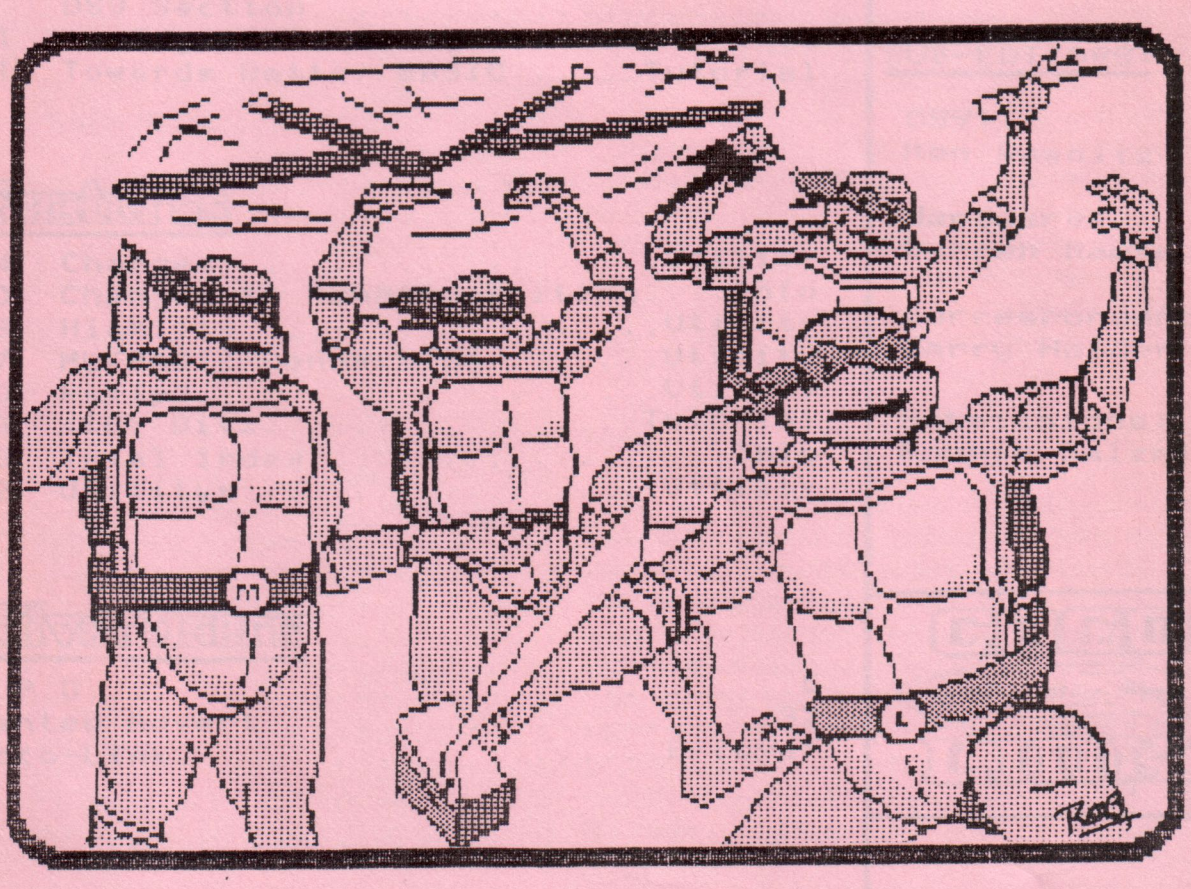
April 1991 Vol 4. No.2

COCO LINK

The Color Computer Magazine



Featuring:

- ML/BASIC Connection
- Script Changes
- Utilities Galore!
- Total Index

Contents

Departments

Columns

Features

Advertisers

COCO LINK

EDITOR:

Robbie Dalzell

ASST. EDITOR:

Garry Holder

SUB-EDITORS:

OS9:
Ken Wagnitz

Hardware:
Darren Ramsey

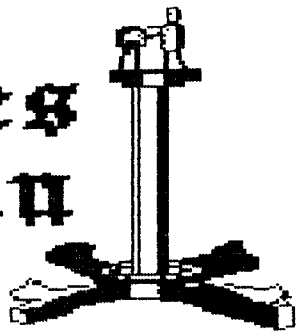
Correspondence:
Garry Holder

Submissions:
Robbie Dalzell

COCO LINK

Copyright Notice:

Robbies Column



HOW IS COCO-LINK GETTING ON?

That is a question we get asked regularly. Well the facts are these. Under the unique circumstances that COCO-LINK operates, I would say that we are doing very well.

Each month we gain new subscribers but it is also true that we lose some subscribers. This loss equals out or even outguns our good work in getting new subscribers therefore causing a slow erosion of our subscriber base. We have not nearly reached any critical stage and are in fact, at this time very close to our highest subscription numbers but you can see the reasons for my continually exhorting our subscribers to send their renewals promptly.

This slow erosion of subscribers is obviously closely related to the fact that Tandy discontinued the Coco and with it any pretence of backup for existing owners. The other main reason people abandon the Coco for other lesser machines is the lack of availability of many of the new programmes available in the USA.

Something has to be done to remedy this situation. More on that later.

****

THE COCO-LINK GUARANTEE

Many Coco users remember other Coco magazines which went broke suddenly and stopped publication without warning. Some even while denying that it was about to happen. Many people got their fingers burned and lost subscription money without ever receiving any of the magazines they had just paid for. Count me in that number...twice.

I think these memories affect some people when it comes to renewing their subscriptions. Therefore I think it is about time to reitterate the guarantee I made to subscribers when I started this magazine to replace the defunct COCO MAGAZINE. That is:

1) The cost of the magazine will be kept as low as possible and will reflect the basic costs of running the magazine.

2) The magazine will be distributed by subscription only thereby avoiding the possibility of throwing money away on unsold copies.

3) Six months warning will be given before the demise of this magazine thereby insuring that no subscriber can lose more than half his/her subscription. (At this time this is a mere $7.00). During this period all subscribers on the books at that time will receive the last three magazines.

When the decision was made to start this magazine, we started it on the premise that it must pay for itself. It must cover the cost of all materials used, postage/packaging, repairs to equipment and all the other overheads that require to be paid by any other type of small business.

Garry and I produce this magazine from home using our own personal equipment. We willingly give our spare time for no recompense except the satisfaction of turning out what we like to think is a near professional product. As we are both devoted Cocoists it is in our own interest to keep the Coco community together. Our only stipulation on producing this magazine has been that it does not cost us personally to do it. So far this has held true.

This said, I wish to assure you that COCO-LINK will not be contemplating closure for quite a long time to come given your continuing support.

I hope this guarantee will allay the fears of the doubtful.

****

NOW FOR THE GOOD NEWS

Harping back to the first paragraph, I think we may have a solution to one problem.

It is our intention to inaugurate a SOFTWARE ORDERING SYSTEM (SOS for short!).

The basis of the idea is that each magazine will contain the names and price (in Australian dollars) of several pieces of software currently available in the USA. The prices will range in the region of 55% to 65% above the US$ prices. This to cover the inherant cost of the rate of exchange, procuring the programmes from the USA and redistributing them throughout Australia.

There will be closing date for the ordering of programmes and all orders and monies must be received by this given date. An order form will be included with each magazine.

COCO-LINK will then order the programmes en bulk and distribute them to the purchasers. This should be some time before the publication of the following magazine.

All programmes will be checked to make sure that they work before being posted to the purchaser.

No refunds will be granted and no returned programmes will be accepted.

The programmes listed in each magazine will only be available during the term of that order.

That is the outline of SOS. To help us get this scheme up and running we need you to write or phone and let us know what type of programmes you are interested in, whether for Coco 2 or 3 and even individual programmes you would like us to include on our bi-monthly list.

Like most things connected to COCO-LINK, the success or failure will depend on you, the reader. Please let us hear from you and don't be shy in adding any ideas you may have to make a better system.

Should SOS be given the green light by you, the reader, we will review its potential after a few months.

Both Garry and I wait with bated breath for your reply to

our SOS.

****

THIS ISSUE

Most of the material submitted to COCO-LINK Magazine is submitted by a few dedicated Cocoists. One of these is George McLintock (although he is also a dedicated IBMist as well). His articles and programmes are both educational and entertaining as well as useful.

The bulk of this issue is dedicated to Utilities from George. The articles accompanying the programmes give detailed insights into the intricacies of interfacing ML and BASIC as well as information on file structure and many other things.

To complement this issue, Public Domain Disk No.022 will feature more utilities and articles by George. These were thought to be too long to fit into the pages of COCO-LINK. George is also willing to do something special for cassette users. this offer appears on page ?? of this issue. If you have any interest in ML at all this will be a good one to have.

Added to that we are issuing a PD Graphics disk for those of you who are interested in looking at and making pictures.

There is also a complete index of our two previous volumes.

As if that were not enough, we have all our regular features to help entertain and educate you.

****

COCO-LINK LISTINGS

The programme listings in COCO-LINK Magazine are our single most critcized item. The reasons for using the three column listing method are both financial and a means of putting more in the pages of our magazine.

I am sure that most of our readers will agree that over the past two yearsm we have improved the readability of the listings 100 fold.

Still, for a variety of reasons we receive complaints on the readability of the listings. With this in mind, and the fact that it is our aim to please wherever possible, we have decided to change the way we print our listings.

We will, in future, print our listings in full letter size, 32 characters wide and two columns per page. In the case of overlong listings we will revert to the three column listing system.

Jhis format will start from this issue and we would like to receive your comments on this change and any other changes you feel would benefit the magazine.

****

MORE CLUBS

It is good to see that people are getting together again to form user groups so as to exchange information and help one another. This helps to keep the Coco Community flourishing.

This month we announce two new groups:

1) A new group has been formed in Brighton Qld.

2) The Whyalla Coco User Group in SA has been resurrected.

Further details on these two groups can be found on the Noticeboard page plus we have added two more BBS's for SA modem users.

****

OSK MACHINES (OS9 for the 68000)

I just read in the OS9 Users Newsletter that another OSK machine has entered the market. Along with the MM1 and Hoggs Tomcat that makes three machines vying for this market.

So far it doesn't look as if the MM1 OSK based machine will be a goer in Australia. At least there doesn't seem to be any way to find out if there is any movement in this direction.

I have written to Kenneth-Leigh Enterprises for further information on any Australian Connection but to no avail. I am not getting a reply. It seems that they do not wish us to know what is going on. I will try again.

It is a pity in a way as I thought that maybe the MM1 was the road for us to travel in the future. At the moment it seems that this road has reached a dead end.

****

AUSTRALIA POST

Not only have Australia post increased the postage on each of our magazines by 6 cents and increased the registration fee for our publication by 50%, but they seem to be losing quite a few of our magazines in transit.

This may or may not coincide with the switching of the address labels to the outside of our packaging, nevertheless, we have decided to revert to sticking the address labels directly to the magazine and let the postie read it through the clear plastic.

At least if the magazines are being lost because of damage to the packaging the address will still be intact on the magazine itself.

****

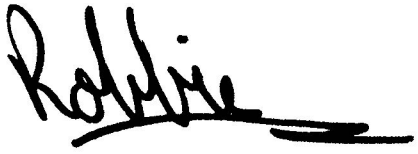
BACK ISSUES

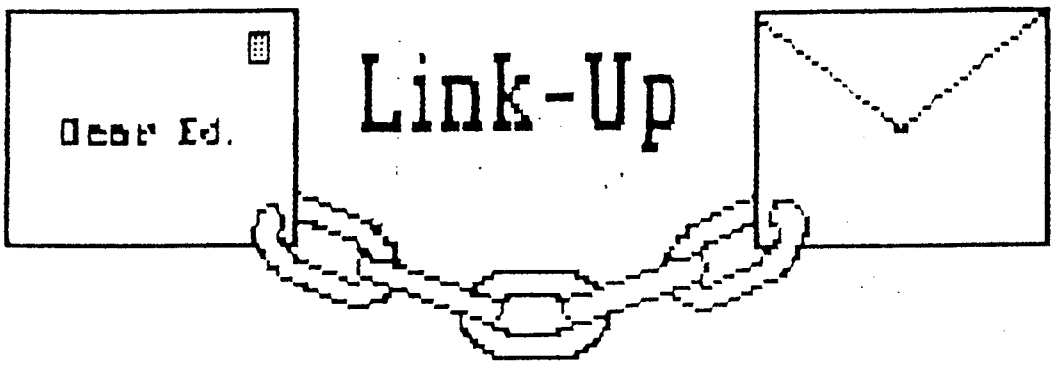
As reported in a previous magazine, Volume 2 of COCO-LINK will no longer be reprinted. We have only one copy left of the following issues. Volume 2 Nos. 2,3 and 6.

$2.50 each. First come first served.

All of volume 3 will be available till the end of this year.

****





Dear Ed,
Re 'C' Compiler; I was just thinking about how I double sided all my disks by changing Rom to Ram, and poking locations for drive 0+2.
Well, a few months back I wanted to use MS2COCO (Aust Rainbow mag pge 30 July '86), but it needed 2 drives, so I swapped the pokes around so that the single sided MS-DOS disk, instead of being inserted in drive 1, was really drive 0 or the front side; then I converted the files to the back side.
Anyway, it sounds confusing, but in OS9 I think it would mean that you leave /d0 as /d0, and /d1 as /d0 as well; OS9 is very strict, so it probably wouldn't work, but it sounded good and that's the main thing.
Tandy sure do have a few nice people working for them, but their system stinks. Anyway, my controller blows up, so they leave it sitting around in their shop before they tell me they can't get parts. Now they have sent it to the USA which apparently takes 3 months. How they dare advertise "We service what we sell"!
I am writing this on TW64, a great program; it seems to be echoing characters on the CoCo3, that's why I just noticed that I said, "Hi, my name is Danny Piczak". Ha!
I guess it's better than Color Scripsit, when TW64 is not supposed to work on the CoCo3.
Now for some advice; if I wanted to get some new CoCo software from the USA (which I do), should I, and exactly how would I? Or should I get it locally, where it is shipped into Australia, then back to Perth for me?

Do you think a game similar to 'California Games' (Commodore 64), could be written for the CoCo? Has anyone done it yet?

Danny Piczak. Lynwood. WA.

