoCo3:

1. Unplug the Pak from the computer and the mains, and pull out any cartridges;
2. Remove the four screws from the bottom & remove the top;
3. Drill or cut a hole somewhere in the bottom case to take a 2 pole changeover switch, either toggle or slide type - I used a slide switch mounted at the front;
4. Short an outside pair of terminals on the switch, solder four wires to the other four contacts;
5. Mount the switch - I glued mine to the rear of the front panel with Selleys 'Acribond' which fastens metal to plastic OK;
6. Unsolder the added wire which was on pin 19 of a 74LS245, solder to that pin, one of the wires to the centre of the switch;
7. Connect that removed wire to the outside contact on the switch which mates with the previous centre wire;
8. Cut the wire which goes from the extra small PCB (the modification) to the large square surface mounted chip;
9. Join the chip end of that wire to the wire from the centre of the switch;
10. Join the PCB end of that wire to the wire from the outside of the switch.

⌽

From P 39

```
3090 RESET(54,V)
3100 NEXTV
3110 FORH=5TO56STEP2
3120 SET(H+2,6,8)
3130 NEXTH
3140 FORH=5TO56STEP2
3150 RESET(2+H,6)
3160 NEXT H
3170 FORV=6TO25STEP2
3180 SET(57,2+V,8)
3190 NEXT V
3200 FORV=6TO23STEP2
3210 RESET(57,2+V)
3220 NEXT V
3230 FORH=56 TO 6 STEP-2
3240 SET(2+H,26,8)
3250 NEXT H
3260 FOR H=54 TO 6 STEP-2
3270 RESET(2+H,26)
3280 NEXT H
3290 FOR V=24 TO 6 STEP -2
3300 SET(6,2+V,8)
3310 NEXT V
3320 FOR V=24 TO 6 STEP-2
3330 RESET(6,2+V)
3340 NEXT V
3350 GOTO 3110
3360 END
3370 PRINT@J,CHR$(143+16*C);
3380 RETURN
```

⌽

M/L Revealed

The 'mysteries' of Machine Language made easy.

by Grahame Pollock

64K ECB CoCo/CoCo3
APPLICATION

A FEW PEOPLE HAVE asked me to write an article about machine language for beginners. Well the level should be just about right, because I'm only a beginner myself. If you know all about machine language then this article is definately not for you.

I'd like to make it plain right at the start that I'm not an expert on machine language. I've only written a few ML programs, and I intend to avoid the parts that I don't understand myself.

I think the first thing I should do is to run through a few things that you need to have before you start your journey to the land of ML.

1. OPCODES LIST - you need a list of opcodes (OPERATION CODES?) which equate stored numbers to computer operations.

This list will let you know which numbers to store in RAM to make the computer do what you want it to. I've included an opcode list for you.

2. MONITOR or ASSEMBLER - you need a program which will enable you to examine and change the contents of a certain location within RAM. These programs will enable you to enter your ML program into the computer memory. An assembler works on a higher level than a monitor. The ASSEMBLER allows you to enter the opcodes which it later assembles into the ML program.

The MONITOR allows you to enter the M.L. program directly, after you have spent some time "hand assembling" (on paper).

For the CoCo, "EDTASM+" is a good assembler and "HUMBUG" or "CBUG" are good monitors. "EDTASM+" has a monitor built in ("ZBUG"). For the MC-10, Nigel Steele's "ASS6803" is a good assembler and "HUMBUG" or "MINIMON" are good monitors. You could even use my "POCSAVER" as a monitor.

3. MEMORY MAP - this tells you which parts of the memory are concerned with such things as BASIC SCRATCHPAD, ROM, BASIC PROGRAM AREA, SCREEN MEMORY, ETC. I've included a simple memory map for the CoCo and the MC-10.

(see table one)

4. ROM DISASSEMBLY-in order to utilise the built in ROM with your ML programs, you need a fully annotated ROM disassembly.

This document tells you what every byte of you ROM does. It also tells you all about the scratch RAM and how BASIC uses it to store numbers. For the CoCo, there's "C.B. UNRAVELLED" and "E.C.B. UNRAVELLED". For the MC-10 there's Ron Wright's "MICO EXPOSED". I really couldn't do without mine!

5. DISASSEMBLER- it's always nice to be able to disassemble someone elses ML program so that you can work out how it ticks. To do this you need a disassembler. One of the best ways to learn something is to see how someone else does it and try to work out what they've done and why they've done it.

You should try to collect some small ML programs first and try to work out how they operate. The disassembler comes in handy if you don't have an assembly language listing for the program.

6. REFERENCE BOOK- there are quite a few good assembly language books on the market, mostly for the 6809. You can use them for the 6803 as long as you know the differences, most of which I'll outline in this article. It is important to have at least one of these books because they contain full descriptions of the meaning of the opcodes.

So for successful ML programming you need:- An opcodes list, a monitor or an assembler, a memory map, a ROM disassembly and a disassembler.

To get started though, you could get by with:-

1. OPCODES LIST
2. MONITOR
3. MEMORY MAP

HERE'S THE OPCODE LIST:-

table 1

HEX	COCO	DECIMAL
$0000-$03FF	BASIC WORKING STORAGE	0-1023
$0400-$05FF	TEXT SCREEN	1024-1535
$0600-?	GRAPHICS PAGES (LIMIT SET BY NUMBER OF PAGES USED)	1536-?
?-$3FFF	BASIC PROGRAM MEMORY	?-16383
$4000-$7FFF	EXTRA PROGRAM MEMORY	16384-32767
$8000-$9FFF	E.C.B. ROM	32768-40959
$A000-$BFFF	C.B. ROM	40960-49151
$C000-$FFFF	CARTRIDGE ROM	49152-65535

HEX	MC-10	DECIMAL
$0000-$001F	INTERNAL REGISTERS	0-31
$0020-$007F	UNUSED	32-127
$0080-$009F	DIRECT PAGE SCRATCH RAM	128-255
$0100-$3FFF	UNUSED	256-16383
$4000-$41FF	TEXT SCREEN	16384-16895
$4200-$4335	BASIC SCRATCH PAD RAM	16896-17221
$4336-$4FFF	INTERNAL RAM	17222-20479
$5000-$8FFF	EXTERNAL RAM EXPANSION	20480-36863
$9000-$BFFF	KEYBOARD INPUT,VDG,SOUND·	36864-49151
$C000-$DFFF	EXTERNAL ROM EXPANSION	49152-57343
$E000-$FFFF	MICROCOLOR BASIC ROM	57344-65535

<see table one>

<see table two>

Note that about 80% of CoCo ML is equivalent to MC-10 machine code.

Before we go on to use some of these machine language instructions, we need to look more closely at the way the microprocessor handles its numbers.

Each memory location can store a single number between 0 and 255 ($00-$FF). This number is a

table 2

DEC	HEX	COCO(6809)			MC-10(6803)		
		OP	MODE	#	OP	MODE	#
*0	00	NEG	DIR	2	-	-	-
*1	01	-	-	-	NOP	INH	1
2	02	-	-	-	-	-	-
*3	03	COM	DIR	2	-	-	-
*4	04	LSR	DIR	2	LSRD	INH	1
*5	05	-	-	-	LSLD	INH	1
*6	06	ASL	DIR	2	TAP	INH	1
*7	07	ASR	DIR	2	TPA	INH	1
*8	08	LSL	DIR	2	INX	INH	1
*9	09	ROL	DIR	2	DEX	INH	1
*10	0A	DEC	DIR	2	CLV	INH	1
*11	0B	-	-	-	SEV	INH	1
*12	0C	INC	DIR	2	CLC	INH	1
*13	0D	TST	DIR	2	SEC	INH	1
*14	0E	JMP	DIR	2	CLI	INH	1
*15	0F	CLR	DIR	2	SEI	INH	1
*16	10	LB**	REL	4	SBA	INH	1
*17	11	-	-	-	CBA	INH	1
*18	12	NOP	INH	1	-	-	-
*19	13	SYNC	INH	1	-	-	-
20	14	-	-	-	-	-	-
21	15	-	-	-	-	-	-
22	16	-	-	-	TAB	INH	1
*23	17	LBSR	REL	3	TBA·	INH	1
24	18	-	-	-	-	-	-
25	19	DAA	INH	1	DAA	INH	1
*26	1A	ORCC	IMM	2	-	-	-
*27	1B	-	-	-	ABA	INH	1
*28	1C	ANDCC	IMM	2	-	-	-
*29	1D	SEX	INH	1	-	-	-
*30	1E	EXG	INH	2	-	-	-
*31	1F	TFR	INH	2	-	-	-
32	20	BRA	REL	2	BRA	REL	2
33	21	BRN	REL	2	BRN	REL	2
34	22	BHI	REL	2	BHI	REL	2
35	23	BLS	REL	2	BLS	REL	2
36	24	BCC	REL	2	BCC	REL	2
37	25	BCS	REL	2	BCS	REL	2
38	26	BNE	REL	2	BNE	REL	2
39	27	BEQ	REL	2	BEQ	REL	2
40	28	BVC	REL	2	BVC	REL	2
41	29	BVS	REL	2	BVS	REL	2
42	2A	BPL	REL	2	BPL	REL	2
43	2B	BMI	REL	2	BMI	REL	2
44	2C	BGE	REL	2	BGE	REL	2
45	2D	BLT	REL	2	BLT	REL	2
46	2E	BGT	REL	2	BGT	REL	2
47	2F	BLE	REL	2	BLE	REL	2
*48	30	LEAX	REL	2	TSX	INH	1
*49	31	LEAY	REL	2	INS	INH	1
*50	32	LEAS	REL	2	PULA	INH	1
*51	33	LEAU	REL	2	PULB	INH	1
*52	34	PSHS	INH	2	DES	INH	1
*53	35	PULS	INH	2	TXS	INH	1
*54	36	PSHO	INH	2	PSHA	INH	1
*55	37	PULU	INH	2	PSHB	INH	1
*56	38	-	-	-	PULX	INH	1
57	39	RTS	INH	2	RTS	INH	1
58	3A	ABX	INH	1	ABX	INH	1
59	3B	RTI	INH	1	RTI	INH	1
*60	3C	CWPI			PSHX	INH	1
61	3D	MUL	INH	1	MUL	INH	1
*62	3E	-	-	-	WAI	INH	1
63	3F	SWI	INH	1	SWI	INH	1
64	40	NEGA	INH	1	NEGA	INH	1
65	41	-	-	-	-	-	-
66	42	-	-	-	-	-	-
67	43	COMA	INH	1	COMA	INH	1
68	44	LSRA	INH	1	LSRA	INH	1
69	45	-	-	-	-	-	-
70	46	RORA	INH	1	RORA	INH	1.
71	47	ASRA	INH	1	ASRA	INH	1
72	48	ASLA	INH	1	ASLA	INH	1
73	49	ROLA	INH	1	ROLA	INH	1
74	4A	DECA	INH	1	DECA	INH	1
75	4B	-	-	-	-	-	-
76	4C	INCA	INH	1	INCA	INH	1
77	4D	TSTA	INH	1	TSTA	INH	1
78	4E	-	-	-	-	-	-
79	4F	CLRA	INH	1	CLRA	INH	1
80	50	NEGB	INH	1	NEGB	INH	1
81	51	-	-	-	-	-	-
82	52	-	-	-	-	-	-
83	53	COMB	INH	1	COMB	INH	1

byte. The microprocessor works by juggling these bytes around.

The microprocessor can store single bytes in its A and B accumulators and double bytes in its index registers (X and Y), stack pointers (U and S) and program counter (PC). The 6803 (MC-10) does not have a Y index register or a U stack pointer. The 6809 (CoCo) also has a single byte direct page pointer (DP). There is also a condition Code register (CC) which uses its separate bits to keep a record of the results of any operations done by the microprocessor.

The single byte registers (A,B and DP in CoCo) will only store numbers from 0 to 255 ($00-$FF), where as the double byte registers (X,S,PC and Y,U in the 6809) can hold any number from 0 through to 65535 ($0000-$FFFF).

If the X register contains the hex number $BFFF then $BF(191) is said to be the most significant byte (M.S.B.) and $FF(255) is said to be the least significant byte (L.S.B.). To work out the decimal number stored in the X register we need to do the following calculations:-

```
MSB * 256 + LSB
=191 * 256 + 255
=48896 + 255
=49151
```

If we want to use the A and B accumulators together, then the two of them fit neatly together as the D accumulator with A as the M.S.B. and B as the L.S.B.

