For your TANDY Color Computer

Registered by Australia Post Reg. No. QBG 4009

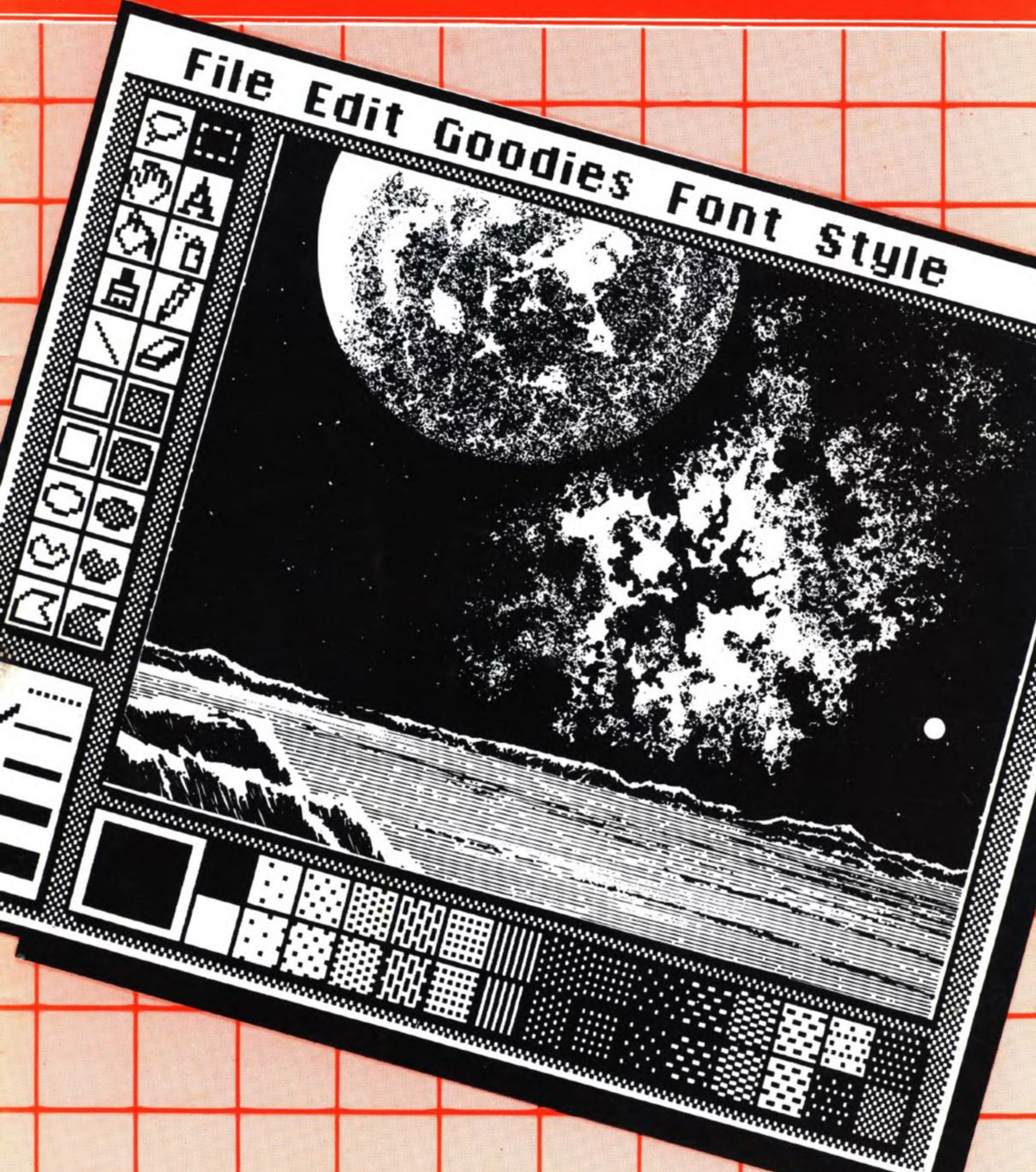
NEW GUINEA K5.95

$4.50

AUSTRALIAN RAINBOW

July, 1985

No.49



PARIS RADIO ELECTRONICS

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

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

COMPILERS

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

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

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

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

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

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

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

DATA BASE'S

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

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

OPERATING SYSTEMS

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

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

DISASSEMBLERS

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

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

WORD PROCESSING

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

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

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

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

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

SPREAD SHEETS

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

UTILITIES

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

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

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

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

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

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

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

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

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

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

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

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

MAGAZINES and BOOKS

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

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

THE OFFICIAL BASIC09 TOUR GUIDE $29.95

HARDWARE

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

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

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

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

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

DISK CONTROLLERS

J&M Disk Controller with JDOS Disk Basic	$219.95
RADIO SHACK Disk Controller Ver 1.1	$199.95

MISC

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

TM FLEX is a trademark of Technical Consultants
TM OS9 is a trademark of Microware
PRICES SUBJECT TO ALTERATION WITHOUT NOTICE

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

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

PRICES DO NOT INCLUDE POSTAGE AND PACKAGING

Rainbow Ware

Sweat shirts

any size

$23.95

with hood and arm length logo

$27.95

NAME ....

ADDRESS ....

ITEM ....

SIZE ....

COLOUR:
Pink Sky Blue Yellow Navy
Blue Red Grey Yellow

Amount Enclosed ....

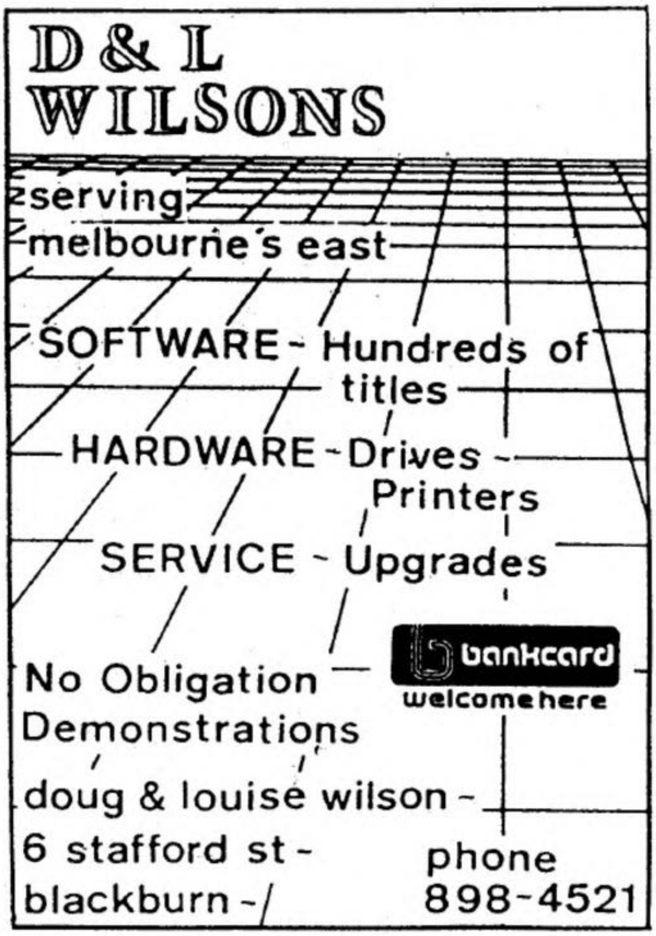
AUSTRALIAN

RAINBOW

index

lower case = article only
UPPER CASE = PROGRAM + ARTICLE

AUSTRALIAN RAINBOW Publisher and Editor Graham Morphett. Co- editor Kevin Mischewski. Assistant Editor Sonya Young. With grateful assistance from Brian Dougan, Richard and Judy, Bob Thomson, Paul Humphries, Alex Hartmann, Michael Horn, Jim and Sheryl Bentick, Annette Morphett. Cover Art Jim Bentick.
ADVERTISING DEADLINES: The 7th of the preceeding month of publication. All advertising is arranged through ToToAdvertising, PO Box 5730, Gold Coast Mail Centre, Qld, 4217.
OS 9: Kevin Holmes is the contact for os 9 information. He also has access to OS 9 Software from the US. His address is: 39 Pearson St., Narara, NSW, 2250.
All programmes in this issue of AUSTRALIAN RAINBOW are available on Rainbow on Tape.
The contents of this magazine is COPYRIGHT. Magazine owners may maintain a copy of each programme plus two back ups, but may not provide others with copies of this magazine.
Telephone: 075 51 0015 Voice; 075 32 6370 CoCoLink.
Printed by: Australian Rainbow Magazine PO Box 1742, Southport, Qld, 4215.
Founder GREG WILSON.



PRINT #-2,

A year ago we lost our friend, Greg Wilson.

Australian Rainbow is still very much his magazine - he set its attitude, its objectives, and its style. But he did much more than that, he gave CoCo a co-orinated start and in doing so, made a definite contribution to the longevity of our favourite computer.

It is with very great pleasure therefore, that we announce that the Delbourgo Family have been awarded the first Greg Wilson Award for Services to the Australian Computer Community.

The Delbourgos were writing programs of an advanced nature and sending them to America to get them published, when the rest of us were just thinking of buying our first computers - so to a certain extent, they got the drop on us! But I think it is the way they have gone about disseminating their knowledge which has earnt them the respect of the Computer Comunity. Much of the work they did some years ago (eg. Expanded Colour Basic - A rewrite of CoCo's ROMS, which provides amoungst other things, 64 colours, text in all modes and colours, extra graphics pages, and very much more), is still state of the art today!

When Australian CoCo magazine started, the strength of the Delbourgo's support was felt immediately and a constant supply of top quality programs has flowed ever since.

Honoured by others too, we want this award to say 'thank you' to you Bob, Tina and Daniel, for all the help you have given others, for showing us the capabilities of our various computers, and for the amazing programs you write. Even though Greg would certainly not approve of the award in the first place, I KNOW he would agree with the choice of recipient!

At CoCoConf, the Delbourgos received two Wall Plaques, one which will be passed to further recipients, and a smaller one to keep.

Speaking of CoCoConf, prize winners in the games contest, which were announced at CoCoConf, will be notified by mail, and their names will appear in Rainbow next issue.

Rainbow on tape has always been a problem. We still have not received the master for July 1984, despite a number of phone calls to America, and a number of promises to send! Those of you who have paid for this issue should therefore contact us, and we will replace that issue with another.

Not that you can have a recent copy either, because April onwards is also unavailable - undoubtedly held up in Australia Post's mincing machine, or on the water between Tokyo and Cairo! Sooner or later, Rainbow on Tape always arrives, but in the case of July 1984, we'll tell you when it does!

16K
ECB

NEAT LITTLE COLUMNS 2 × 2

by Stephen Lai

When I bought a printer, the first thing I did to test it was LLIST a program. The quality of the printing really looked fine, but the program listing was terrible! The lines were jagged at the right side of the sheet and the listing didn't stop for page breaks.

I wanted my listings to look as refined and neat as those that are printed in the RAINBOW magazine, so I decided to write *Two-Column Program LLISTer*. This program requires 16K Extended Color BASIC, a disk drive and a printer with at least 80 columns. I use it every time I make a hard copy of any BASIC program because it provides me with single sheet, easy-to-read two-column program LLISTings.

This program is fairly easy to run. First type in and save *Two-Column Program LLISTer*. Next, find the program that you wish to LLIST. If it's saved in ASCII, go on to the next step; if not, LOAD the program and SAVE it again, but this time in ASCII (e.g., SAVE"PROGRAM",A). After that is done, RUN *Two-Column Program LLISTer*.

Upon running the program, you will be greeted with a few questions about the program that is to be LLISTed. First, it asks for the disk-saved name. If you include an extension, use a slash instead of a period or else the program won't recognize it; if you don't include an extension, the program will assume the extension is /BAS.

Next, enter the full name of the program; the disk-saved name was limited to eight characters plus three more for extensions, but this name, which is printed as the title of the LLISTing, is only limited to the number of columns that your printer has.

Next you may choose to have single sheet pause. If you don't, asterisks will divide each group of lines of the LLISTing. After you have entered all of that information, the program will proceed to read the listing from the disk into the computer, byte by byte. When it is finished reading in one full page, you may start the LLISTing of the program.

I have included a message that is shown on the screen while the data is being read from the disk. It tells you how much longer you must wait to begin printing the next page of the LLISTing.

This program can be broken down into different sections that do different things. Lines 10-30 set up variables and display the title screen. If you have a printer with more than 80 columns, change CO in Line 11 to the number of columns on your printer for a centered LLISTing and the allowance of a longer title. You may change NL in Line 12 to however many lines you want printed on each page. Also, if you want a larger or smaller top margin, change TM in Line 13.

Lines 40-110 receive the input data from you concerning the LLISTing. Line 120 instructs you to insert the disk containing the ASCII-saved program and press ENTER. Lines 130-280 read the program listing from the disk. Lines 300-410 print the listing. Lines 450-460 ask whether you wish to use the program again.

Two-Column Program LLISTer shouldn't be too hard for you to understand. It certainly can be improved, and provisions for specific printers can be added to it to increase its flexibility and value. Someone with a printer that allows for form feeding may wish to make a subroutine using it in this program.

120 186
END 91

The Listing: *2-COLUMN*

```
10 CLEAR4000
11 CO=80
12 NL=50
13 TM=5
20 DIMPL$(100):AL=0:CN=1:IN=INT(
(CO-70)/2):BL$=STRING$(32,32)
30 CLS:PRINT@32,"  TWO COLUMN PR
OGRAM LLISTER":PRINT@104,"BY STE
PHEN LAI"
40 PRINT@166,"INPUT THE PROGRAM'
S          DISK-SAVED (IN ASCII)
NAME:"
50 LINEINPUTDN$:IFLEFT$(RIGHT$(D
N$,4),1)<>"/"THENDN$=DN$+"/BAS"
60 IFLEN(DN$)>12ORLEN(DN$)<5THEN
30
70 PRINT@289,"INPUT THE PROGRAM'
S FULL NAME:"
80 LINEINPUTNM$:LE=LEN(NM$):IF L
E>CO THEN70
90 PN=INT((CO-LE)/2)
100 PRINT@385,"DO YOU WANT SINGL
E-SHEET PAUSE              (Y/N)
?"
110 I$=INKEY$:IFI$="Y"ORI$="y"TH
ENSP=1ELSEIFI$<>"N"ANDI$<>"n"THE
N110ELSESP=0
120 CLS:LINEINPUT"INSERT DISK AN
D PRESS <ENTER>";A$
130 OPEN"D",#1,DN$,1
140 FIELD#1,1 AS RD$
150 LF=LOF(1)
200 CLS:PRINT"  LLISTING WILL BE
GIN/CONTINUE  BEFORE THIS NUMBER
 REACHES"NL*2+1"."
210 FORF=1TONL*2:PL$(F)=BL$:NEXT
F
220 FORF1=1TONL*2:PL$(F1)=STRING
$(32,32)
230 FORF2=1TO32:CN=CN+1:IFCN>LF
THENST=1:GOTO300
240 GET#1,CN
250 IFRD$=CHR$(13)THEN280
260 MID$(PL$(F1),F2,1)=RD$
270 NEXTF2
280 PRINT@109,F1:NEXTF1
300 IFSP=1THENGOSUB400ELSEIFAL=0
THENGOSUB400ELSEPRINT#-2:PRINT#-
2,TAB(IN);STRING$(70,"*"):PRINT#
-2
310 CLS:PRINT"LLISTING...":FORF=
1TONL:PRINT#-2,TAB(IN);PL$(F);"
      ";PL$(F+NL)
320 NEXT
330 IFST=1THEN450
340 GOTO200
400 AL=1:CLS:PRINT"POSITION TOP
OF PAPER TO PRINTERHEAD AND PRES
S <ENTER).";:LINEINPUTI$
410 PRINT#-2,STRING$(TM,13):PRIN
T#-2,TAB(PN);NM$:PRINT#-2:PRINT#
-2:RETURN
450 CLOSE:CLS:PRINT"THE TWO COLU
MN LLISTING IS DONE.PRESS 'Y' FO
R ANOTHER PROGRAM   LISTING OR '
N' TO STOP."
460 I$=INKEY$:IFI$="Y"ORI$="y"TH
ENRUNELSEIFI$="N"ORI$="n"THENEND
ELSE460
```

EDUCATION PAGE

My trip to Canberra, Canowindra, and Sydney involved me in discussions with a number of Educationalists, and provided an opportunity for some recent thoughts on the use of computers in education to mature.

The question most asked by prospective and new users of computers, is "What can I use it for?".

There is no doubt - you can get any amount of educational software for a large range of computers. The quantity available for the Apple IIe alone is staggering. Tandy computers also seem to have boundless quantities available.

But when you boil it down, a teacher, and even parents, must ultimately ask themselves about the aims they have for involving their kids with a computer.

Is it to entertain? Perhaps to motivate? To provide a form of rote learning? Or is it to provide job training or skills of reasoning?

Whatever the answer, it is abundantly clear that the real classroom, or home time, use of the computer is limited by computer availability and the other activities of the class or household.

This being the case, I contend that there is little need to have a great library of software.

If you have a good word processor like Telewriter 64, a good graphics processor like CoCo Max, perhaps a couple of 'motivational' programs like our own Speech Pack Speller, &/or Maths Invaders, &/or Kidwriter; a couple of 'drill' programs for maths and spelling; and, only if you can find one to match the educational needs of your kids, an adventure or two.

In practice, if you get to use half of these properly with the class through one year, I reckon you'll be doing fine!

Take CoCo Max as an example. I had the great pleasure of trying this program out on a special class at Clayfield during the trip. These kids had some small experience with computers, so it was easy to introduce the general start up routines. And they genuinely enjoyed the afternoon. They turned in some good work, and it provided them with a valuable experience.

This Clayfield lot look like trouble to me!

But the thing that stood out to me, was that here were about 20 kids, all motivated to learn, but we could not proceed at that time to use CoCo Max to teach them anything more than how to use CoCo Max.

It could be argued, and was, that this was enough, but my view is that while they were receptive, we should have been slipping in as much information as possible. Motivating kids is a means to an end - not an end in itself, and if the computer is not being used in the classroom to impart knowledge, then we are only achieving half the potential.

If we think about it for a while, there should be a term's computing in CoCo Max alone.

Telewriter 64 could be in daily use in a modern classroom, but what with other classes wanting the computers, and the fact that there are rarely sufficient computers available so that each child in a class can use one concurrently, it will be some time before word processors are in general use. The sooner the better, because until that time, the skills of using a word processor become a subject in themselves, and you can write off another term whilst you grapple with that one!

That leaves the third term (not necessarily meant to mean THIRD TERM), and this is a time to tie the other uses of a computer into the existing computer experence. In particular, if the class can be challenged to think, through the use of a sound adventure or conundrum, or if they can use the computer to prove a principle taught earlier, or if they can use the computer as a tool in an experiment, then again, that should account for most of that term too!

So you can see that before we even attempt to integrate the motivation and rote learning programs, we have really accounted for a full year's computing activities! To do more in my opinion, risks the quality of any response we may be seeking to achieve.

Some of the teachers at Canowindra with (left) Eric Hicks from Tandy (Orange).

Back to the trip - and a special hello to the Sisters and teachers at St Edward's Primary School, Canowindra, who invited us (meaning Tandy's super dooper Australian minus Queensland Education Consultant, Karel Davey, Tandy's Orange Manager, Eric (Ham) Hicks and myself) to demonstrate CoCos at their school. We had a most enjoyable time, not only going into the more usual programs that are shown in schools, but also upon their request, doing some introductory work in BASIC!

I am pleased to report the growing use of CoCos in schools around Australia. Despite not being on contract in most states, many schools are learning that CoCo is ideal, and so, in defiance of what the contract says, we hear of these schools going out on a limb to buy CoCos. This month alone, I heard of four major purchases of computers by schools.

Fortunately Canowindra, being a non state school, doesn't have the contract problem. CoCos will do the job there admirably!

TREASURE ISLAND

by Dean Hodgson

Computer games have been a social phenomenon since the early 1970's. They are played on nearly all computers, whether in sleazy game arcades or sophisticated corporations, and their popularity has achieved almost cult proportions.

Of the range of games an important tool within the classroom. Adventure games require students to use reading skills for a real purpose, and to develop skills of synonyms, or the use of a well-worn thesaurus, are needed to successfully play many of these games.

Beyond the direct skills and learning involved lie many spinoff benefits for learning. Unlike drill and practice packages, it is often necessary for two or more students to work as a team to successfully complete an adventure. Social and communication skills become important. Students are forced to discuss their ideas, explain their reasoning, test their logic. And from the game itself can come a wealth of other related classroom activities -- drawings of locations or events in a game, written accounts of adventurer's tales, social studies research to find information about pyramids, perhaps hustling to the library to find a copy of "The Hobbit", and even music in the guise of "The Hall of the Mountain King."

There are many adventure games available for the Colour Computer today. Most are text-only but some include spectacular graphics and animated displays. Unfortately, when the teacher of children aged between 7 and 10 goes looking for a suitable adventure games, they draw an almost empty hand. Most of the commercial and published games are written for adults, have adult concepts and language, and too frequently include undesirable elements. Typically it is assumed by most game authors that the adventurer is male and armed with an assortment of weapons. The content of the games, too, does not usually refelct the school's own curriculum.

The "Traesure Island" game was written for children between the ages of 7 and 12. It is a beginner's adventure game, not requiring the use of complex verb-noun inputs, and including many overt clues to guide child-adventurers. The theme of the game -- that of finding buried treasure on a deserted tropical island -- is well known to children. Mapping is straight forward, with few "twisted paths" and no mazes. The game includes a "Help" command which displays all the words the game understands. Movement from one location to another is by the usual method -- north, south, east,west, up and down. The game understands one-letter inputs for these commands.

CLASSROOM USE

The following steps have been used to successfully introduce "Treasure Island" as a first adventure game to a class of children.

The theme and object of the game are first introduced to the class as a whole. Children are asked what they might find on a tropical island and their ideas listed.

A large outline map of South Island is drawn and locations and movements explained. Special note is made that north and south are opposite directions, as are, east and west.

Next, students are told there are objects on the island that can be collected and may be needed for later use. There are three objects on South Island: a rope, a paddle and a raft, and three others on the North Island.

"Obstacles" can only be overcome if the player collects the needed object(s). For example, the stretch of water between the two islands is patrolled by the famous JAWS (drawing of shark fin) and cannot be crossed unless the player has found the raft.

Special words the game understands are then listed. These are:

LOOK - clears the screen and shows the location

INVENTORY - lists what you have collected

GET - allows player to pick up objects

TAKE - same as get

DROP - used to drop things picked up, if desired

CLIMB - there are a few places that can be climbed

DRINK - the player may find a source of water and care to refresh themselves

MOVES - tells how many moves have been made

HELP - lists the words the game knows

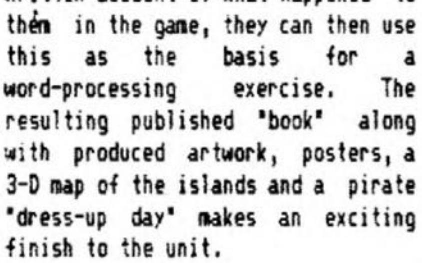
QUIT STOP - ends the game early

Adventurers, as they have nearly become, are then warned of three "death traps" hidden on the islands. Be careful of cliffs. Some can be climbed, others can't.

A map of South Island is then provided, which players can follow. However, North Island is, as yet, completely unmapped. They will have to draw their own maps.

Students then begin to play the game. It is often solved within one hour, in a networked computer lab. The challenge is then to work out the game in as few moves as possible.

If the children keep a short written account of what happened to thém in the game, they can then use this as the basis for a word-processing exercise. The resulting published "book" along with produced artwork, posters, a 3-D map of the islands and a pirate "dress-up day" makes an exciting finish to the unit.



SOME NOTES ON THE PROGRAM

"Treasure Island" was written following traditional BASIC adventure game formats. Location, movement and object DATA is read into a series of arrays. IF...THEN statements are used throughout to test for various conditions and events.

Some special subroutines were used to make the game easier to play.

Lines 5-7 display string A$ without splitting words at the right edge of the screen.

Lines 10-19 are a special INKEY$ routine that makes a blinking cursor.

Lines 20-29 are a special LINEINPUT routine that calls subroutine 10. The maximum length of the input string (A$) is 28 characters (1 line) and the clear comma keys are ignored, which is not the case with the normal INPUT statement. These routines are used only because of this defect with Microsoft BASIC.

A great deal more could, obviously, be done with the game. Use of graphics, especially hi-res, and lower-case display would improve presentation enormously. (Adding these features is left as an exercise for the interested programmer!)

(The author of this article would be interested in getting in touch with other teachers/programmers who have written adventure games especially for children.)

(NOTE: Parts of this article are derived from the Introduction section of "Pathweaver -- An Adventure Game Generator for the Commodore 64 manual, written by Wayne Starick, Dean Hodgson and Alan Goldsmith, programed by Dean Hogson, copyright 1985, South Australian Education Department, Angle Park Computing Centre.)

NOTE:

"Treasure Island" can be converted to an MC-10 with memory expansion if the following lines and changes are entered:

```
10 CX=PEEK(PX):B$=INKEY$:B$="":P
OKE16924,255
20 M=M+1*PRINT") ";:A$="":PX=PEE
K(16912)*256+PEEK(16913):P9=PX-1
6384
56 P9=303:PX=303+16384:PRINT@303
,;:GOSUB10:IF B$="Y" THEN3000
111 IF L=32 THEN PRINT@(PEEK(169
12)*256+PEEK(16913)-16384-32),"Y
OU CAN MOVE UP.":GOTO120
```