Dear Danny,
There is an upgrade of TW64 available for Coco 3. It is called TW128. Try giving APD a call for details.
Whether to buy direct from the states or not is purely a personal decision. It probably pays to check if the programme wanted is available in stock in Australia before sending to the US.
In "Robbie's Column" of this magazine we outline a scheme we are considering inaugurating. It may be just what you need. Let us hear your opinions on this.

****

Dear Ed,
I have been following your adventures with OS-9 with some interest, even though I decided there is no future in it for someone with only 128K and a single disk drive.
One thing I learned along the way was that you can simplify backups of your disks. Just DSKINI a disk in the normal way, then use a program like SUPERDUP to backup your OS-9 system and configuration disks. That saved a lot of hassle.
Last letter I commented on the improved readability of your 3 column listings; that, and how you did it, should be worth a paragraph in COCO-LINK.

Keiran. Cremorne. nsw.

Dear Keiran,
By public demand we have reverted to 2 column by 32 character listings.

****

Dear Ed,
After some poking around on my CoCo3 I discovered how to run double sided drives. This Poke will NOT work on CoCo2.
&HD89F,65 make 2 back of 0 Poke&HD8A0,66 make 3 back of 1.

The next set of Pokes I discovered in an old CoCo OZ magazine. I don't know who contributed it, but I have only seen it in print once.
These Pokes enable the CoCo3's disk drive to run at 6milli seconds. Poke55232,0 : Poke55318,20

Does anybody know why, when you press CTRL+ALT + Reset on a CoCo3 it only clears the lower 28K? Don't believe me? Here are two tests.
You can either find an ML program, and its start address, Load it and EXEC it, press CTRL+ALT and Reset, then EXEC the start address, and away it goes again. This will not work on those darn marbles, as I have discovered that it places its game too far up in memory; (Yes I do own it, and it is quite good).
The 2nd is to run ANY CoCo3 game, except those darn marbles, as it uses 200K above the HSCREEN2 page.
After the program has run for a while, do the old CTRL+ALT Reset. Now, let's do something really tricky;

type in this small program:-

```
10 POKE &HE6C6,18:POKE&HE6C7,18:'DISABLE HCLS
20 HSCREEN2
30 GOTO 30
```

You have the picture of the game screen.
But it ofcourse does NOT have the correct palette registers due to the reset trick.
Now, do some SMOOTH scroll to out show the AMIGA, using HARDWARE SCROLL.
Add this line.

```
25 FOR X = 0 TO 127:POKE&HFF9F,X:NEXT
```

I don't know why it grabs an extra half of the screen, but I have not come up with a solution; it might be because I used too many values in the variable, V.

This next piece needs a 512K COCO3 ONLY.
Want to use about 250K to draw on?
Well, to move around this 250K of memory, using the above program; delete line 25 and instead type in:-

```
30 POKE&HFF9D,X
```

This moves around the memory using the vertical offset vertical register; all you have to do is discover a way how to Poke the screen in; I have had no such luck.

I hope you may be able to use this at sometime in the mag; also look out when I send in an animated CoCo3 picture of Bugs Bunny with DIGITIZED SOUND.

Desmond Rae. Mt Isa.

****

Dear Ed,
Thanks for sending the PD Disk 014, but I am having trouble with (3 boxes) something, loads OK only on RUN, it shows only "OK".
3VAGAS I get UL error in 7; delete 7 and it RUNs improperly. The rest of the programs are fine.
I only backed up once, and used the BU. Both original and BU are the same errors.
I cannot see anything obviously amiss in the programs, as I know they must have been OK originally,

Dr Val Stephen. Camberwell Vic

Dear Val,
The problems with PD 014 caused me to put on my thinking cap and do a bit of experimenting.
Firstly 3BOXES. I really can't explain what happened to your disk. My Master worked OK but I got the same result as you did on the disk you returned. I just recopied the master programme over the bad one and now that seems to be alright.
There is one thing though, it doesn't seem to like following 3VAGAS. This gives it a SN Error at some unknown line. I have found that it is best to run this game after a cold start.
Next we have the UL error in 3VAGAS. What has happened here is that, through some cause unknown, that line has been masked and does not show up when RUN. I'm sorry to say that this problem is on my master disk. I'm not sure how to get rid of it but there is a way round it.
When the programme comes up with the UL Error after RUNning 3VAGAS, just RUN the programme again and it will perform OK. I don't know why this is but it is a solution.
I will be adding a NOTE to the cover of all disk 014's in future.
Thank you for informing me of these problems. I am sorry for any inconvenience they may have caused you.

****

Dear Ed,
I was surprised to find no answer to Bernard Fletcher's letter in October 1990 LINK UP; there is a POKE from Australian Rainbow Aug '85.
POKE 155,n where n is the character limit.
Sounds alright but it doesn't work on my set up. I have had the same trouble myself; having to use a paper that won't fit into a manilla folder, only because of a few long lines. Using condensed font got me through. After reading Dec90 & Feb91 LINK- UP and finding no answers, I put some thought into it, and the answer is simple. Check your printer manual for right margin set. Mine is CHR$(27);CHR$(82);CHR$(n) . Let's assume I want a carriage return at 55 chrs, type in or LOAD the program and follow it with this one liner, using line 0 irrespective of how large the last line number is. Set the printer up NOW!!

```
0 PRINT#-2,CHR$(27);CHR$(82);CHR$(55);:LLIST 1-:END
```

Press ENTER and RUN the program. Do NOT LLIST it!! it's self LLISTing.

I have a problem myself. I'm using a COCO3 attached to a mono VM2 monitor. On the hi res screen only, I get 4 yellow lines across the top of the screen, about 20mm apart. I don't get it on the TV, so it sounds like the monitor.
Can a reader tell me whether it's a workshop repair, or something I haven't done right. I realise the VM2 is fed from the video outlet and the TV goes through the RF adaptor.
I will be one of many readers awaiting Robbie's article on the M.L. for beginners; like Roy Simmons I am lost among the trees before I enter the forest, and there is no one to ask questions as I go; so here's looking forward to it.
Keep up the good work on the magazine.

Sam Thompson. Gold Coast. QLD.

****

END

The ML/BASIC Connection

By George McLintock

INTRODUCTION
****

Machine language routines are used with Basic programs mainly to speed up an operation or procedure. Basic is an easy language to use but some repetitive type operations can be completed a lot faster by using ML routines called from Basic. Another reason for using ML routines is to do things in a way which can't be done with Basic, or simply to experiment with the machine to see what happens.

A desire to 'learn a bit of ML' is common for people with an interest in computers. However, it is not as easy to achieve as a similar desire to 'learn a bit of Basic'. The main problem here is not with learning ML itself (Assembler is a very simple language), but with learning how to do things at the binary level of logic. With ML, all you have to deal with is binary data (bytes) at machine addresses and you have to do all operations at this level. The simple Basic command of PRINT 2 + 2 takes quite a lot of ML to duplicate,

Developing ML routines requires some knowledge of both language, machine specific information on interfacing, hardware etc and an understanding of some general programming techniques and data structures etc. The main problem for the CoCo these days is that it is not a 'current product' so that the main source of information (if you don't already have it) is the second hand market.

When writing programs you can often obtain significant differences in execution time depending on how you actually perform a function. This is particularly so for operations on the graphic screen. Basic provides commands which allow you to do most things you are ever likely to require on the graphic screen. However, these are normally performed on a 'pixil by pixil' basis which can be relatively slow for some operations. You can sometimes speed things up a bit by using PEEKs and POKE's to get the same effect on a 'byte by byte' basis. To do this, you need some knowledge about binary level logic and the specifics of your machine, but you don't need to know anything about Assembler or ML.

In situations where you want the maximum possible speed you have to use ML routines. In most cases you will find that it is only a small part of the total operation which is slow when coded in Basic, so it is normally not necessary to code the whole application in ML, you can code most of it in Basic and only use ML routines for the specific bits which are slow in Basic.

DEMO PROGRAMME
****

As a demonstration of the sort of difference that can be achieved, I've included a small Basic program (Listing 1) that uses three different procedures to invert the PMODE 4 screen. This is a simple operation which involves changing the status of each pixil on the screen. ie if it is on now, turn it off, while if it is off now, then turn it on. The procedures are (1) Normal basic - takes 17 minutes. (2) PEEK/POKE's - takes 98 seconds and (3) ML - takes a small fraction of a second.

For the normal Basic loop, you don't need to know anything about the specifics of the machine or the language interfaces. The operation is performed at a logical level where the same loop will have the same effect on practically any other computer type using Basic. For the other two loops you need to know some specific details about the CoCo graphic screens, memory pointers for Basic and a bit of binary arithmetic. The extra bits required for the ML loop include some knowledge of 6809 Assembler and interfacing between ML routines and Basic.

The relevant bits for the second procedure include: The address of the start of the PMODE screen is held at location Hex BA. The end address is at Hex B7. The Pmode

4 screen is 6K bytes long. (32 bytes per row x 192 rows deep). Each pixil is represented by a single bit (on or off) so can store 8 pixils in a byte. The screen is 256 pixils wide so it requires 32 bytes (256 / 8) to hold a row. Consectitive pixils are held in adjoining memory locations. If all bits in a byte are on, then it has a decimal value of 255, so if you substract the actual byte value from 255 you will effectively invert each bit. This is same effect as the COM instruction in assembler.

For the third procedure, all that the ML routine does is to loop through screen memory and compliment the bits. It uses the start and end address from low memory, with index addressing. The code is relocatable and can be executed from anywhere in memory. It will work the same with any model CoCo.

INTERFACING ML WITH BASIC
****

When using ML routines you need a procedure to set up the ML code in memory and to interface between it and the Basic program. There are lots of different ways of doing this which have been described in magazines from time to time. For myself, I use LOADM files from disk when testing because this is the quickest and easiest to change. For normal use, I prefer the procedure to incorporate the ML code at the end of the Basic program itself because this is the most convenient and efficient to use. The ML routine must be relocatable for this procedure to work.The LOADM procedure is described in the EDTASM manual. You assemble the binary file to disk or tape with EDTASM. You then (C)LOADM the file back to memory from within a Basic program and execute as required.

I consider this to be the best way to include a ML routine which is designed specifically to run with a specific Basic program. It uses the minimum amount of memory and disk space and it is always available when required by the Basic program. Also it has no real adverse impact on future use of the program. You can continue to EDIT it normally, MERGE other programs into it etc and do most things. The only thing you can't do is save the program in ASCII and then reload the ASCII version. This will remove the ML code from the end of the program, and is the best way of doing this particular function.

The other thing which will remove it is ML routines that re-arrange the Basic program in memory, UNLESS the utility is specifically coded to retain any ML code which is included in this way.

When you do a number of ML routines yourself it is easy to include later ammendments to improve things, but it can become tedious to update all the supporting text to incorporate such changes. Naturally enough, it is not always done except for special purposes. I mark my own hardcopy with the changes, but don't always update the text files on disk, so you don't get the changes.

For small and/or 'quickie' routines (like this one), DATA statements POKE'd into memory are more convenient because it requires less effort to set up and implement.

The use of DATA statements is described in the manuals for the CoCos 1 and 2, but not the CoCo 3 manual. It is used in the example here. You will note that the Hex values in the DATA statement are the same as the assembled code on the left hand side of Listing 2. There are quite a few variations of this one which relate mainly to where you put the ML code in memory. eg protected memory, an array, extra memory, one of the buffers etc

I guess that the main point of this article is to emphasise that if you want to do a bit of ML programming then you will need access to varying amounts of other information as well, which has no direct relationship to the ML routines themselves. There is no single source of all the information you are likely to require. You pick it up from magazines, reference material and experimenting to see what happens. The best source of general type information is in old magazines. Quite often it can appear as incidental points in a larger description of how a procedure or utility works. The best reference manual I've seen is 'Assembly Language Programming' by William Barden Jnr. (A Tandy Publication).

LISTING 2
****

NOTE Program is ORG'ed at 32000 for convenience only. It is relocatable and could be ORG'ed anywhere

```
7D00                00100          ORG       32000
7D00 9E   BA      00110 START      LDX       <$BA     START SCN
7D02 A6   84      00120 T1         LDA       ,X       NEXT BYTE
7D04 43             00130           COMA               INVERT
BITS
7D05 A7     80      00140           STA       ,X+      RESTORE
AND INC
7D07 9C   B7      00150          CMPX       <$B7     END SCN
7D09 26   F7      00160          BNE        T1       DO ALL
7D0B 39           00170          RTS                 TO BASIC
          7D00    00180          END        START
```

ML USING DATA STATEMENTS
****

I have included two routines here which are intended to give you a bit of a start with using ML routines and interfacing them with Basic via DATA statements.

GETML/BAS (Listing 3) shows the code required to extract a ML routine from memory, (put there with LOADM) and convert it to DATA statements in a file on disk. This one is set up for COMSBUF, (see PD DISK 022), but only minor changes are required to suit any other program. This ML is 478 bytes long.

The ADDML/BAS (Listing 4) is similar, but for a different routine. It is just another example of the same thing.

The general sequence for putting a ML routine at the end of a Basic program is to RUN GETML to get the ML routine into DATA statements on disk. LOAD ADDML and edit the code to suit the ML routine. ie adjust number of bytes in ML routine, and per data statement etc to suit. Then MERGE the MLDATA file as produced by GETML into the program. You then SAVE the combined program in ASCII (ie with ,A after the file name). You then LOAD the Basic program that you want to add the ML code to, and then MERGE the ADDML/BAS program into it. You then RUN 55000 which will actually put the ML code into the program and adjust the pointers to suit. You can then SAVE the program again, with the ML code included.

Other bits to adjust include:
The length of the ML routine to be included in the program is included in the string in Line # 55110, as a hex number in character positions 9-12. This is a separate little ML routine which changes the pointers for the Basic program. The value to substract from the pointer in location 27 (end of Basic program) is 3 greater than the length of the ML routine as included in line 55110

The procedure itself is based on a routine described by Charles Roslund in 'Charlie's Machine', Australian Rainbow Jan 1983. That article describes how to do it from the keyboard, what I've done here is to develop a small ML routine which does the same thing automatically as part of a program. The manual procedure is too tedeous and difficult to use.