In other words, if the A

84	54	LSRB	INH	1	LSRB	INH	1
85	55	-	-	-	-	-	-
86	56	RORB	INH	1	RORB	INH	1
87	57	ASRB	INH	1	ASRB	INH	1
88	58	ASLB	INH	1	ASLB	INH	1
89	59	ROLB	INH	1	ROLB	INH	1
90	5A	DECB	INH	1	DECB	INH	1
91	5B	-	-	-	-	-	-
92	5C	INCB	INH	1	INCB	INH	1
93	5D	TSTB	INH	1	TSTB	INH	1
94	5E	-	-	-	-	-	-
95	5F	CLRB	INH	1	CLRB	INH	1
96	60	NEG	IND	2	NEG	IND	2
97	61	-	-	-	-	-	-
98	62	-	-	-	-	-	-
99	63	COM	IND	2	COM	IND	2
100	64	LSR	IND	2	LSR	IND	2
101	65	-	-	-	-	-	-
102	66	ROR	IND	2	ROR	IND	2
103	67	ASR	IND	2	ASR	IND	2
104	68	ASL	IND	2	ASL	IND	2
105	69	ROL	IND	2	ROL	IND	2
106	6A	DEC	IND	2	DEC	IND	2
107	6B	-	-	-	-	-	-
108	6C	INC	IND	2	INC	IND	2
109	6D	TST	IND	2	TST	IND	2
110	6E	JMP	IND	2	JMP	IND	2
111	6F	CLR	IND	2	CLR	IND	2
112	70	NEG	EXT	3	NEG	EXT	3
113	71	-	-	-	-	-	-
114	72	-	-	-	-	-	-
115	73	COM	EXT	3	COM	EXT	3
116	74	LSR	EXT	3	LSR	EXT	3
117	75	-	-	-	-	-	-
118	76	ROR	EXT	3	ROR	EXT	3
119	77	ASR	EXT	3	ASR	EXT	3
120	78	ASL	EXT	3	ASL	EXL	3
121	79	ROL	EXT	3	ROL	EXT	3
122	7A	DEC	EXT	3	DEC	EXT	3
123	7B	-	-	-	-	-	-
124	7C	INC	EXT	3	INC	EXT	3
125	7D	TST	EXT	3	TST	EXT	3
126	7E	JMP	EXT	3	JMP	EXT	3
127	7F	CLR	EXT	3	CLR	EXT	3
128	80	SUBA	IMM	2	SUBA	IMM	2
129	81	CMPA	IMM	2	CMPA	IMM	2
130	82	SBCA	IMM	2	SBCA	IMM	2
131	83	SUBD	IMM	2	SUBD	IMM	2
132	84	ANDA	IMM	2	ANDA	IMM	2
133	85	BITA	IMM	2	BITA	IMM	2
134	86	LDA	IMM	2	LDAA	IMM	2
135	87	-	-	-	-	-	-
136	88	EORA	IMM	2	EORA	IMM	2
137	89	ADCA	IMM	2	ADCA	IMM	2
138	8A	ORAA	IMM	2	ORAA	IMM	2
139	8B	ADDA	IMM	2	ADDA	IMM	2
140	8C	CMPX	IMM	3	CPX	IMM	3
141	8D	BSR	REL	2	BSR	REL	2
*142	8E	LDX	IMM	3	LDS	IMM	3
143	8F	-	-	-	-	-	-
144	90	SUBA	DIR	2	SUBA	DIR	2
145	91	CMPA	DIR	2	CMPA	DIR	2
146	92	SBCA	DIR	2	SBCA	DIR	2
147	93	SUBD	DIR	2	SUBD	DIR	2
148	94	ANDA	DIR	2	ANDA	DIR	2
149	95	BITA	DIR	2	BITA	DIR	2
150	96	LDA	DIR	2	LDAA	DIR	2
151	97	STA	DIR	2	STAA	DIR	2
152	98	EORA	DIR	2	EORA	DIR	2
153	99	ADCA	DIR	2	ADCA	DIR	2
154	9A	ORAA	DIR	2	ORAA	DIR	2
155	9B	ADDA	DIR	2	ADDA	DIR	2
156	9C	CMPX	DIR	2	CPX	DIR	2
*157	9D	JSR	DIR	2	-	-	-
*158	9E	LDX	DIR	2	LDS	DIR	2
*159	9F	STX	DIR	2	STS	DIR	2
160	A0	SUBA	IND	2	SUBA	IND	2
161	A1	CMFA	IND	2	CMPA	IND	2
162	A2	SBCA	IND	2	SBCA	IND	2
163	A3	SUBD	IND	2	SUBD	IND	2
164	A4	ANDA	IND	2	ANDA	IND	2
165	A5	BITA	IND	2	BITA	IND	2
166	A6	LDA	IND	2	LDAA	IND	2
167	A7	STA	IND	2	STAA	IND	2
168	A8	EORA	IND	2	EORA	IND	2
169	A9	ADCA	IND	2	ADCA	IND	2
170	AA	ORAA	IND	2	ORAA	IND	2
171	AB	ADDA	IND	2	ADDA	IND	2

accumulator holds $B0 and the B accumulator holds $A4, then the D accumulator holds $B0A4. We can use A and B separately or we can use them together as the D accumulator.

The time has come to start looking more closely at some of the opcodes to see how they operate.

There six different addressing modes:

- inherent(INH),
- direct (DIR),
- extended (EXT),
- immediate (IMM),
- indexed (IND), and
- relative (REL).

Each of these differ in the way an instruction is carried out. The INHERENT opcodes are complete instructions by themselves and need no other numbers to complete their instruction. They usually occupy 1 byte as seen in the # column of the opcode table.

The number in the # column tells you the total number of bytes for the complete operation. Some examples of inherent opcodes are:-

- INX: increment the X register (i.e. add 1 to the number stored in the X register)
- CLRA: CLEAR the A accumulator (i.e. make the A accumulator equal to zero)
- RTS: Return to sender
- INCB: increment the B accumulator.
- DECA: decrement the A accumulator.

In the IMMEDIATE mode, the number needed to complete the operation is found immediately following the opcode. For example:-

```
86 F3
```

... will load the A accumulator with the hex number F3, since $86 is the opcode for the

172	AC	CMPX	IND	2	CPX	IND	2	214	D6	LDB	DIR	2	LDAB	DIR	2
173	AD	JSR	IND	2	JSR	IND	2	215	D7	STB	DIR	2	STAB	DIR	2
*174	AE	LDX	IND	2	LDS	IND	2	216	D8	EORB	DIR	2	EORB	DIR	2
*175	AF	STX	IND	2	STS	IND	2	217	D9	ADCB	DIR	2	ADCB	DIR	2
176	B0	SUBA	EXT	3	SUBA	EXT	3	218	DA	ORAB	DIR	2	ORAB	DIR	2
177	B1	CMPA	EXT	3	CMPA	EXT	3	219	DB	ADDB	DIR	2	ADDB	DIR	2
178	B2	SBCA	EXT	3	SBCA	EXT	3	220	DC	LDD	DIR	2	LDD	DIR	2
179	B3	SUBD	EXT	3	SUBD	EXT	3	221	DD	STD	DIR	2	STD	DIR	2
180	B4	ANDA	EXT	3	ANDA	EXT	3	*222	DE	LDU	DIR	2	LDX	DIR	2
181	B5	BITA	EXT	3	BITA	EXT	3	*223	DF	STU	DIR	2	STX	DIR	2
182	B6	LDA	EXT	3	LDAA	EXT	3	224	E0	SUBB	IND	2	SUBB	IND	2
183	B7	STA	EXT	3	STAA	EXT	3	225	E1	CMPB	IND	2	CMPB	IND	2
184	B8	EORA	EXT	3	EORA	EXT	3	226	E2	SBCB	IND	2	SBCB	IND	2
185	B9	ADCA	EXT	3	ADCA	EXT	3	227	E3	ADDD	IND	2	ADDD	IND	2
186	BA	ORAA	EXT	3	ORAA	EXT	3	228	E4	ANDB	IND	2	ANDB	IND	2
187	BB	ADDA	EXT	3	ADDA	EXT	3	229	E5	BITB	IND	2	BITB	IND	2
188	BC	CMPX	EXT	3	CPX	EXT	3	230	E6	LDB	IND	2	LDAB	IND	2
189	BD	JSR	EXT	3	JSR	EXT	3	231	E7	STB	IND	2	STAB	IND	2
*190	BE	LDX	EXT	3	LDS	EXT	3	232	E8	EORB	IND	2	EORB	IND	2
*191	BF	STX	EXT	3	STS	EXT	3	233	E9	ADCB	IND	2	ADCB	IND	2
192	C0	SUBB	IMM	2	SUBB	IMM	2	234	EA	ORAB	IND	2	ORAB	IND	2
193	C1	CMPB	IMM	2	CMPB	IMM	2	235	EB	ADDB	IND	2	ADDB	IND	2
194	C2	SBCB	IMM	2	SBCB	IMM	2	236	EC	LDD	IND	2	LDD	IND	2
195	C3	ADDD	IMM	3	ADDD	IMM	3	237	ED	STD	IND	2	STD	IND	2
196	C4	ANDB	IMM	2	ANDB	IMM	2	*238	EE	LDU	IND	2	LDX	IND	2
197	C5	BITB	IMM	2	BITB	IMM	2	*239	EF	STU	IND	2	STX	IND	2
198	C6	LDAB	IMM	2	LDAB	IMM	2	240	F0	SUBB	EXT	3	SUBB	EXT	3
199	C7	-	-	-	-	-	-	241	F1	CMPB	EXT	3	CMPB	EXT	3
200	C8	EORB	IMM	2	EORB	IMM	2	242	F2	SBCB	EXT	3	SBCB	EXT	3
201	C9	ADCB	IMM	2	ADCB	IMM	2	243	F3	ADDD	EXT	3	ADDD	EXT	3
202	CA	ORAB	IMM	2	ORAB	IMM	2	244	F4	ANDB	EXT	3	ANDB	EXT	3
203	CB	ADDB	IMM	2	ADDB	IMM	2	245	F5	BITB	EXT	3	BITB	EXT	3
204	CC	LDD	IMM	3	LDD	IMM	3	246	F6	LDB	EXT	3	LDAB	EXT	3
205	CD	-	-	-	-	-	-	247	F7	STB	EXT	3	STAB	EXT	3
*206	CE	LDU	IMM	3	LDX	IMM	3	248	F8	EORB	EXT	3	EORB	EXT	3
207	CF	-	-	-	-	-	-	249	F9	ADCB	EXT	3	ADCB	EXT	3
208	D0	SUBB	DIR	2	SUBB	DIR	2	250	FA	ORAB	EXT	3	ORAB	EXT	3
209	D1	CMPB	DIR	2	CMPB	DIR	2	251	FB	ADDB	EXT	3	ADDB	EXT	3
210	D2	SBCB	DIR	2	SBCB	DIR	2	252	FC	LDD	EXT	3	LDD	EXT	3
211	D3	ADDD	DIR	2	ADDD	DIR	2	253	FD	STD	EXT	3	STD	EXT	3
212	D4	ANDB	DIR	2	ANDB	DIR	2	*254	FE	LDU	EXT	3	LDX	EXT	3
213	D5	BITB	DIR	2	BITB	DIR	2	*255	FF	STU	EXT	3	STX	EXT	3

table 3

EXTRA COCO DOUBLE BYTE OPCODES:-

OP	INH	DIR	EXT	IMM	IND
CMPD		1093 3	10B3 4	1083 4	10A3 3
CMPS		119C 3	11BC 4	118C 4	11AC 3
CMPU		1193 3	11B3 4	1183 4	11A3 3
CMPY		109C 3	10BC 4	108C 4	10AC 3
LDS		10DE 3	10FE 4	10CE 4	10EE 3
LDY		109E 3	10BE 4	108E 4	10AE 3
STS		10DF 3	10FF 4		10EF 3
STY		109F 3	10BF 4		10AF 3
SWI2	103F 2				
SWI3	113F 2				

LONG BRANCHES 102* LB**

mnemonic LDA.

It has a 2 in the # column so that 2 bytes are needed for the complete operation. In assembly language it would be written as:-

```
LDA #$F3
```

... which means "Load the A accumulator with the hex number F3".