The Listing:

```
1 '******TREASURE ISLAND*********
  ******BY DEAN HODGSON*********
2 GOTO10
3 SAVE"TREAISLE":STOP
10 CLEAR1000:DIM R$(33),D(33,4),
OB$(6),OB(6),D$(6)
11 GOTO36
12 '**print a$ left justified**
13 I=31:IFLEN(A$)<I+1THENPRINTA$
:RETURN
14 IFMID$(A$,I,1)<>" "THENI=I-1:
GOTO14
15 PRINTLEFT$(A$,I-1):A$=MID$(A$
,I+1):GOTO13
16 '**inkey$ subroutine**
17 CX=PEEK(PX):B$=INKEY$:B$="":P
OKE282,255
18 I0=0:IFPEEK(PX)=CX THENPOKEPX
,32:GOTO20
19 IFPEEK(PX)<>CX THENPOKEPX,CX
20 I0=I0+1:B$=INKEY$:IFB$<>""THE
NPOKEPX,CX:RETURN
21 IFI0<10THEN20
22 GOTO18
23 '**input a$ subroutine**
24 M=M+1:PRINT") ";:A$="":PX=PEE
K(136)*256+PEEK(137):P9=PX-1024
25 GOSUB17:IFB$=CHR$(13)THENPRIN
T:RETURN
26 IFB$=CHR$(8)THEN30
27 IFLEN(A$)>28 THENSOUND1,2:GOT
025
28 IFB$<" "THEN25
29 A$=A$+B$:PRINT@P9,A$;:PX=PX+1
:GOTO25
30 IFLEN(A$)<1THEN25
31 A$=LEFT$(A$,LEN(A$)-1):PX=PX-
1:POKEPX,CX:GOTO25
32 '**return movement number**
33 D=0:FORI=1TO6:IFA$=MID$("NSEW
UD",I,1) OR A$=D$(I) THEND=I
34 NEXT:RETURN
35 '**game beginning**
36 CLS3:PRINT@40,"treasure islan
d";:POKE1072,32:PRINT@72,"by dea
n hodgson";:POKE1098,32:POKE1103
,32
37 GOSUB169
38 PRINT@227,"DO YOU WANT INSTRU
CTIONS?";
39 P9=303:PX=303+1024:PRINT@303,
;:GOSUB17:IFB$="Y"THEN155
40 IFB$<>"N"THEN39
41 L=1:CLS
42 '**display location**
43 PRINT"-----------------------
---------";:A$=R$(L):GOSUB13
44 IFL=21 ANDOB(3)>0 THENPRINT"T
REES OVERHANG THE CLIFF'S EDGE.Y
OU WILL NEED A ROPE TO CLIMB DOW
N THE CLIFF."
45 IFL=10 AND (OB(1)>0 OR OB(3)>
0) THENPRINT"YOU NEED AN OAR AND
 RAFT TO      CROSS TO THE OTHER
ISLAND."
46 IFL=26 AND OB(4)>0 THENA$="JU
```

```
NGLE VINES TO THE NORTH ARE   TO
O THICK TO MOVE THROUGH. YOU MUS
T FIND SOMETHING TO CUT THEM.":G
OSUB13
47 A$="":FORI=1TO6:IFOB(I)=L THE
NA$=A$+"THERE IS A "+OB$(I)+" HE
RE.":'**objects**
48 NEXT:IFA$<>""THENGOSUB13
49 '**movement**
50 IFL=33 THENPRINT"YOU CAN MOVE
 DOWN.":GOTO55
51 IFL=32 THENPRINT@(PEEK(136)*2
56+PEEK(137)-1024-32),"YOU CAN M
OVE UP.":GOTO55
52 PRINT"YOU CAN MOVE: ";:J=0
53 FORI=1TO4:IFD(L,I)>0 AND D(L,
I)<50 THENPRINTLEFT$(D$(I),1)","
;:J=1
54 NEXTI:PRINTCHR$(8)".":GOSUB15
1:IF L=16 THENPRINT"YOU CAN ALSO
 MOVE UP."
55 GOSUB24:GOSUB33:IFD>0 THEN75:
'**player input**
56 IFA$="L" OR A$="LOOK"THENCLS:
GOTO43
57 IFA$="I" OR A$="INVENTORY"THE
N104
58 IFA$="G" ORA$="GET" ORA$="T"
ORA$="TAKE"THEN110
59 IFA$="DROP" THEN116
60 IFA$="CLIMB" THEN124
61 IFA$="DRINK" THEN128
62 IFA$="DIG" THEN134
63 IFA$="QUIT" ORA$="STOP" THEN1
42
64 IFA$="H" OR A$="HELP" THEN144
65 IFA$="M" OR A$="MOVES" THENPR
INTM;"MOVES.":GOTO55
66 PRINT"I'M SORRY. I DON'T KNOW
 THE      COMMAND '";A$;"'.":PRIN
T"TRY ANOTHER WORD.":GOTO55
67 '**result of traps**
68 PRINT"YOU ARE DEAD."
69 PRINT"DO YOU WANT TO PLAY AGA
IN?"
70 GOSUB24
71 IFLEFT$(A$,1)="Y"THENRUN
72 IFLEFT$(A$,1)<>"N"THENPRINT"
PLEASE ANSWER yes OR no.":GOTO70
73 CLS:END
74 '**movement checking**
75 IFD<>6 THEN82:'**down**
76 IFL=33 THENL=16:GOTO43
77 IFL<>21 THEN82
78 IFOB(2)>0 AND OB$(2)<>"ROPE T
IED TO A TREE" THENPRINT"YOU NEE
D A ROPE TO CLIMB DOWN   THE CLI
FF.":GOTO55
79 GOTO81
80 PRINT"YOU TIE THE ROPE TO A T
REE AND  CLIMB DOWN."
81 OB(2)=21:OB$(2)="ROPE TIED TO
 A TREE":L=32:GOTO43
82 IFA$<>"U" OR (L<>32 AND L<>16
) THEN87:'**up**
83 IFL=32 THEN86
84 IFOB(5)>0 THENPRINT"THE TREE
HOUSE DOOR IS LOCKED.  YOU NEED
TO FIND THE KEY.":GOTO43
85 PRINT"YOU CLIMB THE TREE AND
UNLOCK   THE TREE HOUSE DOOR.":L
=33:GOTO43
86 PRINT"YOU CLIMB UP THE ROPE."
:L=21:GOTO43
87 IF A$<>"S" AND A$<>"E" THEN89
:'**south or east**
88 IFL=3 THENPRINT"GAAAAA......"
:PRINT"YOU JUST FELL OFF THE CLI
FF AND BROKE EVERY BONE IN YOUR
BODY.":GOTO68
89 IF A$<>"N" THEN94:'**north**
90 IFL=13 THENPRINT"AWK! YOU JUS
T FELL INTO THE      QUICKSAND!":
PRINT"GLUB...BLUB...":GOTO68
91 IFL=29 THENPRINT"EEEEEK!!....
.":PRINT"YOU HAVE JUST FALLEN IN
TO THE   VOLCANO!":GOTO68
92 IFL=10 AND(OB(1)>0 OR OB(3)>0
) THENPRINT"YOU NEED TO FIND A R
AFT AND      PADDLE TO CROSS TO T
HE OTHER    ISLAND.":GOTO55
93 IFL=10 AND OB(1)=-1 AND OB(3)
=-1 THENPRINT"YOU INFLATE THE RA
FT AND PADDLE IN IT ACROSS TO TH
E OTHER       ISLAND.":OB(3)=12:
OB(1)=12:L=12:GOTO43
94 IFA$<>"S"THEN97
95 IFL<>12THEN99
96 IF
97 IF(OB(1)<>L AND OB(1)>0) OR (
OB(3)<>L AND OB(3)>0)THENPRINT"Y
OU NEED THE RAFT AND PADDLE TO G
O TO THE SOUTH ISLAND.":GOTO55
98 PRINT"YOU PADDLE THE RAFT BAC
 TO THE  SOUTH ISLAND.":OB(1)=-1
:OB(3)=-1:L=10:GOTO43
99 IFD>4 THEN101
100 IFD(L,D)>0THEN102
101 PRINT"YOU CANNOT MOVE ";D$(D
);" HERE.":GOTO55
102 L=D(L,D):PRINT:GOTO43
103 '**inventory**
104 A$="YOU ARE CARRYING: ":K=0
105 FORI=1TO6:IF OB(I)=-1 THENA$
=A$+OB$(I)+",":K=1
106 NEXTI
107 IFK=0 THENA$=A$+"NOTHING,"
108 A$=A$+CHR$(8)+".":GOSUB13:GO
TO55
109 '**get take**
110 K=0:I=1
111 IFOB(2)=L AND OB$(2)="ROPE T
IED TO A TREE" THENOB$(2)="COIL
OF ROPE"
112 IFOB(I)=L THENOB(I)=-1:PRINT
OB$(I);" TAKEN.":GOTO55
113 I=I+1:IF I<7 THEN112
114 PRINT"THERE IS NOTHING HERE
TO TAKE.":GOTO55
115 '**drop**
116 K=0
117 FORI=1TO6:IFOB(I)=-1 THENK=1
:PRINT" "I"- "OB$(I)
118 NEXTI:IFK<1THENPRINT"YOU HAV
E NOTHING TO DROP.":GOTO55
119 PRINT"DROP WHICH OBJECT (TYP
E NUMBER)"
120 GOSUB24:N=VAL(A$)
121 IFOB(N)=-1 THENPRINTOB$(N)"
DROPPED.":OB(N)=L:GOTO55
122 PRINT"SORRY. YOU AREN'T CARR
YING IT.":GOTO55
123 '**climb**
124 IF L=16 OR L=32 THENA$="U":G
OTO75
125 IF L=3 OR L=21 OR L=22 THENP
RINT"THE CLIFFS CANNOT BE CLIMBE
D.":GOTO55
126 PRINT"THERE IS NOTHING TO CL
IMB HERE.":GOTO55
127 '**drink**
128 IFL=5 THENPRINT"NO. THE WATE
R LOOKS FOUL.":GOTO55
129 IFL=6 THENPRINT"YUK! SALT WA
TER!":GOTO55
130 IFL=27 THENPRINT"NO. YOU CAN
'T DRINK SWAMP WATER!":GOTO55
131 IFL=31 THENPRINT"REFRESHING!
":GOTO55
132 PRINT"THERE IS NOTHING TO DR
INK HERE.":GOTO55
133 '**dig ...the secret word**
 RAFT,32,MACHEEY,17,SHOVEL,133 '
**dig ...the secret word**
134 IFOB(6)<>-1 THENPRINT"YOU NE
ED A SHOVEL TO DIG.":GOTO55
135 IFL<>14 THENPRINT"YOU FIND N
OTHING.":GOTO55
136 PRINT:PRINT"DIG...DIG...PANT
...HUFF..."
137 A$="YOU HAVE DUG UP A LARGE
BROWN CHEST. YOU OPEN IT AND FIN
D THAT IT IS FULL OF TREASURE!!!
!":GOSUB13
138 PRINT"WELL DONE!!  YOU DID I
T!!"
139 PRINT"YOU ARE NOW RICH AND F
AMOUS!"
140 PRINT"YOU TOOK"M"MOVES.":GOT
O69
```

```
141 '**quit stop**
142 GOTO140
143 '**help**
144 GOSUB145:GOTO55
145 PRINT"THIS GAME KNOWS THESE
WORDS:"
146 FORI=1TO6:PRINTD$(I)",";:NEX
T:PRINT
147 PRINT"LOOK,INVENTORY,GET,TAK
E,DROP,   CLIMB,DRINK,QUIT,STOP,
MOVES,HELP";
148 PRINT"AND ONE OTHER SECRET W
ORD."
149 RETURN
150 '**moving down at cliff**
151 IFL<>21 THENRETURN
152 IFOB(2)=-1 OR OB$(2)="ROPE T
IED TO A TREE" THENPRINT"YOU CAN
 MOVE DOWN."
153 RETURN
154 '**instructions**
155 CLS:PRINT@8,"TREASURE ISLAND
":PRINTTAB(8)"======== ======":P
RINT
156 PRINT"YOU ARE ON AN ISLAND L
OOKING FORBURIED TREASURE.":PRIN
T
157 PRINT"YOU CAN MOVE FROM PLAC
E TO PLACEBY TYPING IN A DIRECTI
ON NORTH, SOUTH,EAST,WEST,UP OR
DOWN.":PRINT
158 PRINT"TO MAKE IT EASIER YOU
CAN ALSO  TYPE JUST THE FIRST LE
TTER --      N,S,E,W,U OR D."
159 PRINT:PRINT"              PRESS
ENTER"
160 IFINKEY$<>CHR$(13)THEN160
161 CLS
162 PRINT:PRINT"THIS GAME ALSO K
NOWS THESE      SPECIAL WORDS:":
PRINT:GOSUB147
163 PRINT:PRINT"AND THESE SPECIA
L LETTERS:":PRINT
164 PRINT"L (LOOK), I (INVENTORY
), G (GET)M (MOVES), H (HELP)"
165 PRINT:PRINT"   PRESS ENTER T
O START GAME"
166 IFINKEY$<>CHR$(13)THEN166
167 GOTO41
168 '**initialize game**
169 D$(1)="NORTH":D$(2)="SOUTH":
D$(3)="EAST":D$(4)="WEST":D$(5)=
"UP":D$(6)="DOWN":RESTORE
170 FORI=1TO33:READR$(I)
171 FORJ=1TO4:READD(I,J):NEXTJ
172 NEXTI
173 FORI=1TO6:READ OB$(I),OB(I):
NEXTI
174 L=1
175 RETURN
176 '**location & movement data*
*
177 DATA YOU ARE STANDING ON A W
IDE SANDY BEACH.,2,0,3,18
178 DATA YOU ARE ON A GRASSY PAT
HWAY THAT CUTS BETWEEN TALL PALM
 TREES.,4,1,9,8
179 DATA YOU ARE STANDING ON THE
 EDGE OF HIGH CLIFFS OVERLOOKING
 THE SEA. TO THE SOUTH AND EAST
FAR BELOW WAVES CRASH AGAINST JA
GGED ROCKS.,9,99,99,1
180 DATA YOU ARE IN THE JUNGLE.
THERE IS A RUINED EMPTY NATIVE H
UT HERE.,5,2,0,7
181 DATA YOU ARE AT A SMALL POND
 IN THE MIDDLE OF THE JUNGLE.,10
,4,6,21
182 DATA YOU ARE AT A LOVELY SMA
LL LAGOON. THE WATER IS CALM AND
 YOU CAN HEAR THE ISLAND BIRDS I
N THE TREES.,23,31,0,5
183 DATA YOU ARE IN THE JUNGLE.
A LARGE WOODEN STATUE STANDS BEF
ORE YOU. A MESSAGE CARVED IN THE
 STATUE'S BASE READS 'I LEFT IT
ON THE OTHER ISLAND.  C.K.,21,8,
4,0
184 DATA YOU ARE IN THE JUNGLE.,
7,18,2,0
185 DATA YOU ARE IN THE JUNGLE.
THERE ARE LOTS OF BIRDS IN THE T
REES.,31,3,19,2
186 DATA YOU ARE STANDING ON A R
OCKY BEACH. ACROSS THE WATER YOU
 CAN SEE ANOTHER ISLAND. THERE I
S A SIGN ON THE OTHER ISLAND'S B
EACH. (IT'S TOO FAR AWAY TO READ
.) YOU CAN ALSO SEE SHARK FINS I
N THE WATER.,12,5,23,22
187 DATA YOU ARE IN THE JUNGLE.
IT IS VERY QUIET HERE. YOU NOTIC
E THE ROTTING REMAINS OF A PARAC
HUTE HANGING IN A TREE.,13,12,26
,16
188 DATA YOU ARE ON A BEACH ON T
HE NORTH ISLAND. A LARGER ISLAND
 IS TO THE SOUTH. THERE IS A SIG
N HERE WHICH READS 'CAPTAIN KIDD
 WAS HERE.' THERE IS THICK JUNGL
E TO THE NORTH AND SHARK INFESTE
D WATER SOUTH.,11,10,25,24
189 DATA YOU ARE AT A POOL OF MU
DDY QUICKSAND. THERE ARE NO HAND
Y VINES IN CASE YOU FALL IN. THE
RE IS AN OLD FLYING HELMET LYING
 ON THE OPPOSITE EDGE OF THE POO
L BUT IT'S OUT OF REACH.,99,11,1
1,27
190 DATA YOU ARE IN A CLEARING I
N THE JUNGLE. THERE IS A LARGE X
 ON THE GROUND.,28,0,17,0
191 DATA YOU ARE IN A FIELD FULL
 OF LITTLE YELLOW FLOWERS.,17,26
,26,26
192 DATA YOU ARE IN THICK JUNGLE
. THERE IS AN OLD TREE-HOUSE IN
ONE LARGE CLIMBABLE TREE.,27,24,
11,0
193 DATA YOU HAVE FOUND THE WREC
KAGE OF AN AIRPLANE. THERE IS A
SKELETON INSIDE AND EVERYTHING L
OOKS VERY RUSTED.,28,15,0.14
194 DATA YOU ARE ON A SMALL SAND
Y BEACH.,8,0,1,0
195 DATA YOU ARE AT A ROCKY BEAC
H.,20,0,0,9
196 DATA YOU ARE ON A BEACH WITH
 LOVELY WHITE SAND.,0,19,0,31
197 DATA YOU ARE STANDING ON THE
 EDGE OF A CLIFF.  BELOW IS AN E
NCLOSED BEACH.,0,7,5,0
198 DATA YOU ARE ON A ROCKY BEAC
H WITH HIGH CLIFFS TO THE SOUTH
AND WEST.,0,0,10,0
199 DATA YOU ARE ON A WHITE SAND
Y BEACH.,0,6,0,10
200 DATA YOU ARE AT A SMALL BAY.
,16,0,12,0
201 DATA YOU ARE AT A SMALL COVE
.,26,0,0,12
202 DATA YOU ARE IN VERY THICK J
UNGLE.,15,25,25,11
203 DATA YOU ARE AT THE EDGE OF
A LARGE SWAMP FULL OF CAT-TAILS
AND REEDS. THERE ARE MANY FROGS
CROAKING AND DRAGONFILES BUZZING
 ABOUT. YOU CAN SEE A VOLCANO TO
 THE NORTH.,30,16,13,0
204 DATA YOU ARE ON A LONG SANDY
 BEACH.,0,17,28,28
205 DATA YOU ARE AT THE TOP OF T
HE VOLCANO. STEAM IS RISING OUT
BUT THERE IS NO LAVA.,99,30,30,3
0
206 DATA YOU ARE ON A HILL LEADI
NG UP TO THE TOP OF A VOLCANO.,2
9,27,27,27
207 DATA YOU ARE IN A LITTLE VAL
LEY FULL OF PALM TREES. A STREAM
 RUNS TOWARD THE OCEAN.,6,9,20,0
208 DATA YOU ARE ON A SECRET BEA
CH BELOW SOME LOW CLIFFS.,0,0,0,
0
209 DATA YOU ARE INSIDE THE TREE
 HOUSE.,0,0,0,0
210 '**object data**
211 DATA WOODEN OAR,18,COIL OF R
OPE,19,INFLATABLE RAFT,32,MACHET
TE,29,RUSTY KEY,17,SHOVEL,33
```

New Trends In Educational Computing

By Michael Plog, Ph.D.

Back in April 1982, the Tandy Corporation began a program called "Tandy Educational Grants." The company provides sums of money to educational institutions for research and development of educational uses of computers. Since its beginning, the Tandy Educational Grants program has awarded over $885,000 worth of hardware and software.

The current "cycle" of awards was made for proposals based on "Using Microcomputers to Develop Thinking Skills." Tandy, of course, has several models of computers in its product line. Four awards were given during the current cycle; one involved the Color Computer.

This award went to Mrs. Margaret Perry of Safety Harbor Middle School, Safety Harbor, Fla. Her project is to establish a model program using computers to aid gifted students in improving their thinking and creative skills. Mrs. Perry (and the Safety Harbor school system) received 11 64K Color Computers with monitors and disk drives, a DMP-110 printer, color graphics printer, touch pad and several software packages. (Does that sound like a dream come true?)

At present, we do not know exactly how the hardware and software will be used, or what the curriculum will look like. In the future, we hope to be able to report on the results of this project. The materials and procedures developed in Safety Harbor might be worthwhile to adapt to your local school system.

Possibly, curriculum materials may be developed that you can use at home. Whatever the outcome of the Safety Harbor experience, you should be aware that the Tandy Corporation is taking education seriously, and even providing funds for innovative programs in schools.

If you are interested in preparing a proposal of your own, write to Tandy Educational Grants Program, Radio Shack Education Division, 1400 One Tandy Center, Forth Worth, TX 76102. The educational community needs to experiment with different uses of computers, and we need quality products and procedures to use in schools. Since schools are often short of money, outside sources of funds are important to continue development of curriculum to benefit all students in the country.

Even with the reduction of funds for education from the federal government, there are still some programs which help development of educational experiences. The National Diffusion Network is one such program. This program provides funds for innovative programs, then goes the next step. Funds are also provided to help school systems implement the projects that have been judged successful. Several Diffusion projects in past years have dealt with computers in the classroom.

One of the most recent such projects is the Asbury Park Computer Math Program. The goal of this project is to integrate computers into the entire curriculum of grades 9-12, with 18 hours of instruction in each of six subject areas: general mathematics, algebra I and II, geometry, trigonometry and calculus. The emphasis of this project is on mathematics, but other projects have stressed different aspects of the educational arena. You can find out what National Diffusion Network projects exist by contacting the administration of your local school district.

Another sign of federal involvement in computers for schools is from the National Institute of Education (NIE). This organization has set as one of its priorities for 1986 an investigation into the effective uses of education software and technology. We hope NIE officials are aware of projects similar to the one in Safety Harbor. The report from NIE should be completed in 1986, but interim reports may be released earlier.

One study NIE will probably examine has been conducted by the Office of Bilingual Education and Minority Languages Affairs (of the Department of Education). The Office recently released a report on the use of educational technologies in programs dealing with limited English-proficient students. The study was limited to students with a native language other than English.

Computer assistance has long been thought to be helpful for such students, because some students may be in school districts where no one else (teacher, aide, principal) speaks the same language as the student. Computer assisted instruction could help such students learn English, as well as basic skills in their native language. The study conducted by the Office has several findings. Many of the findings apply to all students, not just those with limited English proficiency.

As might be expected, funding for computer assisted instruction increased from 1982 to 1984, while funding for audio-visual technologies decreased. This is not to imply that schools dealing with limited English proficient (LEP) students are no longer interested in audio-visual technology. Many schools have already purchased this type of equipment, and have no need for more equipment. In a few years, we will probably see less money spent for hardware and more resources used for software.

The study also found that educational technologies can increase the effectiveness of instruction for LEP students. In addition, the study concluded that computer assisted instruction holds a greater educational potential than other technologies, such as audio-visual techniques.

The study also pointed out some concerns for users of computers in the classroom. One finding relates to staff dealing with computer assisted instruction. A lack of planning and staff training have compromised the effectiveness of many CAI programs. As with any educational program, poor staff preparation and poor planning will result in a "hit or miss" outcome.

Positive results are due more to chance than conscious effort. And, many educational computer programs depend on one key person; without that person (the study founder), the project would most likely fail. Again, as with any program, a single individual has difficulty institutionalizing a set of educational experiences.

Two other findings are important from this study, and should be recognized by anyone trying to implement computers in schools. The people initiating the computer assisted instruction program often had objectives that were not specific enough for success. We all know people who are so enamored with the equipment that they do not realize its use.

Finally, the study found what most educators have been saying: A lack of instructionally and technically sound software has reduced the effectiveness of CAI for limited English-proficient students. Naturally, the lack of good software is not limited to students with a native language other than English.

This study, while limited in scope and intent, is worthy of study by people interested in computer assisted instruction. While computer assisted instruction is only one component of computer use in schools, it is an important component.

The federal government may even take a more active role in computer education, if Representative Timothy Wirth, a Democrat from Colorado, gets his way. Congressman Wirth will introduce a computer literacy bill in the House of Representatives this year. The purpose of his bill is to help schools buy microcomputers, train teachers, establish a federal information bank and create a computer consulting service. The proposed legislation covers a broad area of assistance to educational computing.

Last year, Congressman Wirth introduced a similar bill, but it was not passed. He is trying again. Wirth is interested in equity of access. As the nation moves from an industrial to an information economy, Wirth claims, schools must ensure that all children — regardless of wealth — have access to computers.

The issue of equity of access of computers is a priority topic for many people. A coalition of Washington computer educators has established SLICE (Support for Leadership in Computer Education). This group is organizing in-service training for local computer instructors with emphasis on equity. This group is working without any government funds, but has a localized area of interest and effect.

Some efforts for computer literacy are state oriented. After this summer vacation, all schools in Texas will have to begin teaching seventh and eighth graders computer literacy according to standard, state-mandated curriculum. Other states are implementing computer literacy programs, but none that I know of has a state-mandated curriculum.

EDUCATION NOTES

4K

A Serendipitous Learning Experience

By Steve Blyn

Sometimes it is important to present students with an educational program that is mostly for fun. Entertainment remains one of the primary reasons many of us bought computers in the first place. This month's program attempts to combine learning with fun.

Although it is loosely intended as a language arts program, there is really no definite learning that is expected from this program. Many incidental learnings, however, may occur that we are not always aware of at the moment.

Incidental learning is learning that is not necessarily designed to happen, but rather occurs as a side effect of the experience. Typing in computer programs from magazines, for example, often produces the incidental learning of the keyboard. Another example might be shopping with your family in a department store. This may produce incidental learning about using money, travel training, reading signs and a host of others.

The game we are doing this month is a code breaker. The alphabet is written on the screen with a number next to each letter. Next to the letter 'A' is a '1,' next to 'B' is a '2,' and so on down to the letter 'Z' with a "26" next to it. This represents a simple code. Each letter may be associated with a different number. The numbers, of course, range from 1 to 26 to represent each of the letters.

A word should be entered by someone other than the player; this is a good two-player game. The computer will show the child the word in code and the child's job is to decode the secret word. For example, if someone types in the word COCO, the program will convert it into "3 - 15 - 3 - 15." The player must use the chart or his/her memory of the alphabetical order to decode the word back again to its original form.

This game may be played on two levels. You may either choose to have the code visible or invisible while you are decoding. If you choose to hide the code, you will have to review the alphabetical order mentally several times to figure out the word. This is much more difficult, of course, than leaving the code in view.

Younger players will most probably need the code visible at all times. Older players will no doubt hide the code each round. Middle-of-the-road learners will probably combine the two and benefit the most from this program; they can constantly be learning and reviewing the alphabetical order while playing the game.

Lines 400-430 draw the code. Line 450 will hide the code if that option is selected. Lines 120-140 present the option of hiding the code.

An easy possibility for altering this program is to present the letters and numbers in reverse order. The letter 'A' could be equivalent to 26, 'B' to 25, and so on to 'Z' equal to one. This would make the code slightly more difficult and the program more challenging. Two lines must be altered to accomplish this switch.

First, change Line 250 from

```
  PRINT ASC(L$)-64;
       to
  PRINT ASC(L$)-91;
```

Secondly, change the portion of Line 410 which reads

```
  A$(R)
    to
  A$(27-R)
```

These two changes will reverse the position of the numbers. You may get more daring and devise your own schemes to further mix up the numbers, if you desire.

The partner types in the letters of the mystery word on lines 160-210. The computer converts these letters into numbers on lines 220-270. The player then guesses the secret word. If incorrect, the right answer will be displayed by Line 320.

We meant no pressure to be on the student in this program. For this reason, we included no time limit or report card. The game can be ended after each round by pressing 'E' or continued with more examples by pressing 'M.' The game can be played as long as the interest remains. We hope your children have fun as well as incidentally learn at the same time.

The listing: CODEWRDS

```
10 REM"SECRET CODE WORDS"
20 REM"STEVE BLYN,COMPUTER ISLAN
D,NY,1985"
30 DIM N(26),A$(26)
40 CLS
50 C$=""
60 PRINT@10,"SECRET CODES"
70 PRINT@32,STRING$(32,191);
80 FOR A=1TO 26: N(A)=A:NEXTA
90 FORB=1TO 26:A$(B)=CHR$(64+B):
NEXT B
100 GOSUB 390 -
110 SOUND 200,3
120 PRINT@64," DO YOU WANT TO HI
DE THE CODE?"
130 EN$=INKEY$
140 IF EN$="Y" THEN GOSUB 450 EL
SE IF EN$="N" THEN 150 ELSE 130
150 SOUND 220,3
160 PRINT@64," TYPE IN YOUR MYST
ERY WORD NOW."
170 B$=INKEY$
180 IF B$=CHR$(13) THEN 220
190 C$=C$+B$
200 IF B$="" THEN 170
210 GOTO 170
220 REM"PRINT OUT THE WORD USING
 NUMBERS"
230 FOR T=1 TO LEN(C$)
240 L$=MID$(C$,T,1)
250 PRINT ASC(L$)-64;
260 PRINTCHR$(8);:PRINT"-";
270 NEXT T
280 PRINT:PRINT" WHAT DO YOU THI
NK THE WORD IS       ";
290 INPUT M$
300 PRINTSTRING$(32,".");
310 IF M$=C$ THEN PRINT"
   CORRECT":SOUND180,5
320 IF M$<>C$ THEN PRINT"SORRY,T
HE ANSWER IS ";C$:SOUND10,3
330 PRINTSTRINGS$(32,".");
340 PRINT"PRESS 'M' FOR MORE OR
'E' TO END";
350 EN$=INKEY$
360 IF EN$="E" THEN CLS:END
370 IF EN$="M" THEN 40
380 GOTO 350
390 PRINT@321,STRING$(30,236);
400 REM"PRINT THE CODE"
410 FOR R=1 TO 26:PRINTN(R);CHR$
(8);"=";A$(R);:NEXT R
420 PRINT@481,STRING$(30,227);
430 RETURN
440 REM"HIDE THE CODE"
450 PRINT@352,STRING$(128,143);:
RETURN
```

Learn Your Multiplication With MULTO OF MARS

by Richard Ramella

"I don't have to learn the multiplication tables," my 9-year-old announced.

"Yes, you do," I said.

"No, I don't!"

"DO!"

"DON'T!"

My son and I often have such philosophical discussions.

"Don't you want to know why?" he asked between rounds five and six.

"OK, tell me why."

"Because when I grow up, they'll have wrist computers. If I want to know how much something times something is, I'll just punch it into the computer."

"That hasn't happened yet," I said. But in my heart I knew I was fibbing. There are already cheap digital watches with full calculator functions. Some of the newer ones are rumoured to have spreadsheets that run up your arm.

"Besides," my son went on, "when I'm grown I'll probably be living on Mars." He paused, savouring the idea. "And my wrist computer'll have word processing so I won't have to write. And it'll have a full-colour screen that picks up any TV program I want."

"But what if you run into a Martian slime bunny and it vaporizes your wrist computer?" I said. "Then you won't be able to figure the coordinates to return to Mars Base One. You'll be lost out there! And all because you never learned the times tables!"

"Oh, get serious, Dad!"

I am a stern father. I sent my son to bed with only four peanut butter sandwiches and a quart of milk.

That night I wrote *Multo of Mars*. Multo is a computer character that makes a game of multiplication drills.

I remember learning the times tables in a kind of group agony called choral recitation. Thirty of us squirming fourth-graders droned answers as meaningless as telephone numbers we'd never call. I'm sure most of us managed to lapse into fantasies while mouthing the numbers. Like my son, I usually took a rocket ship to Mars, arriving well before "two times two is four."

The next afternoon, I introduced Multo to the pre-adolescent Earthling at my house. Multo helped but didn't do the entire job alone. Young Earthlings must write, recite and think about concepts they are learning, not just punch the answers into a computer.

Multo of Mars is a 16K Extended Color BASIC program. It uses Extended graphics and animation to teach fundamental multiplication skills ranging from "1 × 1" to "9 × 9." The times table is an educational must which is presented at about third grade level and should be mastered by about fifth grade.

Multo is a comic creature with tousled red hair, a huge head and big blue feet. Its mouth moves rapidly, then becomes a rectangle with a multiplication problem. Multo responds to correct answers in random, cheerful ways: dancing, smiling, crossing or blinking its eyes, and lifting an ear to emit colorful lightning bolts.

Play is simple. When a problem is presented, the player types the number answer and presses ENTER. A correct answer produces positive visual cues, and that particular problem is erased from the system. It may seem the same problem is presented more than once, but consider that "4 × 8" and "8 × 4" are a different sequence, and that "3 × 4" and "2 × 6" have the same answer.

A wrong answer offers nonjudgmental correction. The mouth becomes a green rectangle, the correct answer is shown in white, and the problem is once again presented for the player to enter the answer just seen.

This problem is not taken out of the system. It returns in its random turn until the player gets it right. In this way, the pool of problems narrows to those which the learner needs to study.

Multo of Mars keeps score inwardly. About every seventh correct answer, a new letter of a building message appears on the screen. The encouraging message isn't completed until the 81st problem is answered correctly. When this happens, Multo springs its last surprise: a huge smile and an endless series of dancing, eye-crossing and blinking, and fireworks from the ear. The program must be broken into to stop the run.

If your computer does not accept the "speed poke" (POKE 65495,0.), this command should be taken out of Line 110.

If a run of *Multo of Mars* is stopped before the entire series of problems is worked, the problems not yet solved may be seen by typing FOR X=1 TO 81: PRINT A$(X);: NEXT and pressing ENTER.

My advice to adults is to merely tell the young player how to play and leave the rest as a series of surprises. The building message, especially, tends to sustain interest even after the player has seen through the facade of what is after all a math drill.

The program has no sound. I removed the "boops" and "beeps" after a classroom test showed they tended to interfere with the work of students not at the computer.

Finally, I am not a teacher, but I know these things: Telling the answers to a computer, no matter how much fun it can be, is no substitute for writing the answers on paper. There is a learning connection between seeing, saying and writing, and learning the times tables is only the first step to learning how to multiply large numbers by each other — a process that requires pencil, paper and mind.

(Any inquiries regarding this program may be directed to Mr. Ramella at 1493 Mt. View Ave., Chico, CA 95926. Please include a SASE.)