The ML routine to do it, is fairly simple and consists of the following code

```
LDX   <27     Current end of program
LEAX  478,X   Add Adjustment
CLR   ,X+     Make zero bytes
CLR   ,X+     as required for Basic
CLR   ,X+
STX   <27     New values in pointer
RTS
```

The adjustment is the number of bytes of ML code to be included, and is the bit which is changed for each routine. As coded, it must be followed with a CLEAR statement in Basic. This is to reset the pointers in 29 and 31 to the same value, and is a 'better' way of doing it.

If you have a local users group you can probably get some assistance there. Alternatively, if you have a specific problem you can contact me about it. But be aware that I don't have the time or the inclination to provide a detailed tutorial on the basic aspects of Assembler or interfacing. If you want to 'follow up' the subject I suggest you try PD Disk 022, and also go through some old magazines. However, having said that, I will respond to any queries and do my best to assist, but you will need to do a lot of research and experimenting of your own.

Listing 1

```
10 WIDTH 32
20 CLS:PRINT "WAYS OF INVERTING
PMODE SCREEN"
30 PRINT:PRINT "BY GEORGE MCLINT
OCK":PRINT:PRINT "OPTIONS":PRINT
40 PRINT "1 NORMAL BASIC": PRINT
 "2 PEEK / POKE":PRINT "3 ASSEMB
LER"
50 PRINT:PRINT "PRESS ENTER AT A
NY TIME TO":PRINT "RETURN TO THI
S MENU"
60 PRINT:PRINT "ENTER CHOICE"
70 A$=INKEY$:IF A$="" THEN 70
80 IF A$<"1" OR A$>"3" THEN 70
90 PMODE 4,1:SCREEN 1,0:PCLS
100 LINE (0,0)-(128,96),PSET,BF
110 LINE (128,96)-(255,191),PSET
,BF
120 ON VAL(A$) GOSUB 140,190,260
130 SCREEN 0:GOTO 20
140 FOR Y=0 TO 191: FOR X=0 TO 2
55
150 IF PPOINT (X,Y)=0 THEN PSET
(X,Y,1) ELSE PSET (X,Y,0)
160 IF INKEY$<>"" THEN X=300:Y=X
170 NEXT X,Y
180 RETURN
190 M=PEEK(&HBA)*256
200 FOR X=M TO M+&H1800-1
210 POKE X,255-PEEK(X)
220 IF INKEY$<>"" THEN X=&H8000
230 NEXT X
240 RETURN
250 DATA 9E,BA,A6,84,43,A7,80,9C
,B7,26,F7,39
260 RESTORE:FOR X=0 TO 11: READ
A$: POKE &H1DA+X,VAL("&H"+A$)
270 NEXT X: EXEC &H1DA
280 IF INKEY$ = "" THEN 280
290 RETURN
```

Listing 3

```
5 CLEAR 2000
10 LN=58000:OPEN"O",#1,"DATAML1/
BAS"
20 TS=32000:FOR X=TS TO TS+478 S
TEP 25
30 A$=MID$(STR$(LN),2)+" DATA ":
B=0
40 IF X<TS+474 THEN N=25 ELSE N=
3
50 FOR Y=0 TO N-1
60 B=B+PEEK(X+Y)
70 A$=A$+HEX$(PEEK(X+Y))+","
80 NEXT Y
90 A$=A$+HEX$(B):LN=LN+10
```

```
100 PRINTA$
110 PRINT#1,A$
130 NEXT X
140 CLOSE:STOP
```

Listing 4

```
57000 PRINT "CHECKING DATA":LN=5
8000:FOR X=0 TO 483 STEP 50:IF X
<449 THEN N=50 ELSE N=33
57010 PRINT LN;:A=0:FOR Y=0 TO N
-1
57020 READ C$:B=VAL("&H"+C$):A=A
+B
57030 NEXT Y:READ C$:IF A<> VAL(
"&H"+C$) THEN PRINT "ERROR IN LI
NE NO";LN:STOP
57040 LN=LN+10:NEXT X
57050 '
57060 PRINT:PRINT "DATA OK"
57070 '
57080 M$="9E1B308901E36F806F806F
809F1B39":Y=&H01DA
57090 B=0:FOR X=1 TO 30 STEP 2:N
=VAL("&H"+MID$(M$,X,2)):B=B+N:PO
KE Y,N:Y=Y+1:NEXT X
57100 IF B <> &H616 THEN PRINT "
ERROR IN LINE NO 57080":STOP
57110 EXEC &H1DA:CLEAR
57120 PRINT "NOW POKING DATA":RE
STORE
57130 A=PEEK(27)*256+PEEK(28)-48
6:LN=58000
57140 FOR X=0 TO 483 STEP 50:IF
X<449 THEN N=50 ELSE N=33
57150 PRINT LN;:LN=LN+10:FOR Y=0
 TO N-1:READ C$
57160 POKE A,VAL("&H"+C$):A=A+1:
NEXT Y:READ C$
57170 NEXT X
57180 PRINT:PRINT " DUMP2   NOW
ADDED TO END OF    BASIC PROGRAM
":PRINT "AND EXTRA BASIC CODE DE
LETED"
57190 '
57200 DEL 57000-
```

George McLintock
7 Logan Street
NARRABUNDAH ACT 2604
Phone (06) 295 6590
Week ends and 8.30 - 10.30 PM week nights

PUBLIC DOMAIN SOFTWARE

This month we have two Public Domain Disks to add to our Library.

PD 022 McLintock Utilities

The first is a disk crammed with Utilities and extensive comments in the form of Document files. All these are by George McLintock. The explanations of the use of ML in these utilities are a must for anyone interested in in ML.

SPECIAL OFFER FOR CASSETTE OWNERS.

As a special offer George is willing to transfer these files to tape for those cassette users who do not have access to disk systems. There will be a cover charge of $10.00 which will include post and packing. If you are interested in this offer write to:

George McLintock
7 Logan St.
Narrumbundah
ACT 2604

The utilities on disk 022 are:

XCOM	ERASE
COMSBUFF	DIVERT
MKI	TRANSFER
MGEFILES	PRINTDOC

****

PD 041 GRAPHICS

The second disk is a graphics disk for Coco 3. This disk has some really clever graphic effects and displays different modes of accomplishing these effects. The files included:

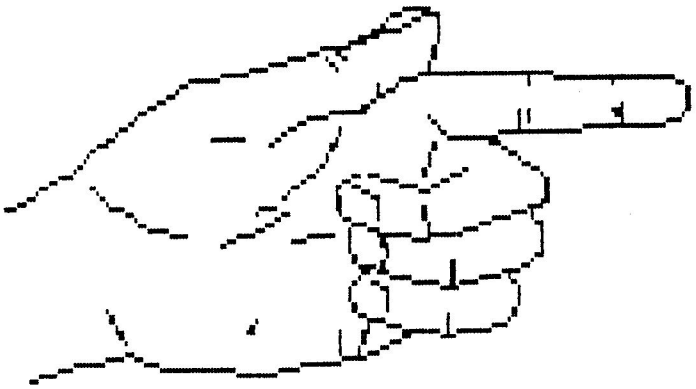
DIR	ROCKFEST
AIRPORT	WATERFALL
BOUNCING BALL	WORLDMAP
NUDE	

The waterfall graphic is animated and really something special.

****

COCO-LINK PD Disks are all $5.00 each.

****



Better BASIC Part 14

Coco 3 Graphics

Bu Keiran Kenny

Following are four short programmes which show different ways of manipulating Coco 3 graphics. The routines in these programmes can all be used in your own programmes to help give them that extra litte bit of pizzaz.

All the programmes require a Coco 3 with 128K.

MULTICOLORED PATCHWORK PIX (Listing 1)

This HSCREEN2 program (Listing 1) divides the screen into a grid of 32 x 24 squares (total 768) that you can fill with 14 different colors plus black to form a colorful patchwork.

In the top lefthand square is a flashing cursor. To move the cursor, hold down an arrow key, or T for north-west, Y for north-east, G forsouth-west or H for south-east.

When the cursor is positioned where you want it, press a key 0, 2 to 9, or A to F to paint the square (F is black). You cannot use color 1 (yellow) for a fill. Colors 0 through 7 are the CoCo 3 default colors (green through orange). Colors 8 and 9, and A through F are as in the DATA values in line 40. You could experiment with changes in these values. You can change the color in a square by pressing another color key.

A composition that is symmetrical in both shape and color patch placement will probably be most attractive, but feel free to exercise your own imagination.

When your composition is finished, press ENTER. The grid will disappear leaving only the color patches on the screen. Press BREAK to end the program. Press any other key to set up the grid again for another composition.

HSCREEN4 ARTIFACT COLORS (Listing 2 & 3)

HSCREEN4 offers the advantage of the highest resolution available on the CoCo 3 graphic screen, but when you use the HPAINT command you are restricted to three colors.

However, you also have available 256 artifact color patterns that can be used as fills using the HPAINT command. Type and run listing 2 to see all patterns from 1 through 255 (pattern 0 is black).

Listing 3 is an example of using the artifact color patterns with the HPAINT command. In line 70, the statement POKE&HE79B,196 accesses the patterns, and POKE&HE79C,RND(255) establishes the number of the pattern to be used. I used RND(255) for demonstration purposes but if you note likely pattern numbers from listing 1, you can substitute the pattern number for RND(255) at each stage in your own listings.

Normally, each HPAINT statement should be followed by POKE&HE79B,212:POKE&HE79C,181, as in line 90 of listing 3, to restore normal operating conditions, otherwise your next figure in the listing will have a stippled outline and your paint will run out all over the screen!

If your program includes an ONBRKGOTO statement, as in line 30 in both listings, include these pokes in the line addressed by the statement.

The patterns show best on a black background on an RGB monitor, but you could experiment with other colors.
For an interesting, if irrelevant, psychodelic display, enter the following line in listing 3:

105 PALETTERND(4)-1,RND(64)-1:GOTO105.

CHOOSE YOUR PALETTE (Listing 4)

When I exercise my very limited artistic talent on CoCo 3, I prefer to begin by leaving the HSCREEN2 default color slots, 0 to 7, as they are and poke palette values into slots 8 to 15 to get whatever other colors I may need.

Thus, say my masterpiece has a deep blue sea, slot 2 and sky, slot 5, but I want another, darker or lighter blue for elsewhere on the screen. So I can run PALETEX, (Listing 4), to put color patches in the default slots, 1 to 15, onto the screen. For demonstration purposes, one slot has to provide the background color so palette 0 is poked into slot 0 to produce a black screen.

Press any key and palette values 0 - 63 are poked into slots 8 - 15 in numerical order, eight at a time. When blueish tints show I can compare them with colors 2 and 5 and choose the one I want. Then I can return to my listing and enter the appropriate PALETTE statement.

You can save and use PALETEX as a stand-alone program, or you could load PALETEX from line 9999. Add a line GOTO 9999, and begin typing your program at line 10 or whatever. When you get stuck for a variant color, just RUN10000 and work your way through PALETEX until you find one that suits your need.

Otherwise, you could save PALETEX as an ASCII file and merge it with a listing you have already begun if you have difficulty in finding the right color for an element on your graphic screen.

END

```
 0 'PIXGRID' Copyright 1990
    Keiran Kenny, Sydney
 1 'CoCo-Link 11.11.90
 10 POKE65497,0
 20 ONBRKGOTO360
 30 FORSL=8TO15:READPL:PALETTESL,
 PL:NEXT
 40 DATA 24,33,40,43,48,52,56,0
 50 HSCREEN2:HCLS15:POKE&HFFB1,54
 60 HLINE(0,0)-(319,191),PSET,B
 70 PK=PEEK(135)
 80 FORX=10TO310STEP10
 90 HLINE(X,0)-(X,191),PSET
 100 NEXT
 110 FORY=8TO184STEP8
 120 HLINE(0,Y)-(319,Y),PSET
 130 NEXT
 140 X=5:Y=4
 150 PK=PEEK(135)
 160 IFPK=48ORPK>49ANDPK<58THENCL
 =PK-48:GOTO330ELSEIFPK>64ANDPK<7
 1THENCL=PK-55:GOTO330
 170 IFPK=13THEN340
 180 IFPEEK(341)=247THENY=Y-8
 190 IFPEEK(342)=247THENY=Y+8
 200 IFPEEK(343)=247THENX=X-10
 210 IFPEEK(342)=251THENX=X-10:Y=
 Y-8
 220 IFPEEK(339)=247THENX=X+10:Y=
 Y-8
 230 IFPEEK(345)=254THENX=X-10:Y=
 Y+8
 240 IFPEEK(338)=253THENX=X+10:Y=
 Y+8
 250 IFPEEK(344)=247THENX=X+10
 260 IFX<5THENX=5
 270 IFX>315THENX=315
 280 IFY<4THENY=4
 290 IFY>188THENY=188
 300 DR=HPOINT(X,Y)
 310 HSET(X,Y,1):FORDL=1TO100:NEX
 T:HSET(X,Y,DR)
 320 GOTO150
 330 POKE135,0:HPAINT(X,Y),CL,1:G
 OTO150
 340 POKE135,0:PALETTE1,0
 350 EXEC44539:K$=INKEY$:GOTO50
 360 POKE65496,0:WIDTH32:RGB:END
```

```
 0 'LISTING1' Copyright Keiran
    Kenny, Sydney, 1990
 10 PALETTE0,0
 20 POKE65497,0
 30 ONBRKGOTO130
 40 HSCREEN4
 50 FORCL=1TO255
 60 POKE&HE79B,196:POKE&HE79C,CL
 70 HLINE(280,76)-(360,116),PSET,
 BF
 80 POKE&HE79B,212:POKE&HE79C,181
 90 HPRINT(28,16),"HI-RES COLOR P
 ATT #:"+STR$(CL)
 100 FORDL=1TO1500:NEXT:HCLS
 110 NEXT
 120 GOTO50
 130 POKE65496,0:POKE&HE79B,212:P
 OKE&HE79C,181:RGB:END
```

```
 0 'LISTING2' Copyright Keiran
    Kenny, Sydney, 1990
 10 PALETTE0,0
 20 POKE65497,0
 30 ONBRKGOTO130
 40 HSCREEN4
```

```
50 FORR=9TO315STEP18
60 HCIRCLE(320,96),R,,.5
70 POKE&HE79B,196:POKE&HE79C,RND
(255)
80 HPAINT(320+R-2,96),,1
90 POKE&HE79B,212:POKE&HE79C,181
100 NEXT
110 FORDL=1TO1500:NEXT:HCLS
120 GOTO50
130 POKE65496,0:POKE&HE79B,212:P
OKE&HE79C,181:RGB:END
```

```
0 'PALETEX' Copyright 1990
   Keiran Kenny, Sydney.
10000 ONBRKGOTO10290
10010 POKE65497,0
10020 PALETTE0,0
10030 HSCREEN2
10040 CL=0
10050 KY$="PRESS ANY KEY."
10060 FORQ=0TO280STEP40
10070 HCOLORCL:HLINE(Q,VT)-(Q+32
,VT+32),PSET,BF
10080 CL=CL+1:IFCL>15THEN10110
10090 NEXT
10100 VT=VT+64:GOTO10060
10110 HCOLOR1:HPRINT(18,5),"SLOT
S"
10120 PR=0:HCOLOR1:FORT=8TO15:HP
RINT(PR,7),T:PR=PR+5:NEXT
10130 HPRINT(13,23),KY$:EXEC4453
9
10140 HPRINT(16,15),"PALETTES"
10150 PL=0:PX=7:SL=8:PR=0
10160 FORP=PL TOPX
10170 PALETTESL,P
10180 HCOLOR1:HPRINT(PR,13),P
10190 IFP=63THEN10260
10200 PR=PR+5
10210 SL=SL+1
10220 IFP=PX THEN10240
10230 NEXT
10240 HPRINT(13,23),KY$:EXEC4453
9:HCOLOR0:HPRINT(13,23),KY$:HCOL
OR1:PR=0:HCOLOR0:FORT=PX-7 TOPX:
HPRINT(PR,13),T:PR=PR+5:NEXT:HCO
LOR1
10250 PR=0:SL=8:PL=PL+8:PX=PX+8:
GOTO10160
10260 HPRINT(11,23),"<R>EPEAT OR
 <E>ND?"
10270 K$=INKEY$:IFK$<>"R"ANDK$<>
"E"THEN10270
10280 IFK$="R"THENFORSL=8TO15:RE
ADPL:PALETTESL,PL:NEXT:RUN
10290 POKE65496,0:RGB:CLS:END
10300 DATA 0,18,0,63,0,18,0,38
```