Some more examples are :-

```
C6 A9 LDA #$A9
 (LOAD B WITH HEX A9)

8E 1E49 LDX #$1E49
 (LOAD X WITH HEX 1E49 (CoCo))

CE 1E49 LDX #$1E49
 (LOAD X WITH HEX 1E49 (MC-10))
```

In the DIRECT addressing mode, the number needed is found in the direct page RAM, most usually between $0000 and $00FF (the CoCo can alter the DP). For example :-

```
96 B7
```

... will load the A accumulator with the number stored in memory location $00B7. This can be written as :-

```
LDA $B7
```

Some more direct operations are:-

```
D6 FE  LDB  $FE
 (LOAD B WITH NO. STORED AT
  $00FE)

9E FB  LDX  $FB
 (LOAD X WITH NO. STORED AT
  $00FB AND $00FC (CoCo))

DE FB  LDX  $FB
 (LOAD X WITH NO. STORED AT
  $00FB AND $00FC (MC-10))

97 7B  STA  $7B
 (STORE THE CONTENTS OF A IN
  LOCATION $007B)

DD A4  STD $A4
 (STORE THE CONTENTS OF B IN
  LOCATIONS $00A4 AND $00A5)
```

Note with double registers or accumulators that the direct number indicates the first of the two bytes that are involved, ie in STD $A4, the $A4 is for the M.S.B. ($00A4) and because D is a double accumulator, $00A5 is where the L.S.B. is stored.

If the D accumulator contains $20F9 then STD $A4 will store $20 in location $00A4 and $F9 in $00A5.

In the INDEXED mode, the index register X (and Y in the 6809) have control over the address involved. The CoCo has a variety of indexing forms but I'll stick to the simplest ones here. For example ...

```
A6 84  (A6 00 FOR MC-10)
```

... will load the A accumulator with the number stored in the memory location which is pointed to by the X register. It is usually shown as:-

```
LDA  ,X
```

If the X register contains $FE73, then this instruction will load the A accumulator with the contents of memory location $FE73. The 84 shown in the CoCo instruction is the code for the X register.

Some more examples of indexed addressing are (MC-10 instructions are in brackets, and what they do are in square brackets, ie <something>)):-

```
A6 A4 (NONE)  LDA  ,Y
 <LOAD A WITH THE CONTENTS OF
  THE LOCATION POINTED TO BY Y>

E6 84 (E6 00) LDB  ,X
 <LOAD B WITH THE CONTENTS OF
  THE LOCATION POINTED TO BY X>

E6 A4 (NONE)  LDB  ,Y
 <LOAD B WITH THE CONTENTS OF
  THE LOCATION POINTED TO BY Y>

A7 84 (A7 00) STA  ,X
 <STORE A IN THE LOCATION
  POINTED TO BY X>

AF 84 (EF00)  STX ,X
 <STORE X IN THE LOCATION
  POINTED TO BY X>
```

Notice that, in the CoCo the code number for the X register (no offset) is $84 and for the Y it is $A4. The MC-10 has $00 for no offset.

For an explanation of the other indexing functions of the CoCo, see "TRS80 COLOR COMPUTER ASSEMBLY LANGUAGE PROGRAMMING" by William Barden Jr Catalogue number 62-2077.

In the EXTENDED mode, the address concerned is given as part of the instruction.

```
B6 FE1F
```

... will load the A accumulator with the contents of memory location $FE1F. This can be written on:-

```
LDA $FE1F
```

Some more extended instructions are:-

```
F6 E614  LDB  $E614
 <Load B with contents of
  $E614>

B7 71AB  STA  $71AB
 <Store A at $71AB>

BF 50FE  STX $50FE
 <Store X at $50FE (CoCo)>

FF 50FE  STX $50FE
 <Store X at $50Fe (MC-10)>
```

The last addressing mode is

the relative one. This one works a bit like IF THEN GOTO in BASIC.

Then again it's more like those board games, where you're told to go 4 spaces backwards or 10 spaces forwards.

The relative branches usually check the condition code register to find the results of the previous operation before it decides what to do.

Some examples are:

- BRA - Branch always
- BNE - Branch if not equal
- BEQ - Branch if equal
- BMI - Branch if minus
- BSR - Branch to subroutine

In order to be able to use relative branches you've got to be able to count forward and backward in HEX. Alternatively you could look up a table of TWO'S COMPLIMENT numbers. The easiest way to use an assembler and let the computer worry about it.

Just to show you the way it works, look at this short ML program.

```
 7000  86 F3         LDA  #$F3
 7002  C6 00         LDB  #$00
 7004  4A      LOOP  DECA
 7005  5C            INCB
 7006  81 00         CMPA #$00
 7008  27 XX         BEQ  END
 700A  20 ZZ         BRA  LOOP
 700C  39       END  RTS
```

The program starts at $7000 in memory although it could be located anywhere in memory. There are 2 relative branching jumps. The one which jumps forwards (BEQ) is no problem because you count forwards starting with 00 as below. The count comes to 02.

```
MEMORY BYTE COUNT

 7008    27    FE
 7009    XX    FF
 700A    20    00
 700B    ZZ    01
 700C    39    02
 700D          03
```

To get to location $700C, so XX is 02. To get the number to go with the backward jump, we need to set out the count in the same way.

```
MEMORY BYTE COUNT

 7004    4A    F8
 7005    5C    F9
```

table 5

```
3F00               00100        ORG    $3F00
3F00 8E   A000     00110 START  LDX    #$A000   LOAD X REG WITH THE START OF ROM
3F03 BF   0400     00120        STX    $0400    STORE ROM START AT $0400
3F06 8E   0420     00130        LDX    #$0420   LOAD X REG WITH THE START OF SECOND SCREEN LINE
3F09 BF   0402     00140 LOOP   STX    $0402    STORE X AT $0402
3F0C BE   0400     00150        LDX    $0400    LOAD X WITH LOCATION STORED IN $0400
3F0F A6   84       00160        LDA    ,X       LOAD A WITH BYTE STORED WHERE X POINTS
3F11 30   01       00170        LEAX   1,X      ADD 1 TO X REGISTER
3F13 BF   0400     00180        STX    $0400    STORE X AT $0400
3F16 BE   0402     00190        LDX    $0402    LOAD X FROM $0402
3F19 A7   84       00200        STA    ,X       STORE A AT THE ADDRESS POINTED TO BY X
3F1B 30   01       00210        LEAX   1,X      ADD 1 TO X
3F1D 8C 0600       00220        CMPX   #$0600   COMPARE TO SCREEN END
3F20 26   E7       00230        BNE    LOOP     BRANCH TO LOOP IF NOT SCREEN END
3F22 39            00240        RTS             RETURN
          0000     00250        END
00000 TOTAL ERRORS


LOOP    3F09
START   3F00
```

MC-10 (4K TO 20K)

```
4F00 CE E000 START LDX #$E000 Load index reg. with start of ROM
4F03 FF  4000      STX  $4000 Store ROM start at $4000
4F06 CE  4020      LDX #$4020 Load X with start of 2nd screen line
4F09 FF 4002  LOOP STX  $4002 Store X at $4002
4F0C FE 4000       LDX  $4000 Load X with ROM location stored in $4000-1
4F0F A6 00         LDAA ,X    Load A with byte stored where X is pointing
4F11 08            INX        Increase X by one
4F12 FF 4000       STX  $4000 Store X reg. at $4000
4F15 FE 4002       LDX  $4002 Load X from 4002
4F18 A7 00         STAA ,X    Store A at address in X reg.
4F1A 08            INX        Increase X by one
4F1B 8C 4200       CPX  #$4200 Compare X to end of screen
4F1E 26 E9         BNE LOOP   Branch if not screen end
4F20 39            RTS        Return
```

From P 30

```
"
50200 '
50500 DATA CC,30,31,FD,FF,A4,CC,
32,33,FD,FF,A6,20,15,CC,3C,3D,FD
,FF,A4,CC,3E,3F,FD,FF,EFF
50510 DATA A6,20,11,1A,50,21,E0,
7F,FF,DF,86,2,1F,8B,39,21,E4,7F,
FF,DE,1C,AF,4F,1F,8B,B2F
50520 DATA 39,59,D8,FE,0,59,D8,8
D,E8,17,0,B5,1F,1,1F,B8,C6,50,1F
,3,CC,B,3,DD,10,9D0
50530 DATA 17,0,A5,ED,C1,30,5,A,
10,26,F5,30,8C,DA,EC,81,ED,C1,A,
11,26,F8,96,5E,9A,B4C
50540 DATA 5F,97,6C,20,C6,8D,B1,
F,9,96,57,97,C,9E,6A,9F,10,CC,0,
2,DD,E,30,2,9F,96F
50550 DATA 6A,DE,58,9E,E,E6,C4,3
A,30,1,9F,E,9E,6A,3A,30,1,9C,68,
24,37,9E,6A,E7,80,A4F
50560 DATA 5D,27,A,10,AE,42,A6,A
0,A7,80,5A,26,F9,9F,6A,33,45,A,C
,26,D5,9E,10,DC,E,99E
50570 DATA ED,84,20,13,17,FF,66,
DE,58,D6,57,6F,C4,33,45,5A,26,F9
,DC,A,FD,0,23,16,FF,BC2
50580 DATA 5F,C,9,20,F9,17,FF,4C
,8D,3D,10,9E,58,9E,5A,96,57,97,1
0,E6,C0,E7,A4,AF,22,B4D
50590 DATA 5D,27,7,A6,C0,A7,80,5
A,26,F9,31,25,A,10,26,EA,20,D3,6
D,84,27,15,86,90,A0,9E7
50600 DATA 84,97,C,EC,1,8A,80,D,
C,27,6,44,56,A,C,26,FA,39,4F,5F,
39,9E,50,DE,66,886
50610 DATA 6D,84,27,E,8D,DA,1F,1
,30,1F,27,6,EC,C4,33,CB,26,F6,33
,42,39,9E,54,6D,84,984
50620 DATA 27,F9,8D,C3,D7,C,4F,E
6,C4,33,CB,33,41,A,C,26,F6,39,17
,FE,DB,F,9,DC,5E,B6B
50630 DATA DD,12,9E,52,8D,A8,DD,
E,9E,62,9F,10,10,9E,5C,9E,10,8D,
B9,8D,CE,9E,12,E6,C0,C5D
50640 DATA D,6C,27,29,3A,9C,60,1
0,24,FF,60,9E,12,E7,A4,AF,22,A6,
C0,A7,80,5A,26,F9,9F,B43
50650 DATA 12,31,25,9E,10,30,5,9
F,10,9E,E,30,1F,9F,E,26,CC,16,FE
,9D,E7,A4,EF,22,20,901
50660 DATA E7,17,FE,88,17,FF,78,
33,5E,1F,31,EC,84,30,8B,A6,80,A7
,C0,9C,6A,25,F8,DF,6A,D17
50670 DATA 20,DD,17,FE,6E,F,9,9E
,52,17,FF,3E,DD,E,10,9E,64,9E,5C
,9F,14,A6,84,97,1B,A62
50680 DATA EC,2,DD,18,6F,A0,DC,1
8,DD,16,96,1B,97,1A,9E,E,30,1F,2
7,B2,9F,E,9E,14,30,99B
50690 DATA 5,9F,14,A6,84,97,1B,E
C,2,DD,18,8D,2,20,DE,D6,1A,D1,1B
,26,15,5D,27,D,9E,94A
50700 DATA 16,DE,18,A6,80,A1,C0,
26,8,5A,26,F7,96,9,A7,A0,39,C,9,
20,F7,17,FE,10,17,9BF
50710 DATA 0,E9,16,FE,16,17,FE,7
,17,2,A0,20,F5,4FD
50720 DEL 50000-50720
```

From P 26

per record in the program area for the pointers etc required for their operations.

The "XSORT/XSEARCH" routines provide a more limited set of options, but require no additional space in the program area for their operations. These routines provide the basic functions required and allow the buffer to be used for any number of records that will physically fit in the buffer.

HINT...

Drawing a shape ?
Don't use so many (H)LINE commands ! - use the (H)DRAW command. It's easy :

```
BM = X,Y STARTING CO-ORDINATES
Rx = RIGHT x PIXELS
Lx = LEFT x PIXELS
Ux = UP x PIXELS
Dx = DOWN x PIXELS
Ex = 45 DEGREE ANGLE x PIXELS
Fx = 135 DEGREE ANGLE x PIXELS
Gx = 225 DEGREE ANGLE x PIXELS
Hx = 315 DEGREE ANGLE x PIXELS
S  = SCALE
C  = COLOR
```

A good idea when drawing a picture on the CoCo 3 hi-res graphics screen is to photocopy the 640 by 192 graphics worksheet in the CoCo 3 manual

```
7006   81   FA
7007   00   FB
7008   27   FC
7009   02   FD
700A   20   FE
700B   ZZ   FF
```

To count in reverse from $700B, back to $7004 the count comes to F8, so ZZ is F8.