The listing: MULTO

```
 100 REM * MULTO OF MARS * TRS-80
  EXTENDED COLOR BASIC / 16K / RI
 CHARD RAMELLA
 110 POKE 65495,0: CLEAR 900: DIM
  Z(1,21): ZL$="U8E5F5D2L10R10D6"
 : GOTO 250
 120 Z$="6020010829697871": RETUR
N
130 Z$="234040482959": RETURN
140 Z$="022020606073737575070709
0979": RETURN
150 Z$="022020606073737575357578
786969292908": RETURN
160 Z$="000606765059": RETURN
170 Z$="700000040464647575777759
59191908": RETURN
180 Z$="702020020207072929797978
78757515": RETURN
190 Z$="00707009": RETURN
200 Z$="011010606071717373646414
14030301140505080819196969787875
7564": RETURN
210 Z$="741414030301011010606071
7178786969191908": RETURN
220 Z$="12721575": RETURN
230 Z$="00790970": RETURN
240 FOR H=1 TO LEN(Z$) STEP 4: L
INE(X+VAL(MID$(Z$,H,1)),Y+VAL(MI
D$(Z$,H+1,1)))-(X+VAL(MID$(Z$,H+
2,1)),Y+VAL(MID$(Z$,H+3,1))),PSE
T: NEXT: RETURN
250 PMODE 3,1: PCLS1: SCREEN 1,1
260 COLOR 3,1: LINE(0,0)-(255,20
),PSET,BF: COLOR 1,1
270 M$="U16R5F7E7R5D16L5U11G7H7D
11L5": DRAW"BM5,18;"+M$: PAINT(7
,15),4,1
280 DRAW"BM33,18;U16R5D11R8U11R5
D16L18": PAINT(35,15),4,1
290 DRAW"BM55,18;U16R5D11R13D5L1
8":  PAINT(57,15),4,1
300 DRAW"BM77,18;U11L7U5R19D5L7D
11L5": PAINT(79,15),4,1
310 DRAW"BM93,18;U16R18D16L18E1C
3E4C1U6R8D6L8": PAINT(95,15),4,1
320 CIRCLE(125,11),8
330 DRAW"BM137,18;U10R3L6R3U3E3R
3F3"
340 DRAW"BM153,18;"+M$: PAINT(15
5,15),4,1
350 DRAW"BM183,18;U16R18D16L5U6L
7D6L5": DRAW"BM189,5;R5D4L5U4":
PAINT(185,15),4,1
360 DRAW"BM205,18;U16R17D10L6F6L
5H6L2D6L5": DRAW"BM211,5;R5D4L5U
4": PAINT(207,15),4,1
370 DRAW"BM227,18;U3R13U3L13U10R
18D4L13D3R15D9L18": PAINT(229,17
),4,1
380 DIM A$(81): C=1: D=81: FOR A
=1 TO 9: FOR B=1 TO 9
390 A$(C)=STR$(A)+"X"+STR$(B): C
=C+1: NEXT B,A
400 COLOR 2,1: CIRCLE(128,96),80
,,.7,.96,.55
410 DRAW"BM50,80;H25R35C1R135C2R
35G25": COLOR 4,1
420 R=75: FOR A=-R+10 TO R-10 ST
EP 2: G=R*R-A*A: Y=INT(SQR(G))
430 LINE(A+128,96-Y)-(A+128-(RND
(20)-10),96-Y+RND(25)),PSET: NEX
T: COLOR 2,1
440 FOR X=100 TO 156 STEP 56: CI
RCLE(X,70),20,,.6: CIRCLE(X,73),
5: NEXT
450 DRAW"BM117,85;F12E12": DRAW"
BM115,178;U27R30D27"
460 FOR X=100 TO 160 STEP 60: CI
RCLE(X,183),20,,.5: PAINT(X,185)
,3,2: NEXT
470 LINE(80,188)-(180,192),PRESE
T,BF: DRAW"BM85,188;R32C1R30C2R3
1"
480 FOR U=1 TO 5+RND(15): ER=1+R
ND(3)
490 Q1=Q: P1=P: P=RND(26): Q=RND
(0): CIRCLE(128,125),P,ER,Q: CIR
CLE(128,125),P1,1,Q1
500 NEXT U: CIRCLE(128,125),P,1,
Q
510 COLOR 2,1: R1=0: C$="": E=RN
D(81): IF D=0 THEN 730
520 IF A$(E)="" THEN 510
530 F=VAL(LEFT$(A$(E),2)): G=VAL
(RIGHT$(A$(E),1))
540 LINE(91,115)-(169,135),PSET,
B: X=95: Y=120: A$=A$(E)+"=": GO
SUB 750
550 W$=INKEY$: IF W$=CHR$(13) TH
EN 570 ELSE IF W$="" OR INSTR("1
234567890",W$)=0 OR R1=2 THEN 55
0 ELSE A$=W$: C$=C$+W$: H1=VAL(C
$): GOSUB 750: R1=R1+1
560 GOTO 550
570 IF H1=F*G THEN FOR T=1 TO 50
0: NEXT T: GOTO 590
580 GOSUB 810: R1=0: E1=1: C$=""
: GOTO 530
590 LINE(91,115)-(169,135),PRESE
T,BF: KL=20+RND(30): CIRCLE(128,
115),KL,2,.5,0,.5
600 GH=RND(10): ON GH GOSUB 840,
860,910,970: IF GH>4 THEN FOR T=
1 TO 600: NEXT T
610 CIRCLE(128,115),KL,1,.5,0,.5
620 IF D=74 THEN DRAW"BM15,85;D5
F5E5U5D5G5D6"
630 IF D=67 THEN DRAW"BM15,103;R
10D12L10U12"
640 IF D=60 THEN DRAW"BM15,118;D
12R10U12"
650 IF D=53 THEN DRAW"BM15,151;"
+ZL$
660 IF D=46 THEN DRAW"BM15,166;U
12R10D6L10R3F6"
670 IF D=39 THEN DRAW"BM25,181;L
10U6R5L5U6R10"
680 IF D=32 THEN DRAW"BM235,101;
"+ZL$
690 IF D=25 THEN DRAW"BM235,123;
R10L10U12R10"
700 IF D=18 THEN DRAW"BM235,139;
U12D6R10U6D12"
710 IF D=11 THEN DRAW"BM235,154;
"+ZL$
720 IF D=4 THEN DRAW"BM235,169;U
12F6E6D12"
730 IF D=0 THEN DRAW"BM235,185;U
12R10D6L10": GOTO 990
740 IF E1=1 THEN E1=0: GOTO 480
ELSE A$(E)="": D=D-1: GOTO 480
750 FOR P=1 TO LEN(A$): Q$=MID$(
A$,P,1)
760 K=ASC(Q$): IF K=61 OR K=88 O
R K>47 AND K<58 THEN 770 ELSE 80
0
770 IF K=61 THEN GOSUB 220 ELSE
IF K=88 THEN GOSUB 230 ELSE IF K
=48 THEN GOSUB 120 ELSE IF K=49
THEN GOSUB 130 ELSE IF K=50 THEN
 GOSUB 140 ELSE IF K=51 THEN GOS
UB 150 ELSE IF K=52 THEN GOSUB 1
60 ELSE IF K=53 THEN GOSUB 170
780 IF K=54 THEN GOSUB 180 ELSE
IF K=55 THEN GOSUB 190 ELSE IF K
=56 THEN GOSUB 200 ELSE IF K=57
THEN GOSUB 210
790 GOSUB 240: X=X+12: NEXT P: R
ETURN
800 NEXT P: RETURN
810 LINE(91,115)-(169,135),PSET,
BF
820 COLOR 1,1: X=95: Y=120: A$=A
$(E)+"="+RIGHT$(STR$(F*G),2): GO
SUB 750
830 FOR T=1 TO 1000: NEXT T: LIN
E(91,115)-(169,135),PRESET,BF: C
OLOR 2,1: RETURN
840 FOR U1=1 TO 5+RND(10): FOR X
1=100 TO 156 STEP 56: PAINT(X1,6
6),RND(2)+2,2: NEXT X1,U1
850 PAINT(100,66),1,2: PAINT(156
,66),1,2: RETURN
860 FOR HG=1 TO 3+RND(5): C1=RND
(2): IF C1=1 THEN L1=79 ELSE L1=
139
870 GET(L1,170)-(L1+56,192),Z
880 FOR J1=170 TO 170-(RND(8)*2)
 STEP -2: GOSUB 900: NEXT J1
890 FOR J1=J1 TO 170 STEP 2: GOS
UB 900: NEXT J1,HG: RETURN
900 PUT(L1,J1)-(L1+56,J1+22),Z:
RETURN
910 GET(205,50)-(230,80),Z
920 FOR J1=50 TO 30 STEP -1: GOS
UB 960: NEXT J1
930 FOR T=1 TO 5+RND(10): P=3+RN
D(10): P$=RIGHT$(STR$(P),1): PL$
="E"+P$+"F"+P$: PL$=PL$+PL$+PL$:
 CO$=STR$(1+RND(3))
940 F$=";BM217,52;": DRAW "C"+CO
$+F$+PL$: FOR T1=1 TO 100: NEXT
T1: DRAW"C1"+F$+PL$: NEXT T
950 FOR J1=30 TO 50: GOSUB 960:
NEXT J1: COLOR 2,1: RETURN
960 PUT(205,J1)-(230,J1+30),Z: R
ETURN
970 FOR WR=1 TO RND(5)*2: IF WR/
2=INT(WR/2) THEN T1=1: T2=2 ELSE
 T1=2: T2=1
980 CIRCLE(100,73),5,T2: CIRCLE(
113,70),5,T1: CIRCLE(156,73),5,T
2: CIRCLE(145,70),5,T1: F0R YT=1
 TO 10: NEXT YT,WR: RETURN
990 FOR X=85 TO 115: CIRCLE(128,
X),40,4,.5,0,.5: NEXT X
1000 FOR X=105 TO 113: CIRCLE(12
8,X),32,1,.4,0,.5: NEXT X
1010 GH=RND(4): ON GH GOSUB 840,
860,910,970: GOTO 1010
1020 END
```

ML UTILITY 16K ECB

Enhance Your Keyboard Input With Buffer Stuffer

By Richard W. Rutter

This program consists of a position independent machine language routine designed to greatly enhance your Colour Computer's keyboard input capability. Its features include:

1) The ability to mask (disable) up to 10 keys.

2) The ability to unmask any key that had been previously masked.

3) The ability to increase or decrease the size of the input text buffer.

4) A resetable right tab key.

5) A resetable left tab key.

6) A repeat key to allow rapid duplication of any printable keypress, and the ability to either increase or decrease the speed of this repeat function.

7) An exchange function that lets you change characters anywhere within the input buffer instead of having to retype the line.

8) The ability to edit BASIC text strings using any or all of the above options.

9) The ability to apply any of all of the above options to Extended Colour BASIC's line statement EDIT function.

10) The ability to enable or disable the entire program, as needed, by entering the command EXEC.

In essence, BUFFER STUFFER provides the capability to both input and edit command lines and program statements and text strings according to user modifiable specifications.

The program will require 1,536 bytes of storage. It may be offset loaded into either an unused graphics page or behind the string pool. There are two ways to create the program: First, process the Assembly Language Source Code with a dependable assembler, or second, use the Object Code Generator to poke the instructions into RAM and have a complete block of memory saved on either cassette or disk.

If you have a 16K computer, you may need to PCLEAR 3 to provide room for the Object Code Generator. Also, you should exclude the comments in the Source Code to assure that it will fit within a 16K computer. A detailed description of how these programs function will be provided later.

Remember that the assembler generated version will always need a loading offset value, but the OCG version may not necessarily require one. Here are two loading examples: CLOADM "BUFBIN",1536 for Extended Colour BASIC or LOADM "BUF.BIN",3541 for Disk Extended Colour BASIC.

After you have loaded it into your computer, enter the command EXEC. The program is now "patched" into your computer's line input routine. To verify this, press the down-arrow key. This key is the control key. When you press it, the cursor will flash yellow, reminding you you're in the control mode. Whenever in this mode, you will have nine keyboard command options available. You may abort the control mode by again pressing the control key. Let's look at each of the nine control mode options.

If not already in the control mode, press the control key to activate it. Now press the right-arrow key. You have just sent a right tab. The value of the right tab has been initially set to five blank spaces.

the right-arrow key. You have just sent a right tab. The value of the right tab has been initially set to five blank spaces.

To reset the right tab, press the control key and then press 'R'. You will see the prompt RTAB:. Enter the desired numerical value. Note that only three-digit key presses will be accepted; anything beyond that will be ignored. Non-digit key presses will not be displayed.

If you key in the wrong value or change your mind for whatever reason, press BREAK and the routine will abort without affecting any current values. Take note that there is no backspace function. Use the BREAK option to start over if you should make a mistake. Press ENTER to return the current value. Note that an entry less than one will cause an automatic abort, and all values will remain unchanged. An entry in excess of 250 will be adjusted equal to 250. To verify all of this, experiment with both setting and sending the right tab.

The left tab is the opposite of the right tab. To send one, press control, and then press the left-arrow. The left tab erases a predetermined number of characters. To reset the left tab value, press control and then press 'L'. You will see the prompt LTAB:. Enter the desired value in precisely the same manner as you would set the right tab.

You may change the buffer size by pressing control and then pressing 'B'. The prompt BUF: will appear. Enter the desired buffer size, one to 250. The buffer size determines how many characters may be entered into the current line. It is difficult to overstate the usefulness of this option.

Now let's try masking a key. Press control, then press 'M'. You see the prompt MASK:. Press whatever key you wish to mask. To verify that the key is masked, try pressing it; any key that is masked will be completely ignored. The main purpose of the mask option is to prevent the loss of data from an accidental key press. You will almost certainly want to mask the BREAK and CLEAR keys. Also, the "line erase" SHIFT-left arrow and ENTER keys are prime candidates for masking.

It is fitting that an unmask option be available. Press the control key, and then press 'U' and you will see the prompt UNMASK:. Press whatever key you wish to unmask. To verify that it is unmasked, press it. You normally would not press keys such as BREAK, ENTER, and CLEAR to test for mask status, for obvious reasons. Also, note that two keys are not completely maskable. If you mask the control key, it will still allow access to one control option, the unmask function. If you mask the 'U' key, it will still respond to an unmask request.

Another feature is the repeat key option. To try it out, press any printable key and press SHIFT-@. The current character will begin to duplicate itself and will continue to do so until you press a key to stop it, or either the beginning or end of the buffer is reached. You may also use the repeat key to repeat delete (left-arrow, SHIFT @).

It is a good idea to use the repeat key to stop and start the repeat process so you will be able to interact with it more swiftly. Practice using the repeat key to familiarize yourself with it.

The speed of the repeat process may be increased or decreased. Press control, then press 'S'. You see the prompt SPEED:. Enter the desired value from one to 250. A setting of one will give you the fastest speed, while a setting of 250 will yield the slowest.

Perhaps the most useful feature is the EXCHANGE command. If at least one character is currently in the buffer, you may activate this mode by pressing control and then pressing 'X'. The cursor is now riding over the last character in the buffer. The cursor is flashing orange, and you will notice the character beneath it can still be seen.

When in the exchange mode, you have six commands available. They are: move left, move right, character delete, character insert, repeat function, and exit using ENTER. To move the cursor to the left, press the left-arrow. To move the cursor to the right, press the right-arrow. To delete the character directly under the cursor, press CLEAR. To insert a printable character, press the desired key, and it will be inserted at the current cursor position. To leave the exchange mode, press ENTER.

The only key checked for mask status in the exchange mode is the repeat key. If you want the repeat option to function, you should unmask it before entering this mode since no control options are available from within exchange. The repeat key is quite useful to quickly position the cursor anywhere within the buffer. Remember that you may enter and exit the exchange mode as needed so as to access the control options. Try experimenting with the exchange mode.

Yet another option is the ability to edit string variables. To use the option you will need Extended Color BASIC and a BASIC program subroutine similar to the sample edit driver program I have provided.

Run this program to test the string entry/edit capabilities. All of the commands discussed apply to the entry and edit of text strings. You may append characters to the end of the string or activate the exchange mode (control X) to make changes anywhere within the string. Press ENTER or BREAK to end the edit session. When you do, you see the prompt A/C/G:.

If you press 'A', the edit session will start again using the same string you originally sought to edit. If you press 'G', the current string will be sent directly into the BASIC variable, and control will return to the calling program. Pressing 'C', or any other key, will continue the edit session using the current string.

The final option available is the ability to edit program statements. If you have Extended Color BASIC, you should first use the *EDIT* command (i.e., *EDIT30*) to access the desired line statement. All of BASIC's line *EDIT* commands are preserved (unless you choose to mask them). *Buffer Stuffer's* commands will also function (unless you choose to mask them). The ability to activate the exchange mode (control X) effectively provides an "editor within an editor." You may prefer the exchange mode when editing your BASIC programs.

There are a few changes to the performance of BASIC's *EDIT* function you should be aware of. The first is the possible effects when using the repeat option to repeat change characters. Since the repeat mode does not know how many changes to make, the key value causing the character change will be sent to BASIC immediately after the specified number of changes have been made, unless you have pressed a key to stop it. You will find it nearly impossible to react that quickly. A problem will occur if the keys 'A', 'E', 'Q', or 'X' are being repeat changed. They are also *EDIT* command keys, and if sent to BASIC could cause needless inconvenience.

If you have any problems with the repeat key when in line *EDIT*, you might consider masking the repeat key or activating the exchange mode. Realistically, this should rarely be a problem since you are unlikely to need a repeat change when editing a program statement.

Notice that if the current buffer limit is less than the length of the program statement being edited, you will need to use the control B option to expand the buffer size. Failure to do so will restrict your ability to edit the line. In fact, the cursor may even be "frozen" at the current position. No need to worry, however, because the control options are available to get you out of such a jam.

When you are in the character insert mode, you will be allowed to insert one character more than the current buffer limit. However, you will not be able to exit the insert mode (using SHIFT^ or ENTER) until you have backspaced at least one position to ensure your line is of legal length. This feature ensures your program lines cannot exceed the buffer size you have preset.

There is a modification to the keyboard that I have not yet mentioned. The right-arrow key now performs as an extra space bar. This simplifies the insertion and deletion of spaces. The right-arrow key does not function as a space bar when you are in the exchange mode; only when appending characters or when in normal line statement *EDIT* is it redefined.

The Assembly Language Source Code

All numerical values to the right of the line numbers are in base 10. Lines 90 to 220 equate ROM referenced memory locations which allow the program to communicate with BASIC on an interactive basis. We will demonstrate the functions of these equated locations as we encounter them throughout the source code.

Lines 260 to 450 define the prompt display strings; the end of each prompt is indicated by a *CHR$(255)*. Each of the control mode prompts starts with a *CHR$(128)*. This ensures that the prompts will not be confused with any other characters currently on the screen. All of these prompts will be erased automatically to prevent the display from becoming a jumbled mess.

Lines 490 to 910 contain the "variable" locations manipulated exclusively by the program. MAXBUF will reside in Location 51,PCR. Its value must never exceed 250, but it may be smaller. It determines just how large the buffer limit may become when using the set buffer control option.

BUFLIM will reside in Location 52,PCR. Its value determines the number of characters that may be entered into the current buffer. The buffer set routine is used to change it to any value between one and MAXBUF. It must never exceed 250.

CONKEY will reside in Location 53,PCR. It is used to define the control key. You may change it to any key you so desire. I chose the down-arrow key because it is unprintable, preventing the loss of any important characters.

REPKEY will reside in Location 54,PCR. It is used to define the repeat key. You may change it, but I chose the SHIFT @ key because it is unprintable.

CONCUR will reside in Location 55,PCR. It determines the cursor character when the control mode has been activated. It may be changed to any printable character.

EXCCUR will reside in Location 56,PCR. It determines the cursor character when the exchange mode has been activated or when the repeat mode is duplicating characters.

LTBSIZ will reside in Location 57,PCR. It determines how many backspace characters will be sent when a left tab is requested.

RTBSIZ will reside in Location 58,PCR. It determines how many blank characters will be sent when a right tab is requested.

RSPEED will reside in Location 59,PCR. Its value determines how quickly or slowly the repeat function will duplicate characters. It may contain any value from one to 255. The smaller its value, the faster the repeat speed.

MINVAL will reside in Location 60,PCR. It determines the minimum value accepted when using any of the control key value set commands. You may reset it to any value between one and 250.

Lines 590 to 800 are to be manipulated exclusively by the program. You should not attempt to change them.

Lines 810 to 900 make up the keyboard mask table. If a key is masked, its value will reside in one of these 10 locations (83,PCR to 92,PCR). The mask and unmask control functions manipulate these bytes. You may also manipulate this table as long as you do not change location 93,PCR since it flags the table's end.

Lines 950 to 1040 effectively patch the program into BASIC's keyboard input routine. A check is made to see if the patch is already in effect. Locations 1533,PCR and 1534,PCR must both contain *CHR$(255)*, or the routine will be deactivated rather than activated. The activation sequence requires that the two-byte memory value at Location 363 be replaced with the program's starting address. The value is first placed in Location RETBAS,PCR so that it may be restored at the next *EXEC* command.

Lines 1080 to 1140 effectively deactivate the program by pulling the return address out of RETBAS,PCR and placing it back into Location 363. Two *CHR$(255)*s are put back into RETBAS,PCR to allow reactivation at the next *EXEC* command.

Lines 1180 to 1730 comprise the routine to access BASIC string variables. The length and location of the variable must be sent to this routine from BASIC. Register Y points to the location of the variable. Register X points to the start of BASIC's input buffer. If the length of the variable is greater than zero, each character of the string will be placed into the BASIC buffer and displayed on the screen. The length of the string is temporarily increased by one to satisfy a ROM input requirement. The ROM subroutine is called, and the BASIC string is treated as keyboard input.

When either the BREAK key or the ENTER key is pressed, the BASIC ROM will return control to this calling location. This allows the options of either continuing, sending the results to BASIC, or reediting the original string. Continuing the edit is accomplished by erasing the prompt and positioning the cursor at the end of the current string.

We must take into account any screen scroll caused by the prompt display and compensate for it if needed. To restart the edit using the original string we must erase the prompt, erase the current string, and pull the original out of the BASIC string by starting anew. To send the current string, we simply erase the prompt, send the string length, and copy the characters into the BASIC string, if any.

Lines 1770 to 1910 contain the primary keyboard scan routines. If not in the exchange mode and not in the repeat mode, the cursor is flashed in the same way that normal BASIC would do it. The ROM POLCAT key scan routine is used to seek a key press. If a key is pressed, we erase the cursor.

Lines 1950 to 1990 provide the ability to send special cursor characters when in the exchange or repeat key modes. VIDPOS contains the current video screen print location.

Lines 2030 to 2270 contain the repeat key activation routine. A check is made to see if the repeat key had been pressed and if the current key value is a valid one. If so, a timing loop is started to search for a request to stop the repeat through any other key press. If the timer expires without any key press, the current key is fetched from CURKEY and returned as the key press. If a key is pressed, a check is made to see if it is the repeat key. If it isn't, that key will be returned as the current value. If it is the repeat key, it is checked for masked status. If masked, it is rejected. Otherwise, the entire process is repeated until either the timer expires and CURKEY is returned as the key press, or a key other than REPKEY is pressed, thereby deactivating the repeat function and returning a new value in CURKEY.

I prefer this repeat method over the kind which requires you hold down a particular key. There are three reasons for this preference: First, having to hold down any key is annoying; second, the problem that can be caused if keys such as BREAK and ENTER are held down too long; and third, the instantaneous response available through a defined repeat key as opposed to the annoying delay by the other method. There is merit in either method, and you may wish to create a repeat routine different from the one provided.

Lines 2320 to 2530 perform a multitude of functions. BASIC's input routine jumps to CHECK whenever BASIC requires keyboard input. The device number must be zero, or the entire operation is aborted, returning directly to BASIC. A check is made to see if the buffer pointer (register X) is either at the beginning or the end of the buffer. Such would be the case if 'X' is pointing to the same previous location, and the repeat function must then be deactivated by setting CURKEY to zero. The input/output buffer is cleared to satisfy a BASIC requirement. The exchange mode indication flag is also cleared.

The current video screen location is saved for later use. The number of characters currently in the buffer is saved in BUFCNT. Tests are also made to see if either the right or left tab counts need to be satisfied, in which case the appropriate tab routine will be executed. A key scan is started and will continue until either a key is pressed or the repeat mode causes CURKEY to be fetched as the current value. The cursor is erased. The key is checked for masked status. If not masked, it is processed normally. If masked, a check is made to see if it is the control key. If it is the control key, we allow it to be processed. Any other masked key press will be hidden from BASIC.

Lines 2570 to 2660 comprise the check for mask routine. Each byte in the mask table is examined until either we find a match or reach the table's end. Register B will contain the search result. If the zero flag is set, the key is not currently in the mask table.

Lines 2700 to 2840 effectively process the current key press. If it is the control key, then we activate the control mode.

If the right-arrow key has been pressed, we convert it to a blank. We fetch the number of characters currently in the buffer and see if the buffer limit has been reached. If there is still room, we send the key to BASIC. If not, we check to see if Extended BASIC's Line Edit is in operation by testing for a character count versus a buffer count mismatch. If the counts are equal, we are not in a Line Edit. Otherwise, we will only accept a backspace to bring the edit count within range. If we are not in Line Edit and the buffer limit has been reached, we will only accept a key press which will not add to the buffer. Any unusable key press will be rejected by hiding the current key press from BASIC and assuring that that character cannot be repeat processed.

Lines 2880 to 3110 process a control key request. The control cursor is flashed according to the special cursor flash timing function. The key scan/flash sequence will continue until a key is pressed. After getting a key press, we will attempt to convert it to uppercase. If the 'U' key has been pressed, the unmask routine is called. Any other keypress is checked for masked status. If masked, we hide it. Next, we check the control key itself for masked status. If it is masked, we abort the control session.

Lines 3150 to 3490 look for a valid control mode request. Any key that does not correspond to one of the control mode options is hidden from BASIC.

Lines 3530 to 3620 either hide or send the key press, as appropriate. If the key press is not repeatable, then no option to repeat it will be allowed. We fetch the current buffer count and save the current buffer pointer. We return to BASIC in a manner that will prevent a redundant key scan.

Lines 3660 to 3790 effectively unmask the desired key press. A prompt is displayed and a key press is looked for. The key press is searched for in the mask table, and if found, will be removed from the table. After successful unmask or reaching the table end, the prompt is erased, and the key press is hidden from BASIC.

Lines 3830 to 4000 effectively mask the desired key press. A prompt is displayed and a key press is looked for. The mask table is searched to find the first free byte. If one is found, the key press is stored in that byte, and the unmask routine is entered to assure that no mask duplications are present. If the end of the table is reached before a free byte is found, the key press will not be masked.

Lines 4040 to 4090 attempt to set a new buffer limit by calling the *Get Number* routine. If the value returned is equal to the maximum, no adjustment is needed, otherwise we must increase it by one to compensate for BASIC's input requirements. The new value of BUFLIM is saved, and the key press is hidden from BASIC.

Lines 4130 to 4150 attempt to set a new left tab value. Lines 4190 to 4210 attempt to set a new right tab value. Lines 4250 to 4270 try to set a new repeat speed.

Lines 4310 to 4350 effectively set to zero those values used by the get number routine.

Lines 4390 to 4460 are used to send prompts to the screen. A count of the number of characters sent is kept in BKUCNT so the prompt may later be erased.

Lines 4500 to 4540 erase the number of characters specified in BKUCNT. This routine is normally used to erase prompts.

Lines 4580 to 4910 get and process numerical value set requests. The appropriate prompt is displayed. Numerical values are set to zero. The key press count is set to three, assuring that no more than three digits may be entered. A key scan is started, and continues until a usable key is pressed. If a digit is pressed, it is sent to the screen, the get number routine is called, and the digit count is updated.

If BREAK has been pressed, the routine is aborted by erasing the prompt, pulling the return location off of the stack and hiding the key press from BASIC. If ENTER has been pressed, the prompt is erased, the number in CURVAL is tested for validity and is adjusted if too large, or the routine is aborted if the value is too small. Any usable numerical value (MINVAL to MAXBUF) will be returned to the calling routine.

Lines 4950 to 5230 figure an ongoing numerical quantity for the set value routine. The current digit is changed to a number and saved in register B. The decimal places will be moved from right to left, and a new value will be computed. Checks are made to see that no attempt will be made to compute a value greater than 255. If the value could exceed 255, it will be set equal to MAXBUF. Upon return from this routine, the current value (CURVAL) will be in register B.

Lines 5270 to 5640 attempt to activate and control the buffer exchange routine. The flag EXCHAN is incremented to indicate exchange mode activation. The current number of characters in the buffer are fetched. The current line end is flagged with a zero. The beginning of the buffer is tested to see if any characters are present; if none are, we abort the exchange request. If at least one character is present, we activate the exchange mode.

Upon activation, we save the current character count in BUFCNT and the current buffer end in EOBUF. Register Y is saved on the hardware stack. The buffer and video pointers are decremented to point to the last character in the buffer. A keyboard scan is then started which will continue until a key is pressed. The cursor is flashed at a rate determined by the *Timer* subroutine. Instead of erasing the cursor, this time we replace it with the current character pointed to by register X.

Whenever a key is pressed, we replace the cursor with the current buffer character and save the buffer pointer in TMPX. We then determine if the key is a usable one; if usable, we process it accordingly. If unusable, we assure that repeat is deactivated and restart the key scan.

Lines 5680 to 5740 respond to a request to move the cursor one place to the left. If at the buffer start, the request will be ignored. Otherwise, both the video pointer and the buffer pointer will be decremented by one.

Lines 5780 to 5840 attempt to move the cursor one place to the right. If the pointers are not current at the line end, they will be incremented by one to accomplish this.

Lines 5880 to 6260 attempt to insert a character at the current cursor position. The buffer count is fetched and checked to see if it is less than the buffer limit. If the count is equal, there is no room, and the request will be ignored.

Having determined that there is room, we set 'Y' to point to the current buffer end. We then move adjacent characters one place to the right until all characters from the current buffer position to the buffer end have been moved. We then insert the new character into the current buffer position. The buffer end is incremented and its value is cleared to indicate a new end of line. The characters on the screen are moved in a similar manner.

We must check to see if the screen will scroll by comparing the video position to the value of SCREND. If a scroll will occur, we must decrement the appropriate pointers by one full line. The new buffer contents from the buffer position rightward are displayed on the screen.

After the screen characters have been moved, the current video position is replaced by the desired position. We also compensate for the additional character by incrementing the old video value, thereby providing the proper return screen location when the exchange mode is exited.

Finally, we increment the buffer pointer and buffer count, and return to the key scan routine.

Lines 6300 to 6600 will attempt to delete the character at the current cursor position. Two or more characters must be present for any to be deleted.

Deleting the character is accomplished by starting at the current buffer position and copying the character which is one position to the right of it into the buffer position. This continues until the end of the line is found, in which case a zero will be placed in the last character of the line.

We then test to see if the character just deleted was the last character of the line. If it was, we decrement the buffer and video pointers. In a manner similar to the one used by the insert function, the old screen characters are replaced by new ones. The last screen character is replaced with a blank.

Lastly, the video and buffer pointers are updated. The video position is reset to its proper place on the screen. The old video position is decremented so the proper return screen location is available when the exchange mode is exited. The end of buffer pointer is decremented to show the new end of buffer.

Lines 6640 to 6740 process the exit from the exchange mode. The video position is reset to one position beyond the last character on the screen. The original value of register Y is restored. The current buffer end is given to register X.

The exchange flag, EXCHAN, is decremented and tested for zero status. If equal to zero, Extended BASIC's Line Edit is not in effect, so the buffer counters must not be adjusted. If Line Edit is in effect, we must fetch and adjust the character count, give it to the edit count, and set the buffer operation count to zero. When in Line Edit, BUFCNT contains the operation count (i.e. the number of moves or changes requested). It should be set to zero upon exit of the exchange mode to assure the operation count will also be set to zero. The key press must also be hidden from BASIC.

Lines 6780 to 6810 pull a character from the current buffer position and send it to the current screen position.

Lines 6850 to 6930 effectively adjust register B for proper screen display.

Lines 6970 to 7000 are used to convert a key press command from lowercase to uppercase. This makes it simpler to check for keypress command matches.

Lines 7040 to 7170 are needed to determine the proper character count depending on which ROM has called *Buffer Stuffer*. We see if Line Edit is in effect by getting the calling address from the hardware stack. If the address is higher than the Line Edit Vector, we know we are not in Extended BASIC, and we simply return the normal buffer count.

If we are in Line Edit, the edit count is used as the character count. We next test the exchange flag to see if the exchange mode has been requested. If so, we call the ROM routine *Getend* to position the cursor at the end of the line. We then fetch the edit count, adjust it for the exchange mode and return it as the character count.

Lines 7210 to 7250 contain the timing routine used to determine when either the control cursor or the exchange mode cursor should be flashed.

Lines 7300 to 7320 contain the return location for normal keyboard input when the program is patched into BASIC or the proper flag to indicate that patching is needed if an *EXEC* command has been entered.

Line 7330 provides a convenient reference point for computing the actual length of the program. *Bottom* is also used as a counter in numerous locations throughout the program.

The Object Code Generator

The OCG is designed expressly for those who do not have an assembler. It contains the same instructions the assembler version would generate. Although essentially self-explanatory, some comments should be helpful.

If you have a disk system, do a *FILES 2,256* to assure that the data values will be poked into usable RAM. The OCG assumes you want a disk save for a disk system and a cassette save for a cassette system. To avoid this, change Line 190 to *DEV$="CASSETTE":GOTO 220.*

Note that if you have a 16K computer, you will need to *PCLEAR* three or fewer graphics pages to assure that the OCG will fit into your computer. Also, if you do not have Extended BASIC, you will need to reserve space behind the string pool and change the values of **FI,LA and EX so they will reference that reserved memory. Here is one way to do it: Change Line 40 to** ***CLEAR 500,31100*** **and add the line** ***75 FI =31100:LA=FI+1535: EX=FI+94***

The String Edit Driver Program

For those of you who have Extended Color BASIC, this program allows you to edit string variables. It is fairly simple, but a few comments should be helpful.

Line 50 contains the execution offset. Some execution offset will always be required. Just what it should be depends on where in memory the 6809 routine currently resides. OF must be equal to whatever loading offset you used. For an OCG version, OF must be equal to FI plus any loading offset. Figuring the proper offset should be quite simple.

Line 10000 contains the essential ingredients of the parameter passing subroutine. EL is the memory location that contains BASIC's machine language execution address. We save this two-byte value by copying it into EA and EB. VP will contain the variable pointer of the parameter string PA$. VL will contain the address inside ***Buffer Stuffer*** **where the location of the BASIC string will be stored.**

We extract the true length of PA$. Next we pad PA$ with trailing blanks. VP is assigned the variable pointer of PA$. We poke the true length of PA$ into VL. We poke the starting address of PA$ into VL+1. Now ***Buffer Stuffer*** **knows how long the string is and where to look for it. We evoke the string editor. Upon return, VL contains the new length. We poke the new length into the variable pointer of PA$. We restore the routine's activation/deactivation execution address. Finally, we return the new value of PA$ to the program's calling routine.**

If you decide to use this string edit option, it is imperative the commands in Line 10000 be preserved.