Kidz Bitz

By Val Stephen

This little short programme is designed to help our younger readers learn a little bit more about entering programmes into the computer.

It helps to show how to position the words on the screen and insert the time gaps between them.

Hopefully we will be able to add more of these little programmes to show how some of the simpler computer tricks are done.

```
10 CLS:POKE&H95C9,&H39:POKE&HFF2
2,&H34
20 '*****MISTAKES******
30 PRINT:PRINT:PRINT"      THIS I
S THE MASTER MIND OF COMPUTERS -
 IT  IS  PERFECT  IN EVERY WAY -
 IT IS TOTALLY AND  COMPLETELY
ERROR FREE!"
40 FOR A=1TO3000:NEXTA:CLS
50 CLS
60 PRINT@98,"THIS COMPUTER NEVER
 EVER MAKES"
70 PRINT@238, "MISSTAKZXES"
80 FORB=1TO2000:NEXTB:PRINT@238,
"MISSTAK-XES"
90 FOR C=1TO500:NEXTC
100 PRINT@238,"MISSTAK--ES"
110 FOR D=1TO500:NEXTD
120 PRINT@238,"MISSTAK-ES"
130 FOR E=1TO500:NEXTE
140 PRINT@238,"MISSTAKES!"
150 FOR X=1TO2000:NEXTX:CLS
160 PRINT@238,"MISTAKES!";
170 FORK=1TO2000:NEXTK
180 CLS
190 PRINT@100,"SEE WHAT I MEANNN
NN!!!";
200 FOR X=0TO2000:NEXTX:CLS
210 PRINT@100,"SEE WHAT I MEAN!!
!";
220 FOR Y=1TO1000:NEXTY
230 CLS:PRINT @ 264,"BYE BYE FOL
KS!"
240 GOTO230
```

CHANGE

To Running Writing Script

By George McLintock

This programme allows you to alter the script for the "Running Writing" programme from last issue

INTRODUCTION
****

There is nothing special about the script used with the "RUNNING WRITING" programme in the last issue of COCO-LINK. It just happens to be one that I settled on at the time. It has changed quite a bit from the one I started with, and is likely to change again in the future. To make it easy to change the script, I've developed another program, called CHANGE, which does all the work required.

This program uses exactly the same code from line 2800 onwards as used by the SCRIPT (Running Writing) program. This is deliberate, and the line numbers used must be maintained as exactly the same between the two programs

To use this, you type in the listing as shown and then merge the lines from 2800 onwards from SCRIPT. To get the lines from script, you load it, delete from 1 to 2790 and then save the results as SAVE "FRED",A. Then load CHANGE again and MERGE "FRED", then save CHANGE again. A similar sort of sequence is used to implement changes you make to the script, so you need to be reasonably familar with the process.

The basis of the program is that it will display any script character on the screen, and allow you to modify it in any way you like. It then writes out to a file on disk a replacement line for the DATA statement that defines that script character in the program. The record written to disk is in the same format as produced by the Basic save in ASCII command, so you can merge that file back into your program and so change the DATA statement that defines any character. This is the reason why the line numbers must be the same in both programs

The program also allows you to add new characters to the script. When this is done, you also have to change one of the Basic program lines as well. eg lines 2800. The program will produce the new program line required as well, so that all associated changes are included when you merge the new file back again.

CHANGE is fairly straight forward. It includes getting commands from the keyboard and positioning the cursor on the screen, the flashing cursor, manipulating individual bits in a byte with Basic and building up a Basic program command line which can be merged back into the Basic program again.

INITIALISATION
****

The initialisation for this program follows much the same procedure as for Script. All characters defined are read into the K$() array so that we can work with them. The extra here is the Q() array which sets up some bit patterns that will be used to pack and unpack the bytes used to define individual script characters. Individual bits within a byte are processed in sequence from the high order to the low order, so Q(0) has the high order bit only on (decimal 128) through to Q(7) which has the low order bit only on (binary 1). I've used decimal numbers here simply for convenience to produce the bit patterns.

HOW THE PROGRAMME WORKS
****

The following is a fairly comprehensive explanation of how the routines in this programme work. With an understanding of how the programme works you will be able to incorporate some of the routines and ideas into your own programmes.

The process requires us to be able to change any single bit in the pattern which defines each character. Individual characters are 21 dots high and up to 23 dots wide. To allow some flexibility, I've set this for a maximum character width of 40 dots. The text screen allows for 25 lines deep, so I've chosen to display individual characters on this screen for modification. When displayed, a full stop '.' is used to indicate a bit off, and a lower case 'x' indicates a bit on, for each dot position. This represents how a character will appear when printed.

After you select a character to be altered, the program extracts the individual bits that form that character and puts them into a numeric array which is defined as DIM V(40,21). Individual elements in the array are set equal to zero if the corresponding bit in the character definition is off, and equal to 1 if the corresponding bit is on. The relationship between array elements and dot positions should be obvious from the DIM statement. The high order elements correspond to columns and the low order elements correspond to rows.

The loop from lines 160 to 210 translates the individual bits in each byte from the definition to its corresponding position in the V() array. To start, all elements in the V() array are set equal to zero. To allow changes to be made in both directions, the existing character is set up towards the middle of the array. It extends from column number ST (start of character) to ED (end of character). ST is set to 10, and ED is calculated according to the value in W, which is the width of the character in dots.

The extraction of individual bits occurs in two stages. The Y loop (outer loop) steps through the three graphic rows used to define each character. The X loop steps across the columns for each row of dots. Individual bytes are extracted from the K$() array by the X loop. Its ASCII value is put into T. The Q loop (inner loop) extracts the individual bits from the byte by operating on the variable T.

Individual bits are extracted using the AND command. T is AND'ed with the corresponding element in the Q() array to see if the bit is on or off. If a bit is on, then the result of the AND will be non zero, and the corresponding element of V() is set to 1. If the bit is off, the value in V() is left unaltered. This is the reason for starting with all elements in V() set to zero.

At the end of this loop the character is now represented in the V() array, and is printed on the screen with the next loop. Some other bits of information are also printed on the screen and the program goes into a control loop to allow you to alter the way that the character is defined. X1 and Y1 track the position of the cursor on the screen and any changes made to the character are held in the V() array.

When you exit from this loop, V() contains the new definition of the character. The extra little bit here covers the start and end position of the new character. You may want to define a character with blank columns on both sides of it so as to separate it from the character next to it; e.g. most punctuation marks require this. The program includes an option, 'S', which allows you to set the start and end points for a character. If you enter 0,0 for this, the program will calculate a start and end column that corresponds to the columns containing the first dot on from the left, and the last dot on the right. If you want a space on either side of it, you re-enter the S option and specify the values you want; e.g. if the default is 12 and 22, to put 2 blanks on both sides you enter 10,24. Note that if you change the width of a character, you should recalculate the start and end points anyway.

If you exit from the loop with a 'Y' then the program builds up a new DATA statement to define that character and writes it out to the disk file. The DATA statement is built up as an ASCII string containing the same characters as you would enter through the keyboard if you typed it into a program. This is fairly straightforward, but you need to remember to include all the numbers, blanks and commas etc. that are required. During this loop it also builds a replacement string in the K$() to match the DATA statement.

To build up the Hex digits required to define the character, the program has to convert the bit pattern that now exists in V() back to their Hex equivalents. This is effectively the reverse of the loop that was used to extract them into V() in the first place. The main difference is that instead of using the AND command to find out if a bit is on or off, we use the OR command to turn the bits on that are required to be on. In this case, T starts with a value of zero. We test each element in V() to see if a bit should be on or not. If it is on, then we OR the corresponding element of Q() into T so as to turn that bit on.

At the end of the Q loop, T contains the value required. It is converted to a Hex value with the HEX$(T) command, which produces a string in the format required. The little IF test here is to ensure that all Hex digits will be two bytes long. This is required for the next bit of the program.

To be consistent, the program now executes a little loop to compress adjoining Hex values of '00' into the format used here; e.g. =12 means that the next 12 bytes have a value of zero. The line is then written to the output

file. To produce a record in the format required, you must use the PRINT command to write it out, you can't use WRITE in this instance.

The program includes a bit to allow you to define a new character, rather than to change an existing one. This works in much the same way except that you start with V() equal to all zeros, and you have to change some of the code in the program itself. The program produces the new line required and writes it to disk as well. The switch NCD is used to ensure that once you define a new character, the program will always write a new matching DATA statement to the disk file. If you define more than one new character in a session then you will get multiple copies of the new program line in the output file. This doesn't matter because the last one written out will be the one which remains effective after the file is merged back into the program. All the earlier ones will be replaced by the merge process.

The main point to remember with this sequence is that to make the new character definitions effective you must merge the file produced back into both programs, i.e. script and change, and save both programs back to disk again.

CHANGING THE HEIGHT OF THE CHARACTERS
****

The characters defined here are 21 dots high, which corresponds to 3 rows of graphics print. If you want to, you can change them to be 14 dots high, which will correspond to two rows of graphics print. The programs will adjust automatically to changes of this nature, but you will have to redefine all the characters to suit.

The easiest way to do this is to delete the lines from 3000 onwards and start again with only two characters defined to be 2 rows deep; e.g. key in

```
3000 DATA 2,8,=16 ' Not Recognised
3010 DATA 2,8,=16 ' Blank
```

and change line 2800 to be K$ = " ". You then run change and define all new characters to suit. They will be automatically added in the same way as done now, except that they will all only be 14 dots high.

ALTERATIONS TO THE PROGRAMME
****

The display could be easily changed to fit on the 40 column screen. you only require about 25 print positions to display the longest character, so the changes would be relatively minor

The SCRIPT programme from last issue of COCO-LINK will work on any Coco. However, the CHANGE programme, to modify the script characters requires a Coco 3 text screen. Again, it would not be unduly difficult to change this bit to work with the PMODE4 graphic screen and hence make it work on any Coco. But again I have chosen not to do so. If you want it to work that way, the changes are not difficult.

Next issue I will give some insight into producing NLQ quality script and also address right justification.