That covers all of the main types of addressing modes. For more detailed explanations see one of the assembly language reference books. Look at as many assembly listings as you can and try to work out how they operate. You will learn a lot from doing that.

Now, before I go on to show you a working ML program, see if you can answer these short questions.

1. Where is the screen memory on your computer?

2. What are each of these HEX numbers in decimal?

(a) $7000
(b) $EFB4
(c) $ABCD

3. What do each of the following do?

(a) LDA #$F2
(b) LDX #$701E
(c) LDA $FE
(d) STX $FB
(e) LDA ,X
(f) STX ,X
(g) LDA $701E
(h) STB $50F9

4. What are the opcodes for all the LDA instructions? (do you see a pattern)

Now for a ML program that you can sink you teeth into. This program will move part of the ROM into screen memory.

<see table 5>

Both of these programs will do the same thing. The CoCo one is slightly longer because it uses LEAX (instead of INX) to allow any value to be added to X. If you want to move a different part of the ROM into screen memory then you need to change the value at the START. ($A000 is the start of COLOR BASIC ROM and $E000 is the start of MICROCOLOR BASIC ROM).

Study the programs and see how the four bytes at the start of the screen memory can be used to keep a record of the memory locations which we are loading and storing.

Just to make sure that all of you can enter this program, I've provided BASIC LOADERS which were written by changing all the HEX numbers to decimal and putting them into DATA lines.

If you want to alter the chunk of memory that is moved onto the screen then you need to alter the numbers in line 70 of the BASIC loader.

HAVE FUN.

CoCoFish

Let's see who the REAL flathead is!

by Aaron West

32K ECB GAME

THE IDEA for this program is to try and catch as much fish as you can before you run out of fuel. I wrote this one weekend when I was bored.

More instructions are contained in the program.

ENJOY!

The Listing:

```
0 GOTO 10
1 '################################
2 '#      LET'S GO FISHING         #
3 '#   HOW MANY FISH CAN YOU       #
4 '#           CATCH               #
5 '#                               #
6 '#  NOVEMBER 1987 WESTSOFT       #
7 '#     BY AARON WEST(10)         #
8 '################################
9 SAVE"95:3":END'1
10 CLS0
11 'SETTING UP
12 B$="F5R5E5L7D5U2U10G5R10H5D7L
7"
13 L=30
14 K=0
15 XX=122:YY=30
16 Q=RND(-TIMER)
17 'NOTE:!!!!!!!!!!!!!!!!!!!!!!!!!
18 '! THROUGH THIS LISTING YOU !
19 '!     WILL SEE LOWERCASE    !
20 '!    WRITING. TO GET THIS   !
21 '!  PRESS SHIFT&0 TOGETHER   !
22 '! BUT YOU WILL SEE THIS AS !
23 '!     INVERSED WRITING.     !
24 '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
25 PRINT"please"+CHR$(128)+"wait
";
26 DIMW$(90)
27 FORA=0TO90:READW$(A):NEXT
28 'TITLE SCREEN
29 PMODE 4:PCLS 1:COLOR 0
30 DRAW"BM65,20":S$="Let's Go Fi
shing.":GOSUB177
31 DRAW"BM30,180":S$="1987 Wests
oft by Aaron West":GOSUB177
32 DRAW"BM15,80":S$="'I' for INS
TRUCTIONS or any other":GOSUB177
33 DRAW"BM80,90":S$="key to begi
n.":GOSUB177
34 DRAW"BM1,1":S$="Westsoft Pres
ents...":GOSUB177
35 SCREEN1,1
36 A$=INKEY$:IF A$="" THEN 36 EL
SE IF A$="I" THEN 38 ELSE 59
37 'INSTRUCTIONS
38 PCLS1:DRAW"BM65,5":S$="Let's
Go Fishing":GOSUB177
39 DRAW"BM75,15":S$="Instruction
s":GOSUB177
40 LINE(0,25)-(255,191),PSET,B
41 DRAW"BM5,30":S$="The object o
f this game is to catch":GOSUB17
7
42 DRAW"BM5,40":S$="the most fis
h before you run out of":GOSUB17
7
43 DRAW"BM5,50":S$="fuel. You ha
ve a grid map to show":GOSUB177
44 DRAW"BM5,60":S$="where you ar
e. In the top left":GOSUB177
45 DRAW"BM5,70":S$="corner of th
e map is the wharf":GOSUB177
46 DRAW"BM5,80":S$="where you st
art off. When your fuel":GOSUB17
7
47 DRAW"BM5,90":S$="supply is ve
ry low or you just want":GOSUB17
7
48 DRAW"BM5,100":S$="to quit go
to the dock and your":GOSUB177
49 DRAW"BM5,110":S$="score will
be displayed and the":GOSUB177
50 DRAW"BM5,120":S$="game will e
nd. To move around the":GOSUB177
51 DRAW"BM5,130":S$="grid map us
e N,S,E,W. You will be":GOSUB177
52 DRAW"BM5,140":S$="told how mu
ch fish you have caught":GOSUB17
7
53 DRAW"BM5,150":S$="and if you
caught nothing. Dont":GOSUB177
54 DRAW"BM5,160":S$="crash into
the edges or else!":GOSUB177
55 DRAW"BM5,170":S$="HAVE FUN...
...........":GOSUB177
56 DRAW"BM70,180":S$="(Press Any
 Key)":GOSUB177
57 EXEC44539
58 'THE GAME
59 PCLS1:DRAW"BM65,5":S$="Let's
Go Fishing":GOSUB177
60 DRAW"BM1,100;R20L3D5L2U5L10D5
L2U5"
61 DRAW"BM5,50":S$="Have a good
trip.":GOSUB177
62 DRAW"BM5,120":S$="You are now
 leaving the wharf.....":GOSUB17
7
63 FOR X=25 TO 235
64 DRAW"BM"+STR$(X)+",100;C0"+B$
65 DRAW"BM"+STR$(X)+",100;C1"+B$
66 NEXT
67 DRAW"C0":GOSUB162
68 GOSUB162
69 GOSUB145
70 'GOING IN A DIRECTION
71 A$=INKEY$:IF A$="" THEN 71
72 IF A$="N" THEN D=1:GOTO78
73 IF A$="S" THEN D=2:GOTO78
74 IF A$="E" THEN D=3:GOTO78
75 IF A$="W" THEN D=4:GOTO78
76 GOTO 71
77 'CHECKS
78 IF D=1 THEN YY=YY-20
79 IF D=2 THEN YY=YY+20
80 IF D=3 THEN XX=XX+20
81 IF D=4 THEN XX=XX-20
82 IF YY<20 OR XX<100 THEN 89
83 IF YY>130 OR XX>240 THEN 89
84 IF XX=102 AND YY=30 THEN GOTO
 109
85 IF L=0 THEN 128
86 IF L<5 THEN M=1
87 GOTO 68
88 'CRASH!!!!!!!!!!!!!!!!!!!!!!!!
89 FOR D=1 TO 100
90 X=RND(255):Y=RND(191)
91 XX=RND(255):YY=RND(191)
92 LINE(X,Y)-(XX,YY),PSET
93 NEXT D
94 SOUND1,10
95 PCLS1:LINE(0,0)-(255,191),PSE
T,B
96 DRAW"BM5,3":S$="I told you no
t to crash!!!":GOSUB177
97 DRAW"BM5,13":S$="All the fish
 went overboard and you":GOSUB17
7:DRAW"BM5,23":S$="died!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!":GOSUB177
98 DRAW"BM100,96D56M152,168U66M-
4,-10M-12,-8M-8,-2L12M-16,+14M+1
6,-14M+14,-6R14M+14,+6M+6,+12D64
M-12,+10"
99 DRAW"BM105,100":S$="REST":GOS
UB177
100 DRAW"BM105,110":S$="IN":GOSU
B177
101 DRAW"BM105,120":S$="PEACE":G
OSUB177
102 DRAW"BM5,180":S$="Another Ga
me Perhaps?":GOSUB177
103 A$=INKEY$:IFA$="" THEN103
```

```
104 IF A$="N" THEN CLS:PRINT"SEE
 YOU AGAIN SOMETIME":END
105 IF A$="Y" THEN RUN
106 GOTO 103
107 GOTO 107
108 'COMING BACK HOME
109 PCLS1:LINE(0,0)-(255,191),PS
ET,B
110 LINE(20,20)-(235,171),PSET,B
111 PAINT(5,5),0
112 DRAW"BM25,25":S$="You have c
aught"+STR$(K)+" KILOGRAMS":GOSU
B177
113 DRAW"BM25,35":S$="of fish.":
GOSUB177
114 DRAW"BM25,50":S$="I hope you
 have had a good":GOSUB177:DRAW"
BM25,60":S$="Trip":GOSUB177
115 DRAW"BM25,120":S$="You are n
ow entering the wharf...":GOSUB1
77
116 DRAW"BM21,100;R20L3D5L2U5L10
D5L2U5"
117 FOR X=215 TO 45 STEP -1
118 DRAW"BM"+STR$(X)+",100;C0"+B
$
119 DRAW"BM"+STR$(X)+",100;C1"+B
$
120 NEXT X
121 DRAW"C0
122 DRAW"BM25,160":S$="Another g
ame perhaps?":GOSUB177
123 A$=INKEY$:IF A$="" THEN 123
124 IF A$="N" THEN CLS:PRINT"SEE
 YOU AGAIN SOMETIME":END
125 IF A$="Y" THEN RUN
126 GOTO 123
127 'WHEN YOUR TANK IS DRY
128 FOR X=1 TO 50:EXEC43350:NEXT
:FORX=1TO200:NEXT
129 FOR X=1 TO 200:EXEC43350:NEX
T:FORX=1 TO100:NEXT
130 FORX=1 TO 10:EXEC43350:NEXT:
FORX=1TO50:NEXT:FORX=1TO5:EXEC43
350:NEXT
131 PCLS1:LINE(0,0)-(255,191),PS
ET,B
132 DRAW"BM5,5":S$="The boat sta
lls to a holt as you":GOSUB177
133 DRAW"BM5,15":S$="Relize that
 you have no fuel. Are":GOSUB177
134 DRAW"BM5,25":S$="you strande
d here forever? Are you":GOSUB17
7
135 DRAW"BM5,35":S$="you stuck o
ut her with nobody but":GOSUB177
136 DRAW"BM5,45":S$="the fish? F
ortunately the rescue":GOSUB177
137 DRAW"BM5,55":S$="service has
 found you, BUT ALL":GOSUB177
138 DRAW"BM5,65":S$="the fish ha
s gone all smelly and":GOSUB 177
139 DRAW"BM5,75":S$="rotten. But
 you were lucky!!!!!!!":GOSUB177
140 DRAW"BM50,150":S$="Another g
ame perhaps?":GOSUB177
141 A$=INKEY$:IF A$="" THEN 141
142 IF A$="Y" THEN RUN
143 IF A$="N" THEN CLS:PRINT"SEE
 YOU AGAIN SOMETIME":END
144 GOTO 141
145 DRAW"BM1,150":S$="FISHING RE
SULTS:":GOSUB177
146 'CATCHING FISH
147 L=L-1
148 Q=RND(1000)
149 IF Q>=500 THEN GOSUB160:GOTO
158
150 Q=RND(5)
151 IF Q<2 THEN GOTO 150
152 F=Q
153 Q=RND(7)
154 FOR X=1 TO Q
155 K=K+F
156 NEXT X
157 DRAW"BM1,170":S$="You caught
"+STR$(Q)+" fish at"+STR$(F)+" K
G.":GOSUB177
158 FOR X=1 TO 500:NEXT:RETURN
159 RETURN
160 DRAW"BM1,160":S$="You caught
 nothing!!!":GOSUB177:RETURN
161 EXEC44539
162 PCLS1:DRAW"BM65,5":S$="Let's
 Go Fishing":GOSUB177
163 'THE MAIN SCREEN
164 FORX=100 TO 250 STEP20:LINE(
X,20)-(X,140),PSET:NEXT
165 FORY=20 TO 150 STEP20:LINE(1
00,Y)-(240,Y),PSET:NEXT
166 DRAW"BM"+STR$(XX)+","+STR$(Y
Y)+";"+B$
167 DRAW"BM105,25":S$="W":GOSUB1
77
168 DRAW"BM1,20":S$="$$$ STATUS
$$$":GOSUB177
169 DRAW"BM1,30":S$="You have"+S
TR$(L)+" li":GOSUB177
170 DRAW"BM1,40":S$="-tres of fu
el":GOSUB177
171 DRAW"BM1,50":S$="You have"+S
TR$(K)+"":GOSUB177
172 DRAW"BM1,60":S$="KG of fish.
...":GOSUB177
173 DRAW"BM1,70":S$="N,S,E,W":GO
SUB177
174 IF M=1 THEN DRAW"BM1,80":S$=
"LOW FUEL WARN-":GOSUB177:DRAW"B
M1,90":S$="ING!!!!!!!!!!!!":GOSUB
177
175 RETURN
176 'DRAWS LETTERS ON THE PMODE4
 GRAPHIC SCREEN
177 FORA=1TOLEN(S$):DRAWW$(ASC(M
ID$(S$,A,1))-32)+EX$:NEXT:RETURN
178 'LETTERING DATA
179 DATA BR7,BR2D4BD2D0BU6BR5,BR
DBR2UBR4,BD2R4HD4EL4FU4BUBR6,BR4
BDL4D2R4D2L4R2DU6BR5,DRUBR3DG4DB
R3URDBU6BR3,BRRFG3DFRE2BD2H4UBUB
R7,BRDRUBR5,BR3G2D2F2BU6BR4,BRF2
D2G2BU6BR6,BD3R4BD2H4BD4E4BUBR3,
BD3R4BG2U4BUBR5,BD6BR2GBU7BR6,BD
3R4BU3BR3
180 DATA BD6BR2R0BU6BR5,BD6UE4UB
R3,BDD4FR2EU4HL2BD3BRR0BE3BR2,BR
2D6RL2BU5EBR5,BDER2FDG4R4BU6BR3,
BDER2FDGL2R2FDGL2HBE5BR2,D3R4LD3
U6BR4,R4L4D3R4D3L4BE6BR,BDD4FR2E
UHL2BU3R2FBEBR2,DUR4D2G3DBE6,BDD
FR2FDGL2HUER2EUHL2BR6,BRR2FD4GL2
HBU4DFR3BE3
181 DATA BD3BR2D0BD3U0BU6BR5,BD3
BR2D0BD3GBU7BR6,BR3G3F3BU6BR4,BD
2R4BD2L4BE4BR3,BRF3G3BE6,BD2UER2
FD2L2DBD2U0BU6BR5,R4D4L2U2R2BD4L
4U6BR7
182 DATA BDD5U2R4D2U5HL2BR6,D6R3
EUHL2R2EUHL2BR6,BDD4FR2UBU4HL2BR
6,D6R2E2U2H2LBR6,D3R3L3D3R4BU6L4
BR7,D6U3R3L3U3R4BR3,BDD4FR2EU2L2
R2BU2HL2BR6,D6U3R4D3U6BR3,R4L2D6
L2R4BU6BR3,BD4DFR2EU5BR3,D6U3RF3
H3E3BR3,D6R4BU6BR3,D6U5RFDUERD5U
6BR3
183 DATA D6U5RFD2F2U6BR3,BDD4FR2
EU4HL2BR6,D6U3R3EUHL2BR6,BDD4FRE
HF2HEU3HL2BR6,D6U4F4H3R2EUHL2BR6
,BDDFR2FDGL2HBE4HL2BR6,R2D6U6R2B
R3,D6R4U6BR3,D3FDFEUEU3BR3,D5FEU
DFEU5BR3,DF4DBL4UE4UBR3,DFDFD2U2
EUEUBR3,R4DG4DR4BU6BR3
184 DATA BRR2L2D6R2BU6BR4,BD8LR6
BU8BR2,BRR2D6L2BE6,BD2E2D6U6F2BU
2BR3,BL7,BR7
185 DATA BD2R3FD3L3HER3BU4BR3,D6
R3EU2HL2BU2BR6,BD3D2FR2EBU2HL2BU
2BR6,BD3D2FR3U4L3R3U2BR3,BD3DR4U
HL2GD2FR3BU6BR3,BD3R3L2D3U5ERFBE
BR2,BD3D2FR3DGL3BR4BUU5L3R3BU2BR
3,D6U4R3FD3BU6BR3,BDBR2D0BD2D3BU
6BR5,BD7FR2EU4BU2U0BUBR3,D6U3F3H
2E2BU2BR4
186 DATA BR2D6RBU6BR4,BD2D4U4R2D
4U4RFD3BU6BR3,BD2D4U4R3FD3BU6BR3
,BD3D2FR2EU2HL2BU2BR6,BD2D6U2R3E
U2HL2BU2BR6,BD3D2FR3D2U6L3BU2BR6
,BD2D4U2E2R2BU2BR3,BD3FR2FGL3BE4
L3BU2BR6,BD2R2LU2D5FEBU5BR4,BD2D
3FR2EU3BU2BR3,BD2DFDFEUEUBU2BR3,
BD2D3FEUDFEU3BU2BR3
187 DATA BD2F4H2G2E4BU2BR3,BD2DF
DFG2E3UEUBU2BR3,BD2R4G4R4BU6BR3
188 'NOTE:!!!!!!!!!!!!!!!!!!!!!!
189 '! IF YOU WISH LEAVE OUT  !
190 '!  ALL THE REMARK LINES  !
191 '!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

[redacted]

From P 43