Concluding Remarks

It is not by chance the program is exactly one graphics page in length. My goal was to pack all those keyboard options into precisely 1,536 bytes of memory. Many more options could be added, but it would be very difficult to do so without requiring more memory. One way to do so would be to use a completely stack oriented approach. I chose not to use that approach because, although it would save memory, the program would become much more difficult to follow, let alone to understand.

In any event, by using ***Buffer Stuffer*****, you'll no longer need to be a huffer or a puffer!**

Listing 1:

```
                        00010 *ASSEMBLY LANGUAGE SOURCE CODE
                        00020 *BUFFER STUFFER (C) 1984
                        00030 *by Richard W. Rutter
                        00040 *
0000                    00050  ORG 0 ;SIMPLIFY OFFSET LOADING
                        00060 *
                        00070 *MISC EQUATES
                        00080 *
              006F      00090 DEVNUM EQU 111 ;DEVICE NUMBER
              0070      00100 IOBUFF EQU 112 ;I/O BUFFER
              0088      00110 VIDPOS EQU 136 ;VIDEO POSITION
              00D7      00120 EDTCNT EQU 215 ;LINE EDIT COUNT
              016A      00130 INPVEC EQU 362 ;ROM INPUT VECT
              02DD      00140 BSTART EQU 733 ;BUFFER START
              05E0      00150 SCROPO EQU 1504 ;SCROLL POS
              05FF      00160 SCREND EQU 1535 ;SCREEN END
              85B4      00170 GETEND EQU 34228 ;GET LINE END
              9FFF      00180 LEDVEC EQU 40959 ;LINE EDIT VEC
              A000      00190 POLCAT EQU 40960 ;SCAN KEYBOARD
              A002      00200 CHROUT EQU 40962 ;PRINT CHARS
              A199      00210 FLASH EQU 41369 ;FLASH CURSOR
              A39A      00220 INPUT EQU 41882 ;BAS ROM INPUT
                        00230 *
                        00240 *MISC PROMPT STRINGS
                        00250 *
0000          41        00260 ASKU FCC "A/C/G:"
              2F
              43
              2F
              47
              3A
0006          FF        00270  FCB 255
0007          80        00280 BUFPRO FCB 128
0008          42        00290  FCC "BUF:"
              55

              46
              3A
000C          FF        00300  FCB 255
000D          80        00310 LTBPRO FCB 128
000E          4C        00320  FCC "LTAB:"
              54
              41
              42
              3A
0013          FF        00330  FCB 255
0014          80        00340 MASPRO FCB 128
0015          4D        00350  FCC "MASK:"
              41
              53
              4B
              3A
001A          FF        00360  FCB 255


001B          80        00370 RTBPRO FCB 128
001C          52        00380  FCC "RTAB:"
              54
              41
              42
              3A
0021          FF        00390  FCB 255
0022          80        00400 RSPPRO FCB 128
0023          53        00410  FCC "SPEED:"
              50
              45
              45
              44
              3A
0029          FF        00420  FCB 255
002A          80        00430 UNMPRO FCB 128
002B          55        00440  FCC "UNMASK:"
              4E
              4D


              41
              53
              4B
              3A
0032          FF        00450  FCB 255
                        00460 *
                        00470 *RESERVED SYMBOLIC LOCATIONS
                        00480 *
0033          FA        00490 MAXBUF FCB 250 ;MAX BUFFER SIZE
0034          FA        00500 BUFLIM FCB 250 ;BUFFER LIMIT
0035          0A        00510 CONKEY FCB 10 ;CONTROL KEY
0036          13        00520 REPKEY FCB 19 ;REPEAT KEY
0037          9F        00530 CONCUR FCB 159 ;CONTROL CURSOR
0038          FF        00540 EXCCUR FCB 255 ;EXCHANGE CURSOR
```

```
0039          05        00550 LTBSIZ FCB 5 ;LEFT TAB SIZE
003A          05        00560 RTBSIZ FCB 5 ;RIGHT TAB SIZE
003B          28        00570 RSPEED FCB 40 ;REPEAT SPEED
003C          01        00580 MINVAL FCB 1 ;MINIMUM VALUE
003D          01        00590 CURVAL FCB 1 ;CURRENT VALUE
003E          00        00600 OLDVID FCB 0 ;OLD VIDEO POS
003F          00        00610  FCB 0
0040          00        00620 EOBUF FCB 0 ;TEMP END OF BUFFER
0041          00        00630  FCB 0
0042          00        00640 TMPX FCB 0 ;FOR REGISTER X
0043          00        00650  FCB 0
0044          00        00660 CURPOS FCB 0 ;CURSOR POSITION
0045          00        00670  FCB 0
0046          00        00680 EXCHAN FCB 0 ;EXCHANGE FLAG
0047          00        00690 UNITS FCB 0 ;DIGIT 0-9
0048          00        00700 TENS FCB 0 ;DIGIT 0-9
0049          00        00710 HUNS FCB 0 ;DIGIT 0-9
004A          00        00720 CURKEY FCB 0 ;CURRENT KEYVALUE
004B          00        00730 REPEAT FCB 0 ;REPEAT INDICATOR
004C          00        00740 RTBCNT FCB 0 ;RIGHT TAB COUNT
004D          00        00750 LTBCNT FCB 0 ;LEFT TAB COUNT
004E          00        00760 VARLEN FCB 0 ;STRING VAR LENGTH
004F          A0        00770 VLOC FCB 160 ;LOCATION OF BASIC
0050          00        00780  FCB 0 ;STRING VARIABLE
0051          00        00790 BUFCNT FCB 0 ;BUFFER CHAR COUNT
0052          00        00800 BKUCNT FCB 0 ;PROMPT BACKUP CNT
0053          00        00810 MASK FCB 0 ;MASK VALUE TABLE OF
0054          00        00820  FCB 0 ;UP TO 10 KEYS
0055          00        00830  FCB 0
0056          00        00840  FCB 0
0057          00        00850  FCB 0
0058          00        00860  FCB 0
0059          00        00870  FCB 0
005A          00        00880  FCB 0
005B          00        00890  FCB 0
005C          00        00900  FCB 0
005D          FF        00910  FCB 255 ;SHOW MASK TABLE END
                        00920 *
                        00930 *ENABLE THE ROUTINE
                        00940 *
005E AE       8D 059B   00950 HOOK LDX 1+RETBAS,PCR ;IS THE
0062 8C       FFFF      00960  CMPX #65535 ;HOOK IN EFFECT?
0065 26       16        00970  BNE UNHOOK ;YES,UNHOOK IT
0067 B6       016A      00980  LDA INPVEC ;GET JMP COMMAND
006A A7       8D 058E   00990  STA RETBAS,PCR ;COPY IT
006E BE       016B      01000  LDX INPVEC+1 ;GET MEMORY LOC
0071 AF       8D 0588   01010  STX 1+RETBAS,PCR ;COPY IT
0075 30       8D 0116   01020  LEAX CHECK,PCR ;GET PROG START
0079 BF       016B      01030  STX INPVEC+1 ;PLUG INTO BASIC
007C 39                 01040  RTS  ;HOOK COMPLETED
                        01050 *
                        01060 *DISABLE THE ROUTINE
                        01070 *
007D AE       8D 057C   01080 UNHOOK LDX 1+RETBAS,PCR ;GET IT
0081 BF       016B      01090  STX INPVEC+1 ;SET NORMAL BASIC
0084 30       8D 0575   01100  LEAX 1+RETBAS,PCR ;GET RET LOC
0088 86       FF        01110  LDA #255 ;RESET HOOK INDICATOR
008A A7       84        01120  STA ,X ;STORE ONE
008C A7       01        01130  STA 1,X ;AND THE OTHER
008E 39                 01140  RTS  ;UNHOOK COMPLETED
                        01150 *
                        01160 *ROUTINE TO EDIT BASIC STRINGS
                        01170 *
008F 10AE 8C BC         01180 GETVAR LDY VLOC,PCR ;BAS VARPTR
0093 8E       02DD      01190  LDX #BSTART ;GET BUFFER START
0096 5F                 01200  CLRB  ;SET COUNTER
0097 6D       8C B4     01210  TST VARLEN,PCR ;NULL STRING?
009A 27       0E        01220  BEQ NTS ;YES,NOTHING TO SEND
009C A6       A0        01230 GET1 LDA ,Y+ ;GET VARIABLE
009E A7       80        01240  STA ,X+ ;PUT INTO BUFFER
00A0 AD       9F A002   01250  JSR >[CHROUT] ;SEND TO SCREEN
00A4 5C                 01260  INCB  ;UPDATE COUNTER
00A5 E1       8C A6     01270  CMPB VARLEN,PCR ;ALL SENT?
00A8 25       F2        01280  BLO GET1 ;CONTINUE
00AA 5C                 01290 NTS INCB  ;BUFFER SIZE FOR ROM
00AB E7       8C A3     01300  STB BUFCNT,PCR ;SAVE IT
00AE 0F       6F        01310 GET2 CLR DEVNUM ;KEYBOARD INPUT
00B0 0F       70        01320  CLR IOBUFF ;CLEAR I/O BUFFER
00B2 BD       A39A      01330  JSR INPUT ;EVOKE ROM INPUT
00B5 33       8D FF47   01340  LEAU ASKU,PCR ;GET PROMPT
00B9 17       0298      01350  LBSR SENPRO ;SEND IT
00BC 8D       61        01360 GET3 BSR GKEY ;SEEK KEYPRESS
00BE 27       FC        01370  BEQ GET3 ;CONT TILL PRESSED
00C0 17       0504      01380  LBSR MAKCAP ;CONVERT TO CAPS
00C3 34       02        01390  PSHS A ;SAVE THE KEYPRESS
00C5 17       02A1      01400  LBSR BKUP ;ERASE PROMPT
00C8 35       02        01410  PULS A ;GET THE KEYPRESS
00CA E6       8C 84     01420  LDB BUFCNT,PCR ;GET #CHARS
00CD 81       47        01430  CMPA #'G ;IS STRING GOOD?
00CF 27       2A        01440  BEQ GIVVAR ;YES,SEND TO BASIC
00D1 AE       8D FF6D   01450  LDX TMPX,PCR ;GET X REGISTER
00D5 EE       8D FF65   01460  LDU OLDVID,PCR ;GET OLD VIDEO
00D9 1183 05E0          01470  CMPU #SCROPO ;SCREEN SCROLL?
```

```
00DD 25    03        01480  BLO GET4 ;NO,IT DID NOT
00DF 33    C8 E0     01490  LEAU -32,U ;BACK UP 1 LINE
00E2 DF    88        01500 GET4 STU VIDPOS ;SET CUR VIDEO
00E4 81    41        01510  CMPA #'A ;EDIT ORIGINAL AGAIN?
00E6 27    02        01520  BEQ GETORG ;YES,GET ORIGINAL
00E8 20    C4        01530  BRA GET2 ;EDIT CURRENT STRING
00EA E6    8D FF63   01540 GETORG LDB BUFCNT,PCR ;GET BUFF
00EE 5A              01550  DECB  ;ADJUST TO TRUE LENGTH
00EF 5D              01560  TSTB  ;LENGTH=0?
00F0 27    9D        01570  BEQ GETVAR ;YES,CAN'T ERASE
00F2 E7    8D FF5C   01580  STB BKUCNT,PCR ;SET COUNTER
00F6 17    0270      01590  LBSR BKUP ;ERASE THE STRING
00F9 20    94        01600  BRA GETVAR ;GET THE ORIGINAL
00FB 8E    02DD      01610 GIVVAR LDX #BSTART ;BUFF START
00FE 10AE 8D FF4C    01620  LDY VLOC,PCR ;BASIC VARPTR
0103 E7    8D FF47   01630  STB VARLEN,PCR ;SET LENGTH
0107 C6    01        01640  LDB #1 ;SET COUNTER
0109 E1    8D FF41   01650  CMPB VARLEN,PCR ;ANY CHARS?
010D 27    0B        01660  BEQ NTG ;IF NOT,NONE TO GIVE
010F A6    80        01670 GIV1 LDA ,X+ ;GET CHAR
0111 A7    A0        01680  STA ,Y+ ;PUT INTO VARIABLE
0113 5C              01690  INCB  ;UPDATE COUNTER
0114 E1    8D FF36   01700  CMPB VARLEN,PCR ;ALL SENT?
0118 25    F5        01710  BLO GIV1 ;CONTINUE
011A 6A    8D FF30   01720 NTG DEC VARLEN,PCR ;TRUE SIZE
011E 39              01730  RTS  ;RETURN TO BASIC PROGRAM
                     01740 *
                     01750 *KEYSCAN ROUTINES
                     01760 *
011F 6D    8D FF23   01770 GKEY TST EXCHAN,PCR ;EXCHANGE?
0123 26    14        01780  BNE GKEY2 ;NO ERASE,NO FLASH
0125 6D    8D FF22   01790  TST REPEAT,PCR ;IN REPEAT?
0129 26    07        01800  BNE GKEY1 ;ALLOW ERASE
012B 34    10        01810  PSHS X ;SAVE X
012D BD    A199      01820  JSR FLASH ;FLASH CURSOR
0130 35    10        01830  PULS X ;GET X
0132 8D    05        01840 GKEY1 BSR GKEY2 ;SEEK KEY
0134 27    02        01850  BEQ KEPCUR ;IF=0,KEEP CURSOR
0136 8D    13        01860  BSR ERCUR ;ERASE CURSOR
0138 39              01870 KEPCUR RTS  ;RETURN KEYPRESS
0139 34    20        01880 GKEY2 PSHS Y ;SAVE Y
013B AD    9F A000   01890  JSR >[POLCAT] ;SEEK KEYPRESS
013F 35    20        01900  PULS Y ;RESTORE Y
0141 39              01910  RTS  ;RETURN KEYSCAN CONDITION
                     01920 *
                     01930 *SPECIAL CURSOR SEND/ERASE
                     01940 *
0142 E6    8D FEF2   01950 SENCUR LDB EXCCUR,PCR ;GET CURS
0146 E7    9F 0088   01960 SENCC STB [VIDPOS] ;ON SCREEN
014A 39              01970  RTS  ;RETURN
014B C6    60        01980 ERCUR LDB #96 ;GET SCREEN BLANK
014D 20    F7        01990  BRA SENCC ;ERASE CURSOR
                     02000 *
                     02010 *AUTO KEY REPEAT ROUTINE
                     02020 *
014F A6    8D FEF8   02030 TRYREP LDA REPEAT,PCR ;CHECK IT
0153 A1    8D FEDF   02040  CMPA REPKEY,PCR ;REPEAT ON?
0157 26    19        02050  BNE TR3 ;NO MATCH=NO REPEAT
0159 A6    8D FEED   02060  LDA CURKEY,PCR ;FETCH KEYVALUE
015D 27    13        02070  BEQ TR3 ;IF NULL,REJECT IT
015F 8D    E1        02080  BSR SENCUR ;SEND CURSOR
0161 5F              02090  CLRB  ;SET REPEAT TIMER
0162 8D    BB        02100 TR2 BSR GKEY ;SEEK KEYPRESS
0164 26    0C        02110  BNE TR3 ;IF PRESSED,REPEAT OFF
0166 5C              02120  INCB  ;UPDATE TIMER
0167 E1    8D FED0   02130  CMPB RSPEED,PCR ;TIME ELAPSED?
016B 25    F5        02140  BLO TR2 ;LOOP RSPEED TIMES
016D A6    8D FED9   02150  LDA CURKEY,PCR ;GET KEYVALUE
0171 39              02160  RTS  ;SEND KEYVALUE
0172 6F    8D FED5   02170 TR3 CLR REPEAT,PCR ;STOP REPEAT
0176 8D    A7        02180  BSR GKEY ;SEEK NEW KEYPRESS
0178 27    0A        02190  BEQ TR4 ;IF NO KEY,RETURN
017A A1    8D FEB8   02200  CMPA REPKEY,PCR ;START REPEAT?
017E 27    05        02210  BEQ TR5 ;YES,TRY IT
0180 A7    8D FEC6   02220  STA CURKEY,PCR ;NEW KEYVALUE
0184 39              02230 TR4 RTS  ;SEND KEYPRESS VALUE
0185 8D    4B        02240 TR5 BSR CHKMAS ;REPEAT MASKED?
0187 26    E9        02250  BNE TR3 ;IF SO,SEEK ANOTHER
0189 A7    8D FEBE   02260  STA REPEAT,PCR ;REPEAT ON
018D 20    C0        02270  BRA TRYREP ;REACTIVATE LOOP
                     02280 *
                     02290 *IF IN STANDARD KEYBOARD INPUT
                     02300 *MODE, PROCESS INPUT VALUES
                     02310 *
018F 0D    6F        02320 CHECK TST DEVNUM ;DEVICE=0?
0191 1026 0467       02330  LBNE RETBAS ;IF NOT,ABORT
0195 0F    70        02340  CLR IOBUFF ;CLR I/O BUFF
0197 6F    8D FEAB   02350  CLR EXCHAN,PCR ;NO EXCHANGE
019B DE    88        02360  LDU VIDPOS ;GET VIDEO POSITION
019D EF    8D FE9D   02370  STU OLDVID,PCR ;SAVE FOR LATER
01A1 E7    8D FEAC   02380  STB BUFCNT,PCR ;SAVE CHR COUNT
01A5 6D    8D FEA4   02390  TST LTBCNT,PCR ;LEFT TAB?
01A9 1026 00F3       02400  LBNE SALTAB ;YES,SATISFY IT
```

```
01AD 6D    8D FE9B   02410  TST RTBCNT,PCR ;RIGHT TAB?
01B1 1026 00F3       02420  LBNE SARTAB ;YES,SATISFY IT
01B5 AC    8D FE89   02430  CMPX TMPX,PCR ;CURSOR FROZEN?
01B9 26    04        02440  BNE CHECK1 ;NO,ALLOW REPEAT
01BB 6F    8D FE8B   02450  CLR CURKEY,PCR ;REPEAT OFF
01BF 8D    8E        02460 CHECK1 BSR TRYREP ;KEYSCAN
01C1 27    FC        02470  BEQ CHECK1 ;CONT TILL KEYPRESS
01C3 8D    86        02480  BSR ERCUR ;ERASE CURSOR
01C5 8D    0B        02490  BSR CHKMAS ;IS KEY MASKED?
01C7 27    1C        02500  BEQ CHFCON ;IF=0,NOT MASKED
01C9 A1    8D FE68   02510  CMPA CONKEY,PCR ;CNTRL MASKED?
01CD 27    16        02520  BEQ CHFCON ;ALLOW UNMASK
01CF 16    00E6      02530  LBRA HIDKEY ;HIDE MASKED KEY
                     02540 *
                     02550 *SEE IF KEYPRESS IS MASKED
                     02560 *
01D2 33    8D FE7D   02570 CHKMAS LEAU MASK,PCR ;GET TABLE
01D6 E6    C4        02580 CHKMA1 LDB ,U ;GET MASK VALUE
01D8 C1    FF        02590  CMPB #255 ;AT END OF LIST?
01DA 27    06        02600  BEQ NOMSK ;NO MASK FOUND
01DC A1    C0        02610  CMPA ,U+ ;CHECK FOR MATCH
01DE 27    03        02620  BEQ MASCHK ;THE KEY IS MASKED
01E0 20    F4        02630  BRA CHKMA1 ;CHECK EACH LOC
01E2 5F              02640 NOMSK CLRB ;SET NO MASK COND
01E3 5D              02650 MASCHK TSTB  ;SET CC
01E4 39              02660  RTS  ;RETURN RESULTS
                     02670 *
                     02680 *PROCESS THE KEYPRESS
                     02690 *
01E5 A1    8D FE4C   02700 CHFCON CMPA CONKEY,PCR ;CNTRL?
01E9 27    26        02710  BEQ PCKEY ;PROCESS CONTROL KEY
01EB 81    09        02720  CMPA #9 ;RIGHT ARROW?
01ED 26    02        02730  BNE CHF1 ;IF NOT,DON'T CONVERT
01EF 86    20        02740  LDA #32 ;CONVERT TO BLANK
01F1 17    03DA      02750 CHF1 LBSR GNCHRS ;GET #OF CHARS
01F4 E1    8D FE3C   02760  CMPB BUFLIM,PCR ;AT LIMIT?
01F8 1025 00BD       02770  LBLO SENKEY ;WE HAVE ROOM
01FC E1    8D FE51   02780  CMPB BUFCNT,PCR ;IN LINE EDIT?
0200 27    06        02790  BEQ CHF2 ;IF COUNTS MATCH,NO
0202 81    08        02800  CMPA #8 ;IS IT BACKSPACE?
0204 1026 00B0       02810  LBNE HIDKEY ;MUST BE BACKSPACE
0208 81    20        02820 CHF2 CMPA #32 ;ADD TO BUFFER?
020A 1025 00AB       02830  LBLO SENKEY ;IF NOT, SEND IT
020E 16    00A7      02840  LBRA HIDKEY ;NO ROOM,HIDE IT
                     02850 *
                     02860 *PROCESS CONTROL KEY REQUEST
                     02870 *
0211 6F    8D 03EA   02880 PCKEY CLR BOTTOM,PCR ;SET COUNT
0215 E6    8D FE1E   02890  LDB CONCUR,PCR ;CONTROL CURSOR
0219 17    FF2A      02900  LBSR SENCC ;SEND IT
021C 17    03D1      02910 GNK1 LBSR TIMER ;UPDATE TIMER
021F 27    06        02920  BEQ ECURS ;TIME TO ERASE
0221 C1    FF        02930  CMPB #255 ;TIME FOR CHANGE?
0223 27    EC        02940  BEQ PCKEY ;YES,START OVER
0225 20    03        02950  BRA GNK2 ;SEEK KEYPRESS
0227 17    FF21      02960 ECURS LBSR ERCUR ;ERASE CURSOR
022A 17    FF05      02970 GNK2 LBSR GKEY1 ;SEEK KEY
022D 27    ED        02980  BEQ GNK1 ;CONT TILL KEYPRESS
022F 6F    8D FE17   02990  CLR CURKEY,PCR ;REPEAT OFF
0233 17    0391      03000  LBSR MAKCAP ;CONVERT TO CAPS
0236 81    55        03010  CMPA #'U ;UNMASK?
0238 1027 0094       03020  LBEQ UNSMSK ;IF SO, ALLOW IT
023C 8D    94        03030  BSR CHKMAS ;CHECK FOR MASK
023E 27    02        03040  BEQ GCVAL ;IF=0,NOT MASKED
0240 20    76        03050  BRA HIDKEY ;HIDE MASKED KEY
0242 34    02        03060 GCVAL PSHS A ;SAVE THE KEYPRESS
0244 A6    8D FDED   03070  LDA CONKEY,PCR ;GET CNTRL KEY
0248 8D    88        03080  BSR CHKMAS ;TEST FOR MASK
024A 35    02        03090  PULS A ;GET THE KEYPRESS
024C 27    02        03100  BEQ FULOPT ;ALLOW FULL OPTIONS
024E 20    68        03110  BRA HIDKEY ;CONTROL WAS MASKED
                     03120 *
                     03130 *PROCESS CONTROL KEY OPTIONS
                     03140 *
0250 81    4D        03150 FULOPT CMPA #'M ;MASK A KEY?
0252 1027 009B       03160  LBEQ SETMAS ;SET MASK VALUE
0256 81    58        03170  CMPA #'X ;EXCHANGE REQUEST?
0258 1027 01C4       03180  LBEQ EXCHAR ;TRY EXCHANGE CHAR
025C 33    8D FDA7   03190  LEAU BUFPRO,PCR ;GET PROMPT
0260 81    42        03200  CMPA #'B ;SET BUFFER SIZE?
0262 1027 00B4       03210  LBEQ SETBUF ;YES
0266 33    8D FDA3   03220  LEAU LTBPRO,PCR ;GET PROMPT
026A 81    4C        03230  CMPA #'L ;SET LEFT TAB?
026C 1027 00B9       03240  LBEQ SETLTB ;YES
0270 33    8D FDA7   03250  LEAU RTBPRO,PCR ;GET PROMPT
0274 81    52        03260  CMPA #'R ;SET RIGHT TAB?
0276 1027 00B7       03270  LBEQ SETRTB ;YES
027A 33    8D FDA4   03280  LEAU RSPPRO,PCR ;GET PROMPT
027E 81    53        03290  CMPA #'S ;SET REPEAT SPEED?
0280 1027 00B6       03300  LBEQ SETREP ;YES
0284 81    08        03310  CMPA #8 ;SEND A LEFT TAB?
0286 26    0A        03320  BNE CFRTAB ;NO,CHECK FOR RIGHT
0288 E6    8D FDAD   03330  LDB LTBSIZ,PCR ;GET LEFT TAB
```

```
028C E7    8D FDBD  03340  STB LTBCNT,PCR ;SET COUNT
0290 20    0E       03350  BRA SALTAB ;SEND LEFT TAB
0292 81    09       03360 CFRTAB CMPA #9 ;SEND RIGHT TAB?
0294 26    22       03370  BNE HIDKEY ;HIDE UNUSABLE KEY
0296 E6    8D FDA0  03380  LDB RTBSIZ,PCR ;GET RIGHT TAB
029A E7    8D FDAE  03390  STB RTBCNT,PCR ;SET COUNT
029E 20    08       03400  BRA SARTAB ;SEND RIGHT TAB
02A0 86    08       03410 SALTAB LDA #8 ;GET BACKSPACE
02A2 6A    8D FDA7  03420  DFC LTBCNT,PCR ;CNT=CNT-1
02A6 20    11       03430  BRA SENKEY ;SEND THE BACKSPACE
02A8 86    20       03440 SARTAB LDA #32 ;GET BLANK
02AA 6A    8D FD9E  03450  DEC RTBCNT,PCR ;CNT=CNT-1
02AE 17    031D     03460  LBSR GNCHRS ;GET #OF CHARS
02B1 5C             03470  INCB  ;UPDATE TAB COUNTER
02B2 E1    8D FD7E  03480  CMPB BUFLIM,PCR ;AT LIMIT?
02B6 23    01       03490  BLS SENKEY ;SEND THE BLANK
                    03500 *
                    03510 *SEND VALUES TO THE ROM ROUTINE
                    03520 *
02B8 4F             03530 HIDKEY CLRA ;HIDE THE KEY
02B9 E6    8D FD94  03540 SENKEY LDB BUFCNT,PCR ;GET CNT
02BD 81    1F       03550  CMPA #31 ;REPEATABLE KEY?
02BF 22    08       03560  BHI SENK1 ;YES,PRESERVE CURKEY
02C1 81    08       03570  CMPA #8 ;REPEATABLE KEY?
02C3 27    04       03580  BEQ SENK1 ;YES,PRESERVE CURKEY
02C5 6F    8D FD81  03590  CLR CURKEY,PCR ;NO REPEAT
02C9 AF    8D FD75  03600 SENK1 STX TMPX,PCR ;COPY X
02CD 32    64       03610  LEAS 4,S ;CLEAR 2 RTS'S
02CF 39             03620  RTS  ;MAKE BASIC PROCESS KEY
                    03630 *
                    03640 *UNMASK A KEYBOARD CHAR
                    03650 *
02D0 33    8D FD56  03660 UNSMSK LEAU UNMPRO,PCR ;PROMPT
02D4 8D    7E       03670  BSR SENPRO ;SEND THE PROMPT
02D6 17    FE46     03680 UNM1 LBSR GKEY ;GET KEY
02D9 27    FB       03690  BEQ UNM1 ;MUST HAVE KEY
02DB 33    8D FD74  03700  LEAU MASK,PCR ;GET TABLE START
02DF E6    C4       03710 FINMAS LDB ,U ;GET MASK VALUE
02E1 C1    FF       03720  CMPB #255 ;AT LIST END?
02E3 27    2D       03730  BEQ MASDON ;UNMASK COMPLETE
02E5 A1    C4       03740  CMPA ,U ;MASK MATCH?
02E7 27    04       03750  BEQ FOUMSK ;IF SO,UNMASK IT
02E9 33    41       03760  LEAU 1,U ;NEXT MASK POSITION
02EB 20    F2       03770  BRA FINMAS ;CHECK ALL LOCS
02ED 6F    C4       03780 FOUMSK CLR ,U ;UNMASK THE KEY
02EF 20    21       03790  BRA MASDON ;UNMASK IS DONE
                    03800 *
                    03810 *MASK A KEYBOARD CHAR, IF ROOM
                    03820 *
02F1 33    8D FD1F  03830 SETMAS LEAU MASPRO,PCR ;PROMPT
02F5 8D    5D       03840  BSR SENPRO ;SEND PROMPT
02F7 17    FE25     03850 SET1 LBSR GKEY ;SEEK MASK VALUE
02FA 27    FB       03860  BEQ SET1 ;MUST HAVE KEY
02FC 33    8D FD53  03870  LEAU MASK,PCR ;GET TABLE START
0300 E6    C4       03880 FINFRE LDB ,U ;SEEK FREE BYTE
0302 C1    FF       03890  CMPB #255 ;AT LIST END?
0304 27    0C       03900  BEQ MASDON ;NO MORE ROOM
0306 A1    C4       03910  CMPA ,U ;ALREADY MASKED?
0308 27    08       03920  BEQ MASDON ;IF YES,WE'RE DONE
030A 6D    C4       03930  TST ,U ;FREE BYTE?
030C 27    08       03940  BEQ GOTFRE ;IF SO,USE IT
030E 33    41       03950  LEAU 1,U ;NEXT MASK POSITION

0310 20    EE       03960  BRA FINFRE ;CONTINUE ATTEMPT
0312 8D    55       03970 MASDON BSR BKUP ;REMOVE PROMPT
0314 20    A2       03980  BRA HIDKEY ;HIDE CURRENT KEY
0316 A7    C0       03990 GOTFRE STA ,U+ ;SET THE MASK
0318 20    C5       04000  BRA FINMAS ;NO DUPLICATIONS
                    04010 *
                    04020 *SET NEW BUFFER LIMIT
                    04030 *
031A 8D    5A       04040 SETBUF BSR GN0 ;GET BUFFER LIM
031C E1    8D FD13  04050  CMPB MAXBUF,PCR ;AT MAX?
0320 27    01       04060  BEQ SETB1 ;YES,CAN'T ADJUST
0322 5C             04070  INCB  ;EXPAND TO TRUE VALUE
0323 E7    8D FD0D  04080 SETB1 STB BUFLIM,PCR ;SAVE IT
0327 20    8F       04090  BRA HIDKEY ;HIDE THE KEYPRESS
                    04100 *
                    04110 *SET NEW LEFT TAB
                    04120 *
0329 8D    4B       04130 SETLTB BSR GN0 ;GET LEFT TAB
032B E7    8D FD0A  04140  STB LTBSIZ,PCR ;SAVE IT
032F 20    87       04150  BRA HIDKEY ;HIDE THE KEYPRESS
                    04160 *
                    04170 *SET NEW RIGHT TAB
                    04180 *
0331 8D    43       04190 SETRTB BSR GN0 ;GET RIGHT TAB
0333 E7    8D FD03  04200  STB RTBSIZ,PCR ;SAVE IT
0337 16    FF7E     04210  LBRA HIDKEY ;HIDE THE KEYPRESS
                    04220 *
                    04230 *SET NEW REPEAT SPEED
                    04240 *
033A 8D    3A       04250 SETREP BSR GN0 ;GET NEW SPEED
033C E7    8D FCFB  04260  STB RSPEED,PCR ;SAVE IT
```

```
0340 16    FF75     04270  LBRA HIDKEY ;HIDE THE KEYPRESS
                    04280 *
                    04290 *ROUTINE TO CLEAR OLD VALUES
                    04300 *
0343 6F    8D FD00  04310 CLRVAL CLR UNITS,PCR ;NO UNITS
0347 6F    8D FCFD  04320  CLR TENS,PCR ;NO TENS
034B 6F    8D FCFA  04330  CLR HUNS,PCR ;NO HUNDREDS
034F 6F    8D FCEA  04340  CLR CURVAL,PCR ;VALUE=0
0353 39             04350  RTS  ;RETURN ZERO VALUES
                    04360 *
                    04370 *ROUTINE TO SEND PROMPTS
                    04380 *
0354 6F    8D FCFA  04390 SENPRO CLR BKUCNT,PCR ;SET TO 0
0358 A6    C0       04400 SEN1 LDA ,U+ ;GET CHAR
035A 81    FF       04410  CMPA #255 ;END OF PROMPT?
035C 27    0A       04420  BEQ SEN2 ;IF YES,NO MORE CHARS
035E AD    9F A002  04430  JSR >[CHROUT] ;SEND TO SCREEN
0362 6C    8D FCEC  04440  INC BKUCNT,PCR ;UPDATE COUNTER
0366 20    F0       04450  BRA SEN1 ;SEND ALL CHARS
0368 39             04460 SEN2 RTS  ;RETURN
                    04470 *
                    04480 *ROUTINE TO ERASE PROMPTS
                    04490 *
0369 86    08       04500 BKUP LDA #8 ;GET ERASE CHAR
036B AD    9F A002  04510 BK1 JSR >[CHROUT] ;ERASE A CHAR
036F 6A    8D FCDF  04520  DEC BKUCNT,PCR ;DECREASE COUNT
0373 26    F6       04530  BNE BK1 ;CONTINUE TILL 0
0375 39             04540  RTS  ;RETURN
                    04550 *
                    04560 *PROCESS SET VALUE REQUESTS
                    04570 *
0376 8D    DC       04580 GN0 BSR SENPRO ;SEND PROMPT
0378 8D    C9       04590  BSR CLRVAL ;RESET VALUES
037A C6    03       04600  LDB #3 ;GET MAX KEYPRESS COUNT
037C E7    8D 027F  04610  STB BOTTOM,PCR ;SET IT
0380 17    FD9C     04620 GN1 LBSR GKEY ;SEEK KEYPRESS
0383 27    FB       04630  BEQ GN1 ;UNTIL PRESSED
0385 81    39       04640  CMPA #'9 ;A DIGIT?
0387 22    1A       04650  BHI GN2 ;TOO BIG
0389 81    30       04660  CMPA #'0 ;A DIGIT?
038B 25    16       04670  BLO GN2 ;TOO SMALL
038D 6D    8D 026E  04680  TST BOTTOM,PCR ;AT DIGIT LIM?
0391 27    10       04690  BEQ GN2 ;3 DIGITS ENTERED
0393 6A    8D 0268  04700  DEC BOTTOM,PCR ;NEW DIGIT CNT
0397 AD    9F A002  04710  JSR >[CHROUT] ;SEND THE DIGIT
039B 6C    8D FCB3  04720  INC BKUCNT,PCR ;UPDATE COUNTER
039F 8D    29       04730  BSR GVAL ;UPDATE VALUE
03A1 20    DD       04740  BRA GN1 ;SEEK ANOTHER DIGIT
03A3 81    03       04750 GN2 CMPA #3 ;ABORT WITH BREAK?
03A5 27    1C       04760  BEQ GN3 ;YES,RETAIN PREV VALS
03A7 81    0D       04770  CMPA #13 ;RETURN REQUEST?
03A9 26    D5       04780  BNE GN1 ;IF NOT,CONTINUE
03AB 8D    BC       04790  BSR BKUP ;ERASE PROMPT
03AD E6    8D FC8C  04800  LDB CURVAL,PCR ;CURRENT VALUE
03B1 E1    8D FC87  04810  CMPB MINVAL,PCR ;IS IT>=MIN?
03B5 24    02       04820  BHS GODVAL ;GOOD VALUE
03B7 20    0C       04830  BRA GN4 ;RETAIN PREVIOUS VALS
03B9 E1    8D FC76  04840 GODVAL CMPB MAXBUF,PCR ;<=MAX?
03BD 23    03       04850  BLS ATMVAL ;YES,VALUE IS OK
03BF 5A             04860  DECB  ;ADJUST TO WITHIN RANGE
03C0 20    F7       04870  BRA GODVAL ;CONTINUE TILL GOOD
03C2 39             04880 ATMVAL RTS  ;RETURN VALUE IN B
03C3 8D    A4       04890 GN3 BSR BKUP ;ERASE PROMPT
03C5 32    62       04900 GN4 LEAS 2,S ;REMOVE 1 RTS
03C7 16    FEEE     04910  LBRA HIDKEY ;HIDE THE KEYPRESS
                    04920 *
                    04930 *COMPUTE VALUE FOR SET REQUEST
                    04940 *
03CA 80    30       04950 GVAL SUBA #48 ;MAKE INTO NUMBER
03CC 1F    89       04960  TFR A,B ;SAVE A REGISTER
03CE 6F    8D FC6B  04970  CLR CURVAL,PCR ;VALUE=0
03D2 A6    8D FC72  04980  LDA TENS,PCR ;GET TENS
03D6 A7    8D FC6F  04990  STA HUNS,PCR ;HUNS=TENS
03DA A6    8D FC69  05000  LDA UNITS,PCR ;GET UNITS
03DE A7    8D FC66  05010  STA TENS,PCR ;TENS=UNITS
03E2 E7    8D FC61  05020  STB UNITS,PCR ;SET NEW UNITS
03E6 E7    8D FC53  05030  STB CURVAL,PCR ;SAVE UNITS
03EA A6    8D FC5A  05040  LDA TENS,PCR ;GET # OF TENS
03EE C6    0A       05050  LDB #10 ;TEN MULTIPLIER
03F0 3D             05060  MUL  ;COMPUTE TENS
03F1 EB    8D FC48  05070  ADDB CURVAL,PCR ;ADD TO UNITS
03F5 E7    8D FC44  05080  STB CURVAL,PCR ;UPDATE VALUE
03F9 A6    8D FC4C  05090  LDA HUNS,PCR ;GET HUNDREDS
03FD 81    02       05100  CMPA #2 ;HOW MANY?
03FF 22    12       05110  BHI SATMAX ;MAX OF 2 HUNDREDS
0401 25    04       05120  BLO GHUNS ;IF < 2,IT'S OK
0403 C1    37       05130  CMPB #55 ;VALUE BE > 255?
0405 22    0C       05140  BHI SATMAX ;DON'T ALLOW IT
0407 C6    64       05150 GHUNS LDB #100 ;HUNDRED MUL
0409 3D             05160  MUL  ;COMPUTE HUNDREDS
040A EB    8D FC2F  05170  ADDB CURVAL,PCR ;ADD TO VALUE
040E E7    8D FC2B  05180  STB CURVAL,PCR ;NEW VALUE
0412 39             05190  RTS  ;RETURN WITH VALUE
```

```
0413 E6    8D FC1C 05200 SATMAX LDB MAXBUF,PCR ;GET MAX
0417 E7    8D FC22 05210  STB CURVAL,PCR ;VALUE=MAXIMUM
041B E6    8D FC1E 05220 MXSVAL LDB CURVAL,PCR ;B=VALUE
041F 39            05230  RTS  ;RETURN THE NUMBER
                   05240 *
                   05250 *BUFFER CHAR EXCHANGE ROUTINE
                   05260 *
0420 6C    8D FC22 05270 EXCHAR INC EXCHAN,PCR ;FLAG IT
0424 17    01A7    05280  LBSR GNCHRS ;GET #CHARS
0427 6F    84      05290  CLR ,X ;CLEAR END OF LINE AND
0429 6F    01      05300  CLR 1,X ;END OF LINE + 1
042B 7D    02DD    05310  TST BSTART ;ANYTHING TO EDIT?
042E 1027 FE86     05320  LBEQ HIDKEY ;NO,BUFFER EMPTY
0432 E7    8D FC1B 05330  STB BUFCNT,PCR ;SAVE COUNTER
0436 AF    8D FC06 05340  STX EOBUF,PCR ;SET END OF BUF
043A 34    20      05350  PSHS Y ;SAVE Y
043C 30    1F      05360  LEAX -1,X ;POINT TO LAST CHAR
043E 109E 88       05370  LDY VIDPOS ;GET VIDEO POS
0441 10AF 8D FBF8  05380  STY OLDVID,PCR ;SAVE IT
0446 31    3F      05390  LEAY -1,Y ;POINT TO LAST CHAR
0448 109F 88       05400  STY VIDPOS ;SET TEMP VIDEO POS
044B 6F    8D FBFB 05410 EX0 CLR CURKEY,PCR ;STOP REPEAT
044F 6F    8D 01AC 05420 EX1 CLR BOTTOM,PCR ;SET COUNT
0453 17    FCEC    05430  LBSR SENCUR ;SEND THE CURSOR
0456 17    0197    05440 EX2 LBSR TIMER ;UPDATE TIMER
0459 27    06      05450  BEQ EX3 ;SEND CURRENT CHAR
045B C1    FF      05460  CMPB #255 ;TIME FOR CHANGE?
045D 27    F0      05470  BEQ EX1 ;RESET COUNTER
045F 20    03      05480  BRA ED4 ;SEEK KEYPRESS
0461 17    014B    05490 EX3 LBSR PUTSCR ;CHAR ON SCREEN
0464 17    FCE8    05500 ED4 LBSR TRYREP ;ALLOW REPEAT
0467 27    ED      05510  BEQ EX2 ;MUST HAVE KEYPRESS
0469 17    0143    05520  LBSR PUTSCR ;CHAR ON SCREEN
046C AF    8D FBD2 05530  STX TMPX,PCR ;SAVE X REGISTER
0470 81    08      05540  CMPA #8 ;BACK UP?
0472 27    16      05550  BEQ BKUP1 ;YES,BACK UP 1
0474 81    09      05560  CMPA #9 ;MOVE FORWARD?
0476 27    23      05570  BEQ FO1 ;YES,MOVE FORWARD 1
0478 81    0C      05580  CMPA #12 ;DELETE A CHAR?
047A 1027 00AE     05590  LBEQ TRYDEL ;TRY TO DELETE
047E 81    0D      05600  CMPA #13 ;DONE?
0480 1027 0106     05610  LBEQ EXDONE ;YES,EXCHANGE DONE
0484 81    20      05620  CMPA #32 ;PRINTABLE?
0486 24    26      05630  BHS TRYINS ;YES,TRY TO INSERT
0488 20    C1      05640  BRA EX0 ;INVALID COMMAND
                   05650 *
                   05660 *MOVE CURSOR TO LEFT
                   05670 *
048A 8C    02DD    05680 BKUP1 CMPX #BSTART ;BUFF START?
048D 27    BC      05690  BEQ EX0 ;IF YES,LEFT JUSTIFIED
048F 30    1F      05700  LEAX -1,X ;BACK UP X
0491 109E 88       05710  LDY VIDPOS ;GET VIDEO POSITION
0494 31    3F      05720  LEAY -1,Y ;BACK IT UP
0496 109F 88       05730  STY VIDPOS ;UPDATE IT
0499 20    B4      05740  BRA EX1 ;CONTINUE
                   05750 *
                   05760 *MOVE CURSOR TO RIGHT
                   05770 *
049B 109E 88       05780 FO1 LDY VIDPOS ;GET VIDEO POS
049E 31    21      05790  LEAY 1,Y ;POINT TO NEXT POS
04A0 10AC 8D FB99  05800  CMPY OLDVID,PCR ;AT END?
04A5 27    A4      05810  BEQ EX0 ;YES,REJECT
04A7 109F 88       05820  STY VIDPOS ;NEW VIDEO POS
04AA 30    01      05830  LEAX 1,X ;NEW BUFFER POINTER
04AC 20    A1      05840  BRA EX1 ;CONTINUE
                   05850 *
                   05860 *INSERT A CHAR
                   05870 *
04AE E6    8D FB9F 05880 TRYINS LDB BUFCNT,PCR ;GET CHRS
04B2 E1    8D FB7E 05890  CMPB BUFLIM,PCR ;ANY ROOM?
04B6 24    93      05900  BHS EX0 ;NO,REJECT
04B8 10AE 8D FB83  05910  LDY EOBUF,PCR ;GET END OF BUF
04BD E6    A2      05920 MOVINB LDB ,-Y ;GET LEFT CHAR
04BF E7    21      05930  STB 1,Y ;PUT IN CURR BUF POS
04C1 10AC 8D FB7C  05940  CMPY TMPX,PCR ;Y=X?
04C6 22    F5      05950  BHI MOVINB ;REPEAT TILL Y=X
04C8 A7    84      05960  STA ,X ;INSERT THE CHAR
04CA 10AE 8D FB71  05970  LDY EOBUF,PCR ;GET END OF BUF
04CF 31    21      05980  LEAY 1,Y ;UPDATE IT
04D1 10AF 8D FB6A  05990  STY EOBUF,PCR ;SAVE IT
04D6 6F    A4      06000  CLR ,Y ;SHOW END OF LINE
04D8 109E 88       06010  LDY VIDPOS ;GET VIDEO POSITION
04DB 31    21      06020  LEAY 1,Y ;UPDATE IT
04DD 10AF 8D FB62  06030  STY CURPOS,PCR ;SAVE IT
04E2 1F    12      06040  TFR X,Y ;GIVE X TO Y
04E4 A6    A0      06050 MOVONS LDA ,Y+ ;GET A CHAR
04E6 27    23      06060  BEQ ALLMOV ;IF=0,ALL MOVED
04E8 9E    88      06070  LDX VIDPOS ;GET VIDEO POSITION
04EA 8C    05FF    06080  CMPX #SCREND ;AT SCREEN END
04ED 25    16      06090  BLO WNSCR ;CHROUT WON'T SCROLL
04EF AE    8D FB4B 06100  LDX OLDVID,PCR ;GET OLD VIDEO
04F3 30    88 E0   06110  LEAX -32,X ;BACK UP 1 LINE
04F6 AF    8D FB44 06120  STX OLDVID,PCR ;SAVE IT
```

```
04FA AE    8D FB46 06130  LDX CURPOS,PCR ;GET CUR POS
04FE 30    88 E0   06140  LEAX -32,X ;BACK UP 1 LINE
0501 AF    8D FB3F 06150  STX CURPOS,PCR ;SAVE IT
0505 AD    9F A002 06160 WNSCR JSR >[CHROUT] ;TO SCREEN
0509 20    D9      06170  BRA MOVONS ;MOVE THEM ALL
050B 10AE 8D FB34  06180 ALLMOV LDY CURPOS,PCR ;CURS POS
0510 109F 88       06190  STY VIDPOS ;SET VIDEO POSITION
0513 10AE 8D FB26  06200  LDY OLDVID,PCR ;GET OLD VIDEO
0518 31    21      06210  LEAY 1,Y ;UPDATE IT
051A 10AF 8D FB1F  06220  STY OLDVID,PCR ;SAVE IT

051F AE    8D FB1F 06230  LDX TMPX,PCR ;GET REG X
0523 30    01      06240  LEAX 1,X ;NEW POINTER
0525 6C    8D FB28 06250  INC BUFCNT,PCR ;UPDATE COUNT
0529 16    FF23    06260  LBRA EX1 ;CONTINUE
                   06270 *
                   06280 *DELETE A CHAR
                   06290 *
052C 7D    02DE    06300 TRYDEL TST 1+BSTART ;#CHARS?
052F 1027 FF18     06310  LBEQ EX0 ;MUST BE > 1 CHAR
0533 109E 88       06320  LDY VIDPOS ;GET VIDEO POS
0536 E6    01      06330 DELINB LDB 1,X ;GET NEXT CHAR
0538 E7    80      06340  STB ,X+ ;PUT IN CURRENT LOC
053A 5D            06350  TSTB  ;B=0?
053B 26    F9      06360  BNE DELINB ;CONT IF NOT=0
053D AE    8D FB01 06370  LDX TMPX,PCR ;GET POINTER
0541 6D    84      06380  TST ,X ;AT LINE END?
0543 26    04      06390  BNE KEEPOS ;IF NOT,KEEP POS
0545 30    1F      06400  LEAX -1,X ;BACK UP ONE
0547 31    3F      06410  LEAY -1,Y ;BACK UP ONE
0549 109F 88       06420 KEEPOS STY VIDPOS ;UPDATE IT
054C 10AF 8D FAF3  06430  STY CURPOS,PCR ;SAVE IT
0551 1F    12      06440  TFR X,Y ;GIVE TO Y REG
0553 A6    A0      06450 DELONS LDA ,Y+ ;GET A CHAR
0555 27    06      06460  BEQ DOS ;IF=0,SCREEN FIXED
0557 AD    9F A002 06470  JSR >[CHROUT] ;SEND TO SCREEN
055B 20    F6      06480  BRA DELONS ;MOVE ALL CHARS
055D 86    20      06490 DOS LDA #32 ;GET BLANK
055F AD    9F A002 06500  JSR >[CHROUT] ;ERASE LAST CHAR
0563 10AE 8D FADC  06510  LDY CURPOS,PCR ;GET CURSOR POS
0568 109F 88       06520  STY VIDPOS ;SET NEW POSITION
056B 10AE 8D FACE  06530  LDY OLDVID,PCR ;GET OLD VIDEO
0570 31    3F      06540  LEAY -1,Y ;BACK IT UP
0572 10AF 8D FAC7  06550  STY OLDVID,PCR ;SAVE IT
0577 10AE 8D FAC4  06560  LDY EOBUF,PCR ;GET END OF BUF
057C 31    3F      06570  LEAY -1,Y ;BACK IT UP
057E 10AF 8D FABD  06580  STY EOBUF,PCR ;SAVE IT
0583 6A    8D FACA 06590  DEC BUFCNT,PCR ;UPDATE COUNTER
0587 16    FEC5    06600  LBRA EX1 ;CONTINUE
                   06610 *
                   06620 *EXIT EXCHANGE ROUTINE
                   06630 *
058A 10AE 8D FAAF  06640 EXDONE LDY OLDVID,PCR ;OLD VID
058F 109F 88       06650  STY VIDPOS ;RESTORE IT
0592 35    20      06660  PULS Y ;RESTORE Y
0594 AE    8D FAA8 06670  LDX EOBUF,PCR ;GET END OF BUFF
0598 6A    8D FAAA 06680  DEC EXCHAN,PCR ;ADJUST FLAG
059C 1027 FD18     06690  LBEQ HIDKEY ;NOT IN LINE EDIT
05A0 E6    8D FAAD 06700  LDB BUFCNT,PCR ;GET BUFFER CNT
05A4 C0    02      06710  SUBB #2 ;ADJUST FOR LINE EDIT
05A6 D7    D7      06720  STB EDTCNT ;UPDATE EDIT COUNT
05A8 6F    8D FAA5 06730  CLR BUFCNT,PCR ;SET TO ZERO
05AC 16    FD09    06740  LBRA HIDKEY ;HIDE THE KEY
                   06750 *
                   06760 *SHOW CHARACTER DURING EXCHANGE
                   06770 *
05AF E6    84      06780 PUTSCR LDB ,X ;GET CHAR
05B1 8D    05      06790  BSR FIXIT ;CONVERT FOR SCREEN
05B3 E7    9F 0088 06800  STB [VIDPOS] ;PUT ON SCREEN
05B7 39            06810  RTS
                   06820 *
                   06830 *CONVERT FOR SCREEN
                   06840 *
05B8 C1    40      06850 FIXIT CMPB #64 ;SCREEN ADJUST
05BA 25    05      06860  BLO INC64 ;TOO SMALL
05BC C1    61      06870  CMPB #97
05BE 24    04      06880  BHS DEC96 ;TOO BIG
05C0 39            06890  RTS  ;JUST RIGHT
05C1 CB    40      06900 INC64 ADDB #64
05C3 39            06910  RTS
05C4 C0    60      06920 DEC96 SUBB #96
05C6 39            06930  RTS
                   06940 *
                   06950 *CONVERT LOWER TO UPPER CASE
                   06960 *
05C7 81    61      06970 MAKCAP CMPA #97 ;LOWER CASE?
05C9 25    02      06980  BLO ISUPP ;NO,IT'S UPPER
05CB 80    20      06990  SUBA #32 ;CONVERT TO UPPER
05CD 39            07000 ISUPP RTS  ;RETURN USABLE KEY
                   07010 *
                   07020 *SEE IF LINE EDIT IS IN CONTROL
                   07030 *
05CE E6    8D FA7F 07040 GNCHRS LDB BUFCNT,PCR ;GET CNT
05D2 EE    66      07050  LDU 6,S ;GET STACK LOCATION
```

```
05D4 1183 9FFF      07060  CMPU #LEDVEC ;IN LINE EDIT?
05D8 22    09       07070  BHI NLEDIT ;NOT IN LINE EDIT
05DA D6    D7       07080  LDB EDTCNT ;GET EDIT COUNT
05DC 5C             07090  INCB   ;LINE EDIT ADJUST
05DD 6D    8D FA65  07100  TST EXCHAN,PCR ;DESIRE EXCHAN?
05E1 26    01       07110  BNE DOEXCH ;YES,DO EXCHANGE
05E3 39             07120 NLEDIT RTS   ;RETURN CHAR COUNT
05E4 6C    8D FA5E  07130 DOEXCH INC EXCHAN,PCR ;ADJUST
05E8 BD    85B4     07140  JSR GETEND ;GET LINE END
05EB D6    D7       07150  LDB EDTCNT ;GET EDIT COUNT
05ED CB    02       07160  ADDB #2 ;ADJUST FOR EXCHANGE
05EF 39             07170  RTS   ;RETURN COUNT IN B
                    07180 *
                    07190 *SPECIAL CURSOR FLASH TIMER
                    07200 *
05F0 E6    8D 000B  07210 TIMER LDB BOTTOM;PCR ;GET COUNT
05F4 5C             07220  INCB   ;UPDATE IT
05F5 E7    8D 0006  07230  STB BOTTOM,PCR ;SAVE IT
05F9 C1    7F       07240  CMPB #127 ;CHECK CONDITON
05FB 39             07250  RTS   ;RETURN CONDITION
                    07260 *
                    07270 *IF BUFFER STUFFER CANNOT BE
                    07280 *USED, RETURN IS MADE HERE
                    07290 *
05FC       FF       07300 RETBAS FCB 255 ;ALLOW
05FD       FF       07310  FCB 255 ;ROUTINE
05FE       FF       07320  FCB 255 ;DEACTIVATION
05FF       FF       07330 BOTTOM FCB 255 ;OBJECT CODE END
                    07340 *"BOTTOM" IS ALSO USED AS A
                    07350 *COUNTER FOR SEVERAL ROUTINES
           005E     07360  END HOOK
00000 TOTAL ERRORS
```