END

```
10 'Called CHANGE - To change ch
aracters in matrix form
20 ' By George McLintock - FOR E
pson compatable printers
25 'Modified for Tandy printers
30 PCLEAR 1: CLEAR 6000
40 DIM K$(90),K(132),V(40,24),Q(
8),N$(90)
50 CLS: PRINT "Change character
definitions - by George McLintoc
k":PRINT
60 PRINT "ENTER File Name for Ou
tput ";:LINE INPUT N1$
70 OPEN "O",#1,N1$
80 GOSUB 2800 'Set up Graphic st
rings
90 Q(0)=1: Q(1)=2:Q(2)=4:Q(3)=8:
Q(4)=16:Q(5)=32:Q(6)=64:Q(7)=128
95 WIDTH 80
100 KK$=CHR$(9)+CHR$(255)+CHR$(8
)+CHR$(255)+CHR$(94)+CHR$(255)+C
HR$(10)+CHR$(255)+"NSYQX10 nsyqx
" 'The CHR$(255) is to keep spac
ings
110 CLS: PRINT " Press ENTER to
Exit - or":PRINT "Enter characte
r to change"
120 INPUT N$: IF N$ = "" THEN CL
OSE: CLS: PRINT "MERGE ";N1$" wi
th Programs to make changes effe
ctive": STOP
130 TX = INSTR(K$,N$): C$ = K$(T
X): ST = 10: BS=13
140 W = ASC(MID$(C$,2,1)): ED =
ST+W-1 'Pointers to char on scre
en
150 X1=ST+INT(W/2): Y1=12 'Start
 cursor position
160 FOR X = 1 TO 40: FOR Y = 1 T
O CD: V(X,Y)=0: NEXT Y,X
170 FOR Y = 0 TO CX 'Extract fro
m strings to numeric array
180 FOR X = 1 TO W: T$ = MID$(C$
,3 + Y * W + X-1,1) : T = ASC(T$
)
190 FOR Q = 0 TO 6 'Do each bit
200 IF (T AND Q(Q)) > 0 THEN V(X
+ST-1,Y*7 + Q + 1) = 1
210 NEXT Q,X,Y
220 CLS: FOR X = 1 TO CD: LOCATE
 0,X-1 'Print on screen
```

```
230 FOR Y = 1 TO 40: IF V(Y,X) =
 0 THEN PRINT "."; ELSE PRINT "x
";
240 NEXT Y,X
250 LOCATE 3,BS: PRINT "B";
260 LOCATE ST-1,23: PRINT "S";:
LOCATE ED-1,23: PRINT "E";:LOCAT
E 80-LEN(N$(TX))-2,23: PRINT N$(
TX);
270 LOCATE 45,2: PRINT "N = New
Character": LOCATE 45,4: PRINT "
S = Set Start and End": LOCATE 4
5,6: PRINT "Y = Yes - Accept new
 definition"
280 LOCATE 45,8: PRINT "Q = Exit
 unchanged": LOCATE 45,10: PRINT
 "X or 1 = Turn on": LOCATE 45,1
2: PRINT "Space Bar or 0 = Turn
off"
290 LOCATE 45,14: PRINT "Arrow K
eys move cursor"
300 GOSUB 335: TT=LPEEK(XX):LPOK
E XX+1,7:CP=2:T=TIMER +23
310 M1$ = INKEY$: IF M1$ <> "" T
HEN 340
320 IF TIMER > T THEN T = TIMER
+ 23: IF CP = 1 THEN CP=2: LPOKE
 XX+1,0: ELSE CP=1: LPOKE XX+1,7
 'FLASH CURSOR
330 GOTO 310
335 XX=&H6C000 + (X1-1)*2 + (Y1-
1)*160 'CURSOR POSITION
337 RETURN
340 LPOKE XX+1,0: T=INSTR(KK$,M1
$): IF T>16 THEN T = T-8
350 IF T = 1 THEN X1=X1+1: IF X1
>40 THEN X1=1
360 IF T = 3 THEN X1=X1-1: IF X1
<1 THEN X1=40
370 IF T = 5 THEN Y1=Y1-1: IF Y1
<1 THEN Y1=CD
380 IF T = 7 THEN Y1=Y1+1: IF Y1
>CD THEN Y1=1
390 IF T = 9 THEN IF NCD = 0 THE
N 810 ELSE 590 'New
400 IF T = 10 THEN 460 'Start En
d
410 IF T = 11 THEN 590 'Yes
420 IF T = 12 THEN IF NCD = 0 TH
EN 110 ELSE 590 'Quit this
430 IF T = 13 OR T = 14 THEN V(X
1,Y1)=1: TT=120:LPOKE XX,TT 'CHA
NGE IT
440 IF T = 15 OR T = 16 THEN V(X
1,Y1)=0: TT=46: LPOKE XX,TT: 'CH
ANGE IT
450 GOTO 300
460 CLS: PRINT "Current Start ="
;ST: PRINT "Current End   = ";ED
470 PRINT "Use  0,0  to calculat
e default values"
480 PRINT: INPUT "Enter new Start
 , End Values";X,Y
490 IF X > 0 THEN ST = X ELSE GO
SUB 520
500 IF Y > 0 THEN ED = Y ELSE GO
SUB 550
510 GOTO 220
520 FOR ST = 1 TO 40:T=0:FOR X =
 1 TO CD: T = T+V(ST,X): NEXT X
530 IF T > 0 THEN RETURN
540 NEXT ST: ST = 10: RETURN
550 FOR ED = 40 TO 1 STEP -1: T=
0: FOR Y = 1 TO CD: T = T + V(ED
,Y): NEXT Y
560 IF T > 0 THEN RETURN
570 NEXT ED: ED=10+W: RETURN
580 'Generate new DATA Line
590 LN = 3000 + TX * 10: K$(TX)
= CHR$(CX+1)+CHR$(ED-ST+1)
600 B$ = MID$(STR$(LN),2)+" DATA
" + STR$(LCX+1)+"," + MID$(STR$(
ED-ST+1),2) + ","
610 FOR Y = 0 TO CX 'Each row
620 FOR X = ST TO ED: A$="":T=12
8 'Each column
630 FOR Q = 0 TO 6 'Each bit
640 IF V(X,Y*7+Q+1) > 0 THEN T =
 (T OR Q(Q))
650 NEXT Q: T$ = HEX$(T): IF LEN
(T$)=1 THEN T$="0"+T$
660 B$ = B$ + T$ + ",": K$(TX) =
 K$(TX) + CHR$(T)
670 NEXT X,Y: B$ = B$+N$(TX)
680 'Compress values of Zero
690 A$=B$: B$="": Q = 1: C = 0:
T$=""
700 T = INSTR(Q,A$,","): IF T =
0 THEN 770
710 T$=MID$(A$,Q,T-Q+1) : Q = T
+ 1
720 IF T$ <> "80," THEN 740
730 C = C + 1: GOTO 760
740 IF C > 0 THEN B$ = B$ + "="
+ MID$(STR$(C),2)+",": C = 0
750 B$ = B$ + T$: Q = T + 1
760 GOTO 700
770 IF C > 0 THEN B$ = B$ + "="+
MID$(STR$(C),2)+",": C = 0
780 B$ = B$ + MID$(A$,Q)
790 PRINT #1,B$ 'Write to Output
 File
800 NCD = 0: GOTO 220
810 CLS: PRINT "Defining new ele
ment": PRINT "Press enter to abo
rt, OR enter Key for New Charact
er": INPUT A$
820 IF A$="" THEN 220
830 IF LEN(A$)>1 THEN PRINT "Mus
t be single character":GOTO 810
840 K$ = K$ + A$: TX = LEN(K$):
N$(TX)=A$: IF A$="," THEN N$(TX)
```

Continued on Page 25

I purchased a DMP-105 when they were on special a while back. Its print quality is not best, but it provides an acceptable quality for a lot of the word processing type of work that I do.
One of the more interesting features of the DMP-105 is its range of print styles and the amount of variations that can be obtained through software printer control characters.

I have submitted a small utility, called HIGHLITE, which allows the use of some of these features when listing a Basic program. One of the problems with larger Basic programs is to be able to identify those parts of the code which perform the various functions within the total program Basic provides the REM statement to allow comments to be included in a program, but with larger programs it can be difficult to find the comments in the total listing.

The utility submitted inserts three extra bytes of printer control codes at the start and end of each comment in a Basic program. Specific printer control codes are set up in the utility by POKE's to certain addresses. To further highlight comments, the utility also changes the normal upper case characters for comments in the program to lower case characters. This feature can be by-passed by other POKE's.

HIGHLIGHTING OPTIONS
****
Three printer control codes appear sufficient for the various options available with the DMP-105. It also appears sufficient to provide a change of character stlye for most other printers . For the CGP-115 printer plotter it allows for a change in pen color for comments, and you can still come back to the original pen for normal printing. The option to change the case of characters in the comment is independent of the printer control characters.

Underlining comments is the simple option, but you can also change to elongated or condensed print as well as underline with three print control characters. You can also insert an extra line feed before and after each comment if you pass up the underline

A useful feature of the DMP-105, which is not adequately described in the manual, is the ability to vary the length of the line feed to any depth. You are not restricted to half or three quarter spacing, but can select any spacing you like The 27 90 n and 27 91 n control sequences provide this capability. eg 27 91 18 will provide one and a half spacing for all printing that follows until it is changed. A 27 90 18 will give an immediate one and a half line feed, after which the printer returns to normal spacing. This feature allows for a variable gap at the start and end of each comment line

USING THE UTILITY
****
The utility is a machine language program (168 bytes) which is relocatable and can be loaded into any convenient area of memory for execution. It is submitted as a Basic program with DATA statements to be POKE'd into CLEAR'ed memory at 32000. (ML source code also provided) If you have a 16K CoCo change the value of 32000 to 16000 throughout,

To use, RUN this Basic program. Then (C)LOAD the Basic program to be altered and EXEC 32006. The modified program can then be LLIST'ed and all comments will be emphasised in the listing You can also (C)SAVE the modified version of the program and keep it in this form. It will continue to execute normally and does not require any further special treatment. (Except that if you later EDIT a comment line, use insert and delete to do so. Don't use H as this will remove the control codes that return the printer to normal)

This utility increases the size of the Basic program by 6 bytes for each comment. There is no check that there is sufficient memory available to hold the larger program. On the basis that if you are that short of memory, there will be no comments in the program anyway.

CHANGING OPTIONS
****

The printer control codes to be inserted into the Basic program are stored at locations 32000 to 32005, when the utility is loaded from address 32000. If you change the location in memory, they are at the same relative position with respect to the execute address.

The first three bytes are inserted into the Basic program immediately following a REM token. The last three are inserted immediately before the zero which marks the end of the Basic line. Whatever characters are POKE'd into these locations will be inserted into the program at these positions. If your printer does not provide variable character style, you can use them to put a couple of ** at the start and end of each comment. Don't ever put zeros in any of these bytes, it upsets Basic.

Control characters set here are 27 20 15 (decimal) to start condensed printing and underlining, and 14 27 19 to restore the printer to normal. Any other control codes can be POKE'd to replace these

The routine to convert upper case to lower case can be eliminated by POKE'ing 18 (decimal - no op) into locations 32103 and 32104

RESTORING THE PROGRAM TO NORMAL
****

If you retain the program in its ammended form, there may be circumstances when you want to return it to normal. For example, to submit it to these magazines, or to change the print control characters to suit a different printer. It is also useful if you want to add more comments to an existing modified program. Remove the adjustments, add the comments, and then re-adjust again. If you want to add new comments without restoring it first, leave 3 blanks at the start and end of them, so that it will work OK when you do restore to normal

The utility includes a feature to do this. It works the same as before, except that you EXEC 32010 instead of 32006. This will restore the program to normal. If you wish to remove the print control characters only, and leave the comments as lower case characters, POKE 18 (decimal) into locations 32134 and 32135.

If you execute the undo routine with a program that has not been modified, you will lose the first and last three characters of all comment lines. If the original comment is less than 6 bytes long, then the following program line will be corrupted. If this happens to be the last line of the program, you will also lose the end of program marker for Basic.

The utility uses memory locations 64 to 66 for tempory storage of values during its operation. This is part of the memory used by Basic for its operations with floating point numbers, and can be used as tempory storage for operations of this nature. For this utility, it saves a few bytes of ML code to use this memory instead of memory within the utility itself.

END

```
1 '**  HIGHLITE
       BY GEORGE MCLINTOCK
       8/04/87
2 GOTO 10
3 SAVE "HIGHLITE":STOP
4 ' THIS PROGRAM SETS UP A ML UT
ILITY TO EMPHASISE COMMENT STATE
MENTS IN A BASIC PROGRAM
5 ' FOR A 16K MACHINE CHANGE 320
00 TO 16000
6 ' PRINTER CONTROL CODES ARE AT
 M+163 TO M+168
10 CLEAR 200,32000
20 M=32000:M1=M
30 LN=150:FOR X=0 TO 168 STEP 25
:IF X<149 THEN N=25 ELSE N=18
40 GOSUB 100:NEXT X
50 CLS:PRINT "HIGHLITE MACHINE L
ANGUAGE NOW INMEMORY FROM";M1;"
TO";M1+167:PRINT
60 PRINT "IT CAN BE (C)SAVEM'ED
FROM THEREFOR LATER USE":PRINT
70 PRINT "OR USED NOW ON ANY BAS
IC PROGRAM"
75 PRINT:PRINT "EXEC ADDRESS";M1
+6:PRINT "RESTORE EXEC";M1+10
80 STOP
90 '
100 PRINT LN;:A=0:FOR Y=0 TO N-1
110 READ C$:B=VAL("&H"+C$):A=A+B
:POKE M,B:M=M+1
120 NEXT Y:READ C$:IF A<> VAL("&
H"+C$) THEN PRINT "ERROR IN LINE
 NO";LN:STOP
130 LN=LN+10:RETURN
140 '
150 DATA 1B,14,F,E,1B,13,F,40,20
,4,86,1,97,40,33,8C,EF,9E,19,30,
4,A6,80,26,1D,64D
160 DATA EC,84,26,F6,9E,19,1F,12
,30,4,A6,80,26,FC,AF,A4,EC,84,26
,F2,30,2,9F,1B,9F,B56
170 DATA 1D,9F,1F,39,81,82,27,4,
81,83,26,D7,E6,1E,C1,FF,27,D1,9F
,41,D,40,26,38,9E,A28
180 DATA 1B,1F,12,31,26,10,9F,1B
,A6,82,A7,A2,9C,41,26,F8,EC,40,E
D,81,A6,42,A7,80,A6,B28
```

PSKIP 2

A utility by GEORGE McLINTOCK to shift listings a few extra spaces to the Right

I have a paper filing problem, in that I have lots of program listings around, and I often have trouble finding the one I want when I want it. I've tried different methods of storing them, but none have been really satisfactory. Mainly I think, because I don't spend the time necessary to keep them organised. When I got the DMP-105, which uses normal A4 paper, I thought I would try the normal 2 ring binders for the listings and documents I do on that printer. So far it seems OK, but one of the problems with it is that when you punch the holes in the paper, you remove part of the listing. While this is no real problem, (you can still work out whats there), I decided to do something about it.

This programme is a variation of PSKIP (CoCo, July 86), which provided the ability to specify a left margin in addition to other features. Naturally enough, I call it PSKIP2. The utility still works as before, but the entry addresses have changed. I have added another parameter at M+6 (and a corresponding switch at M+9), to hold the width of the left margin, and this moves the rest of the program up in memory.

We now also have code to initialise and restore RAM hook routines to prevent these being executed out of sequence.. If you do happen to execute them out of sequence with the old one, you require a cold start to restore things to normal. The extra code prevents this by placing an RTS instruction as the first byte of the routine which should not be executed, and restoring it to normal when the other routine is executed.

The new entry addresses are To initialise the routine - EXEC M+10 To restore to normal - EXEC M+46

The extra parameter works as expected. POKE the character width for the left margin into M+6, and the utility will insert this many blanks at the start of each line of print. To disable the feature, POKE zero into M+6

The only other requirement for this feature to work is that memory location Hex 9B MUST be set for the correct width of the paper. This location is set by the ROM to be 132, and for the left margin option to work correctly with an 80 character printer, this location must be set to 80 (Or 78 if you would like a small right margin as well) If you are also using the option to limit the line length (parameter M+3 non-zero) then location Hex 9B will be set to the correct value anyway. However, it is NOT necessary to activate the limit line length feature to obtain a left margin

The only other thing to note is that to get a left margin for the first line printed after activating the utility, you have to send a CR to the printer before starting to print. ie the left margin will only start after the first CR is received. Again this only applies to immediately after you have activated the utility.