```
548 HPRINT(Y,17),AA:GOSUB514:RET
URN
549 '
550 '
551 ' ** DRAW GREEN INFO BOX
552 HCOLOR0:HLINE(3,116)-(316,17
0),PSET,B:HCOLOR1:HPRINT(10,22),
"<BREAK> TO RE-START":RETURN
553 '
554 '
555 '** GOTO IF 'BRK' PRESSED OR
 ERROR
556 GOTO32
```

DR COCO

Dear Dr CoCo,

Below this letter is a copy of a Flowchart Template ... wonderful little shapes, aren't they? Now, I have this aching problem - I am the proud owner of this template, but I don't know how to use it!

Now, how about a few ideas on how to use it, through our great magazine, CoCo?

Then I, and hundreds of others (when they buy their template) can set to write lots and lots of programs to send to you. Then into the mag they go and more ideas get around and more programs are written and and and ... soon we will need a bigger magazine.

Gee, just enough hints on it to enable me to get one program in print would be great - then I can feel as if I have really done something.

Ron Munro,
Woomera, SA

Ron,

Using flowcharts to design programs is one of the most difficult ways that I know of to write programs, apart from writing them on paper.

They are easy to write, but as soon as the program gets big or needs improvements, then it becomes a pain in the neck to modify.

Flowcharts, in programming and non-programming situations, are used to describe, or show, the flow of data. They can be (and often are) used in more ways other than showing how a program works.

To prove my point, I'll give you a few examples in two forms:

One: to show how programs can be written using flowcharts, and...

Two: how, say, an application can be described using flowcharts.

One: A few programs:

First example: The first program will flip a coin 50 times, counting the number of heads and the number of tails.

Program version: coinflip

```
10 IF I>49 THEN 70
20 I=I+1
30 C=RND(2)
40 IF C=1 THEN
    PRINT"HEADS!":H=H+1:GOTO60
50 PRINT"TAILS!":T=T+1
60 GOTO10
70 PRINT"NUMBER OF HEADS:"H
80 PRINT"NUMBER OF TAILS:"T
90 END
```

Flowchart version: coinflip

<flow chart one>

Small program, medium-sized flowchart. The flowchart was more work and took me 2 minutes to draw (neatly) as opposed to writing the above program, which took me under half a minute.

How it works:

It first asks whether or not 'I' is greater than 49 - one greater than 49 is 50, right? (See line 10.)

If 'I' is greater than 49, it will print the number of heads and the number of tails (lines 70 to 90) and end the program.

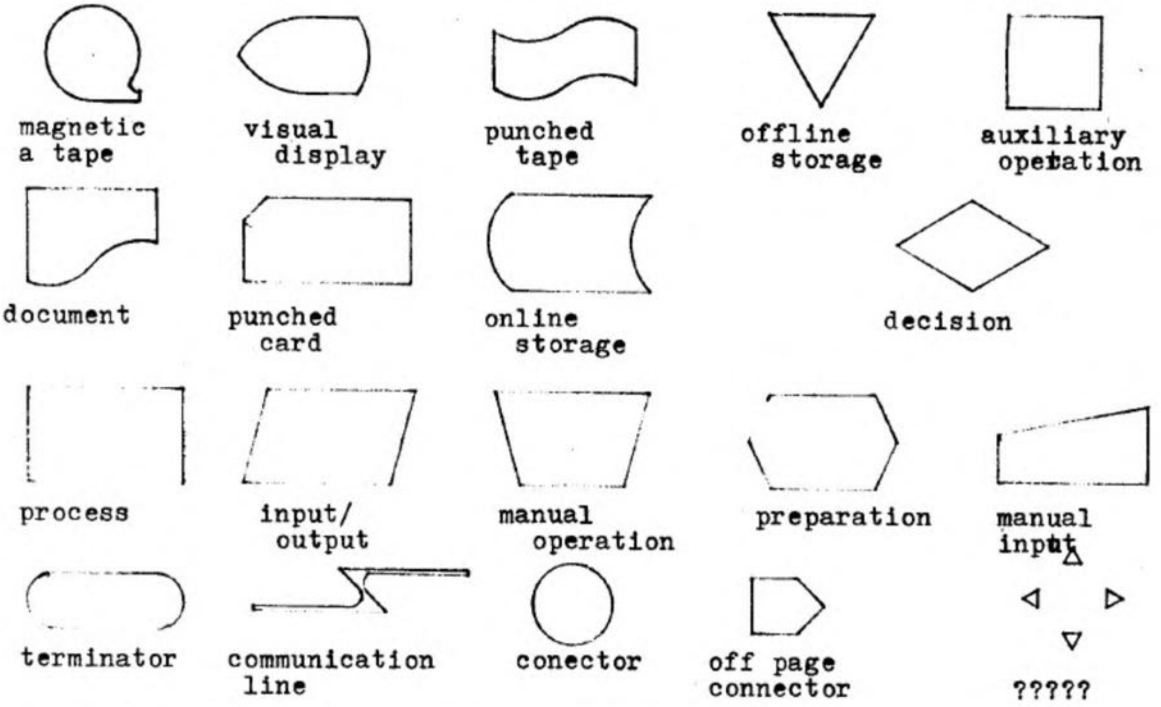
If 'I' is not greater than 49 (ie, 3, 5 or 49) then it adds one to 'I' (line 20).

It then picks a random number (line 30) and determines if the result was a '1'; if it was, it prints the word "HEADS!" and adds one to 'H' (line 40).

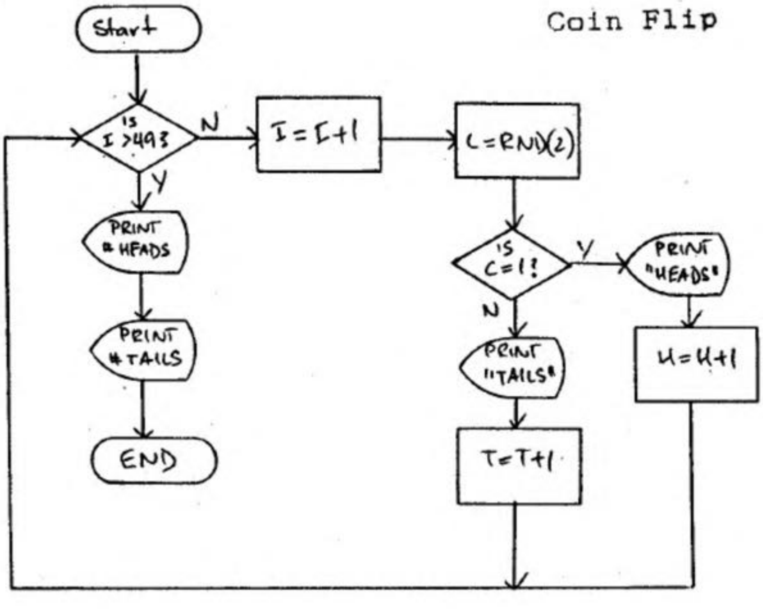
Because there can only be two answers, and, assuming it failed the first test, there can only be one other answer; it therefore prints the word "TAILS!" and adds one more to 'T' (line 50).

The whole process is then

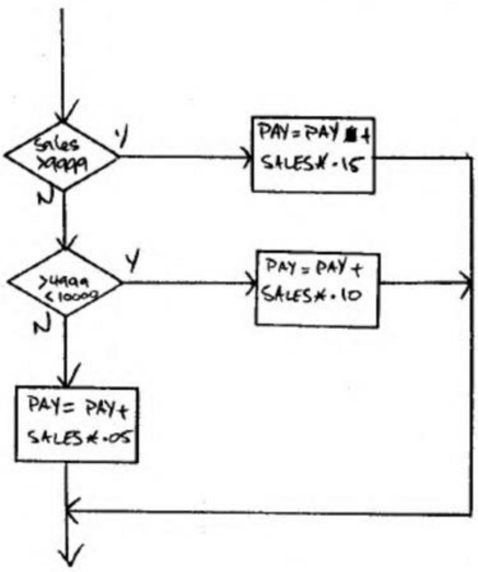
Flow Chart



Coin Flip



Commission



repeated ('GOTO10').

Second example: Let's say Tandy uses a program to calculate the amount of commission each salesperson will receive.

The salesperson who sells $10,000 or more will get 15% commission added to his pay; the salesman who sells from $5,000 to $9,999 will get 10% commission added to his pay; and anyone who sells less than $5,000 gets only 5% commission added to his pay.

We're also assuming · the following:

* it is over a two month period
* this is only a small part of the program we are looking at.
* that the pay is also calculated.
* Tandy does not really use this program.

Program version: commission