150	235	740	127
300	88	850	61
400	148	960	198
520	0	END	97
620	121		

Listing 2:

```
10 'OBJECT CODE GENERATOR
20 'BUFFER STUFFER (C) 1984
30 'BY Richard W. Rutter
40 CLEAR500
50 SP=49446:EP=49449'SET ROM ADD
RESSES
60 DEV$="":FORA=SP TOEP:DEV$=DEV
$+CHR$(PEEK(A)):NEXTA:IFDEV$<>"D
ISK"ANDSP<49465THENSP=49465:EP=4
9468:GOTO60'(LOOK FOR DISK 1.0 O
R 1.1)
70 IFDEV$="DISK"THENFI=3541:LA=5
076:EX=3634ELSEFI=1536:LA=3071:E
X=1629:DEV$="CASSETTE"'SET FIRST
 AND LAST ADDRESSES FOR EITHER A
 DISK OR A NON DISK SYSTEM
80 CLS:PRINT"CREATING OBJECT COD
E.":PRINT"PLEASE WAIT."
90 FORA=FI TOLA'USE FREE LOCATIO
NS AS DETERMINED IN LINE 70
100 READB'GET THE DATA VALUE
110 CS=CS+B'UPDATE CHECKSUM
120 POKEA,B'STORE EACH VALUE
130 NEXTA
140 PRINT
150 IFCS=180207THENPRINT"CHECKSU
M IS GOOD."ELSEPRINT"SORRY, CHEC
KSUM IS BAD!":PRINT"EXAMINE YOUR
 DATA STATEMENTS.":GOTO250
160 PRINT"IS 'DEV$' READY (Y/N)?
:";:LINEINPUTQ$:Q$=LEFT$(Q$,1):I
FQ$<>"Y"THEN140
170 PRINT
180 PRINT"SAVING FILE 'BUFBIN'."
:PRINT"PLEASE WAIT."
190 IFDEV$<>"DISK"THEN220
200 SAVEM"BUFBIN.BIN",FI,LA,EX
210 GOTO230
220 CSAVEM"BUFBIN",FI,LA,EX
230 PRINT:PRINT"FILE 'BUFBIN' NO
W ON "DEV$"."
240 'THE FOLLOWING 1536 DATA VAL
UES ARE USED TO CREATE THE OBJEC
T CODE FOR BUFFER STUFFER.  BE C
ERTAIN THAT YOUR DATA IS THE SAM
E AS THIS DATA.
250 END
260 DATA 65,47,67,47,71,58,255,1
28,66,85,70,58,255,128,76,84,65,
66,58,255
270 DATA 128,77,65,83,75,58,255,
128,82,84,65,66,58,255,128,83,80
,69,69,68
280 DATA 58,255,128,85,78,77,65,
83,75,58,255,250,250,10,19,159,2
55,5,5,40
290 DATA 1,1,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,160,0,0,0,0,0,0,0
,0,0,0,0
300 DATA 0,0,255,174,141,5,155,1
40,255,255,38,22,182,1,106,167,1
41,5,142
310 DATA 190,1,107,175,141,5,136
,48,141,1,22,191,1,107,57,174,14
1,5,124,191
320 DATA 1,107,48,141,5,117,134,
255,167,132,167,1,57,16,174,140,
188,142,2
330 DATA 221,95,109,140,180,39,1
4,166,160,167,128,173,159,160,2,
92,225,140
340 DATA 166,37,242,92,231,140,1
63,15,111,15,112,189,163,154,51,
141,255,71
350 DATA 23,2,152,141,97,39,252,
23,5,4,52,2,23,2,161,53,2,230,14
0,132,129
360 DATA 71,39,42,174,141,255,10
9,238,141,255,101,17,131,5,224,3
7,3,51,200
370 DATA 224,223,136,129,65,39,2
,32,196,230,141,255,99,90,93,39,
157,231,141
380 DATA 255,92,23,2,112,32,148,
142,2,221,16,174,141,255,76,231,
141,255,71
390 DATA 198,1,225,141,255,65,39
,11,166,128,167.160.92,225,141,2
55.54,37
400 DATA 245,106,141,255,48,57,1
09,141,255,35,38,20,109,141,255,
34,38,7,52
410 DATA 16,189,161,153,53,16,14
1,5,39,2,141,19,57,52,32,173,159
,160,0,53
420 DATA 32,57,230,141,254,242,2
31,159,0,136,57,198,96,32,247,16
6,141,254
430 DATA 248,161,141,254,223,38,
25,166,141,254,237,39,19,141,225
,95,141,187
440 DATA 38,12,92,225,141,254,20,
8,37,245,166,141,254,217,57,111,
141,254,213
450 DATA 141,167,39,10,161,141,2
54,184,39,5,167,141,254,198,57,1
41,75,38
460 DATA 233,167,141,254,190,32,
192,13,111,16,38,4,103,15,112,11
1,141,254
470 DATA 171,222,136,239,141,254
,157,231,141,254,172,109,141,254
,164,16,38
480 DATA 0,243,109,141,254,155,1
6,38,0,243,172,141,254,137,38,4,
111,141,254
490 DATA 139,141,142,39,252,141,
134,141,11,39,28,161,141,254,104
,39,22,22
500 DATA 0,230,51,141,254,125,23
0,196,193,255,39.6,161,192,39,3,
32,244.95
510 DATA 93,57,161,141,254,76,39
,38,129,9,38,2,134,32,23,3,218,2
25,141,254
520 DATA 60,16,37,0,189,225,141,
254,81,39,6,129,8,16,38,0,176,12
9,32,16,37
530 DATA 0,171,22,0,167,111,141,
3,234,230,141,254,30,23,255,42,2
3,3,209,39
540 DATA 6,193,255,39,236,32,3,2
3,255,33,23,255,5,39,237,111,141
,254,23,23
550 DATA 3,145,129,85,16,39,0,14
8,141,148,39,2,32,118,52,2,166,1
41,253,237
560 DATA 141,136,53,2,39,2,32,10
```

```
4,129,77,16,39,0,155,129,88,16,3
9,1,196,51
570 DATA 141,253,167,129,66,16,3
9,0,180,51,141,253,163,129,76,16
,39,0,185
580 DATA 51,141,253,167,129,82,1
6,39,0,183,51,141,253,164,129,83
,16,39,0
590 DATA 182,129,8,38,10,230,141
,253,173,231,141,253,189,32,14,1
29,9,38,34
600 DATA 230,141,253,160,231,141
,253,174,32,8,134,8,106,141,253,
167,32,17
610 DATA 134,32,106,141,253,158,
23,3,29,92,225,141,253,126,35,1,
79,230,141
620 DATA 253,148,129,31,34,8,129
,8,39,4,111,141,253,129,175,141,
253,117,50
630 DATA 100,57,51,141,253,86,14
1,126,23,254,70,39,251,51,141,25
3,116,230
640 DATA 196,193,255,39,45,161,1
96,39,4,51,65,32,242,111,196,32,
33,51,141
650 DATA 253,31,141,93,23,254,37
,39,251,51,141,253,83,230,196,19
3,255,39
660 DATA 12,161,196,39,8,109,196
,39,8,51,65,32,238,141,85,32,162
,167,192
670 DATA 32,197,141,90,225,141,2
53,19,39,1,92,231,141,253,13,32,
143,141,75
680 DATA 231,141,253,10,32,135,1
41,67,231,141,253,3,22,255,126,1
41,58,231
690 DATA 141,252,251,22,255,117,
111,141,253,0,111,141,252,253,11
1,141,252
700 DATA 250,111,141,252,234,57,
111,141,252,250,166,192,129,255,
39,10,172
710 DATA 159,160,2,108,141,252,2
36,32,240,57,134,8,173,159,160,2
,106,141
720 DATA 252,223,38,246,57,141,2
20,141,201,198,3,231,141,2,127,2
3,253,156
730 DATA 39,251,129,57,34,26,129
,48,37,22,109,141,2,110,39,16,10
6,141,2,104
740 DATA 173,159,160,2,108,141,2
52,179,141,41,32,221,129,3,39,28
,129,13,38
750 DATA 213,141,188,230,141,252
,140,225,141,252,135,36,2,32,12,
225,141,252
760 DATA 118,35,3,90,32.247,57,1
41,164,50,98,22,254,238,128,48,3
1,137,111
770 DATA 141,252,107,166,141,252
,114,167,141,252,111,166,141,252
,105,167
780 DATA 141,252,102,231,141,252
,97,231,141,252,83,166,141,252,9
0,198,10
790 DATA 61,235,141,252,72,231,1
41,252,68,166,141,252,76,129,2,3
4,18,37,4
800 DATA 193,55,34,12,198,100,61
,235,141,252,47,231,141,252,43,5
7,230,141
810 DATA 252,28,231,141,252,34,2
30,141,252,30,57,108,141,252,34,
23,1,167
820 DATA 111,132,111,1,125,2,221
,16,39,254,134,231,141,252,27,17
5,141,252
830 DATA 6,52,32,48,31,16,158,13
6,16,175,141,251,248,49,63,16,15
9,136,111
840 DATA 141,251,251,111,141,1,1
72,23,252,236,23,1,151,39,6,193,
255,39,240
850 DATA 32,3,23,1,75,23,252,232
,39,237,23,1,67,175,141,251,210,
129,8,39
860 DATA 22,129,9,39,35,129,12,1
6,39,0,174,129,13,16,39,1,6,129,
32,36,38
870 DATA 32,193,140,2,221,39,188
,48,31,16,158,136,49,63,16,159,1
36,32,180
880 DATA 16,158,136,49,33,16,172
,141,251,153,39,164,16,159,136,4
8,1,32,161
890 DATA 230,141,251,159,225,141
,251,126,36,147,16,174,141,251,1
31,230,162
900 DATA 231,33,16,172,141,251,1
24,34,245,167,132,16,174,141,251
,113,49,33
910 DATA 16,175,141,251,106,111,
164,16,158,136,49,33,16,175,141,
251,98,31
920 DATA 18,166,160,39,35,158,13
6,140,5,255,37,22,174,141,251,75
,48,136,224
930 DATA 175,141,251,68,174,141,
251,70,48,136,224,175,141,251,63
,173,159
940 DATA 160,2,32,217,16,174,141
,251,52,16,159,136,16,174,141,25
1,38,49,33
950 DATA 16,175,141,251,31,174,1
41,251,31,48,1,108,141,251,40,22
,255,35,125
960 DATA 2,222,16,39,255,24,16,1
58,136,230,1,231,128,93,38,249,1
74,141,251
970 DATA 1,109,132,38,4,48,31,49
,63,16,159,136,16,175,141,250,24
3,31,18,166
980 DATA 160,39,6,173,159,160,2,
32,246,134,32,173,159,160,2,16,1
74,141,250
990 DATA 220,16,159,136,16,174,1
41,250,206,49,63,16,175,141,250,
199,16,174
1000 DATA 141,250,196,49,63,16,1
75,141,250,189,106,141,250,202,2
2,254,197
1010 DATA 16,174,141,250,175,16,
159,136,53,32,174,141,250,168,10
6,141,250
1020 DATA 170,16,39,253,24,230,1
41,250,173,192,2,215,215,111,141
,250,165,22
1030 DATA 253,9,230,132,141,5,23
1,159,0,136,57,193,64,37,5,193,9
7,36,4,57
1040 DATA 203,64,57,192,96,57,12
9,97,37,2,128,32,57,230,141,250,
127,238,102
1050 DATA 17,131,159,255,34,9,21
4,215,92,109,141,250,101,38,1,57
,108,141,250
1060 DATA 94,189,133,180,214,215
,203,2,57,230,141,0,11,92,231,14
1,0,6,193
1070 DATA 127,57,255,255,255,255
```

Listing 3:

```
10 'STRING EDIT DRIVER PROGRAM
20 'BUFFER STUFFER, (C) 1984,
30 'by Richard W. Rutter
40 CLEAR1000
50 OF=3541'MANDATORY EXECUTION O
FFSET FOR DISK: FOR CASSETTE SY
STEMS, USE "OF=1536"
60 LINEINPUT"QUIT OR STRING ENTR
Y?:";ST$
70 Q$=LEFT$(ST$,3):IFQ$="QUI"THE
NEND
80 PA$=ST$
90 PRINT:PRINT"[IN STRING EDIT M
ODE]":GOSUB1000
100 ST$=PA$
110 PRINT"STRING EDIT RESULTS:":
PRINT"["ST$"]"
120 GOTO60
10000 EL=157:EA=PEEK(EL):EB=PEEK
(EL+1):VP=0:VL=78+OF:LE=LEN(PA$)
:PA$=PA$+STRING$(255-LE,32):VP=V
ARPTR(PA$):POKEVL,LE:POKEVL+1,PE
EK(VP+2):POKEVL+2,PEEK(VP+3):EXE
C143+OF:POKEVP,PEEK(VL)*POKEEL,E
A:POKEL+1,EB:RETURN
```

GAME 16K ECB

GRANNY'S PEG — GAME CHALLENGE

by Daryl Judd

One of the memories of going to my grandmother's house is playing the puzzle-type game called *Hi-Q*. It's a small, white board with 44 red pegs that are jumped back and forth in checker-type moves. The object (which I could never seem to master) is to end up with only one peg in the middle.