Another change I have made is to not count characters sent to the printer which are less than 32 (decimal). This is done by adjusting the ROM character counter at Hex 9C. This feature can be disabled by POKE'ing 18 (decimal) into locations m+152 and M+153 if it causes other problems.

However, it does mean that if you are mixing character widths in a print line, then the routine may not perform as expected. If the left margin is set to zero, and M+3=0 (ie the routine is not actively limiting line length), then the printer is allowed to perform its normal automatic line feed according to its own criteria for when it has reached the end of line. However, if either of these locations (M+3 or M+6) is non-zero, then the utility will send its own CR when its internal use of the counter at Hex 9C indicates that an end of line is reached. This is required to ensure that the overall layout is maintained. In effect, this means that if either M+3 or M+6 is non-zero then the actual width of

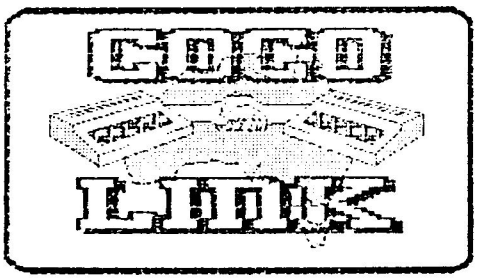
each line printed will be based on the 'normal' character width used to set the value in Hex 9B. If you then print some characters in a narrower width then the line will not reach the normal right margin. While characters of wider width will 'wrap around' the normal right and left margin. But if both these locations are zero, then the width of each line printed will be normal for that printer.

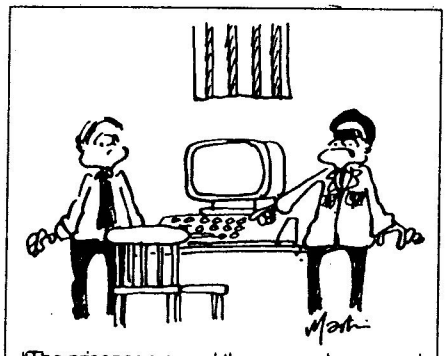
The character value for the software skip to top of page (stored at M+5) is no longer sent to the printer. I had second thoughts about this particular aspect, and now feel that it is more appropriate to do it this way Hence if your printer recognises a control code for skip to top of page, (say 12) then you can POKE 12 into M+4 to use it normally, or POKE 12 into M+5 to use the software function instead. This will now work OK because the value in M+5 is not in fact sent to the printer. However the value in the other location (M+4 or M+5) must still be one that is not recognised by your printer as a control character.

A problem with the routine is that if you are using 'Highlite' to underline comments in a Basic program, and the comment extends past the end of line, then the blanks that are inserted for the left margin will be underlined. At this stage, I don't see any easy way of avoiding this. While it could be done by maintaining a switch for underline on or off, and generating extra printer control codes when required, at this stage I don't think the problem is serious enough to warrant the extra coding complexity required to eliminate it

END

```
1 '**  PSKIP2
       BY GEORGE MCLINTOCK
       19/4/87
2 GOTO 10
3 SAVE "PSKIP2":STOP
4 ' THIS PROGRAM SETS UP A ML UT
ILITY TO CONTROL NORMAL PRINTING
. INCLUDING LEFT MARGIN
5 ' FOR A 16K MACHINE CHANGE 320
00 TO 16000
6 ' EXEC ADDRESS TO INITIALISE I
S M+10 TO RESTORE M+46
10 CLEAR 200,32000
20 M=32000:M1=M
30 LN=150:FOR X=0 TO 239 STEP 25
:IF X<224 THEN N=25 ELSE N=14
40 GOSUB 100:NEXT X
50 CLS:PRINT "PSKIP2 MACHINE LAN
GUAGE NOW IN  MEMORY FROM";M1;"
TO";M1+239:PRINT
60 PRINT "IT CAN BE (C)SAVEM'ED
FROM THEREFOR LATER USE":PRINT
70 PRINT "OR USED NOW BY EXEC";M
1+10
80 STOP
90 '
100 PRINT LN;:A=0:FOR Y=0 TO N-1
110 READ C$:B=VAL("&H"+C$):A=A+B
:POKE M,B:M=M+1
120 NEXT Y:READ C$:IF A<> VAL("&
H"+C$) THEN PRINT "ERROR IN LINE
 NO";LN:STOP
130 LN=LN+10:RETURN
140 '
150 DATA 40,2,0,0,B,C,0,0,0,0,8E
,1,67,33,8C,40,A6,84,A7,C4,EC,1,
ED,41,FC,7FA
160 DATA A0,2,C3,0,3,ED,44,86,7E
,A7,84,33,8C,1F,EF,1,CC,39,8E,20
,11,39,1,67,33,92E
170 DATA 8C,1C,A6,C0,A7,80,EC,C4
,ED,84,CC,8E,39,A7,8C,C8,E7,8C,E
9,39,34,45,D6,6F,C1,EF8
180 DATA FE,27,42,35,45,FF,FF,FF
,BD,0,0,5A,26,FA,39,6F,47,6F,42,
86,20,E6,46,27,2,AB0
190 DATA 8D,ED,A6,48,6F,48,6F,49
,39,A7,48,86,D,E6,40,E0,42,27,2,
8D,DA,E6,41,27,2,ABA
200 DATA 8D,D4,8D,D9,35,45,32,62
,39,6F,42,6F,47,6F,48,6C,49,20,B
E,33,8D,FF,6C,81,20,A8B
210 DATA 24,2,A,9C,A1,44,27,E8,A
1,45,27,CB,6D,48,27,D,81,D,27,9,
A7,48,CC,D,1,808
220 DATA 8D,A2,8D,AB,6D,47,27,C,
E6,41,27,8,A7,48,86,D,8D,92,8D,9
7,6D,49,27,4,A7,9F1
230 DATA 48,8D,93,81,D,27,12,D6,
9B,D0,9C,C1,1,2E,B7,6D,43,26,4,6
D,46,27,2,6C,48,91D
240 DATA 6C,42,6C,49,E6,40,E1,42
,22,A3,6C,47,20,9F,5E3
```





"The prisoner pressed the escape key — and that's the last we've seen of him, Warden!"

Utilities

A Utility Programme for keeping and indexing all those handy utility routines together for easy use.

By George McIntock

USING MACHINE LANGUAGE UTILITIES WITH A DISK SYSTEM

Over the years there have been a number of useful machine language (ML) utilities published in magazines, and you may have developed some yourself, or modified others. One of the problems with using them is that after you have collected a number of them, you require some way to have them conveniently available for when you might want to use them. For example, if you are in the middle of de-bugging a section of a Basic program and want to use BSEARCH (Hot CoCo, Sept 83) to find something, it's anoying to then have to go looking for a disk or tape with BSEARCH on it before you can use it. Even if it is on the same disk, if you have an auto line number routine active in memory, then you still have to switch utilities to be able to use it, and then switch back again when you have finished with it. You have similar problems when you think you are nearly finished and want to get a special listing or remove blanks or something. You have to load and activate another utility, when you might still be using a different one.

I started using the direct access buffer file space for certain utilities. This procedure is fairly common with the Model 1 and compatable machines, however, the basic procedure can be used quite effectively on the CoCo.

The CoCo provides a fixed area of memory for direct access files, starting from location Hex 989. The default file size is 256 bytes, which is the same as a sector on disk. The basis of the procedure is that the Basic command GET #1,1 will transfer the binary image of the first sector of the disk file into the direct access buffer. Likewise, a PUT #1,1 will transfer the binary image in the buffer to the first sector of the file on disk. If a ML routine is poked into the direct access buffer before the PUT, then any subsequent GET for that record will restore that ML routine to a known area of memory for use.

If you depart from the default parameters for the first direct access file, you may have to use the Field and Varptr commands to find the start of the ML code. This procedure will only always work with a direct access file. While you can do the same thing with serial files for some ML routines, it won't work with any routine that contains a bit pattern in a byte which Basic treats as a delimeter, eg CR, comma, quote etc. Hence the basic principle for the procedure is to create a single direct access file that contains the various utilities you might want, and have a copy of this file on any disk that you use while developing a new program.

The effect is much the same as having a separate file for each one on disk that you can LOADM, but this takes up more space, and puts a lot of entries in the directory before you start your own programs. It also makes it less convenient to start up a new disk. With LOADM files you need a clean master and then backup to a new disk. With a single file you can copy it across to the new disk from any other disk that has it. You are therefore more likely to put them all on a new disk before you start. With the default record length, a direct access file will provide 9 records per granule and 18 records per track. A single loadm file uses a full granule on disk

FILE ORGANISATION

One of the problems with this approach is to remember which utility is in which record. You can keep a record on paper, but you have to find the right piece of paper when you need it. Over a period of time you will probably build different versions of a file called 'UTILITYS', and will need to know what is on each one. I've built a few

different versions, but keep the same file name as it is the name I remember.

For my own use, I've adopted a file size of two granules. This provides 18 records. I use records 1 & 2 as index/reminder records, and the other 16 for utilities. I initially selected this size because of the 16 lines of display for the CoCo, but have found it quite adequate. The only time I've exceeded 16 utilities was when I wanted a number of small, special purpose things there. So rather then extend the main file, I put them in other utility files called UTIL1 and UTIL2, which I only copy if I particularly want them.

SUPPORTING PROGRAMS

The main requirement for a supporting program is to build and maintain the index/reminder records. I've organised these so that I can enter OPEN "D",#1,"UTILITYS":GET #1,1 :EXEC &H989 and get an index of what's available, together with the EXEC addresses for each one. If I GET and EXEC record 2, I can display extra reminder information where this might be necessary. With my memory resident single key entry routine, I have control Z set to execute this commend automatically.

The Basic program submitted is the one used for setting up a new utility file and for adding or changing utilities within an existing file. It is the only supporting program required. New utilities are added to the file by POKE'ing the ML code into the buffer area, and PUT'ing the record to the open file. Opening the file does not alter whats in the buffer. Utilities that I have submitted, like REMOVE, can be put there by changing the M to &H989, and running the Basic program with the DATA statements in it. I guess that this is one of the reasons I've adopted this procedure for my own utilities. It sets them up for convenient addition to my utility file if I want to put them there.

INDEX/REMINDER RECORDS.

These records contain a small ML routine in the first 20 bytes which displays the contents of the remaining 236 bytes of the record on the screen. The display on the screen stops when it reaches a zero bute. This routine transfers the bytes directly through the CHROUT routine, and will send them to the printer if Hex 6F is set to -2 before executing it. You have to do this with a single Basic line to prevent it being reset by Basic. eg POKE &H6F,254:EXEC &H989.

The ML routine does no formatting, and the messages to be displayed should contain it's own carriage returns as required. For my own index, I use the record number, utility name, and entry points as off sets from the start, with a space between each entity and followed by a CR. eg

```
3 PSKIP 8 42 CR
4 REMOVE 0 CR
5 HIGHLITE 6 10 CR
```

If the full index looks likely to exceed the space available, I shorten the name of the utility to make them all fit.

OTHER RECORDS

The other records in the file basically contain the ML code for the utilities only. However, very few utilities actually occupy 256 bytes, and I include as standard another small ML routine in the last 15 bytes of each of these records. This routine simply moves the contents of the direct access buffer into the cassette buffer (at Hex 1DA)

This allows me to have two utilities from the file in memory at the same time if I want to. By GET'ing the first, and EXEC &HA79, will move it to the cassette buffer, and allows another one to be loaded into the direct access buffer. The code to display the index record is also relocatable and can be executed from the cassette buffer as well. The move routine is also included as the last 15 bytes of these records as well, but if you want to use it that way you have to ensure that these bytes are not overwritten by the message to be stored there.

The program submitted tells you if this will happen. It can be useful at times to keep the index in the cassette buffer where it can be executed even if you have another utility in the direct access buffer. eg if you need to know the restore to normal execute address for a utility in control. The program used to build the utility file puts this ML code at the end of all of the records when the file is created. To keep it there you have to GET the record before POKE'ing the utility into the buffer and PUT'ing the record again.

It would be possible to combine a number of small separate utilities into a single record of 256 bytes, but I've not felt the need to do this. I prefer to create a separate file if required. It is also possible to spread larger utilities (greater than 256 bytes) accross both the direct access and cassette buffers. The only time I've actually used this approach is when I converted BSEARCH (A utility from Hot CoCo) to work with the utility file approach.

SUITABLE UTILITIES

Most utilities which are less than 256 bytes long are suitable for use in this way. Its easier to put them there if they are relocatable, but if they are not you can either re-assemble them or convert them so that they are relocatable. The ones you have to be more careful with are those which alter RAM hooks to make them work. You can still use these, but they should include a routine which restores the RAM hook to normal again. And you must remember to EXEC this 'undo' routine before changing the utility.

I find that the 'restore to normal' routine is not common with utilities in magazines. I include them with my own

utilities, but then I have a requirement for it. The more common approach is to alter the RAM hook and then put a RTS instruction at the start of the initialise routine. This is a safer approach and could still be used with a restore to normal routine, I haven't bothered to do so in the past, perhaps I should. I have since included this feature with PSKIP2 and it added 15 bytes to the code. I guess this is not an unreasonable price for the extra security. If you want to use one of these routines in the direct access buffer, its desireable to add your own restore to normal routine to it. The alternative is to remember the POKE's required to restore the RAM hook to normal for your machine.