```
600 IF SALES>9999 THEN
      PAY=PAY+SALES*.15:
      GOTO615
605 IF SALES>4999 AND
      SALES<10000 THEN
      PAY=PAY+SALES*.1:
      GOTO615
610 PAY=PAY+SALES*.05
615  ... continue with program
```

Flowchart version: commission

<table 2>

This flowchart is fairly easy to follow, so I won't explain it.

Flowcharts have their good points ... like I said above, they show the flow of data, or how the program operates. You would very quickly be able to see what was happening.

If you look at the part of the program above and compare it to the flowchart version, you'd probably choose the flowchart because you could see what is happening, at any time.

But as for creating programs using flowcharts, that's something I don't recommend. It is very hard to ...

* Expand the code,
* Keep it legible (readable).

There are a few other reasons why one should not create code using a flowchart, but they are minor.

But flowcharts are here with us for a purpose, and the most useful purpose has already been given above: it shows what is happening, and what could possibly happen.

In the second half of the example, I'll show how useful they can get. Mind you, the following example is only a small application of what you can do.

I will give a demonstration of how to make a cup of tea or coffee for any amount of people.

Conditions:

* Everyone wants their coffee different.
* Available ingredients are sugar (and how many sugars they like), milk (and how much), and coffee (again, how much they want).
* The cups have been put out on the table, as has the tea, the coffee, sugar, milk, stirring implements, and so on.
* Everyone likes a full cup.
* Everyone removes their teabags (if they choose tea), ie this process is not shown.
* the water stays hot, after it has been heated.

<table 3>

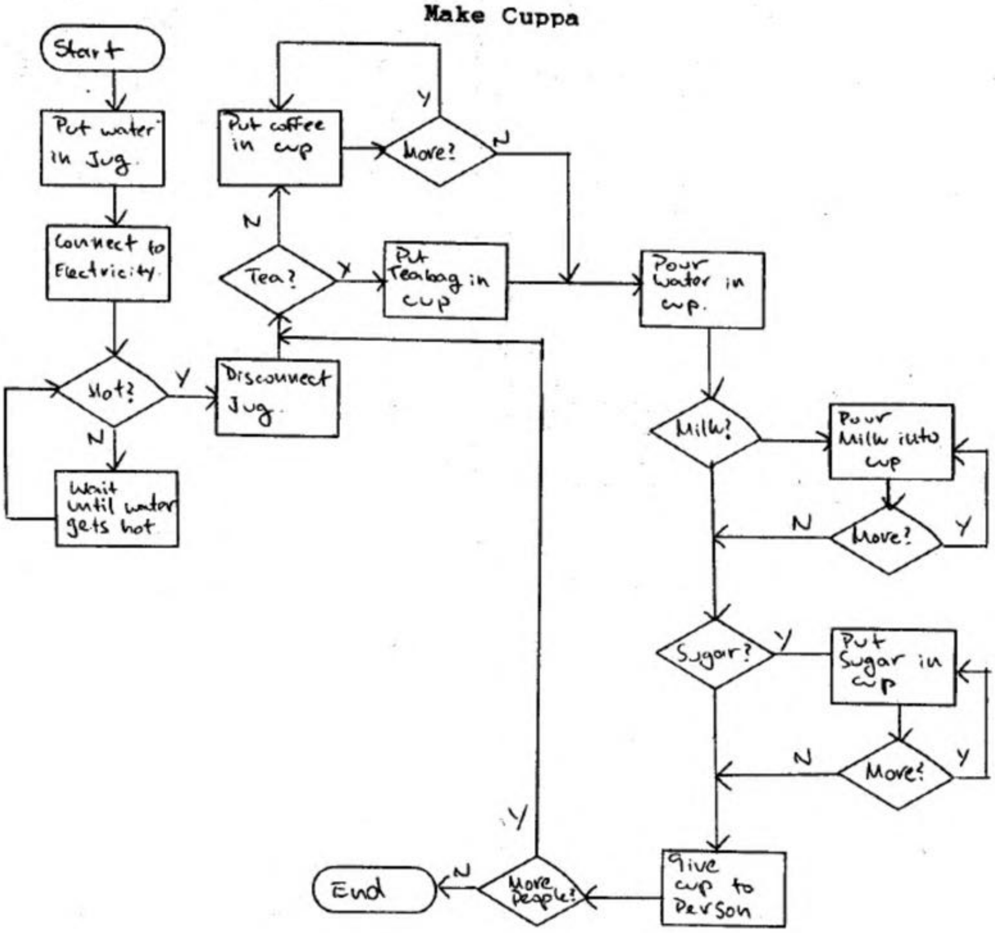
It's getting bigger, isn't it? I could go into much more detail, like how much is 'More?', ie a spoonful of sugar or a bagful of sugar, but we won't, because it would get too big. (And anyway, who'd want a bag of sugar in their tea? -ed).

This type of process can be applied to turning on your CoCo System ...

Assumptions ...

* Everything is off
* You want everything on.
* Even though you may have a TV

Make Cuppa



and a monitor, you only want to turn on the TV (makes this flowchart simple!)

(table 4)

See? This is a very simple flowchart. We can make it more easier ... thus:

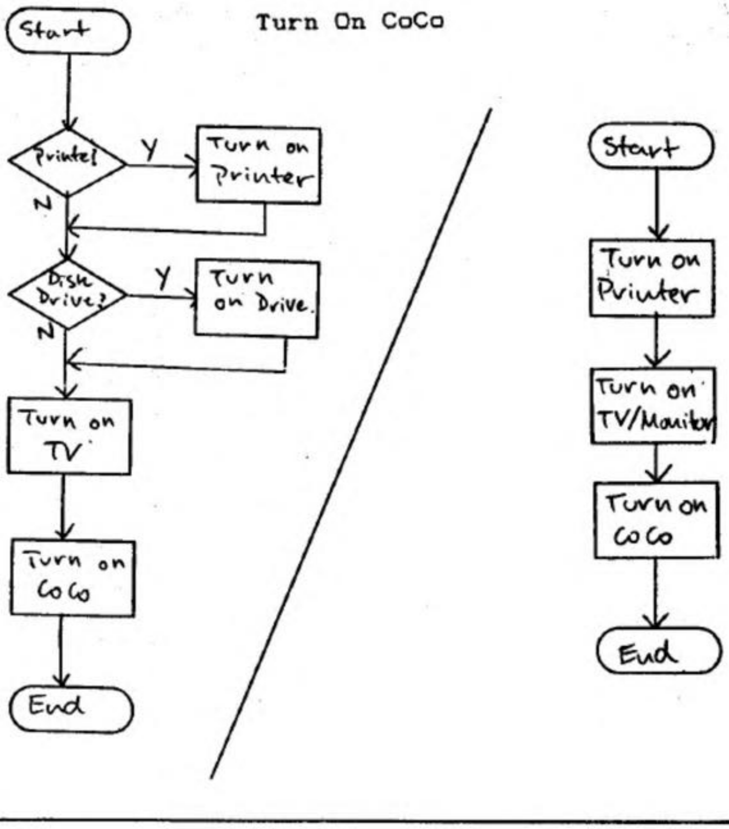
(table 5)

I can, if I get enough response, tell you more about flowcharts and the such in an article. But .. like I said, if I get enough response.

If you are interested in more of this flowchart business, then write to me ...

DR CoCo,
C/o Goldsoft,
PO Box 1742,
Southport,
Qld, 4215

I hope I have somehow answered your letter.


Turn On CoCo

Special Offer

AUSTRALIAN

COCO

or

Softgold

November 1987
MAGAZINES

just $2.00!!!

Sculptor

4GL + DATABASE

RAPID SOFTWARE DEVELOPMENT

This professional-quality fourth generation language with integral database management used by system developers worldwide is now available to you both at home and at work at an affordable price. Sculptor features:—

- VERSIONS FOR 100 COMPUTERS AND A DOZEN OPERATING SYSTEMS
  — Supporting from 1 user up to 1000 users
  — Available for all current Tandy models, all MS-DOS PCs.
  — All 68000 machines under OS-9/68K (Atari ST, Amiga, etc)
  — OS9 MS DOS (+LAN) UNIX XENIX QNX VMS ULTRIX UNIFLEX etc
- TOTALLY PORTABLE APPLICATIONS AND DATABASES
  — All versions compatible at program and data level
  — Develop at home on your Coco and run the system on your PC, mini, supemini at work AND vice versa!
- LOW RESOURCE CONSUMPTION, FAST COMPILE + EXECUTION
  — You can begin with as little as 128KB RAM + 1 diskette (hard disk highly recommened for full performance)
  — Compiles and runs lightning-fast, even on small systems
  — Fast storage/retrieval in database of millions of records

6809 OS -9 level II	(Multi User)	r.r.p. inc. tax $495.00
All MS-DOS systems	(Single User)	r.r.p. inc. tax $1134.95
All MS-DOS L.A.N.S.	(Multi User)	r.r.p. inc. tax $2046.95
SCO XENIX	(Multi User)	r.r.p. inc. tax $2046.95

- New* Sculptor Starter Pack for the above machines. *New* Limited-size versions of eath of the above — only $69.95

Available at your local Tandy Store

MPD Australia Pty Ltd
119 Willoughby Road,
Crows Nest 2065

Australasian importer and distributor

COLOUR 5¼ DISKETTES

Available in Lavender, Dark Blue Light Blue, Grey, Beige, Pink, Yellow White, Orange, Red, Burgundy, Black Brown and Green.

TOP QUALITY CENTECH BRAND with LIFETIME WARRANTY

SOLVE the problem of trying to locate your programs....

COLOUR CODING MAKES IT EASIER!

SSDD	$23.50	for 10
DSDD	$25.00	for 10

(postage free)
Please state colours required

JANDA (AUS)
P.O. BOX 239
MARYBOROUGH Q 4650
106 NEPTUNE STREET (071) 23 1369

WOW! WOW!

- PORT 4. CoCo2 and 3 – four port Serial Serial I/O. Finally push your CoCo3, with Level 2, to the max! Run multiple printers, terminals, modems, includes all driver patches. **$179⁹⁵**
- CoCo3 Multi–pak extender kit. Three piece kit. Finally all your desk space recovered. **$49⁹⁵**
- Hard disk Interface kit – incl. Interface and drivers for 5, 10, 20, ZXREC, WD, DTC, SHUGART controllers. **$129⁹⁵**
- Total OS9 support. Software programm–ing and hardware design. Custom Sculpture base. Software development systems available.

Bayreap P/L

Telephone: 02–9082036
PO Box 1021, Neutral Bay Junction,
2089, NSW.

GOLDSOFT

P.O. BOX 1742,
SOUTHPORT, QLD. 4215
Telephone: (075) 39 6177.

Please tick (✓) your requirements.

Office use only

Goldsoft Price List as at February, 1988.

SOFTWARE:

Magazines, Tapes & Disks

AustralianCoCo (Advanced Programs for your CoCo):

Magazines:		Tape ( ) Disk ( )	
12 Months	$39.95 ( )	12 Months	$123.75 ( )
6 Months	$24.95 ( )	6 Months	$ 74.25 ( )
1 Month	$ 4.50 ( )	1 Month	$ 16.50 ( )

Softgold (Programs for your CoCo):

Magazines		Tape ( ) Disk ( )	
12 Months	$39.95 ( )	12 Months	$123.75 ( )
6 Months	$24.95 ( )	6 Months	$ 74.25 ( )
1 Month	$ 4.50 ( )	1 Month	$ 16.50 ( )

Gold Disk - Available Quarterly:

# 1	$16.00	( )
# 2	$16.00	( )
# 3	$16.00	( )
# 4	$16.00	( )
# 5	Coming Soon !!!	

The CoCo3 Tape/Disk:

#2	Tape: $16.00 ( )	Disk: $16.00 ( )
#3	Tape: $16.00 ( )	Disk: $16.00 ( )
#4	Tape: $16.00 ( )	Disk: $16.00 ( )

"Say the Voidz":
Two Curriculum based speller programs for your Tandy Speech/Sound pack: $29.95
Req: 32K + Tandy Speech Pack ( )

Best of CoCoOz - $16.00
A selection of programs from Australian CoCo Magazine.

		Tape:	Disk:
# 1	- Education:	( )	( )
# 2.1	- Games 16K:	( )	( )
# 2.2	- Games 32K:	( )	( )
# 4	- Business :	( )	( )
# 5	- Adventure:	( )	( )
# 6	- Preschool:	( )	( )
# 7	- Graphics :	( )	( )
# 8	- Games 16K:	( )	( )
# 9	- Games 32K:	( )	( )
#10	- Education:	( )	( )
#11	- Education:	N/A	( )

HARDWARE:

CoCoConnection: $219.95 ( )
Video Amp: Vith sound - $35.00 ( )
Vithout sound - $25.00 ( )
The Probe: $49.95 ( )

GOLDLINK
Access Goldlink *642# on Viatel with a 1200/75 baud modem. Annual subscription: $44.95 ( )

BRIC-A-BRAC:

Blank Tapes:	12 @ $18.00	( )
(C-30)	1 @ $ 2.00	( )
Tape cases:	12 @ $ 5.00	( )