I recently found out that my wife's grandmother also has the game. Is it possible this game is a requirement of some grandmothers' union? Perplexed, I pondered over this thought for several days. Then, I realized my mission: to bring the CoCo world the game of *Hi-Q* for those whose grandmothers didn't belong to the union.

I added sight and sound and in completing my mission, I had to call on several tactics I have picked up in the past (past issues of RAINBOW, that is) such as the false colors of PMODE 3 and GET and PUT statements.

The variables are as follows:

'A' is the array used to draw the pegs
'B' is the array used to erase the pegs
Num is the number of pegs left
'M' is the x starting point of the cursor square
'L' is the y starting point of the cursor square

20	69
40	179
73	8
104	191
130	117
156	157
END	33

The listing: HI-Q

```
1 'H-Q BY DARYL JUDD
2 PMODE3,1:PCLS:SCREEN1,0:COLOR2
,2
3 DRAW"BM0,30;D120;R30;U50;R50;D
50;R30;U120;L30;D50;L50;U50;L30"
4 PAINT(2,32),3,2
5 CIRCLE(190,89),56,2,1.15,.1705
,.11
6 CIRCLE(190,89),36,2,1.15,.2,.1
7 DRAW"BM217,111;H10;G15;F10"
8 DRAW"BM219,142;F10;E13;H11"
9 PAINT(190,28),3,2
10 FORX=1TO400:NEXTX
11 PLAY"T3;L8D;G;P8G;A;P8;A;B;O4
D;O3B;G;P8"
12 PLAY"D;G;P8;G;A;P8;A;L4.B;L6G
;P8"
13 PLAY"L8;D;G;P8;G;A;P8;A;B;O4D
;O3B;G;P4"
14 SCREEN1,1
15 PLAY"L8;O2E;P4;L8.;O1A;P16;L8
;O2C;L4;O1B;P8;L8G"
16 FORX=1TO700:NEXTX
17 CLS:PRINT@7,"**INSTRUCTIONS**
"
18 PRINT" THE OBJECT OF THIS GAM
E IS TO"
19 PRINT"  END UP WITH ONE PEG I
N THE"
20 PRINT"CENTER HOLE. PEGS ARE S
UBTRACTED";
21 PRINT" FROM THE BOARD BY JUMP
ING, LIKE";
22 PRINT"  IN THE GAME OF CHECKE
RS. TO"
23 PRINT" MOVE THE SQUARE WHERE
YOU WANT"
24 PRINT" IT, PRESS THE ARROW KE
YS. TO"
25 PRINT"  JUMP, PRESS THE 'J' K
EY. AND"
26 PRINT"  THEN THE ARROW KEY IN
 THE"
27 PRINT"  DIRECTION YOU WANT TO
 MOVE."
28 PRINT" WHEN THERE ARE NO MORE
 MOVES,"
29 PRINT" PRESS THE 'N' FOR YOUR
 RATING."
30 PRINT" AND IF WANT TO QUIT, P
RESS THE"
31 PRINT"        'Q' KEY AND YOU WI
LL."
32 PRINT"            **ANY KEY**";
33 I$=INKEY$:IFI$=""THEN33
34 PMODE3,1:PCLS0
35 CIRCLE(10,10),7,3,.9
36 PAINT(10,10),3,3
37 DIMA(14,10),B(14,10)
38 GET(3,5)-(17,15),A,G
39 GET(33,5)-(47,15),B,G
40 CLS3:PMODE4,1:PCLS:SCREEN1,1:
PMODE3
41 PCLS0:NUM=44
42 COLOR1,1
43 LINE(10,0)-(246,185),PSET,B
44 LINE(10,0)-(88,62),PSET,BF
45 LINE(166,0)-(244,62),PSET,BF
46 LINE(10,123)-(88,185),PSET,BF
47 LINE(166,123)-(246,185),PSET,
BF
48 FORX=96TO146STEP25
49 FORY=8TO48STEP20
50 PUT(X,Y)-(X+14,Y+10),A,PSET
51 NEXTY:NEXTX
52 FORX=21TO221STEP25
53 FORY=68TO108STEP20
54 IFX=121ANDY=88THEN56
55 PUT(X,Y)-(X+14,Y+10),A,PSET
56 NEXTY:NEXTX
57 FORX=96TO146STEP25
58 FORY=128TO168STEP20
59 PUT(X,Y)-(X+14,Y+10),A,PSET
60 NEXTY:NEXTX
61 COLOR1,1
62 M=119:L=86
63 GOSUB106
64 'WAIT FOR COMMAND
65 I$=INKEY$:IFI$=""THEN65
66 IFI$=CHR$(94)THEN74
67 IFI$=CHR$(10)THEN82
68 IFI$=CHR$(8)THEN90
69 IFI$=CHR$(9)THEN98
70 IFI$="J"THEN108
71 IFI$="N"THEN163
72 IFI$="Q"THEN175
73 GOTO64
74 'MOVE UP
75 IFL=66ANDM<94THEN78
76 IFL=66ANDM>144THEN78
77 IFL>6THEN79
78 SOUND10,2:GOTO64
79 COLOR4,4:GOSUB106
80 COLOR1,1:L=L-20:GOSUB106
81 GOTO64
82 'MOVE DOWN
83 IFL=106ANDM<94THEN86
84 IFL=106ANDM>144THEN86
85 IFL<166THEN87
86 SOUND10,2:GOTO64
87 COLOR4,4:GOSUB106
88 COLOR1,1:L=L+20:GOSUB106
89 GOTO64
90 'MOVE LEFT
91 IFM=94ANDL<66THEN94
92 IFM=94ANDL>106THEN94
93 IFM>19THEN95
94 SOUND10,2:GOTO64
95 COLOR4,4:GOSUB106
96 COLOR1,1:M=M-25:GOSUB106
97 GOTO64
98 'MOVE RIGHT
99 IFM=144ANDL<66THEN102
100 IFM=144ANDL>106THEN102
101 IFM<219THEN103
102 SOUND10,2:GOTO64
103 COLOR4,4:GOSUB106
104 COLOR1,1:M=M+25:GOSUB106
105 GOTO64
106 LINE(M,L)-(M+18,L+14),PSET,B
107 RETURN
108 'JUMP
109 IFPPOINT(M+9,L+7)<>7THEN64
110 I$=INKEY$:IFI$=""THEN110
111 IFI$=CHR$(94)THEN116
112 IFI$=CHR$(10)THEN127
113 IFI$=CHR$(8)THEN138
114 IFI$=CHR$(9)THEN149
115 SOUND10,2:GOTO110
116 'JUMP UP
117 IFL<46THEN160
118 IFPPOINT(M+12,L-13)<>7THEN16
0
119 IFPPOINT(M+12,L-33)<>8THEN16
0
120 COLOR4,4:GOSUB106
121 PUT(M+2,L+2)-(M+16,L+12),B,P
SET
122 PUT(M+2,L-18)-(M+16,L-8),B,P
SET
123 PUT(M+2,L-38)-(M+16,L-28),A,
PSET
124 COLOR1,1:L=L-40:GOSUB106
125 NUM=NUM-1
126 GOTO64
127 'JUMP DOWN
128 IFL>130THEN160
129 IFPPOINT(M+12,L+27)<>7THEN16
0
130 IFPPOINT(M+12,L+47)<>8THEN16
0
131 COLOR4,4:GOSUB106
132 PUT(M+2,L+2)-(M+16,L+12),B,P
SET
133 PUT(M+2,L+22)-(M+16,L+32),B,
PSET
134 PUT(M+2,L+42)-(M+16,L+52),A,
PSET
135 COLOR1,1:L=L+40:GOSUB106
136 NUM=NUM-1
137 GOTO64
138 'JUMP LEFT
139 IFM<69THEN160
140 IFPPOINT(M-14,L+7)<>7THEN160
141 IFPPOINT(M-39,L+7)<>8THEN160
142 COLOR4,4:GOSUB106
143 PUT(M+2,L+2)-(M+16,L+12),B,P
SET
144 PUT(M-23,L+2)-(M-9,L+12),B,P
SET
145 PUT(M-48,L+2)-(M-34,L+12),A,
PSET
146 COLOR1,1:M=M-50:GOSUB106
147 NUM=NUM-1
148 GOTO64
149 'JUMP RIGHT
150 IFM>169THEN160
151 IFPPOINT(M+35,L+7)<>7THEN160
152 IFPPOINT(M+60,L+7)<>8THEN160
153 COLOR4,4:GOSUB106
154 PUT(M+2,L+2)-(M+16,L+12),B,P
```

```
SET
155 PUT(M+27,L+2)-(M+41,L+12),B,
PSET
156 PUT(M+52,L+2)-(M+66,L+12),A,
PSET
157 COLOR1,1:M=M+50:GOSUB106
158 NUM=NUM-1
159 GOTO64
160 'REJECT MOVE
161 SOUND10,2
162 GOTO64
163 'NO MORE MOVES
164 CLS:PRINT@36,"YOU FINISHED W
ITH";NUM"PEGS"
165 IFNUM>7THENR$="IT'S ONLY A G
AME"
166 IFNUM<8ANDNUM>5THENR$="KEEP
TRYING"
167 IFNUM<6ANDNUM>3THENR$="GOOD
SCORE!"
168 IFNUM<4ANDNUM>1THENR$="VERY
GOOD!"
169 IFNUM=1THENR$="OLYMPIC HOPEF
UL"
170 IFNUM=1ANDPPOINT(128,93)=7TH
ENR$="YOUR PERFECT!"
171 PRINT@105,R$
172 PRINT@294,"ANOTHER ROUND (Y/
N)"
173 I$=INKEY$:IFI$=""THEN173
174 IFI$="Y"THEN40
175 'QUIT
176 CLS:SCREEN 0,1
177 PLAY"T4;O3;L4E-;L3E;G;O4;C;P
4"
178 CLS(4)
179 PLAY"O3;L4E;L3D;G;B;P4"
180 CLS(2)
181 PLAY"L5G;L2G-;L5G;L3A;L8A-;L
5A;O4;C;O3;L2B;L8B-;A;A-;L2G;P4"
182 CLS(3)
183 PLAY"L4E-;L3E;G;O4;C;P4"
184 CLS(5)
185 PLAY"O3;L4E;L3D;G;B;P4"
186 CLS(8)
187 PLAY"L5G;L2G-;L5G;B;L3A;L4G-
;L2.G;O4;L8;T12;D;E;G-;L2G"
188 CLS(1)
```

GAME 16K ECB

Chopper Assault

By Jens Petersen

A 16K ECB Color Computer game, *Chopper Assault* requires a joystick to play. The object is to stop enemy spies from gathering too much information; if they do, you die!

First CLOAD and RUN the program, then you will be asked for either levels 1, 2 or 3, depending on your level of play. Type in your name and press ENTER, which will then show the title screen. Press the firebutton to start the game.

You will see from inside your own helicopter your four cannon sites, with a box in the center of the sites showing where the cannon will shoot. Your timer is at the top, indicated by a line or bar. Your score is there too, in the middle. You move the box, or center site, around the screen with the joystick.

You have five shots at the enemy; when he gathers enough information to leave, another comes to take his place. If you shoot one, your score increments by the amount of time left. If your score is above the high score, the program displays some graphics to show you this, but it can only happen once in your game.

You die if your time runs out, meaning that the enemies have gathered enough information to destroy you. If you're dead, the program goes into text and you see your name and score, and the top three names and scores. Press the firebutton to play again or press 'Q' to quit. (*Chopper Assault* does not work on a disk-based system.)

The listing: CHOPPER

```
10 '******CHOPPER ASSAULT*******
20 'JENS PETERSEN  JANUARY14/84*
30 '****************************
40 POKE65495,0
50 CLEAR300:DIMH(18),J(18),EX(10
)
60 A$(0)="BDER2FD4GL2HU4":A$(1)=
"BD6BR2RNRU6G":A$(2)="BDER2FDGL2
GD2R4":A$(3)="BDER2FDGNLFDGL2H":
A$(4)="BR4ND6G3R4":A$(5)="BRNR4D
3ER2FD2GL2H":A$(6)="BRNR3GD4FR2E
UHL2":A$(7)="R4G3D3"
70 A$(8)="BRR2FDGFDGL2HUEHUE":A$
(9)="BD6R2EU4HL2GDFR2"
80 GOSUB780:SC=0
90 GOTO210
100 T$=STR$(SC)
110 COLOR5,0:LINE(104,3)-(D,13),
PSET,BF
120 D=106
130 FORT=2TOLEN(T$)
140 E$=MID$(T$,T,1)
150 E=VAL(E$)
160 DRAW"C0BM"+STR$(D)+",5"+A$(E
)
170 D=D+7
180 NEXT
190 COLOR5,0
200 RETURN
210 PMODE2,1:COLOR0,5:PCLS:SCREE
N1,1
220 D$="NR5D10R5BU10BR3D10U5R5NU
5D5BU10BR3D10R5U10L5BR8ND10R5D5L
5BR8BU5ND10R5D5L5BR8BU5NR5D5NR3D
5R5BU10BR3ND10R5D5L5RF4D":DRAW"B
M92,30"+D$
230 DR$="ND10R5D5NL5D5BR8BU10L5D
5R5D5L5BR12BU10L3D5R3D5L3BR8U10R
3D5NL3D5BR4NU10R5U10BR4D10R3BR6U
10NL2R2":DRAW"BM92,44"+DR$
240 DRAW"BM20,120D10R5U5L5BR8D5R
5NU5D5BR12R5D10G2L3H2BR11BU10NR4
D6NR3D6R4BU12BR4ND12F6ND6U6BR4NR
4D6R4D6NL4BR12BU12ND12R4D6NL4BU6
BR4NR4D6NR3D6R4BU12BR4R2NR2ND12B
R6NR4D6NR2D6R4BU12BR4ND12R4D6L4R
1F3D3BU12BR4NR4D6R4D6NL4BU12BR4N
R4D6NR2D6R4BU12BR4ND12F6ND6U6
250 DRAW"BM160,150D2BR4BU2R4D12L
4U6NR4U6BR6D6R4NU4D6"
260 P=PEEK(65280):IFP=254ORP=126
THEN270ELSE260
270 PMODE4,1:PCLS:SCREEN1,1:COLO
R5,0:FORCF=1TO2:Q1=127:Q2=96:Q3=
96:FORT=127TO0STEP-3:Q1=Q1+3:Q2=
Q2+2.2:Q3=Q3-2.2:LINE(T,Q3)-(Q1,
Q2),PSET,B:NEXT:COLOR0,0:NEXT:CO
LOR5,0
280 PMODE0,1:SCREEN1,1
290 PMODE4,1:PCLS
300 DRAW"BM50,50R4NR4D2LG2FR4EH2
":PAINT(53,54),5,5
310 GET(50,50)-(38,55),H,G:GET(1
00,100)-(108,105),J,G:PCLS:FORX=
1TO20:PSET(RND(10)+100,RND(10)+1
50,5):NEXT:GET(100,150)-(110,160
),EX,G
320 PCLS:SCREEN1,1
330 V1=RND(191):V=RND(255):O1=10
0:O2=100:EM=200:AS=127:SD=96:DS=
96
340 LINE(0,0)-(255,16);PSET,BF
350 COLOR0,1
360 A=0:B=0
370 FORT=1TO2
380 A=A+1:B=B+1
390 DRAW"BM"+STR$(A)+","+STR$(B)
+D$
400 NEXT
410 A=200:B=0:FORT=1TO2:A=A+1:B=
B+1:DRAW"BM"+STR$(A)+","+STR$(B)
+DR$:NEXT
420 D=104:GOSUB100
430 SCREEN1,1
440 X=RND(5)+2:X1=RND(5)+2
450 EM=EM-F3
460 IFEM<5THEN760ELSELINE(200,14
)-(EM,14),PRESET
470 JH=JOYSTK(0):JV=JOYSTK(1)
480 WE=JH*255/63
490 EW=JV*191/63+16+5
500 IFWE<5THENWE=5ELSEIFWE>250TH
ENWE=250
510 IFEW>191THENEW=191ELSEIFEW<1
9+5THENEW=19+5
520 LINE(AS-5,SD-5)-(AS+5,SD+5),
PRESET,B
530 LINE(0,SD)-(8,SD),PRESET:LIN
E(255,DS)-(247,DS),PRESET:LINE(0
,EW)-(8,EW),PSET:SD=EW:LINE(255,
EW)-(247,EW),PSET:DS=EW
540 LINE(AS,17)-(AS,22),PRESET:L
INE(AS,191)-(AS,185),PRESET:LINE
(WE,17)-(WE,22),PSET:LINE(WE,191
)-(WE,185),PSET:AS=WE
550 LINE(WE-5,EW-5)-(WE+5,EW+5),
PSET,B
560 P=PEEK(65280):IFP=126ORP=254
GOSUB640
570 RN=RND(20):IFRN=1THENX1=-X1:
PLAY"L255V31O1:CD" ELSE IF RN=2T
HENX=-X:PLAY"L255V31O1:CD"
580 IFX4=>5THENFORT=1TO7STEP2:CI
RCLE(V+4,V1+3),T:NEXT:FORT=1TO7S
TEP2:CIRCLE(V+4,V1+3),T,0:NEXT:V
=RND(255):V1=RND(191):PLAY"L50O5
DGDGDGDGD":X4=0
590 V=V+X:IFV>247THENV=7ELSEIFV<
7THENV=247
600 V1=V1+X1:IFV1>185THENV1=24EL
SEIFV1<24THENV1=185
610 PUT(O1,O2)-(O1+8,O2+5),J,PSE
T:PUT(V,V1)-(V+8,V1+5),H,PSET:O1
=V:O2=V1
620 IFINKEY$="Q"THEN910
630 GOTO450
640 PRESET(WE,EW):PUT(V,V1)-(V+8
,V1+5),H,PSET:PH=PPOINT(WE,EW):L
INE(WE,22)-(WE,185),PSET:LINE(10
,EW)-(245,EW),PSET:LINE(WE,22)-(
WE,185),PRESET:LINE(10,EW)-(245,
EW),PRESET:PLAY"L255V31O1;12:11;
10;9;8"
650 IFPH<>0THEN660ELSEEM=EM-5:X4
=X4+1:RETURN
660 PUT(WE-5,EW-5)-(WE+5,EW+5),E
X,PSET
670 DRAW"C0BM72,3D4ND4R4NU4D4BR3
NU8BR4U8NL2R2BR3D5BD2D"
680 PLAY"L30V31O1;12;1;12;1;12;1
;12;1;L255O1;4:3;2;1;4;3;2;1;4;3
;2;1;4;3;2;1;4;3;2;1;4;3;2;1;4;3
;2;1;4;3;2;1":PUT(WE-5,EW-5)-(WE
+5,EW+5),J,PSET
690 SC=SC+EM:GOSUB100:EM=200
700 LINE(200,14)-(0,14),PSET
710 V=RND(255):V1=RND(191):IFSGN
(X)=-1THENX=RND(5)+2ELSEX=(RND(5
)+2)*-1
720 IFSGN(X1)=-1THENX1=RND(5)+2E
LSEX1=(RND(5)+2)*-1
```

continued on Page 45

NEW! TANDY 1000

The Best $3000 Computer You Can Buy for $1999 — Includes DeskMate™!

The Extra Cost Options Are Built-In

$1999*

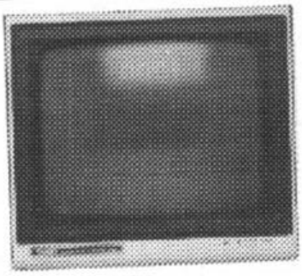
- **Complete with Ready-to-Run DeskMate™ Software**
- **User-Friendly Operation**

Solve difficult business situations faster and more efficiently than ever before! The advanced TANDY 1000 simplifies operation and creates potentially unlimited software possibilities.

The MS-DOS system gives access to leading computer programs such as Lotus 1-2-3 and ATTACHE Business Software. MS-DOS is IBM† compatible for even greater software options! Expand your memory from 128K RAM to a powerful 640K RAM. Connect a monitor and create superb monochrome or color graphics. A three-voice sound circuit and built-in speaker help you create sophisticated sounds and music. With DeskMate you'll start computing from day one, avoiding costly delays. **25-1000**

†IBM is a trademark of International Business Machines

*| Monitor Not Included



$349.95

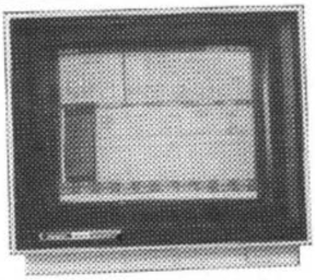
HIGH RESOLUTION DISPLAY MONITORS

VM-2 Monochrome Monitor

Features an easy on the eyes, 80-character by 25-line display for text. 640 by 200 display for high resolution graphics. **26-3211**

CM-2 Color Monitor

Produces red/green/blue color on the 80-character by 25-line screen, with white lettering. **26-3212**



$899.00

Special Offer To "RAINBOW" Readers. Bring "RAINBOW" In To Take Advantage Of These Bargain Prices!

Save $20

Reg 49.95

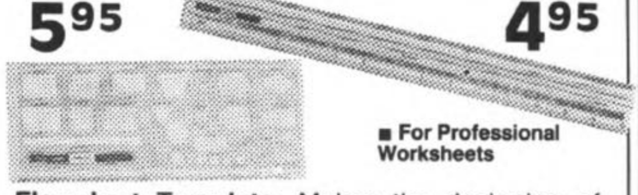
29.95 PK. 10

Durable 5¼" 40-Track Diskettes

At this price you can afford backup disks for all your color computer data! All 10 are single sided, double density, soft sector diskettes. They are made to exacting standards of excellence and certified error free. The specially cured oxide coating ensures surface stability, minimal media wear and longer disk life. For Tandy Color Computers. Plus Model 1, III, 4 and 4P users. **26-406**

SAVE $1 on Helpful Programming Aids

5.95 4.95



■ For Professional Worksheets

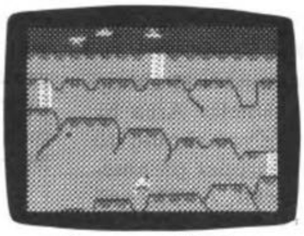
Flowchart Template. Makes the designing of programs easy. Was 6.95. **26-1312**

Printview Ruler. Highlight lines of printout for simple debugging of programs. Was 5.95. **26-1313**

Sale Ends 31st July 1985

Color Computer Games for a Sense of Adventure

Clowns and Balloons



Canyon Climber

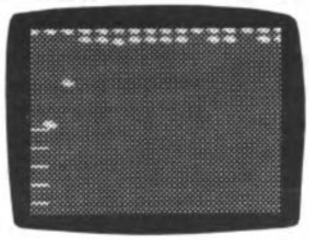
Save $30

Reg 49.95

19.95 EACH

Clowns and Balloons. Bounce the clown with your net to pop the balloons overhead. Disaster strikes if you let the clown hit the ground! Requires joystick. **26-3087**

Canyon Climber. As a cliff hanger you're challenged by one test after another. You must survive the kicking goats, fatal arrows and falling objects! Requires joystick. **26-3089**



Popcorn

Double Back

Save $20

Reg 39.95

19.95 EACH

Popcorn. Catch the falling rows of popcorn with your frying pan. Miss one and you lose a pan. Catch them all to win a bonus pan. Requires joystick. **26-3090**

Double Back. Stategy rules with a sense of humour as you double back to catch your own tail. Encircle the "safe" objects to gain extra points. Requires joystick. **26-3090**

Tandy ELECTRONICS

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

Nearly 350 Stores Australia-Wide

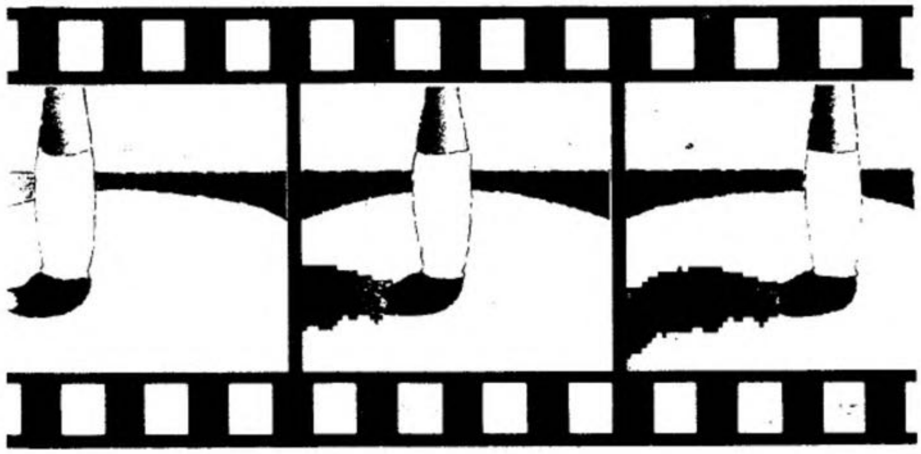
WE SERVICE WHAT WE SELL!

Available from 350 Stores Australiawide including Tandy Computer Centres

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

Animatic:

Automatic Animation



By Rita Sabo

Automatic animation (Animatic) is a set of graphics animation subroutines that can be called from BASIC or Assembler. With Animatic, the cumbersome process of writing animated graphics is minimized. In addition, when written in Assembler, Animatic will provide smoother and faster animation.

Animatic takes advantage of the fact that most animation programs follow roughly the same logic (save previous screen contents, get object from old position, put object in new position, etc.) and it automatically performs many of these steps.

To access Animatic from BASIC, you will make use of a "new" function called ANIM. The syntax for ANIM is:

```
X = ANIM(P0,P1, ...P7)
```

'X' is a numeric variable, and P0-P7 are the parameters described in Table 1. The variable 'Y' will contain return codes and status information relevant to the selected function.

Depending on the selected function (value of P0), you may not need to specify all of the parameters. Zero is assumed when a parameter value is omitted: Y = ANIM(P0,,P2), but if you omit coding double-commas, then the last used value for the missing parameter is used. Example: Y - ANIM(P0,P1,,P3) is the same as Y = ANIM(P0,P1,0,P3), and Y = ANIM(P0,P1) will use the last used values for P2...P7 (if applicable to the function indicated by P0).

To access *Animatic* from an Assembler program, you must first obtain the address of the Parameter Area by doing JSR INFO. There you should do a JSR ANIM with the proper parameters in this area. Upon exit, ANIM will set the 'D' register with the relevant operation status.

Description of Functions

Following find the description for each of the functions shown in Table I. For an example of a program using these functions, refer to program listings 1 and 2. Compare program Listing 1 with the "do-it-yourself #8-1" program of Radio Shack's *Going Ahead with Extended BASIC*.

DEFINE (P0=0)

It must be the first used ANIM function. It defines in P1 the maximum number of figures (a.k.a. objects) to be created in your program.

CREATE (P0=1)

A CREATE is required for each of the figures to be moved in your program. The figure will behave according to the values of P2 and P7.

You don't have to specify anything in P1. A sequential number (starting with 1) is assigned to each object as it is being created. Any further reference to this object will use this "object I D" instead of the traditional XY coordinates.

If P2 equals zero, the object will be placed on the screen exactly as it was created. If P2 is not zero, the object will be MIXed with the screen background. MIX is similar to the OR function for PMODEs 0, 2 and 4. See pictures 1 and 2 for a description of MIX effects in several PMODEs.

P3 and P4 indicate the XY coordinates of the object's upper-left corner. P5 and P6 indicate the width and height of the figure. P5 and P6 should not exceed 100.