PROGRAM OPERATION

The program submitted, called INDEXREM, is a Basic program which allows you to set up a new utility file along the lines that I have outlined. It also allows you to update and modify the index and reminder records for an existing utility file. The ML code at the end of the index record is automatically restored after each update, so that if you go over it once, but then shorten the message later, this ML code is restored. This method of operation won't suit everyone, but I find it convenient

END

```
1 '** INDEXREM
      BY GEORGE MCLINTOCK
2 GOTO 10
3 SAVE "INDEXREM":STOP
4 ' PROGRAM TO CREATE UTILITYS F
ILE AND
5 ' MAINTAIN THE INDEX REM RECOR
DS
10 CLEAR 1000:DIM A$(16)
20 V1=&H989+20:V2=&HA78:V3=&HA88
:B=236
30 CLS:PRINT "PROGRAM TO CREATE
OR CHANGE A   UTILITY FILE":PRIN
T:PRINT"BY GEORGE MCLINTOCK"
40 PRINT:PRINT "ENTER  N  TO CRE
ATE A NEW FILE  OR PRESS ENTER T
O CHANGE AN":INPUT "EXISTING ONE
";A$
50 PRINT:INPUT "ENTER NAME OF FI
LE";N$
60 OPEN "D",#1,N$:IF A$="N" THEN
 GOSUB 630
70 'SELECT RECORD
80 PRINT:PRINT "ENTER RECORD NUM
BER OF INDEX TO ALTER":INPUT "US
E 0 TO FINISH";N:  IF N <= 0 THEN
 CLOSE:  STOP
90 IF N>2 THEN PRINT:INPUT "DO Y
OU MEAN IT";A$:IF LEFT$(A$,1) <>
 "Y" THEN 80
100 GET #1,N
110 'EXTRACT MESSAGE
120 PRINT:PRINT "EXTRACTING MESS
AGE FROM RECORD  NO";N
130 Y=V1:C=V3:FOR X=1 TO 16:A$(X
)=""
140 A=PEEK(Y):IF A=0 THEN 180
150 Y=Y+1:IF Y > C THEN 180
160 IF A=13 THEN 180
170 A$(X)=A$(X) + CHR$(A):GOTO 1
40
180 NEXT X
190 'EDIT LOOP
200 GOSUB 520:GOSUB 570
210 PRINT:PRINT "EDIT INDEX RECO
RD NO";N:PRINT "A TO DISPLAY ALL
":PRINT"S TO SAVE AND TO MENU":P
RINT"Q TO MENU AND NOT SAVE IT":
PRINT"ENTER TO STEP THROUGH MESG
"
220 INPUT A$:IF A$ <> "A" THEN 2
90
230 'DISPLAY FULL SCREEN
240 CLS:FOR X=1 TO 15
250 IF LEN(A$(X)) > 0 THEN PRINT
 A$(X)
260 NEXT X:IF LEN(A$(16)) > 0 TH
EN PRINT A$(16);
270 IF INKEY$="" THEN 270 ELSE 2
00
280 'EXIT THIS RECORD
290 IF A$="Q" THEN 80 ELSE IF A$
="S" THEN 390
300 'STEP THROUGH LINES
310 FOR X=1 TO 16:PRINT:PRINT "L
INE NO";X
320 IF LEN(A$(X))=0 THEN PRINT "
NOT USED" ELSE PRINT A$(X)
330 PRINT:PRINT "ENTER  R  TO RE
PLACE IT":PRINT"        Q  TO RET
URN TO MENU":PRINT "OR PRESS ENT
ER FOR NEXT"
340 INPUT A$:IF A$="Q" THEN X=17
:GOTO 370
350 IF A$="R" THEN PRINT "OLD LI
NE IS":PRINT A$(X):PRINT "ENTER
NEW LINE":LINE INPUT A$(X)
360 IF LEN(A$)>1 THEN PRINT:PRIN
T"DID YOU STUFF IT AGAIN":PRINT"
DID YOU REALLY WANT TO REPLACE":
PRINTA$(X):PRINT"WITH":PRINT A$:
INPUT B$:IF LEFT$(B$,1)="Y" THEN
 A$(X)=A$:PRINT "THOUGHT SO - IS
 DONE"
370 NEXT X:GOTO 200
380 'REPLACE MESSAGE IN RECORD
390 FOR X=V1 TO V2:POKE X,0:NEXT
 X
400 Y=V1:FOR X=1 TO 16:  IF LEN(A
$(X))=0 THEN 440
```

Continued Overleaf

```
410 FOR C=1 TO LEN(A$(X))
420 POKE Y,ASC(MID$(A$(X),C,1)):
Y=Y+1:IF Y>V3 THEN 490
430 NEXT C:IF X<16 THEN POKE Y,1
3:Y=Y+1:IF Y>V3 THEN 490
440 NEXT X:POKE Y,0:IF Y > V2 TH
EN A$="W" ELSE A$="K"
450 IF Y <= V2 THEN RESTORE:FOR
X=1 TO 21:READ C:NEXT X:GOSUB 73
0
460 PUT #1,N
470 IF A$="W" THEN PRINT:PRINT "
ML CODE TO MOVE THIS RECORD TO
THE CASSETTE BUFFER HAS BEEN
OVERWRITTEN":PRINT:INPUT "PRESS
ENTER TO CONTINUE";A$
480 GOTO 80
490 PRINT:PRINT "MESSAGE TO LARG
E FOR BUFFER":PRINT"YOU WILL HAV
E TO REDUCE IT":INPUT "PRESS ENT
ER TO RETURN TO MENU";A$
500 GOTO 200
510 'CALCULATE SIZE
520 T=0:C=0:FOR X=1 TO 16
530 IF LEN(A$(X))>0 THEN T=T+LEN
(A$(X))+1:C=C+1
540 NEXT X:IF LEN(A$(16))>0 THEN
 T=T-1
550 RETURN
560 'PRINT SIZE
570 PRINT:PRINT "TOTAL LENGTH ME
SG =";T
580 PRINT "FREE SPACE AVAIL =";S
-T
590 PRINT "NO OF CR'S =";C
600 IF S-T < 15 THEN PRINT:PRINT
 "WILL DESTROY ML MOVE"
610 RETURN
620 'SET UP NEW FILE
630 FOR X=V1 TO V2:POKE X,0:NEXT
 X
640 RESTORE:C=0:Y=&H989
650 FOR X=0 TO 19
660 READ T:POKE Y+X,T:C=C+T
670 NEXT X:READ X:IF X <> C THEN
 PRINT "ERROR IN FIRST DATA LINE
":STOP
680 GOSUB 730
690 FOR X=1 TO 18
700 PUT #1,X
710 NEXT X
720 RETURN
730 C=0:Y=&HA79
740 FOR X=0 TO 14
750 READ T:POKE Y+X,T:C=C+T
760 NEXT X:READ X:IF X <> C THEN
 PRINT "ERROR IN SECOND DATA LIN
E":STOP
770 RETURN
780 DATA 48,140,17,166,128,39,6,
173,159,160,2,32,246,173,159,160
,0,39,250,57,2154
790 DATA 142,9,137,206,1,218,95,
166,128,167,192,90,38,249,57,189
5
```

END

CHANGE

Continued from Page 17

```
="Comma" ELSE IF A$=CHR$(34) THE
N N$(TX)="Quote"
850 FOR X = 1 TO 40: FOR Y = 1 T
O CD: V(X,Y)=0: NEXT Y,X
860 T = INSTR(K$,CHR$(34)): IF T
 = 0 THEN A$ = "2800 K$ = " + CH
R$(34) + K$ + CHR$(34): GOTO 880
870 A$ = "2890 K$ = "+CHR$(34)+
LEFT$(K$,T-1) + CHR$(34) + "+CHR
$(34)+" + CHR$(34) + MID$(K$,T+1
) + CHR$(34)
880 PRINT #1,A$: NCD = 1: GOTO 2
20
2790 'Set up script characters
```

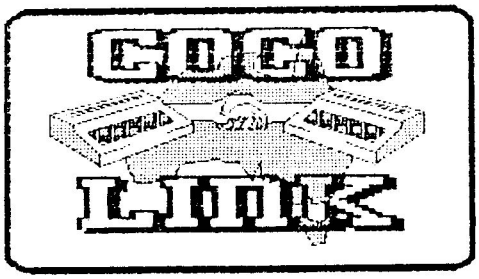
End

HIGHLITE

Continued from Page 19

```
190 DATA 3,27,6,8D,E,A7,80,20,F6
,A6,43,A7,80,EC,44,ED,81,20,9E,8
1,41,2D,6,81,5A,A44
200 DATA 2E,2,8B,20,39,A6,3,27,8
,8D,15,A7,80,20,F6,A6,3,A7,80,26
,FA,EC,3,26,F6,9C6
210 DATA ED,84,9E,41,16,FF,78,81
,61,2D,6,81,7A,2E,2,80,20,39,6F6
```

End



TOTAL INDEX

ARTICLE	TYPE	PAGE	VOL	NO
POWER PROBLEMS	INFO	22	2	6
PRINTER CODE CHART	INFO	17	3	6
QUESTIONS AND ANSWERS	INFO	16	2	1
READER'S SURVEY	REVIEW	10	3	3
RUNNING WRITING	APPLICATION	16	4	1
SCROLL FIX	UTILITY	11	2	6
SEE2	GRAPHICS	24	3	5
SHAPES	EDUCATION	7	4	1
SHORTS	MISC	28	2	6
SPINNIT	GAME	25	3	2
STRING VALUE SHEET	UTILITY	22	3	3
SURVEY RESULTS	INFO	23	3	6
TALKING BEAR	HARDWARE	7	3	3
TANDY REPLY	INFO	17	3	2
TAPE FIX FOR LLISTER	UTILITY	27	2	4
TAX DEPRECIATION	APPLICATION	20	3	3
THE BANK ROBBERY	FICTION	24	2	2
TOTE BET	APPLICATION	21	2	5
TRANSFER	UTILITY	20	3	2
TWO EDUCATIONAL PROGS	EDUCATIONAL	29	3	2
TWO OR THREE COLUMN LLISTer	UTILITY	17	2	2
TWO TEXT GAMES	GAMES	21	3	6
VIRUSES	INFO	13	3	3
WINNERS PART 1	APPLICATION	5	2	5
WINNERS PART 3	APPLICATION	6	2	6
WINNERS PART 3	APPLICATION	6	3	1
WINNERS PART 4	APPLICATION	6	3	2
WINNERS PART 5	APPLICATION	6	3	3
WONDERFUL WORLD OF CHAOS	GAME	29	3	2
21 CARD TRICK	GAME	29	2	6
65K PATTERNS	GRAPHICS	26	3	5

HOW TO SUBMIT MATERIAL TO COCO-LINK

PROGRAMMES: On tape or disk.
At least two copies should be on the tape/disk one of which should be saved in ASCII format.
Where possible include a description of your prodgramme saved as below for articles.

ML PROGRAMMES:
These require Source code saved on a suitable word processor. Two copies should be made.
A working copy of the programme should be included for checking by COCO-LINK.

ARTICLES:
At least one copy saved in ASCII format plus one copy on a commercial word processor where possible. (VIP Writer etc.)

HINTS AND TIPS:
Hand written or typed is acceptable.

LETTERS TO THE EDITOR:
Hand written letters will be accepted subject to the length. Long letters should be submitted on disk in the manner above for articles.

All disks and cassettes will be returned in due course.

Club Noticeboard

COCO SUPPORTING BBS's

Country Club	Ph.(085)-224-434
Decadence	Ph.(03)-794-7949
Happy Hacking	Ph.(03)-787-8759
Hard Rock Cafe	Ph.(03)-894-2815
J&M Systems	Ph.(03)-749-1935
Nemisis	Ph.(03)-331-1155
Real Connection #1	Ph.(03)-808-0910
Real Connection #2	Ph.(03)-808-0331
Tan-80	Ph.(08)-326 1132

WANTED URGENTLY

Programmes, Articles, Hints and tips for COCO-LINK Magazine.

NORTHSIDE USERS GROUP

This Queensland Group meet in the Brighton State School at 1.00PM on the 2nd sunday of each month.

For further details please contact:
Nev Winter 07 269 4373

NATIONAL OS9 USER GROUP

The fullest OS9 information service in Australia.

Monthly magazine included in annual subscription of $18.00

Write now to:

Gordon Bentzen
8 Odin St.
Sunnybank
Qld. 4109

WHyalla Coco Users Group

This group has just reformed and is looking for members in the iron triange of South Australia.

For further details please contact:
Fred Porter 086 45 0607

GENERAL NOTICE

This page is provided free for the use of clubs to let people know who, what, and where you are and to let them know what you are doing.
Please send your notices for the following two months no later than the 1st of the month previous to publication.

CLUB CONTACTS

Adelaide.....Laurie O'Shea
. 08 363 2647
. (after 7.30pm)
. Glenys Ferres
. 08 332 4264

AMUG.....Dick Burke
. 08 296 2995

Basic.....Johanna Vagg
. 068 522 943

Brighton.....N.Winter
. 07 269 4373

Brisbane North....M.Webster
. 07 285 6551

Brisbane S/W.....Bob Devries
. 07 372 7816

Geelong.....Alan Murrells
. 052 753 065

Moe User Group....Joseph Hester
. 051 277 817
. Ian Taffs
. 051 275 751

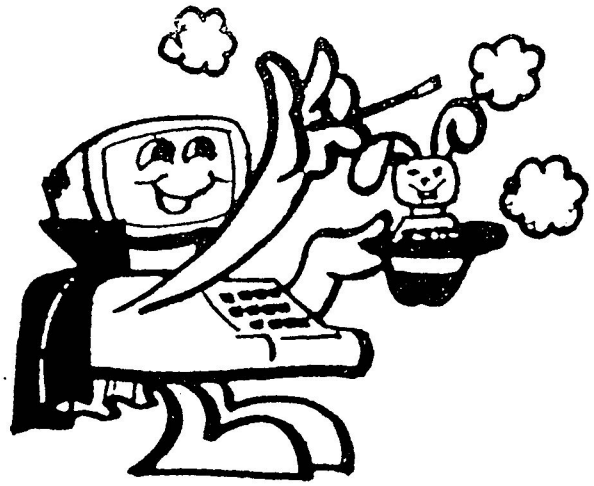
OS9 User Group....Gordon Bentzen
. 07 344 3881

Peninsula CCC.....Bob Charleston
. 059 791 922
. Sue Jones
. 059 773 292
. Gordon Chase
. 059 711 553

Whyalla.....Fred Porter
. 086 450 607

Clubs or persons wishing to be added to this list please inform the editor.

Presto Partner



PRESTO PARTNER is a RAM resident utility for OS9 Level 2. This means that it will not usually be on your screen but, using the windowing facilities of Level 2, it will always be ready and waiting for use at the press of a key.
When Presto Partner is executed it creates a new window for itself.
Presto Partner can be used for writing notes, doing calculations, scheduling/reminding of events and storing names, addresses and phone numbers. All these features are executed by pressing various <ALT> key sequences.