Disk DSDD:	10 @ $20.00	( )
	1 @ $ 2.50	( )

BOOKS:

Help (for your CoCo): $ 9.95 ( )
Mico Help (for your MC-10): $ 9.95 ( )

BACK ISSUES:

Australian CoCo: Sep 84 - Dec 85: $2.00 ( ) each
Australian CoCo: Jan 86 - Feb 87: $3.75 ( ) each
Australian Mico: Aug 84 - Dec 85: $2.00 ( ) each

ADDITIONAL REQUIREMENTS:

....

....

....

....

....

SUBSCRIBE NOW!

New Subscription: ( ) Renewal ( )
Sub No:.......

Name:......................................

Address:...................................

....

....................P/Code........

Phone (...)........

Please find enclosed: a. Cheque: ( )
b. Money Order: ( ) c. Credit card: ( )

Credit Card Type & Number:

Bankcard ( ):.......-.......-.......-.......

Mastercard ( ):.......-.......-.......-.......

Visa ( ):.......-.......-.......-.......

Expiry Date:......./19...

Authorised Amount: $...........

Signed:....................................

Mar.88

user contacts

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

ACT:		
CANBERRA NTH	JOHN BURGER	062 58 3924
CANBERRA STH	LES THURBON	062 88 9226
NSW:		
SYDNEY:		
BANKSTOWN	PAT DORSETT	02 646 3619
BLACKTOWN	KEITH GALLAGHER	02-627-4627
CARLINGFORD	ROSS MCKAY	02 662 1951
CHATSWOOD	BILL O'DONNELL	02 419 6081
COLYTON	HERMAN FREDRIKSSON	02 623 6379
FAIRFIELD	ARTH PITTARD	02 72 2881
GLADESVILLE	MARK ROTHWELL	02 817 4627
HILLS DIST	ARTHUR SLADE	02 674 5620
HORNSBY	ATHALIE SMART	02 848 8830
INGLEBURN	STEPHEN RIDGEWAY	02 605 7382
KENTHURST	TOM STUART	02 654 2178
LEICHHARDT	STEVEN CHICOS	02 560 6207
or	GORGE ECHEGARAY	02 560 9665
MACQUARIE FIELDS	BARRY DARNTON	02 618 1909
MINTO	GRAHAM POLLOCK	02 603 5028
SUTHERLAND	IAN ANNABEL	02 528 3391
SYDNEY EAST	JACKY COCKINOS	02 344 9111
ALBURY	RON DUNCAN	060 43 1031
ARMIDALE	DOUG BARBER	067 72 7647
BLAXLAND	BRUCE SULLIVAN	047 39 3903
BROKEN HILL	TERRY NOONAN	080 88 2382
CAMDEN	SEAN MURDOCH	047 74 8291
COFFS HARBOUR	BOB KENNY	066 51 2205
COOMA	ROSS PRATT	064 52 3065
COORANBONG	GEORGE SAVAGE	047 77 1054
COOTAMUNDRA	CHERYLE WILLIS	069 42 2204
DENILIQUIN	WAYNE PATTERSON	058 81 3014
DUBBO	GRAEME CLARKE	068 89 6549
FORBES	JOHANNA WAGG	068 52 2943
GOSFORD	PETER SEIFERT	043 32 7874
GRAFTON	PETER LINDSAY	066 42 2503
GUYRA	MICHAEL J. HARTMANN	067 79 7547
JUNEE	PAUL MALONEY	069 24 1860
KEMPSEY	RICK FULLER	065-62-7222
LEETON	BRETT WALLACE	069-53-2081
LISMORE	ROB HILLARD	066 24 3089
LITHGOW	DAVID BERGER	063 52 2282
MAITLAND	BILL SNOW	049 66 2557
MOREE	ALF BATE	067 52 2465
NAMBUCCA HDS	WENDY PETERSON	065 68 6723
NARROMINE	GRAEME CLARKE	068 89 6549
NEWCASTLE	LYN DAWSON	049 49 8144
NOWRA	ROY LOPEZ	044 46 5449
ORANGE	DAVID KEMP	063 62 2270
PARKES	DAVID SMALL	068 62 2682
PORT MACQUARIE	RON LALOR	065 62 2682
SPRINGWOOD	JIM HOPPITT	047 54 1474
TAHMOOR	GARY SYLVESTER	046 81 9318
UPPER HUNTER	TERRY GRAVOLIN	065 45 1698
URALLA	FRANK MUDFORD	067 78 4391
WAGGA WAGGA	CES JENKINSON	069 25 2263
WYONG	JOHN WALLACE	043 90 0312
NT:		
DARWIN	BRENTON PRIOR	089 81 7766
QLD:		
BRISBANE:		
BIRKDALE	COLIN NORTH	07 824 2128
CANNON HILL	ROSEMARY LITZOW	07 395 0863
CLAYFIELD	JACK FRICKER	07 262 8869
COLL'WOOD PK	ANDREW SIMPSON	07 288 5206
IPSWICH	MICK MURPHY	07 271 1777
PINE RIVERS	BARRY CLARKE	07 204 2606
SOUTH	JOHN POXON	07 208 7820
SOUTH WEST	BOB DEVRIES	07 372 7816
SCARBOROUGH	PETER MAY	07 203 6723
AIRLIE BEACH	GLEN EVANS	079 46 1264
BIGGENDEN	ALAN MENHAM	071 27 1272
BOWEN	TERRY COTTON C/O	077 85 2220
BUNDABERG	RON SIMPKIN	071 71 5301
CAIRNS	JEFF LARSEN	070 54 7127
DALBY	MERRICK TANSKY	074 62 3228
GLADSTONE	CAROL CATHCART	079 78 3594
GOLD COAST	GRAHAM MORPHETT	075 39 6177
GYMPIE	BERT LLOYD	071 8219100
HERVEY BAY	LESLEY HORWOOD	071 22 4989
MACKAY	LEN MALONEY	079511333x782
MARYBOROUGH	JOHN EPPER	071 21 5638
MT ISA	JACK RAE	077 43 3486
MURGON	PETER ANGEL	071 68 1528
ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
TARA	DEBBIE DORFIELD	074 65 3177
TOOWOOMBA	LEN GERSEKOWSKI	076 35 8204
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064
SA:		
ADELAIDE	JOHN HAINES	08 278 3560
PORT NOARLUNGA	ROB DALZELL	08 386 1647
SEACOMBE HTS	GLENN DAVIS	08 296 7477
PORT LINCOLN	BILL BOARDMAN	086 82 2385
PORT PIRIE	VIC KNAUERHASE	086 32 1230
WHYALLA	MALCOLM PATRICK	086 45 7637
TAS:		
DEVONPORT	JEFF BEST	004 24 6759
HOBART	ROB DELBOURGO	002 25 3896
KINGSTON	WIM DE PUIT	002 29 4950
LAUNCESTON	BILL BOWER	003 44 1584
SMITHTON	HARRY CHRISAFIS	004 52 1590
WYNYARD	ANDREW WYLLIE	004 35 1839
VIC:		
MELBOURNE:		
MELBOURNE CCC	LES LEISHMAN	03 484 0822
DANDENONG	DAVID HORROCKS	03 707 5870
DONCASTER	JUSTIN LIPTON	03 857 5149
FRANKSTON	BOB HAYTER	03 783 9748
NARRE WARREN	LEIGH EAMES	03 704 6650
STH EASTERN	PETER WOOD	03 435 2018
M'TON PENNINSULA	GORDON CHASE	059 71 1553
MELTON	MARIO GERADA	03 743 1323
PAKENHAM	JASON HALL	059 41 1398
RINGWOOD	IVOR DAVIES	03 758 4496
SUNBURY	JACK SMIT	03 744 1355
SUNSHINE	IAN BUTTRISS	03 314 8242
UPR F'TREE GLY	RORY DOYLE	03 758 2671
BAIRNSDALE	COLIN LEHMANN	051 57 1545
BALLARAT	MARK BEVELANDER	053 32 6733
DAYLESFORD	DANNY HEDJI	054 24 8329
GEELONG	DAVID COLLEN	052 43 2128
MAFFRA	MAX HUCKERBY	051 45 4315
MOE	JOSEPH HESTER	051 27 7817
MORNINGTON	MICHAEL MONCK	03 789 7997
MORWELL	JEFF SHEEN	051 33 9904
SHEPPARTON	RUSS FARRAR	058 25 1007
SKYIPESDALE	TONY PATTERSON	053 42 8815
SWAN HILL	BARRIE GERRAND	050 32 2838
TONGALA	TONY HILLIS	058 59 2251
TRARALGON	LEIGH DAWES	051 74 5552
WONTHAGGI	LOIS O'MEARA	056 72 1593
WA:		
PERTH	IAIN MACLEOD	09 448 2136
GIRRAWHEEN	MARK WILLEMSEN	09 342 7639
KALGOORLIE	TERRY BURNETT	090.21.5212
CANADA - CoCo:		
Ontario	Richard Hobson	416 293 2346
Toronto	Franz Lichtenberg	416 845 2889

special interest groups

TEACHERS' INTEREST GROUP		
BRISBANE	BOB HORNE	07 261 8151
BUSINESS:		
BRIZBIZ	BRIAN BERE-STREETER	07 349 4696
OS9 GROUPS:		
NATIONAL OS9 USERS' GROUP	GRAEME NICHOLS	02 451 2954
BRISBANE NON-SMOKERS	JOHN POXON	07 208 7820
NSW		
SYDNEY		
BANKSTOWN	CARL STERN	02 646 3619
CARLINGFORD	ROSKO MCKAY	02 624 3353
GLADESVILLE	MARK ROTHWELL	02 817 4627
SYDNEY EAST	JACKY COCKINOS	02 344 9111
COOMA	ROSS PRATT	064 52 3065
QLD		
BRISBANE	JACK FRICKER	07 262 8869
VIC		
LATROBE VLY	GEORGE FRANCIS	051 34 5175
WA		
KALGOORLIE	TERRY BURNETT	090 21 5212
MC-10 CONTACTS:		
LISMORE	BOB HILLARD	066 24 3089
SYDNEY	GRAHAM POLLOCK	02 603 5028
TANDY 1000 / MS DOS:		
NSW:		
GLADESVILLE	MARK ROTHWELL	02 817 4627
SYDNEY WEST	ROGER RUTHEN	047 39 3903
WYONG	JOHN WALLACE	043 90 0312
QLD:		
BRISBANE		
NORTH	BRIAN DOUGAN	07 30 2072
SOUTH	BARRY CAWLEY	07 390 7946
GOLD COAST	DEON GEORGE	075 39 6177
SA		
PORT LINCOLN	BILL BOARDMAN	086 82 2385
VIC:		
LA TROBE VALLEY	PETER FOLEY	051 74 5791
MELBOURNE	TONY LLOYD	03 882 4664
FORTH:		
SYDNEY	JOHN REDMOND	02 85 3751
ROBOTICS:		
GOLD COAST	GRAHAM MORPHETT	075 51 0577
SYDNEY	GEOFF FIALA	02 84 3172
CHRISTIAN USERS' GROUP		
MOE	RAYMOND L. ISAAC	051 27 2695
MSX		
FRANKSTON	ALAN HASSELL	03 786 6290
MODEL RAILWAY CLUBS:		
BEENLEIGH	DAVID PHILLIPS	07 807 2663

300 BAUD BULLETIN BOARDS

SYDNEY:	
INFOCENTRE	02 344 9511
TANDY ACCESS	02 625 8971
THE COCOCONNECTION	02 618 3591

VIDEOTEX SYSTEMS

VIATEL	01955
MOUSETEX	059 42 5528
VTX 4000	03 741 3295

TANDY INFO ON VIATEL

GOLDLINK	VIATEL *542#
POWER CODE	VIATEL #64265#
TANDY	VIATEL *64261#

SOME TANDY USERS ON VIATEL

ALLAN BEALE	720353300
FRED BISSELING	648232030