P7 represents the action to be taken in the event that this object is moved to an XY position unfit for the size of the object. (For example: attempting to move an object 20 pixels wide to positions X=244, Y=14.) This condition will, from now on, be referred to as "overflow." With P7 = 0, *Animatic* will signal an error in overflow.

When P7 = 1, the object will be "frozen" on the nearest possible position on the border of the screen. In our example: X=235, Y=14.

If P7 = 2 the object would disappear in overflow. You can make it reappear by moving it to a legal position.

With P7 = 3, the object will "wraparound," henceforth appearing on the extreme side of the screen (in our example: X=0, Y=14).

Regardless of the P7 selection, you will receive notice of overflows through the status of the operation.

MOVE (P0=2)

In P1, specify the number of the object to be moved. P2 represents the criteria for obtaining the new XY coordinates.

P2=0: The object will move to the absolute X-Y values specified in P3 and P4.

P2=1: The movement will be relative to the actual position. The P3 and P4 values will be added to the actual XY coordinates to obtain the destination. P3 and P4 can be negative.

P2=2: The object will move to the absolute XY coordinates pointed out by the left joystick. Because the joystick readings cover a 0-63 range, the 'X' reading is multiplied by four and the 'Y' reading by three.

P2=3: Same as in P2=2, but using the right joystick.

P3=4: The object has a relative movement with the displacements calculated from the left joystick readings.

The 'X' and 'Y' coordinates are calculated as follows:

```
X=X0+((XJ-32)*P3)/8
Y=Y0+((YJ-32)*P4)/8
```

Where XO and YO = actual coordinates.

XJ and YJ = X-Y joystick readings.

P3 and P4 = Values given for parameters 3 and 4. These values can be negative. However, the ANIM instruction will only accept negative values in Hex form, i.e., specify &HFF instead of -1.

Using this option, you can move the object with the direction and acceleration represented by the position of the joystick (i.e., P3 = 3 will give the effect of greater accelerations than P3 = 2).

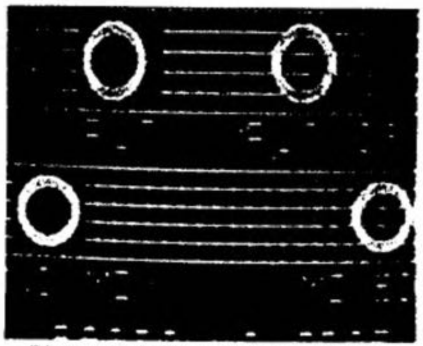
P2=5: Same as in P2=4, but using the right joystick.

P2=6: *Animatic* will select XY values at random. P3 and P4 represent the maximum random value for 'X' and 'Y.' P5 and P6 will be added to the generated 'X' and 'Y' values, respectively.

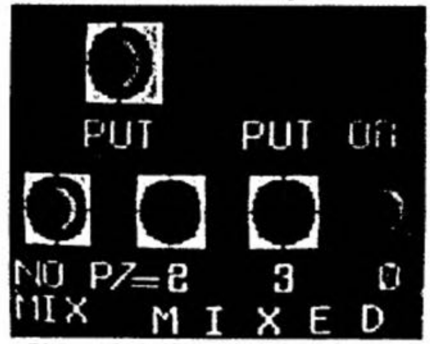
If you specify P3 and P4 = 0, *Animatic* will use P3=255; P4=191; P5=0; P6=0 as a default.

The random sequence has a period of 256, but *Animatic* reseeds itself once the period is exhausted by taking the timer value as a seed number. If you are calling *Animatic* from ML, write an interrupt routine to modify storage addresses $112-$113 accordingly.

P3=7: Keyboard controlled movement can be obtained by selecting this option.



Picture 1: MIX option in PMODEs 0, 2 and 4. Top using PUT (with and without OR). Bottom using *Animatic*.



Picture 2: MIX option in PMODEs 1 and 3. Top using PUT. Bottom using *Animatic* with several MIX color combinations.

ANIMATIC
TABLE#I

FUNCTION	P0	P1	P2	P3	P4	P5	P6 -	P7
DEFINE	0	# FIGS	.	.	.	.	.	.
CREATE	1	.	0=NOMIX ¢0 MIX	X COORD	Y COORD	# COLS	# ROWS	0=ERROR 1=FREEZE 2=DISAPP. 3=WRAP
MOVE	2	# FIG	0=ABS.	X	Y	.	.	IF TYPE MIX COLOR CODE (0-3)
			1=REL.	+X	+Y			
			2=LJOYSK	.	.			
			3=RJOYSK	..	.			
			4=LJOYSK	(X)	(Y)			
			5=RJOYSK	(X)	(Y)			
			6=RANDOM	– X	I Y	+X	+Y	
			7=KEYBRD	+X	+Y	.	.	
			8=REDISP	.	.			
PLACE	3	◄···········	SAME	AS	MOVE	···········	···········	···········►
REMOVE	4	# FIG	.	.	.	.	.	.
COPY	5	TO FIG	FROM FIG	.	.	.	.	.
OPERATE	6	# FIG	0=CLEAR	OPERATION BYTE	.	.	.	.
			1=NOT					
			2=AND					
			3=OR					
DOMAIN	7	# FIG	0	X –	Y I	– X	I Y	.
			# FIG	.	.	.	.	
INFO	8	A	B					

The left/right arrows will generate a relative movement from the value in P4 and the up/down arrows from P5. The values in P4 and P5 should be positive since *Animatic* already knows the left/up arrows represent a negative displacement.

P2>7: Selecting P2 with a value greater than seven will redisplay the object in the current X-Y location.

P3-P6 as discussed above have different meanings depending on the value of P2.

P7 is used only if the object was CREATEd with MIX. P7 indicates the color to be omitted when the object is being mixed with the screen. P7=0 removes buff/green, P7=1 removes cyan/yellow, P7=2 magenta/blue, and P7=3 orange/red. Refer to Picture 2 for results with different P7 values.

As a result of the MOVE function, the variable at the left of the ANIM instruction will be set as follows:

0 = No screen overflow
1 = Upper screen overflow
2 = Bottom screen overflow
4 = Left overflow
5 = Upper left corner
6 = Bottom left corner
8 = Right screen overflow
9 = Upper right corner
10 = Bottom right corner

ML programs can get these values from 'B' register.

PLACE (P0=4)

Unlike MOVE, PLACE does not assume that the object being moved is already on the screen. PLACE is more like PUT as it limits itself to copying object from storage onto the screen. The options for PLACE are exactly the same as these of MOVE.

REMOVE (P0=4)

With REMOVE, you simply "swap" the contents of the screen with the contents of storage. This function differs from MOVE (P2>7) in that the object in storage is displayed "as is," whereas MOVE performs internal pixel and mix adjustments. REMOVE is fast and it can be used to simulate blinking. The figure to be removed is specified in P1.

COPY OBJECT (P0=5)

Sometimes you may want to perform a "tricky" effect or simply substitute one object for another. COPY duplicates an object. COPY does not like it when the new object has not been CREATEd, and when the size of the new object is less than the size of the object being copied. Both the object being copied and the new object must have the same MIX or NO-MIX definition.

In P1, specify the destination object. In P2, specify the object being copied.

OPERATE (P0=6)

Used to directly modify an object. In P2, specify the operation to be performed upon the object defined in P1.

P2=0: Clear the object to the value specified in P3.

P2=1: Perform a logical "NOT" operation on the object. If in a two-color PMODE, this will convert the object into its reverse colors.

P2=2: Make an "AND" operation against the value specified in P3.

P3=3: Make an "OR" against the value in P3.

For NO-MIX objects, the changes will immediately be represented on the screen, but for MIX objects the changes will not appear until the next time you move your object.

CHECK DOMAIN (P0=7)

With DOMAIN you can test if an object "touches" a specific screen area. This function is used in program Listing 2 to check for asteroids crashing with the spaceship.

Specify the object to be tested in P1. If P2 is not zero, this number will represent an object whose coordinates will be used to define the screen area. For example, to know if object 2 "touches" object 5, code P1=2, P2=5. If P2=0, then P3 through P6 define the X-Y cordinates of the area's corners. If the object touches a point within this square, a '1' value is returned.

GENERAL INFO (P0=8)

To call this function from ML programs, make a JSR INFO. The arguments should be given in registers 'A' and 'B.'

With INFO you can obtain information about *Animatic* depending upon the P1 and P2 values. "NZ" represents a value other than zero in the table below.

P1	P2	Result
=0	=0	Address of an internal parameter table (required by ML programs). Also clears to zero the parameter table.
=0	NZ	XY coordinates for the NZ object. The result of the XY coordinates has the format &HXXYY.
=1	NZ	Address of internal Figure Definitions for object NZ. (Do not expect to use this function too often.)

Error Messages

Animatic returns error codes with the following format:

```
"WW ERROR ON FIGURE YYY ACTION Z"
```

YYY is the number of the object you were using when the error occurred and 'Z' is the number of the attempted function. If in BASIC, you'll also get an ?FC Error. When calling *Animatic* from Assembler, the error will be displayed and control returns to your program. You will be notified through a non-zero value in the 'A' register. (This does not apply for calls to INFO.)

What about WW? Following find its meaning:

XOS= Out of Screen. You selected P7=0 during function P0=1 for this object and have attempted to move the object to an overflow position.

XOM= There is not enough memory to create the object. If possible, relocate *Animatic* to a lower address. The program in Listing 3 will help you to determine an appropiate offset for *Animatic*.

XOF= You are trying to CREATE more objects than specified in DEFINE.

XIO= Invalid option. The requested function does not exist (valid options are 0-8 for BASIC and 0-7 for ML programs).

XEX= You are trying to CREATE an object more than 100 pixels wide or with a width of zero pixels.

XEY= You are trying to CREATE an object more than 100 pixels high or with a height of zero.

XNC= Object not created. The object you are trying to use has not been CREATEd.

XNI= You forgot to DEFINE (P0=0) *Animatic*.

XIC= Can't copy object. See description for the COPY function and see if you are violating some of the restrictions.

Some Things to Know

Animatic takes about 2.5K of storage, plus the required storage to keep the objects. It is written in PIC code and works on any CoCo with at least 16K and Extended BASIC. Disk is not required.

Although *Animatic* runs in 16K systems, you will need a 32K system and *EDTASM+* to enter and assemble the program. RAINBOW ON TAPE is an excellent alternative. You may also send me a SASE with a formatted diskette plus $4 (U.S. currency).

Listing 4 contains the source code. The program is so large that I do not recommend typing all the comments.

Program 3 will estimate the required size for your figures, and it suggests a load address for *Animatic*. After assembling the code, make a CLEAR 200,LOAD ADDRESS-1.

If using *Animatic* from BASIC, type in EXEC after loading it. Nothing should happen after typing EXEC and the cursor must continue blinking as normal. At this point, BASIC already recognizes the ANIM instruction. Because of this new instruction, avoid the use of USR0 and USR1 while in BASIC.

For a start, you may try sample programs 1 and 2. If after running a BASIC-*Animatic* program and you get ?SN Errors or you see '!' instead of ANIM when listing your program, this means you forgot to type EXEC after loading *Animatic*.

With *Animatic*, I have tried to provide a lot of functions and an easy interface for animation purposes. However, when used in complex animation environments, several considerations and restrictions inherent in its design have to be taken into account.

The potential for combinations in the animation functions here provided is such that it would require a more lengthy article to describe all possible effects, restrictions and techniques. I do encourage you to experiment whenever you have doubts. Of course, I would like to hear from you if you have questions, comments or problems regarding *Animatic*. You may contact me at 20819 Via Valencia, Boca Raton, FL 33433.

Listing 1: ROCKET

```
10 'THIS PROGRAMS MOVES A ROCKET
 FROM LEFT TO RIGHT OF THE SCREE
N
20 'PREPARE GRAPHICS AND DRAW RO
CKET
30 PCLEAR 4
40 PMODE 4,1
50 PCLS
60 SCREEN 1,1
70 X=10:Y=10
80 DRAW "BM10,10; S2;H10;R15;F10
;R20;F10;G10;L20;G10;L15;E10;U20
;D4;NL8;D4:NL12;D4NL16;D4:NL12:D
4;NL8"
90 'DEFINE ANIMATIC. MAX 1 FIGUR
E
100 A=ANIM(0,1)
110 'CREATE FIGURE: NO MIX, FROM
 X=0 Y=0, X SIZE=35, Y SIZE=35,
IF OUT OF SCREEN WRAP AROUND
120 A=ANIM(1,,0,0,0,X*3.5,Y*3.5,
3)
130 A$=INKEY$:IF A$="" THEN 130
140 PCLS
150 'MOVE FIGURE #1. RELATIVE MO
VEMENT OF +5 IN X AND 0 IN Y
160 X=ANIM(2,1,1,5,0)
170 GOTO 160
```

```
170 .......77
320 .......155
500 .......177
720 .........1
END .....210
```

Listing 2: PROMNADE

```
10 'SPACE PROMENADE WITH ANIMATI
C
20 GOTO 420
30 'DEFINE ANIMATIC. #FIGS=D+SPA
CESHIP+BOMB (D=#ASTEROIDS)
40 A=ANIM(0,D+2)
50 'CREATE SPACESHIP. NO MIX. PO
SITIONS X=0/Y=0, SIZE=35/20. IF
OUT OF SCREEN FREEZE
60 A=ANIM(1,,0,0,5,X*3.5,Y*2,1)
70 'CREATE ASTEROIDS. NO MIX. FR
OM POSITION 95,95. SIZE 11/11. I
F OUT OF SCREEN WRAP-AROUND
80 FOR I=1 TO D
90 A=ANIM(1,,0,95,95,11,11,3)
100 NEXT
110 'CREATE BOMB. NO MIX. FROM P
OSITION 200,184. SIZE=6 X 6. IF
OUT OF SCREEN WRAP
120 A=ANIM(1,,0,200,184,6,6,3)
130 'PREPARE SCREEN'S BACKGROUND
 (PLANET + STARS)
140 PCLS:CIRCLE(255,191),10:PAIN
T(250,189),1,1
150 FOR I=1 TO 60:PSET(RND(255),
RND(191),1):NEXT:SCREEN 1,1
160 ' SET ORIGINAL ASTEROIDS POS
ITIONS.
170 FOR I=2 TO D+1:S=INT(240/D)*
(I-1):X=ANIM(2,I,0,S,0):NEXT
180 'MAIN LOOP. MOVE SPACESHIP (
OBJECT#1). F2 CAN BE 4 IF JOYSTI
CK OR 7 IF KEYBOARD. F3 AND F4 A
RE X AND Y INCREMENTS
190 R=ANIM(2,1,F2,F3,F4):GOSUB31
0:'GO TO CHECK FOR CRASH
200 'MOVE ASTEROIDS. RELATIVE WI
TH X AND Y INCREMENTS DEPENDING
ON THE NUMBER OF THE OBJECT
210 FOR I=2 TO D+1:XA=ANIM(2,I,1
,&HFE,8+I*2):NEXT
220 'MOVE BOMB. RANDOM X=RND(30)
+150, Y=RND(151)+20
230 RA=ANIM(2,D+2,6,30,151,150,2
0)
240 'CHECK FOR CRASH
250 GOSUB 310
260 'REMOVE BOMB TO PREVENT OVER
LAPS WITH ASTEROIDS
270 RA=ANIM(4,D+2)
280 'IF R=10 THEN SPACESHIP REAC
HED BOTTOM/RIGHT CORNER
290 IF R<>10 THEN 190 ELSE 370
300 'CHECK IF SPACESHIP IS IN SA
ME DOMAIN THAT ANY OF THE ASTERO
IDS OR BOMB
310 FOR I=2 TO D+2:XA=ANIM(7,I,1
):IF XA<>0 THEN GOTO 360 ELSE NE
XT:RETURN
320 'OPERATE THE CRASHING ASTERO
ID BY CLEARING IT TO RED (TO SIM
ULATE FIRING)
330 A=ANIM(6,I,0,&HAA)
340 'MAKE SOUNDS AND FLASH SCREE
N
350 'ALMOST ALL THE CODE FROM HE
RE TO THE END IS COSMETIC
360 FOR I=1 TO 3:PLAY"T100;01:F#
C":SCREEN 1,0:FOR J=1 TO20:NEXT:
SCREEN 1,1:NEXT:W=0:GOTO 380
370 FOR I=1 TO 2:PLAY"T250CDEFG"
:NEXT:PLAY "O3;L4;C;L2;D;A":W=8
380 CLS(W):PRINT@290,"";:INPUT "
ANOTHER GAME (Y/N)";A$
390 IF A$="N" THEN CLS:PRINT"I'L
L SEE YOU LATER":END
400 FL=1:GOTO 440
410 'INITIALIZE
420 PCLEAR 4
430 PMODE 4,1
440 PCLS
450 X=10:Y=10
460 DRAW "BM10,10; S2;H10;R15;F1
0;R20;F10;G10;L20;G10;L15;E10;U2
0;D4;NL8;D4;NL12;D4NL16;D4;NL12;
D4;NL8"
470 PAINT (12,12),1,1
480 CIRCLE (100,100),5
490 LINE (200,180)-(205,185),PSE
T,BF
500 'IF NOT FIRST TIME CONTINUE
510 IF FL=1 THEN 40
520 'SHOW PRESENTATION SCREEN
530 CLS(0)
540 'PRINTPEEK(&HFF00):A$=INKEY$
:IF A$="" THEN 301 ELSE POKE &HF
F02,&H00:PRINTPEEK(&HFF00):END
550 PRINT@8,"space";:PRINT@14,"p
romenade";
560 PRINT@64,"a";:PRINT@66,"grap
hics";:PRINT@75,"ANIMATIC";:PRIN
T@84,"program";
570 PRINT@106,"by";:PRINT@109,"r
ita";:PRINT@114,"sabo";
580 FOR I=0 TO 63:SET(I,10,7):SE
T(I,31,7):NEXT
590 FOR I=10 TO 31:SET(0,I,7):SE
T(63,I,7):NEXT
600 PRINT@230,"INSTRUCTIONS (Y/N
)?";
610 A$=INKEY$:IF A$="" THEN 610
620 IF A$<>"Y" THEN GOTO 770
630 'PRESENT INSTRUCTIONS
640 PRINT@230,STRING$(20,CHR$(12
8));
650 TX$(0)="your mission is to m
aneuver  "
660 TX$(1)="the spaceship thru t
he meteors"
670 TX$(2)="rain and successfuly
 cross the"
680 TX$(3)="contact bomb barrier
 to safely"
690 TX$(4)="arrive on the planet
 earth in"
700 TX$(5)="the bottom right--go
od luck!!"
710 TX$(6)="  press ENTER to con
tinue  "
720 PO=225:FOR I=0 TO 6:FOR J=1T
O 30:A$=MID$(TX$(I),J,1):IF A$="
 " THEN A$="":PLAY"T25004D"  ELS
E PLAY"T250L101C"
730 PRINT@PO,A$;:FORH=1TO10:NEXT
:PO=PO+1:GOSUB860:FOR K=1 TO 50:
NEXT:NEXT:PO=PO+2:IF I=5 THEN PO
=PO+32
740 NEXT
750 A$=INKEY$:IF A$="" THEN GOSU
B 860:GOTO 750
760 'PRESENT GAME OPTIONS
770 CLS(5):PRINT@290,"";:INPUT "
HOW MANY ASTEROIDS";D
780 IF D<1 THEN 770 ELSE IF D>6
THEN SOUND 1,1:PRINT@362,"MASOCH
ISTIC?!!";:PRINT@384,"above 6 is
 too much even for you":FORI=1 T
O 1500:NEXT:GOTO770
790 PRINT@360,"jOYSTICK/kEYBOARD
";
800 PRINT@389,"(WITH JOYSTICK IS
 EASIER)";
810 A$=INKEY$:IF A$="" THEN 810
820 IF A$<>"K" AND A$<>"J" THEN
SOUND 1,1:GOTO 790
830 SOUND 200,1
840 IF A$="K" THEN F2=7:F3=8:F4=
8 ELSE F2=4:F3=4:F4=4
850 GOTO 40
860 IF SW=0 THEN PRINT@75,"ANIMA
TIC"; ELSE PRINT@75,STRING$(8,CH
R$(128));
870 SW=NOT SW:RETURN
```

```
170 .......53
400 .......46
END .....100
```

Listing 3: ANIMCALC

```
10 'THIS PROGRAM WILL CALCULATE
THE REQUIRED SIZES FOR ANIMATIC'
```

```
S OBJECTS.
20 'IT WILL ALSO SUGGEST A START
 ADDRESS FOR ANIMATIC'S CODE
30 CLS:PRINT"ANIMATIC'S WORK ARE
AS SIZES"
40 INPUT"NUMBER OF OBJECTS";OB
50 IF OB<1 OR OB>255 THEN 40
60 DIM OB(OB),PM(OB),X(OB),Y(OB)
,MX$(OB)
70 FOR I=1 TO OB
80 CLS(5)
90 PRINTTAB(20);"object #";:PRIN
TUSING"###";I
100 INPUT "OBJECT TO BE MIXED (Y
/N)";MX$
110 IF MX$<>"Y" AND MX$<>"N" THE
N 100
120 INPUT "PMODE (Ø-4)";PM
130 IF PM<Ø OR PM>4 THEN 120
140 INPUT "WIDTH IN PIXELS (1-10
Ø)";X
150 IF X<1 OR X>100 THEN 140
160 INPUT "HEIGHT IN PIXELS (1-1
ØØ)";Y
170 IF MX$="Y" THEN MX=1 ELSE MX
=Ø
180 PM(I)=PM:X(I)=X:Y(I)=Y:MX$(I
)=MX$
190 IF Y<1 OR Y>100 THEN 160
200 IF INT(X/2)<>X/2 AND PM<>4 T
HEN X=X+1
210 IF PM=Ø OR PM=2 THEN X=INT(X
/2)
220 X=INT(X/8)
230 RM=7-X
240 X=X+1:IF RM>1 THEN X=X+1
250 IF (Y/2 <> INT(Y/2)) AND PM<
2 THEN Y=Y+1
260 IF PM<2 THEN Y=INT(Y/2)
270 T=X*Y
280 IF MX=1 THEN T=T*2
290 OB(I)=T
300 TT=TT+T
310 NEXT
320 CLS(7)
330 INPUT "ACTUAL OFFSET OF ANIM
ATIC";OF
340 INPUT"OUTPUT TO PRINTER";DV$
350 IF DV$="Y" THEN DV=-2 ELSE D
V=Ø
360 CLS
370 PRINT"OBJ# PMODE MIX    X
Y  BYTES"
380 FOR I=1 TO OB
390 PRINT#DV,USING"###     ";I;:PR
INT#DV,USING" #     ";PM(I);:PRINT
#DV," ";MX$;" ";:PRINT#DV,USING"
 ### ";X(I);:PRINT#DV,USING" ###
 ";Y(I);:PRINT#DV,USING" ####";O
B(I)
400 NEXT
410 PRINT#DV:PRINT#DV,TAB(11),"S
WAPS==>";:PRINT#DV,USING" ####";
TT
420 PRINT#DV:PRINT#DV,"         RE
QUIRED FDTS ==>";:PRINT#DV,USING
" ####";OB*24
430 TX=TT+OB*24
440 PRINT#DV:PRINT#DV,TAB(10),"T
OTAL ==>";:PRINT#DV,USING"####";
TX
450 PRINT#DV," ":PRINT#DV,"YOU C
AN RELOCATE ANIMATIC AT"
460 SZ=PEEK(&H74)*256+PEEK(&H75)
:AD=SZ-TX-2800
470 PRINT#DV,"ADDRESS: *";AD;"(H
EX=";HEX$(AD);") *"
480 AJ=AD-OF:IF AJ<Ø THEN AJ=&HF
FFF+AJ+1
490 PRINT#DV,"MAKE:":PRINT#DV,"
LOADM 'ANIMATIC',";"&H"+HEX$(AJ)
+","+"&H"+HEX$(AJ+2800)+","+"&H"
+HEX$(AJ)
500 GOTO 500
```

Listing 4: ANIMATIC

```
                 00010 *-- ANIMATIC. (C) 1983 BY RITA SABO --*
                 00020 * BAS IS THE ROUTINE THAT HANDLES ANIM INST. *
          0000   00030 BAS      EQU     *
0000 CE   0139   00040          LDU     #$139
0003 A6   C4     00050          LDA     ,U
0005 81   02     00060          CMPA    #2
0007 2D   09     00070          BLT     NODSK
0009 33   4A     00080          LEAU    10,U
000B 6F   5B     00090          CLR     -5,U
000D 8E   8277   00100          LDX     #$8277
0010 AF   5E     00110          STX     -2,U
0012 86   01     00120 NODSK    LDA     #1
0014 A7   C0     00130          STA     ,U+
0016 30   8D 008E 00140         LEAX    BANIM,PCR
001A AF   C1     00150          STX     ,U++
001C 30   8D 000E 00160         LEAX    DUHEX,PCR
0020 AF   C1     00170          STX     ,U++
0022 8E   8277   00180          LDX     #$8277
0025 AF   43     00190          STX     3,U
0027 AF   48     00200          STX     8,U
0029 6F   40     00210          CLR     0,U
002B 6F   45     00220          CLR     5,U
002D 39          00230          RTS
          002E   00240 DUHEX    EQU     *
002E C0   44     00250          SUBB    #68      (80 IF DISK)
0030 B6   0139   00260          LDA     $139
0033 81   01     00270          CMPA    #1
0035 27   02     00280          BEQ     DUH2
0037 C0   0C     00290          SUBB    #12
0039 34   04     00300 DUH2     PSHS    B
003B BD   826A   00310          JSR     $826A
003E 35   04     00320          PULS    B
0040 BD   873D   00330          JSR     $873D
0043 1F   10     00340          TFR     X,D
0045 31   8D 09D9 00350         LEAY    PARMS,PCR
0049 E7   A0     00360          STB     ,Y+
004B 86   07     00370          LDA     #7
004D 34   22     00380 BMORE    PSHS    A,Y
004F BD   826D   00390          JSR     $826D
```

```
0052 E6   9F 00A6 00400         LDB     >[$A6]
0056 C1   2C     00410          CMPB    #',
0058 27   2C     00420          BEQ     BZERO
005A C1   29     00430          CMPB    #')
005C 26   06     00440          BNE     BL2
005E 86   01     00450          LDA     #1
0060 A7   E4     00460          STA     ,S
0062 20   22     00470          BRA     BZERO
          0064   00480 BL2      EQU     *
0064 BE   00A6   00490          LDX     >$A6
0067 86   08     00500          LDA     #8
0069 E6   80     00510 BL12     LDB     ,X+
006B C1   2C     00520          CMPB    #',
006D 27   10     00530          BEQ     BL3
006F C1   29     00540          CMPB    #')
0071 26   06     00550          BNE     BL22
0073 86   01     00560          LDA     #1
0075 A7   E4     00570          STA     ,S
0077 20   06     00580          BRA     BL3
0079 4A          00590 BL22     DECA
007A 26   ED     00600          BNE     BL12
007C 7E   8277   00610          JMP     $8277
007F BD   873D   00620 BL3      JSR     $873D
0082 1F   10     00630          TFR     X,D
0084 20   01     00640          BRA     BOUTL
0086 5F          00650 BZERO    CLRB
0087 35   22     00660 BOUTL    PULS    A,Y
0089 E7   A0     00670          STB     ,Y+
008B 4A          00680          DECA
008C 26   BF     00690          BNE     BMORE
008E BD   8267   00700          JSR     $8267
0091 86   01     00710          LDA     #1
0093 A7   8D 0998 00720         STA     5+FCTAB,PCR
0097 31   8D 0987 00730         LEAY    PARMS,PCR
009B A6   A0     00740          LDA     ,Y+
009D 81   08     00750          CMPA    #8
009F 1026 004F   00760          LBNE    ANIM
00A3 EC   A4     00770          LDD     ,Y
00A5 16   0008   00780          LBRA    INFO
00A8      41     00790 BANIM    FCC     /ANI/
00AB      CD     00800          FCB     $CD
```

CORRECTIONS

"PERT" (April 1985, Page 24) : Jorge Mir tells us he's had some reports of problems having to do with various printers. PERT was written for the Okidata Microline 92 printer, and these special effects codes are used:

CHR$(12)	Feed paper to beginning of next page (most printers have this)
CHR$(28)	Select elite font (96 chars/line)
CHR$(29)	Select compressed font (132 chars/line)
CHR$(30)	Select normal font (80 chars/line)
CHR$(31)	Switches on double-emphasized mode

If you have some other printer, you will need to change the printer codes contained in lines 1740, 1800, 1810, 2320, 2330, 2470, 2480 and 2500 to make the special modes work with your printer. If your printer does not have the elite (96 characters per line) font, the compressed font will work. Also, on most other printers you will need to use two modes (emphasized and double-strike) in combination to create the double-emphasized mode.

The Okidata printers automatically clear the double-emphasized mode when changing fonts; if your printer doesn't, you will need to insert the necessary codes as well.