The programme comes with a 19 page manual which is well written and easy to understand.

SETTING UP the programme caused me a slight problem for starters. As is normal for such programmes it is recommended that you back up the master disk and then put it away in a safe place. You can then use your backup copy for all future operations. The backup routine is clearly described in the manual but did not take into consideration the fact that the 35 TSSD supplied would not backup onto my 40 TDSD
I decided to copy the programme on to the CMDS directory of my Source disk. This operation is also fully documented. On my first attempt I got a #214 Error indicating that the ATTRibutes would need to be altered to allow me to do this. This was done and then the disk copied OK. (Remember to remove the Write Protect tab to do this and then replace it on completion).
With the programme on my CMDS directory I was able to execute the programme simply by OS9: pp
Instructions are given on how to incorporate the automatic loading of the programme by adding it to your startup file. You can then have it there on screen instantly or just sitting there waiting in the background.

DSAVE can also be used to transfer Directories between different configured disks.

The NOTEPAD page is first to appear on screen. This has the time in the top right hand corner with a calendar of the month directly underneath. The day's date is highlighted on the calendar along with any other scheduled days. These functions appear in each of the options open in Presto Partner.

NOTEPAD: These notes are written in small outlined overlay windows. you can have up to seven of these pages. All can appear on screen at the same time by overlapping one another.
Again, the instuctions in the manual are easy to understand and I had no trouble in jotting down a few notes. The editing system is pretty agricultural but suffices for this purpose. The notes are saved to a file on disk with a simple <ALT><S> command.
If you have any difficulty remembering any of the command sequences used in this section there is a handy HELP feature which lists all the commands on a <ALT><?> command. (You do not need to use the shift key for this).

The Notepad incorporates a calculation feature which allows you to do quite intricate calculations. There is a fair list of operators for use with this feature. This in itself could be a handy feature when working on other programmes. ie calculating costs for a variety of items for inclusion in a letter being worked on in another window.

CALENDAR. The Calendar option is used for noting things which you wish to be reminded of on a certain date and time.

By using the arrow keys to move about the calendar illustrated on the right-hand side of the screen you can pinpoint the date you wish the reminder for.

Pressing "S" puts you in the scheduling mode. The schedule system asks the required questions and on

reaching the end asks "Is this correct Y/N"". This allows you to return to the beginning of this schedule if you have made a mistake.
One problem I stumbled on was, when I accidentally found myself in the scheduling mode, I couldn't get out of it again. I eventually had to input a dummy run (time only) and (enter) for all the other questions and then pressed "Y" for OK. I then went through the routine included to Unschedule it. Some escape mechanism should be built into the programme to avoid this sort of problem.
There are quite a few options incorporated in the Calendar schedule which allow you to be forewarned days before the event. You can have various types of warning. ie Visual, audible or both.
Events which occur at regular intervals, eg days,weeks,months etc can be scheduled to give you prior warning each time.

PHONE BOOK. This allows you to store names, addresses and phone numbers of the most used variety. The phone book allows access by name, address or phone number. You can also remove names etc. which are no longer required.

If you have an auto-dial modem you can have the Phone Book ring the number for you.

Each name and address etc. is added to a file on disk on entering it. This file is obviously a Direct access file. When accessing a name or leafing through the phone book page by page the programme has to access the disk for each successive name. This seems to be an excessive use of the drives for my liking and also makes for pretty slow action.

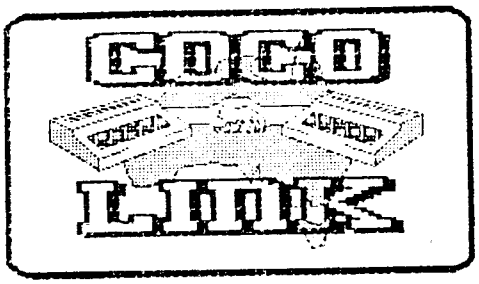
GENERAL COMMENTS
I use a green screen monitor 99% of the time and everything looks fine and clear on it. I am sure it will be the same for a colour monitor. I am not so sure how the 80 column screen used by the programme will turn out on the normal TV screen
The other main drawbacks with the programme are the exessive drive useage with the Phone Book and a doubt as to whether 7 pages in the Notebook would be enough in some situations.
I also feel a bit of upgrading on some of the editing and escape features would enhance the programme.

On the whole I found the programme easy to use and if used in a regular basis by someone who uses OS9 exclusively would be a handy utility to install in your system.
I can see some potential for it in a small business situation where any of its functions can be called up instantly from any other window functioning in OS9. Even handy to make notes or schedule dates while listening to instructions over the phone.

PRESTO PARTNER, while being a fairly basic example of this genre of utility programme, does do what it claims to do and could be a useful addition to any regular OS9 users library.

(PRESTO PARTNER is available from Microcom Software in the USA or through their official Australian agent, APD Springwood QLD).

Robbie Dalzell



Changes to Running Writing

By George McLintock

The following changes are required to the Script program as it appeared in CoCo Link Feb 1991. I apologise for these errors, which were in the program submitted to the magazine. As background, I have 4 versions of this program, for Tandy and Epson printers on the CoCo and IBM PC's. Thevarious versions are not all tested to the same extent as the original.

Line 90: Add LG=792 'Depth of page in dots
Line 460 'Not required for Tandy printers
Line 500 and 640: Change the LD in FOR .. NEXT loop to LG. eg FOR Y=DY TO LG STEP 7
Line 640: Add the STEP 7 to the FOR NEXT loop
Line 550: Change last IF test to IF FY >= FX THEN GOSUB 620 etc
Line 670 Change LPRINT to PRINT #-2,

If your printer recognises a control code to skip to top of page, replace the FOR .. NEXT loop in lines 500 and 640 with it. eg use PRINT #-2,CHR$(12)

The margin settings in the program effectively fill the page with script. To get larger margins, change the values in Line 90. For a 60 dot per inch printer, LG should be 660.

A Beginner's

Well, nothing much has been done over the holiday period. It just doesn't seem right to be sitting in front of a VDU when the weather is lovely and sunny outside. It is hard to get one's brain to think about things like OS9 and "C". Still, one must get back into it.

I received a couple of Public Domain OS9 disks from the OS9 User Group. The disks are full of files with names like "tom.h", "dick.c", "harry.doc" and "help.ar".
What do they do? What do they mean? How do you get them to run?
There's a wide variety of extensions used to indicate what language is being used or what type of file it indicates. It is all very confusing for a beginner like me. It is hard to find out what each file is supposed to do. Is it a command, a utility or just a plain old instruction?

Basic to any system must be the ability to load programmes, files or whatever with ease. One should be able to tell instantly what a particular file does or does not do. With OS9 this is not always the case.

Many extensions are standard and it is just a case of learning them as time goes on. When in doubt call on the help of someone who is a regular user of OS9. Some extensions are easy to deduce. Here is a sample of those:

.doc = Documentation file.
.c = File written in "C"
.ar = An Archival file.

A WHAT?? "What is an Archival file?", you ask. (As I did).
My understanding of an Archival file is that it goes back originally to the old C/MP operating system. Archived files are used under most operating systems nowadays.

What happens is that the file/files to be archived are fed through the archiving utility (called "ar"). What this does is to compress the files so that much more can be stored on a disk. When the files are to be decompressed (or unarchived) they are passed through "ar" again. the files are then available for use.

Most, if not all, the disks in the OS9 Users Library are archived files. This means that there is an awful lot on each disk.

If you ever order files from them, make sure you have the archive utility file "ar". If not be sure to ask for a copy to be included.

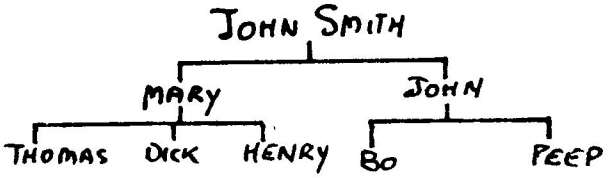
This little piece of information may save others from trying for hours to execute files with .ar at the end of them. It is very fustrating, believe me.

****

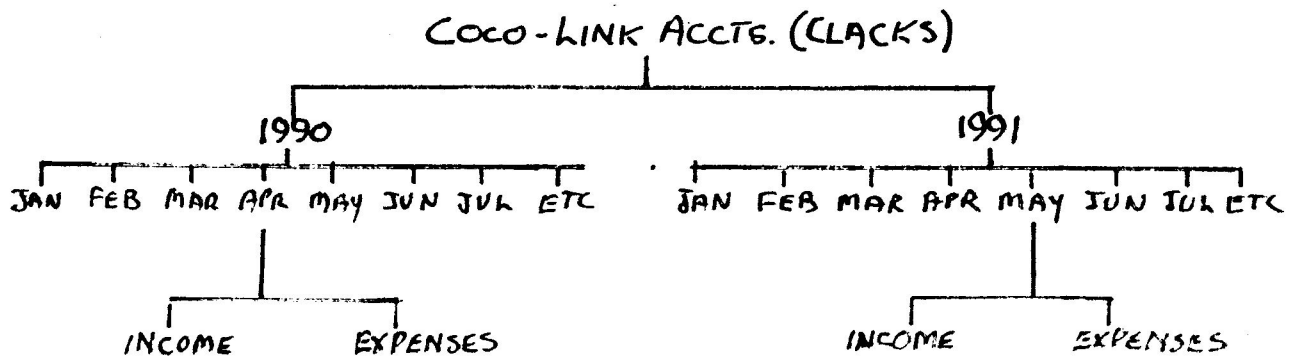
For the beginner into OS9 I think the most fustrating part is the continual errors regarding the pathlist. eg. "Pathlist not found" (my favourite).

To be able to avoid the continual errors of this type one must have at least a basic understanding of how OS9 directories work. The system of directories used in OS9 is called the Hierarchical directory system.

The easiest way to picture this is by imagining your family tree in the form used in historical novels and genealogical publications. Something like this:



If you transfer this to a Dynacalc file for instance and wish to use it for accounting for your business's journey into bankrupcy, your graph would look like this:



COCO-LINK ACCTS (or CLACKS for short) would be your main directory with all the following parts leading off in logical sequence. Some of these in turn would be directories. Always keep in mind that a directory in OS9 can direct you to other directories and so forth.

If, for instance, you wanted to look at your expenses for January 1990 to get a clear view of how bad things really are, you would first go to CLACKS then the YEAR, the month and then expenses. ie. CLACKS/1990/jan/expenses. This is called the pathlist and is merely the path you take through the maze to reach your final destination. Remember that you are the one who will build the maze. You can see how easy it is.....when you know how.

There are ways of getting to your final destination which are quicker than following the complete pathlist. These are described in the manual and various books. The main point of this simplified directory description is to try and impress that you have to know how your directories are formatted to be able to get to the information you seek. I reckon that this is one of the most important lessons for the beginner in OS9.

****

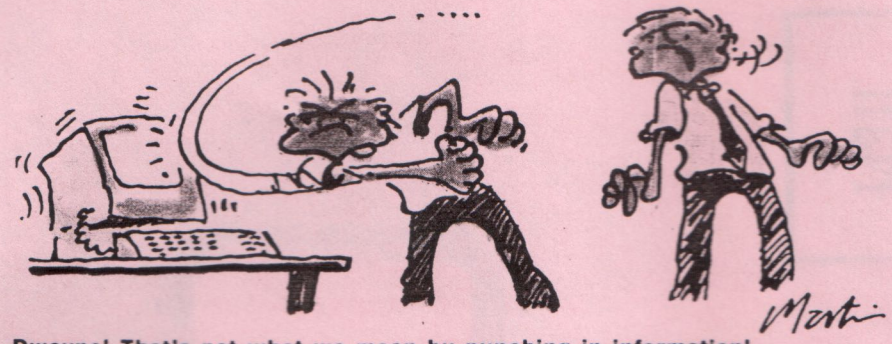
These ruminations of odd bits of information have helped me to get my mind back on track for the next step in this journey of discovery. This is to learn to programme in "C". With luck I should get into it real soon.
It's really more like a "Magical Mystery Tour".

END

Back Issues.....$2.50each

ADVERTISING RATES:

$12.00 per full page
$8.00 per half page
$5.00 per Quarter page



Dwayne! That's not what we mean by punching in information!

COCO-LINK PD SOFTWARE

DISK 001 EDUCATION

1) Australian Geography
2) Australian Explorers
3) Fractutor
4) Decimal
5) Spellit
6) Times Table

DISK 002 EDUCATION #2

BINARY	MATHSMT
COCOHOME	MEMORY
COINDEMO	NUMFUN
FORMULA	PUZZLE
MATCHEM	TRIGSHOW
MATH	WORD

DISK 011 GAME

CoCo Trivia
Trivial Pursuit game.
(Takes up 2 sides of disk)

DISK 012 GAME

Computer Tote
Complete with races and tote betting.
Marvelous for club fund raising!

DISK 013 13 GAMES

21 Card Trick	25 Square
Bobo	Build
Centrit	Cypher
Germ	Life
Max	Maze
Reversi	Tanks
Yancc	

DISK 014 11 GAMES

3Boxes	3Vagas
About	King Tut
Memory	Nausea
Patience	Pong
Puzzle	Slither
Wigworm	

DISK 021 UTILITIES

3CLMLIST	3HBUFF
3PRNTDOC	3QKMEN40
3QKMEN80	3VIPCOCO
CATLOGUE	DIRSORT
DSKDET	GOSUBBER
HASH	MENU
MULTUTIL	PRNTDOC
QKMEN32	

DISK 022 McLINTOCK UTILITIES

XCOM	ERASE
COMSBUF	DIVERT
MKI	TRANSFER
MGEFILES	PRINTDOC

DISK 031 HOME APPLICATIONS

Homehelp	Shoplist
Budget	Loan
Will	

DISK 041 COCO 3 GRAPHICS

DIR	ROCKFEST
AIRPORT	WATERFALL
BOUNCING BALL	WORLDMAP
NUDE	

All Disks

$5.00 each

Registered Publication No. SBH 1944

COCO-LINK MAGAZINE

31 NEDLAND CRES.,
PT. NOARLUNGA STH.,
S.A. 5167
(08) 386 1647

Surface Mail

POSTAGE PAID
CHRISTIES BEACH

DESMOND RAE
PO BOX 2076
MT. ISA
QLD 4825