JACK FRICKER	726283690
JOHN GRIGSBY	945872030
STUART HALL	939765790
BOB KENNY	635122050
JEFF LARSEN	705471270
G LEWIS	954811900
IAIN MACLEOD	944821360
CHRIS NAGLE	689523360
RICHARD PARKHURST	250717870
ROSS PRATT	648230650
RICCAY SCHMAHL	298151500
ARTHUR SLADE	262269400
DARCY O'TOOLE	755100150
RON WRIGHT	352924510

BLAXLAND COMPUTER SERVICES PTY. LTD.

SERVICING MODERN TECHNOLOGY

Tandy
ELECTRONICS
DEALER
NO 9254

We specialise in assisting Tandy Computer Users! Phone or write for information regarding our new range of IBM Compatible soft/hardware.

OS-9 DISKFIX & UTILITIES	CC3 - $59.95
OS-9 UTILITIES & DISKFIX ADVANCED	CC3 - $99.95
DYNASTAR/DYNAFORM - OS-9 Word Processor, Multi-user	CC2/3 - $299.00
COLOR SCRIBE - RS/DOS Wordprocessor	CC3 - $99.95
RGB PATCH	CC3 - $49.90
THE WILD WEST - by Tom Mix (Disk)	CC3 - $69.95
NUKE THE LOVE BOAT - Adventure using 512k as ramdisk	CC3 - $59.95
MAGIC OF ZANTH - Adventure with top graphics	CC3 - $69.95
RETURN OF JUNIOR'S REVENGE - All time favourite	CC3 - $59.95
INTERBANK INCIDENT	CC2/3 - $69.95
ROGUE	CC3 - $69.95
RESCUE ON FRACTULUS	CC3 - $69.95
FLIGHT 16	CC3 - $69.95
DESERT RIDER	CC2 - $59.95
BIOSPHERE	CC2 - $59.95
CAVE WALKER	CC2/3 - $59.95
VEGAS SLOTS	CC3 - $69.95
KORONIS RIFT	CC3 - $69.95
COCO-MAX 3 - Only for Coco3, Animation plus more	CC3 - $160.00
ZONE RUNNER	CC3 - $69.95
MR. DIG - Disk	CC2 - $40.00
ESCAPE: 2012 - 64K Adventure (137 rooms)	CC2 - $56.00
TREASURE OF THE AZTECS - 4 voice music, 50 hires screens	CC2 - $56.00
ROBOT ODYSSEY - 64K Adventure (Disk)	CC2 - $69.95
COMPLETE RAINBOW GUIDE TO OS-9 - 2 Disks, no book	CC2 - $59.95
FIRST RAINBOW BOOK OF ADVENTURES - Book and Tape	CC2 - $32.00
SECOND RAINBOW BOOK OF ADVENTURES - Book and Tape	CC2 - $56.00
THIRD RAINBOW BOOK OF ADVENTURES - Book and Tape	CC2 - $44.00
RAINBOW BOOK OF SIMULATIONS - Book and Tape	CC2 - $40.00
SECOND RAINBOW BOOK OF SIMULATIONS - Book and Tape	CC2 - $40.00
INTRODUCTORY GUIDE TO STATISTICS - Book and Tape or Disk	CC2 - $29.95
COCO GRAPHICS DESIGNER - print your cards, signs,banners	CC2/3 - $59.95
" " " - 100 pre-drawn pictures	CC2/3 - $29.95
" " " - Font disk #1 extra fonts	CC2/3 - $39.95
MICRO ILLUSTRATOR	CC2 - $39.95
COCO MAX II Tape or Disk	CC2 - $100.00
APPROACH CONTROL Simulation	CC2 - $49.95
3D-GRAPHIMATOR - 3 Dimensional Drawing Design	CC2 - $79.95
PROGRAMMERS UTILITY - Commands & Utilities by keystroke	CC2 - $39.95
SECTOR INSPECTOR - RSDOS Hackers Tools	CC2 - $39.95
DEPUTY INSPECTOR - RSDOS Hackers Tools	CC2 - $39.95
PRINTMASTER	IBM - $96.35
ART GALLERY 1	IBM - $56.65
ART GALLERY 2	IBM - $56.65
NEWSROOM	IBM - $101.95
CLIP ART 1	IBM - $50.95
CLIP ART 2	IBM - $62.35
CLIP ART 3	IBM - $62.35
NEWSMASTER (DESKTOP PUBLISHING)	IBM - $170.00
PRODESIGN (CAD SYSTEM WITH HI-RES PRINTOUTS)	IBM - $630.00
MATH BLASTER	IBM - $90.65
SPEED READER II	IBM - $125.00
" " " - Data Disk	IBM - $39.00
CLIPPER (DBASE III COMPILER)	IBM - $599.00
DAC-EASY ACCOUNTING AUSTRALIAN VERSION	IBM - $250.00
DAC-EASY WORD PROCESSOR	IBM - $129.00
ABILITY - Top value Intergrated Package	IBM - $220.00
IN HOUSE ACCOUNTANT - Easy to use accounting	IBM - $199.00
FBS ACCOUNTING SUITE (Inventory,Debtors,Creditors,G/L)	IBM - $830.00
FBS PAYROLL	IBM - $630.00
FBS BILL OF MATERIALS	IBM - $630.00
FBS JOB COSTING	IBM - $630.00
A1 TYPIST WORD PROCESSOR	IBM - $159.00
PEEKS N' POKES FOR IBM	IBM - $66.00
INSIDE TRACK FOR IBM	IBM - $85.00
SPEEDIT PROGRAM EDITOR	IBM - $79.00
WORDSTAR HELP 3 - Complete memory resident help system	IBM - $89.00
PROKEY	IBM - $195.00
LYRA - The Musical Coco Max (pull down menus)	CC2/3 - $110.00
MUSICA II DISK OR TAPE	CC2 - $59.00

MONEYPOLY	CC2 - $33.45
SAILOR MAN	CC2 - $33.45
COCOTEX VIATEL SOFTWARE	CC2/3 - $79.95
BUSINESS ACCOUNTING SYSTEM	CC2 - $99.95
FAMILY TREE GENEOLOGY	CC2 - $33.45
SUPER BACKUP UTILITY	CC2 - $59.95
TELEWRITER 64 DISK + TAPE	CC2 - $84.45
VIP-WRITER DISK + TAPE	CC2 - $98.95
VIP-DATABASE DISK ONLY	CC2 - $84.45
VIP-CALC DISK + TAPE	CC2 - $84.45
VIP-SPELLER DISK ONLY	CC2 - $77.45
VIP-TERMINAL DISK + TAPE	CC2 - $84.45

****

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

****

10MEGABYTE HARD DISK DRIVE - Speed and Mass Storage	CC2/3 - $1299.00
20MEGABYTE HARD DISK DRIVE - Large Business Storage	CC2/3 - $1599.00
512k COCO 3 UPGRADE Includes P.D Software	CC3 - $220.00
TEAC DSDD DRIVE 40T PLUS CONTROLLER - Double Height	CC2/3 - $680.00
TEACx2 DSDD DRIVES 40T PLUS CONTR. - Equal to 4 Tandy.	CC2/3 - $950.00
BARE TEAC DSDD 40T DRIVES - Hi-tech reliability	CC2/3 - $300.00
BARE TEAC DSDD 80T DRIVES - 1,4MB Storage per drive	CC2/3 - $350.00
BARE TANDY SSDD DRIVES - not $519 (subject to stock)	CC2/3 - $250.00
T1000 20MB INTERNAL HD - 1 year warranty, formatted	T1000 - $1399.00
T1000 COMBI RS232/640K/CLOCK/CALENDAR	T1000 - $750.00
THOMSON AMBER/GREEN COMPOSITE MONITORS WITH SOUND	ALL - $350.00
TANDY 1000 SX - 384K, 2 DRIVE	T1000 - $1999.00
TANDY 1000 EX	T1000 - $1299.00
HI-RES INTERFACE	CC3 - $23.00
TANDY COLOUR COMPUTER 3 - 128K	CC3 - $370.00
TANDY 102 PORTABLE	T102 - $999.00
TANDY 200 PORTABLE	T200 - $1399.00
MULTIPAK INTERFACE - SWITCHABLE COCO 2/3, LED ON/OFF	CC2/3 - $199.95
VIDEO DRIVER WITH SOUND	CC2 - $38.00
VIDEO DRIVER WITHOUT SOUND (If your monitor has sound)	CC2 - $32.00
MODIFIED ARCHER JOYSTICKS - SUIT COCO 2/3	CC2/3 - $27.00
MODIFIED QUICKSHOT JOYSTICKS - " "	CC2/3 - $35.00
CITIZEN 120D DOT MATRIX PRINTER - EXTRA TYPEFONTS	ALL - $599.00
DMP-106 DOT MATRIX PRINTER	ALL - $399.00
DMP-130 DOT MATRIX PRINTER	ALL - $499.00
AVTEK MINI MODEM - 300 & 1200/75 baud, economy plus	ALL - $250.00
TANDY INTERNAL MODEM - 300 & 1200/75 + VTEX2 software	T1000 - $699.00
NICE MODEM II - auto ans/dial, Hayes compatible	ALL - $749.00
WYSE 30 TERMINALS	CC3 - $999.00

****

-MISCELLANEOUS-MISCELLANEOUS-MISCELLANEOUS-MISCELLANEOUS-MISCELLANEOUS-

****

PRINTER PAPER 9-1/16" x 11" PLAIN per 1000 sheets	$17.00
PRINTER PAPER " x " PLAIN per box 4000 sheets	$60.00
PRINTER PAPER 9-1/2" x 11" PLAIN per 1000 sheets clean edge	$25.00
PRINTER PAPER 9-1/2" x 11-2/3" PLAIN A4 per 1000 sheets	$29.00
PRINTER PAPER 11" x 15" PLAIN per 1000	$30.00
PRINTER PAPER 11" x 15" BLUE RULE per 1000 (also SCREEN)	$27.00
NCR 2 PART 9-1/2" x 11" (no carbon required) per 1000	$74.00
RAINBOW PACK 500 each blue,green,pink,yellow 9-1/2" x 11"	$55.00
MAILING LABELS 11mm x 50mm (4 up) per 1000	$6.00
MAILING LABELS 23mm x 89mm (1 up) per 1000	$16.00
MAILING LABELS 36mm x 102mm (1 up) per 1000	$18.00
CASSETTE LABELS (orange) per100 (10 per sheet)$5.00 (white)	$4.00
BLANK CASSETTES C-10's (10 min.) $1.10 each. C-30's	$1.30
STORAGE CASE, HINGED, LOCKABLE 5-1/4" holds 100 $25; holds 60	$22.00
STORAGE CASE, HINGED 3-1/2" holds 60	$15.00
3-1/2" DSDD DISKETTES box 10	$55.00
5-1/4" DSDD DISKETTES box 10	$25.00
DISK HEAD CLEANING KIT 3-1/2"	$11.95
DISK HEAD CLEANING KIT 5-1/4"	$11.95
LARGE RANGE PRINTER RIBBON/CARTRIDGES (Tandy/non Tandy)	P.O.A.
LARGE RANGE PRE-PRINTED INVOICE/STATEMENT FORMS AVAILABLE	P.O.A.
DISKETTE NOTCHER	$9.95
PERFECT 5-1/4" DSDD DISKETTES BOX 10	$25.00
DATA PERFECT 5-1/4" DSDD DISKETTES BOX 10	$19.00

bankcard

MasterCard

VISA

AGC CREDIT LINE

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

PRICES SUBJECT TO CHANGE WITHOUT NOTICE.

POSTAGE AND PACKAGING EXTRA