If your printer doesn't have the form feed function, change the following two lines to read as follows:

```
1800 IF INT(I/58) = I/58 THEN FO
R XX=1TO6:PRINT#-2,"":NEXTX
1810 NEXT I
```

Finally, all users should change the word PAINTRICAL in Line 2400 to read CRITICAL.

```
00AC                00810             RMB    4        TO MAKE INFO=BAS+$BU
                    00820 ************
                    00830 * INFO.
                    00840 *  ON ENTRY: D REGISTER WITH OPTIONS
                    00850 *   ON EXIT: D WITH ADDRESS OF FDT, PARMS OR XY
                    00860 ************
          0080      00870 INFO    EQU    *
0080 10EF 8D 0939   00880         STS    SAVSTK,PCR
0085 30   8D 0969   00890         LEAX   PARMS,PCR
0089 5D             00900         TSTB            PARMS ADDR?
008A 26   0D        00910         BNE    ADFI     FDT ADDR
008C 34   10        00920         PSHS   X
008E 86   08        00930         LDA    #8       CLEAR PARMS
00C0 6F   80        00940 MCLPA   CLR    ,X+
00C2 4A             00950         DECA
00C3 26   FB        00960         BNE    MCLPA
00C5 35   06        00970         PULS   D
00C7 20   1E        00980         BRA    EXINF
00C9 34   02        00990 ADFI    PSHS   A
00CB A6   01        01000         LDA    1,X
00CD 34   02        01010         PSHS   A
00CF E7   01        01020         STB    1,X
00D1 17   0773      01030         LBSR   GETFDT
00D4 35   02        01040         PULS   A
00D6 A7   01        01050         STA    1,X
00D8 35   02        01060         PULS   A
00DA 4D             01070         TSTA            XY-COORD.
00DB 26   08        01080         BNE    ADFI2
00DD A6   C8 08     01090         LDA    <AUTOX,U       X-COORD
00E0 E6   C8 09     01100         LDB    <AUTOY,U       Y-COORD
00E3 20   02        01110         BRA    EXINF
00E5 1F   30        01120 ADFI2   TFR    U,D      RESULT
00E7 6D   8D 0944   01130 EXINF   TST    5+FCTAB,PCR    RETURN TO ML?
00EB 27   03        01140         BEQ    RETINF
00ED 7E   B4F4      01150         JMP    $B4F4    TO BASIC
00F0 39             01160 RETINF  RTS             TO ML
00F1 12             01170         NOP             ANIM=INFO+$40
                    01180 *----------------------------*
                    01190 * ANIMATIC. ASSEMBLY ENTRY POINT*
                    01200 *    ON ENTRY: PARMS SET
                    01210 *    ON EXIT: X= ADDR. OF PARMS  *
                    01220 *----------------------------*
          00F2      01230 ANIM    EQU    *
00F2 10EF 8D 08F7   01240         STS    SAVSTK,PCR     SAVE STACK ADDRESS
00F7 32   8D 0926   01250         LEAS   49+STACK,PCR   NEW STACK ADDRESS
00FB 6F   8D 08B9   01260         CLR    STATUS,PCR     CLEAR STATUS AREA
00FF 6F   8D 08B6   01270         CLR    1+STATUS,PCR
0103 30   8D 091B   01280         LEAX   PARMS,PCR      LOAD PARMS ADDR.
0107 A6   84        01290         LDA    ,X       GET REQUESTED FUNCTION
0109 A7   8D 089B   01300         STA    ACTION,PCR     SAVE REQUESTED ACTION
010D C6   08        01310         LDB    #8       CHECK FOR ACTION
010F E1   84        01320         CMPB   ,X
0111 22   05        01330         BHI    GRA2     IF,OK CONTINUE
0113 C6   03        01340         LDB    #XIO     ELSE ERROR
0115 16   0815      01350         LBRA   ERROR
0118 C6   03        01360 GRA2    LDB    #3       GET DISPLACEMENT
011A 3D             01370         MUL
011B 31   8D 0002   01380         LEAY   CALLS,PCR      ADDR. OF CALL LIST
011F 6E   A5        01390         JMP    B,Y      GO TO APPROPIATE CALL
          0121      01400 CALLS   EQU    *
0121 16   0015      01410         LBRA   INIT     INITIALIZE
0124 16   003B      01420         LBRA   CREATE   CREATE FIGURE
0127 16   00F9      01430         LBRA   MOVE     MOVE FIGURE
012A 16   00F6      01440         LBRA   PLACE    PLACE FIG. ON SCREEN
012D 16   02B8      01450         LBRA   REMOVE   ERASE FIG. FROM SCREEN
0130 16   02C4      01460         LBRA   COPYFI   COPY FIGURE
0133 16   0372      01470         LBRA   OPERAT   OPERATE SWAP AREA WITH FUNCTION
0136 16   0320      01480         LBRA   DOMAIN   FIND IN A DOMAIN
                    01490 *------------------------*
                    01500 *INITIALIZE FCT (FIGURE  *
                    01510 * CONTROL TABLE) ACT. 0  *
                    01520 *  ON ENTRY: X-ADDR. OF  *
                    01530 *          PARMS         *
                    01540 *------------------------*
          0139      01550 INIT    EQU    *
0139 A6   01        01560         LDA    1,X      NUMBER OF FIGS.
013B 26   05        01570         BNE    INI2     NUMBER OF FIGS.
013D C6   02        01580         LDB    #XOF     CAN'T BE ZERO
013F 16   07EB      01590         LBRA   ERROR
0142 A7   8D 08E4   01600 INI2    STA    FCTAB,PCR
0146 C6   18        01610         LDB    #24      GET ADDRESS FOR SWAP AREA
0148 3D             01620         MUL             #BYTES FOR FFDT'S
0149 31   8D 08E3   01630         LEAY   FFDT,PCR       FIRST FFDT ADDRESS
014D 34   20        01640         PSHS   Y
014F E3   E1        01650         ADDD   ,S++     ADD TO BYTES FOR FFDT'S
0151 ED   8D 08D8   01660         STD    3+FCTAB,PCR    ADDR. OF FIRST SWAP AREA
0155 6F   8D 08D2   01670         CLR    1+FCTAB,PCR    NUMBER OF CREATED FIGS.
0159 86   AA        01680         LDA    #$AA     INIT FLAG
015B A7   8D 08CD   01690         STA    2+FCTAB,PCR
015F 16   082E      01700         LBRA   EXIT
                    01710 *------------------------*
                    01720 * CREATE FIGURE (ACT. 1) *
                    01730 * ON ENTRY: X= ADDR. OF  *
                    01740 *         PARMLIST       *
                    01750 *------------------------*
          0162      01760 CREATE  EQU    *
0162 31   8D 08C4   01770         LEAY   FCTAB,PCR      ADDR. OF FCT
0166 6C   8D 08C1   01780         INC    1+FCTAB,PCR    NUMBER OF FIGS CREATED
016A A6   8D 08BD   01790         LDA    1+FCTAB,PCR
016E A7   01        01800         STA    1,X      STORE IT IN PARMS
0170 17   06D4      01810         LBSR   GETFDT
0173 EC   A8 03     01820         LDD    <NEXTSW,Y      NEXT SWAP AREA
0176 ED   C8 00     01830         STD    <ASW,U   STORE IT IN FDT
0179 A6   03        01840         LDA    3,X      X-COORD.
017B A7   8D 0825   01850         STA    CADX,PCR
017F A6   04        01860         LDA    4,X      Y-COORD.
0181 A7   8D 0821   01870         STA    CADY,PCR
0185 A6   07        01880         LDA    7,X      ACTION IF OUT-OF-SCREEN
0187 A7   C8 10     01890         STA    <OUTSCR,U
018A E6   05        01900         LDB    5,X      #COLS.
018C 27   04        01910         BEQ    CER1     =0?
018E C1   65        01920         CMPB   #101     MASX. NUMBER OF X PIXELS
0190 25   05        01930         BLO    CRA      OK
0192 C6   04        01940 CER1    LDB    #XEX     ELSE ERROR
0194 16   0796      01950         LBRA   ERROR
0197 E7   C8 04     01960 CRA     STB    <WIDTH,U
019A 1F   98        01970         TFR    B,A      USE A REG.
019C 84   01        01980         ANDA   #1       TO SEE IF ODD NUMBER
```

```
019E 27   0A        01990         BEQ    CR1      IF EVEN CONTINUE
01A0 96   B6        02000         LDA    <$B6     ELSE GET PMODE
01A2 81   04        02010         CMPA   #4       IF PMODE 4
01A4 27   04        02020         BEQ    CR1      DO NOTHING
01A6 6C   C8 04     02030         INC    <WIDTH,U        ELSE ROUND-UP WIDTH
01A9 5C             02040         INCB
01AA 17   0668      02050 CR1     LBSR   CMAXBY   FIND MAX. # OF BYTES
01AD A6   06        02060         LDA    6,X      #ROWS
01AF 27   04        02070         BEQ    CER2     =0?
01B1 81   65        02080         CMPA   #101     MAX. NUMBER OF Y PIXELS
01B3 25   05        02090         BLO    CRB      OK
01B5 C6   05        02100 CER2    LDB    #XEY     ELSE ERROR
01B7 16   0773      02110         LBRA   ERROR
01BA A7   C8 05     02120 CRB     STA    <HEIGHT,U
01BD 84   01        02130         ANDA   #1       TO SEE IF ODD NUMBER
01BF 27   09        02140         BEQ    CR2      IF EVEN CONTINUE
01C1 96   B6        02150         LDA    <$B6     ELSE GET PMODE
01C3 81   01        02160         CMPA   #1       IF PMODES 2,3,4
01C5 22   03        02170         BHI    CR2      DO NOTHING
01C7 6C   C8 05     02180         INC    <HEIGHT,U       ELSE ROUND-UP HEIGHT
01CA 17   0664      02190 CR2     LBSR   NORMY    FIND #BYTES FOR ROWS
01CD 3D             02200         MUL             TOTAL BYTES FOR FIGURE
01CE ED   C9 0012   02210         STD    FIGBYT,U
01D2 6D   02        02220         TST    2,X      MIXABLE?
01D4 27   0C        02230         BEQ    CR3      NO
01D6 AE   A8 03     02240         LDX    <NEXTSW,Y
01D9 30   8B        02250         LEAX   D,X      POINT TO SWAP FOR MIX.
01DB AF   C8 02     02260         STX    <ORFLAG,U
01DE 58             02270         ASLB            MULTIPLY BYTES BY 2
01DF 49             02280         ROLA
01E0 20   03        02290         BRA    CR32
01E2 6F   C8 02     02300 CR3     CLR    <ORFLAG,U       NO-MIX
01E5 AE   A8 03     02310 CR32    LDX    <NEXTSW,Y
01E8 6F   80        02320 CR4     CLR    ,X+      CLEAR TO ZERO
01EA 83   0001      02330         SUBD   #1       NUMBER OF BYTES FOR NEW SWAP
01ED 26   F9        02340         BNE    CR4      NOT DONE YET
01EF AF   A8 03     02350         STX    <NEXTSW,Y       TO NEXT SWAP AREA
01F2 9C   74        02360         CMPX   $74      EXCEEDS AVAILABLE MEMORY?
01F4 23   05        02370         BLS    CRC      NO, OK
01F6 C6   01        02380         LDB    #XOM     ELSE ERROR
01F8 16   0732      02390         LBRA   ERROR
          01FB      02400 CRC     EQU    *
01FB 17   038A      02410         LBSR   MOVGEN
01FE 4F             02420         CLRA
01FF 17   04FE      02430         LBSR   SWAP2    GET FIGURE
0202 6D   C8 02     02440         TST    <ORFLAG,U       OK-ABLE?
0205 27   13        02450         BEQ    ENDCRE   ...NO EXIT
0207 A6   C8 0A     02460         LDA    <LMASK,U
020A A7   C8 0C     02470         STA    <OLMASK,U
020D A6   C8 0B     02480         LDA    <RMASK,U
0210 A7   C8 0D     02490         STA    <ORMASK,U
0213 A6   C8 0E     02500         LDA    <WIDBYT,U
0216 A7   C9 0014   02510         STA    OWID,U
021A 86   01        02520 ENDCRE  LDA    #1       FLAG AS NEW
021C A7   C9 0011   02530         STA    FLAGCR,U
0220 16   076D      02540         LBRA   EXIT
                    02550 *------------------------*
                    02560 *   MOVE FIGURE (ACT. 2) *
                    02570 * ON ENTRY: X= ADDR. OF  *
                    02580 *          PARMLIST      *
                    02590 *------------------------*
          0223      02600 MOVE    EQU    *
          0223      02610 PLACE   EQU    *        PLACE ALSO BEGINS HERE
0223 6F   8D 077C   02620         CLR    CACX,PCR
0227 6F   8D 077A   02630         CLR    CACY,PCR
022B 17   0619      02640         LBSR   GETFDT
022E 6D   C9 0011   02650         TST    FLAGCR,U        FIG. JUST CREATED?
0232 27   0A        02660         BEQ    MO1      NO GO AHEAD
0234 86   03        02670         LDA    #3       FORCE ACTION 3 (PLACE)
0236 A7   8D 076E   02680         STA    ACTION,PCR
023A 6F   C9 0011   02690         CLR    FLAGCR,U        DELETE JUST CREATED FLAG
          023E      02700 MO1     EQU    *
023E A6   02        02710         LDA    2,X
0240 26   0F        02720         BNE    SPEMOV
0242 E6   03        02730         LDB    3,X
0244 E7   8D 075C   02740         STB    CADX,PCR        X-DEST
0248 E6   04        02750         LDB    4,X
024A E7   8D 0758   02760         STB    CADY,PCR        Y-DEST
024E 16   014D      02770         LBRA   XMOVE
0251 81   01        02780 SPEMOV  CMPA   #1
0253 26   25        02790         BNE    RNDXY
0255 4F             02800         CLRA
0256 E6   C8 08     02810         LDB    <AUTOX,U        RELATIVE MOVEMENT
0259 34   06        02820         PSHS   D
025B E6   03        02830         LDB    3,X      GET INCR. IN X
025D 2A   01        02840         BPL    CONX     NEGATIVE?
025F 43             02850         COMA
0260 E3   E1        02860 CONX    ADDD   ,S++     ADD IT
0262 ED   8D 073D   02870         STD    CACX,PCR        X-DEST.
0266 4F             02880         CLRA
0267 E6   C8 09     02890         LDB    <AUTOY,U        UPDATE Y
026A 34   06        02900         PSHS   D
026C E6   04        02910         LDB    4,X      GET INCR. IN Y
026E 2A   01        02920         BPL    CONY     NEGATIVE?
0270 43             02930         COMA
0271 E3   E1        02940 CONY    ADDD   ,S++     ADD IT
0273 ED   8D 072E   02950         STD    CACY,PCR        Y-DEST.
0277 16   0124      02960         LBRA   XMOVE
027A 81   06        02970 RNDXY   CMPA   #6       IS X-Y RANDOM REQUESTED?
027C 26   25        02980         BNE    NORAN
027E A6   03        02990         LDA    3,X      GET MAX. ALLOWED COLUMN
0280 26   04        03000         BNE    NODFX    IF NOT ZERO CONTINUE
0282 86   FF        03010         LDA    #255     ELSE PUT DEFAULT
0284 6F   05        03020         CLR    5,X
0286 17   05F4      03030 NODFX   LBSR   RANDOM   GET RANDOM VALUE
0289 EB   05        03040         ADDB   5,X
028B E7   8D 0715   03050         STB    CADX,PCR        AND STORE NEW X-COORD.
028F A6   04        03060         LDA    4,X      MAX. ALLOWED ROW
0291 26   04        03070         BNE    NODFY    IF NOT ZERO CONTINUE
0293 86   BF        03080         LDA    #191     ELSE PUT DEFAULT
0295 6F   06        03090         CLR    6,X
0297 17   05E3      03100 NODFY   LBSR   RANDOM   GET RANDOM VALUE
029A EB   06        03110         ADDB   6,X
029C E7   8D 0706   03120         STB    CADY,PCR        NEW Y-COORD.
02A0 16   00FB      03130         LBRA   XMOVE    CONTINUE
02A3 102D 0080      03140 NORAN   LBLT   RDJOY
02A7 81   07        03150         CMPA   #7
02A9 26   6B        03160         BNE    STAY
```

```
02AB 86    BF        03170          LDA    #$BF     KEYBOARD CONTROL
02AD B7    FF02      03180          STA    $FF02
02B0 4F              03190          CLRA
02B1 E6    C8 08     03200          LDB    <AUTOX,U
02B4 34    06        03210          PSHS   D
02B6 E6    03        03220          LDB    3,X
02B8 B6    FF00      03230          LDA    $FF00
02BB 81    F7        03240          CMPA   #247
02BD 26    07        03250          BNE    KBXP2
02BF 4F              03260          CLRA
02C0 E3    E4        03270          ADDD   ,S
02C2 ED    E4        03280          STD    ,S
02C4 E6    03        03290          LDB    3,X
02C6 77    FF02      03300 KBXP2    ASR    $FF02
02C9 B6    FF00      03310          LDA    $FF00
02CC 81    F7        03320          CMPA   #247
02CE 26    0A        03330          BNE    KBXEX
02D0 4F              03340          CLRA
02D1 50              03350          NEGB
02D2 27    02        03360          BEQ    KB2X
02D4 86    FF        03370          LDA    #$FF
02D6 E3    E4        03380 KB2X     ADDD   ,S
02D8 ED    E4        03390          STD    ,S
02DA EC    E1        03400 KBXEX    LDD    ,S++
02DC ED    8D 06C3   03410          STD    CACX,PCR
02E0 4F              03420          CLRA
02E1 E6    C8 09     03430          LDB    <AUTOY,U
02E4 34    06        03440          PSHS   D
02E6 E6    04        03450          LDB    4,X
02E8 77    FF02      03460          ASR    $FF02
02EB B6    FF00      03470          LDA    $FF00
02EE 81    F7        03480          CMPA   #247
02F0 26    07        03490          BNE    KBYP2
02F2 4F              03500          CLRA
02F3 E3    E4        03510          ADDD   ,S
02F5 ED    E4        03520          STD    ,S
02F7 E6    04        03530          LDB    4,X
02F9 77    FF02      03540 KBYP2    ASR    $FF02
02FC B6    FF00      03550          LDA    $FF00
02FF 81    F7        03560          CMPA   #247
0301 26    0A        03570          BNE    KBYEX
0303 4F              03580          CLRA
0304 50              03590          NEGB
0305 27    02        03600          BEQ    KB2Y
0307 86    FF        03610          LDA    #$FF
0309 E3    E4        03620 KB2Y     ADDD   ,S
030B ED    E4        03630          STD    ,S
030D EC    E1        03640 KBYEX    LDD    ,S++
030F ED    8D 0692   03650          STD    CACY,PCR
0313 16    0088      03660          LBRA   XMOVE
0316 A6    C8 08     03670 STAY     LDA    <AUTOX,U
0319 A7    8D 0687   03680          STA    CADX,PCR
031D A6    C8 09     03690          LDA    <AUTOY,U
0320 A7    8D 0682   03700          STA    CADY,PCR
0324 16    00BB      03710          LBRA   MO3
0327 34    52        03720 RDJOY    PSHS   U,X,A
0329 AD    9F A00A   03730          JSR    [$A00A] READ JOYSTICK
032D 35    52        03740          PULS   U,X,A   RESTORE
032F 108E  015A      03750          LDY    #$015A  ADDRESS OF VALUES
0333 81    02        03760          CMPA   #2      IF LEFT JOYST.
0335 27    06        03770          BEQ    EXPAND
0337 81    04        03780          CMPA   #4      LEFT JOYSTK?
0339 27    02        03790          BEQ    EXPAND  YES
033B 31    22        03800          LEAY   2,Y     POINT TO RIGHT JOYS VALUES
033D 80    04        03810 EXPAND   SUBA   #4      XY INDICATOR-4
033F 2A    1C        03820          BPL    INCJY   JOYSTK. INCREMENT
0341 68    A4        03830          LSL    ,Y      MULTIPLY X-READING BY 4
0343 68    A4        03840          LSL    ,Y
0345 68    A4        03850          LSL    ,Y
0347 68    A4        03860          LSL    ,Y
0349 68    21        03870          LSL    1,Y     MULTIPLY Y READING BY 3
034B 68    21        03880          LSL    1,Y
034D 68    21        03890          LSL    1,Y-
034F E6    A4        03900          LDB    ,Y      GET Y-COORD
0351 E7    8D 064F   03910          STB    CADX,PCR        STORE IT
0355 E6    21        03920          LDB    1,Y     GET     Y-COORD
0357 E7    8D 064B   03930          STB    CADY,PCR        STORE IT
035B 20    41        03940          BRA    XMOVE
           035D      03950 INCJY    EQU    *
035D A6    A4        03960          LDA    ,Y      GET X READING
035F 80    20        03970          SUBA   #32     MINUS 32
0361 A7    A4        03980          STA    ,Y
0363 A6    21        03990          LDA    1,Y     NOW Y-READING
0365 80    20        04000          SUBA   #32
0367 A7    21        04010          STA    1,Y
0369 86    03        04020          LDA    #3      NOW DIVIDE BY 8
036B 67    A4        04030 DIVCN    ASR    ,Y
036D 67    21        04040          ASR    1,Y
036F 4A              04050          DECA
0370 26    F9        04060          BNE    DIVCN
0372 A6    03        04070          LDA    3,X     GET X MULTIPLIER
0374 E6    A4        04080          LDB    ,Y      AND X READING
0376 3D              04090          MUL
0377 4D              04100          TSTA           CHECK IF RESULT SHOULD BE NEG.
0378 27    02        04110          BEQ    DIV1    POSITIVE
037A 86    FF        04120          LDA    #$FF    MAKE IT NEG.
037C 34    06        04130 DIV1     PSHS   D
037E 4F              04140          CLRA
037F E6    C8 08     04150          LDB    <AUTOX,U        ACTUAL X LOCATION
0382 E3    E1        04160          ADDD   ,S++    NEW LOCATION
0384 ED    8D 061B   04170          STD    CACX,PCR        STORE IT
0388 A6    04        04180          LDA    4,X     Y-MULTIPLIER
038A E6    21        04190          LDB    1,Y     Y-READING
038C 3D              04200          MUL
038D 4D              04210          TSTA           IS RESULT NEGATIVE?
038E 27    02        04220          BEQ    DIV2    NO
0390 86    FF        04230          LDA    #$FF    MAKE IT NEGATIVE
0392 34    06        04240 DIV2     PSHS   D
0394 4F              04250          CLRA
0395 E6    C8 09     04260          LDB    <AUTOY,U        ACTUAL Y LOCATION
0398 E3    E1        04270          ADDD   ,S++    NEW Y POSITION
039A ED    8D 0607   04280          STD    CACY,PCR        STORE IT
           039E      04290 XMOVE    EQU    *
039E 6F    8D 0608   04300          CLR    FLG,PCR
03A2 6F    8D 0608   04310          CLR    1+FLG,PCR       MARK FLAG TO CHECK X
03A6 31    8D 05F9   04320          LEAY   CACX,PCR        ADDR. OF TENTATIVE X
03AA 17    0524      04330          LBSR   ACTOSC  VERIFY X<0 OR X>255
03AD 68    8D 0608   04340          LSL    1+STATUS,PCR    PREPARE FOR Y STATUS FLAG
03B1 68    8D 0604   04350          LSL    1+STATUS,PCR
03B5 86    40        04360          LDA    #64
03B7 A7    8D 05F3   04370          STA    1+FLG,PCR       FLAG TO CKECK Y
03BB 31    8D 05E6   04380          LEAY   CACY,PCR        ADDR. OF TENTATIVE Y
03BF 17    050F      04390          LBSR   ACTOSC  VERIFY Y<0 OR Y>191
03C2 6D    C8 02     04400          TST    <ORFLAG,U       MOVE WITH MIX?
03C5 27    1B        04410          BEQ    MO3     NO
03C7 A6    07        04420          LDA    7,X     IF MOVE WITH MIX
03C9 A7    8D 05E5   04430          STA    BKCOLO,PCR      STORE THE VALUE
03CD C6    03        04440          LDB    #3      COUNTER
           03CF      04450 MO2      EQU    *
03CF 68    8D 05DF   04460          ASL    BKCOLO,PCR      TO CONVERT
03D3 68    8D 05DB   04470          ASL    BKCOLO,PCR      COLOR CODE
03D7 AA    8D 05D7   04480          ORA    BKCOLO,PCR      TO $00,$55,$AA OR $FF
03DB 5A              04490          DECB
03DC 26    F1        04500          BNE    MO2
03DE A7    8D 05D0   04510          STA    BKCOLO,PCR
03E2 17    01A3      04520 MO3      LBSR   MOVGEN  GO TO MOVE FIG.
03E5 16    05A8      04530          LBRA   EXIT
                     04540 *--- --------------------*
                     04550 * REMOVE FIGURE (ACT. 4) *.
                     04560 * ON ENTRY: X= ADDR. OF  *
                     04570 *            PARMLIST    *
                     04580 *------------------------*
           03E8      04590 REMOVE   EQU    *
03E8 17    045C      04600          LBSR   GETFDT  GET FDT ADDRESS
           03EB      04610 REMOV2   EQU    *
03EB 17    0312      04620          LBSR   SWAP2   GO AND REMOVE
03EE 86    01        04630          LDA    #1      MARK AS NEW
03F0 A7    C9 0011   04640          STA    FLAGCR,U
03F4 16    0599      04650          LBRA   EXIT
                     04660 *------------------------*
                     04670 * COPY   FIGURE (ACT. 5) *
                     04680 * ON ENTRY: X= ADDR. OF  *
                     04690 *            PARMLIST    *
                     04700 *------------------------*
           03F7      04710 COPYFI   EQU    *
03F7 E6    01        04720          LDB    1,X
03F9 34    04        04730          PSHS   B
03FB E6    02        04740          LDB    2,X     GET ADDR. OF FFDT FOR FROM-FIG.
03FD E7    01        04750          STB    1,X
03FF 17    0445      04760          LBSR   GETFDT
0402 1F    32        04770          TFR    U,Y     ADDR. OF FFDR
0404 35    04        04780          PULS   B       GET TO-FIG
0406 E7    01        04790          STB    1,X     AND RESTORE
0408 17    043C      04800          LBSR   GETFDT  ADDR. OF FFDT FRO TO-FIG
040B 1F    21        04810          TFR    Y,X     FFDT FOR FROM-FIG
040D C6    08        04820          LDB    #X1C    IF ERROR PREPARE ERROR CODE
040F 10AE 89 0012    04830          LDY    FIGBYT,X        NUMBER OF BYTES IN FIG.
0414 10AC C9 0012    04840          CMPY   FIGBYT,U        IF MORE THAN DESTINATION
0419 1022 0510       04850          LBHI   ERROR
041D A6    88 02     04860          LDA    <ORFLAG,X       IS FROM-FIG. ORABLE?
0420 A1    C8 02     04870          CMPA   <ORFLAG,U       THEY MUST BE SAME CLASS
0423 1026 0506       04880          LBNE   ERROR
0427 34    50        04890          PSHS   U,X
0429 4D              04900          TSTA           IF FIGS ARE MIXABLE
042A 27    08        04910          BEQ    CO2     NO
042C EE    C8 02     04920          LDU    <ORFLAG,U       DEST.
042F AE    88 02     04930          LDX    <ORFLAG,X       ORIGIN
0432 20    06        04940          BRA    CO22
0434 EE    C8 00     04950 CO2      LDU    <ASW,U  DESTINATION SWAP AREA
0437 AE    88 00     04960          LDX    <ASW,X  ORIGIN SWAP AREA
043A 17    048B      04970 CO22     LBSR   COPYSW  COPY AREAS
043D 35    50        04980          PULS   U,X     RESTORE FFDT ADDRESSES
043F 32    7C        04990          LEAS   -4,S    BUT KEEP IN STACK
0441 86    14        05000          LDA    #20     WE'LL COPY FFDT NOW
0443 33    44        05010          LEAU   4,U     EXCEPT FOR SWAP AREA ADDRESS
0445 30    04        05020          LEAX   4,X
0447 10AE 81         05030 CO3      LDY    ,X++    COPYING FFDT
044A 10AF C1         05040          STY    ,U++
044D 4A              05050          DECA
044E 26    F7        05060          BNE    CO3     IF NOT FINISHED CONTINUE
0450 35    50        05070          PULS   U,X     RESTORE FFDT ADDRESSES
0452 6F    C9 0011   05080          CLR    FLAGCR,U        FLAG AS NEW
           0456      05090 EXCOP    EQU    *
0456 16    0537      05100          LBRA   EXIT    THAT'S IT
                     05110 *------------------------*
                     05120 * VERIFY IF FIGURE INSIDE*
                     05130 * AN SPECIFIED DOMAIN    *
                     05140 * ON ENTRY: X= ADDR. OF  *
                     05150 *            PARMLIST    *
                     05160 *------------------------*
           0459      05170 DOMAIN   EQU    *
0459 A6    02        05180          LDA    2,X     IS DOMAIN DIRECTLY GIVEN?
045B 27    21        05190          BEQ    DOM2    YES
045D E6    01        05200          LDB    1,X     NO, DOMAIN BELONGS TO A FIG.
045F 34    04        05210          PSHS   B       GET FFDT FOR THAT FIG.
0461 A7    01        05220          STA    1,X
0463 17    03E1      05230          LBSR   GETFDT
0466 35    04        05240          PULS   B       RESTORE FIG. NUMBER
0468 E7    01        05250          STB    1,X
046A A6    C8 08     05260          LDA    <AUTOX,U        LEFT COL. OF DOMAIN
046D A7    03        05270          STA    3,X
046F AB    C8 04     05280          ADDA   <WIDTH,U        RIGHT COL. OF DOMAIN
0472 A7    05        05290          STA    5,X
0474 A6    C8 09     05300          LDA    <AUTOY,U        TOP ROW OF DOMAIN
0477 A7    04        05310          STA    4,X
0479 AB    C8 05     05320          ADDA   <HEIGHT,U       BOTTOM ROW OF DOMAIN
047C A7    06        05330          STA    6,X
           047E      05340 DOM2     EQU    *       VERIFY IF IN DOMAIN
047E 17    03C6      05350          LBSR   GETFDT  FOR REQUESTED FIG.
0481 5F              05360          CLRB
0482 A6    C8 08     05370          LDA    <AUTOX,U        LEFT COL IS
0485 A1    05        05380          CMPA   5,X
0487 22    17        05390          BHI    OUTDOM  DOMAIN?
0489 AB    C8 04     05400          ADDA   <WIDTH,U
048C A1    03        05410          CMPA   3,X
048E 25    10        05420          BLO    OUTDOM
0490 A6    C8 09     05430          LDA    <AUTOY,U        IS TOP ROW BELOW BOTTOM
0493 A1    06        05440          CMPA   6,X     ROW OF DOMAIN?
0495 22    09        05450          BHI    OUTDOM
0497 AB    C8 05     05460          ADDA   <HEIGHT,U       IS BOTTOM ROW ABOVE TOP
049A A1    04        05470          CMPA   4,X     ROW OF DOMAIN?
049C 25    02        05480          BLO    OUTDOM
049E C6    01        05490          LDB    #1      WELL, INDEED IT TOUCHES DOMAIN
04A0 4F              05500 OUTDOM   CLRA
04A1 ED    8D 0513   05510          STD    STATUS,PCR      LET IT BE KNOWN
04A5 16    04E8      05520          LBRA   EXIT
                     05530 *----- -----------------*
                     05540 * OPERATE FIGURE(ACT. 6) *
```

```
                   05550 * ON ENTRY: X= ADDR. OF  *
                   05560 *             PARMLIST   *
                   05570 *------------------------*
          04A8     05580 OPERAT  EQU     *
04A8 17   039C     05590         LBSR    GETFDT  GET ADDR. OF FDT
04AB 6D   C8 02    05600         TST     <ORFLAG,U       MIXABLE?
04AE 27   05       05610         BEQ     NOMIX
04B0 EC   C8 02    05620         LDD     <ORFLAG,U
04B3 20   16       05630         BRA     MIXC
04B5 6D   C9 0011  05640 NOMIX   TST     FLAGCR,U
04B9 26   0D       05650         BN      NOMI2
04BB 86   04       05660         LDA     #4
04BD A7   8D 04E7  05670         STA     ACTION,PCR
04C1 34   10       05680         PSHS    X
04C3 17   023A     05690         LBSR    SWAP2
04C6 35   10       05700         PULS    X
04C8 EC   C8 00    05710 NOMI2   LDD     <ASW,U  GET ADDR. OF SWAP AREA
04CB 10AE C9 0012  05720 MIXC    LDY     FIGBYT,U        NUMBER OF BYTES IN FIG.
04D0 34   46       05730         PSHS    U,D
04D2 1F   03       05740         TFR     D,U     GOODBYE ADDR. OF FDT
04D4 E6   02       05750         LDB     2,X     OPERATOR
04D6 5A            05760         DECB            IT'S EASIER TO EVALUATE OPERATOR
          04D7     05770 OPLOP   EQU     *
04D7 A6   C4       05780         LDA     ,U      BYTE FROM SWAP
04D9 5D            05790         TSTB            OPERATOR IS:
04DA 2A   04       05800         BPL     NOCLR   NOT CLEAR
04DC A6   03       05810         LDA     3,X     ELSE LOAD CLEAR BYTE
04DE 20   0F       05820         BRA     OPOLO   TO COMMON EXECUTION
04E0 26   03       05830 NOCLR   BNE     NONOT   NOT A NOT OPERATION
04E2 43            05840         COMA            ELSE MAKE A NOT
04E3 20   0A       05850         BRA     OPOLO   AND GET NEXT BYTE
04E5 C1   01       05860 NONOT   CMPB    #1      IF 1 IS AND
04E7 26   04       05870         BNE     NOAND   ELSE IS AN OR
04E9 A4   03       05880         ANDA    3,X     AND WITH MASK
04EB 20   02       05890         BRA     OPOLO   AND CONTINUE
04ED AA   03       05900 NOAND   ORA     3,X     OR WITH MASK
          04EF     05910 OPOLO   EQU     *       HERE COME ALL OPTIONS
04EF A7   C0       05920         STA     ,U+     STORE NEW VALUE
04F1 31   3F       05930         LEAY    -1,Y    NUMBER OF BYTES REACHED?
04F3 26   E2       05940         BNE     OPLOP   NO, CONTINUE
04F5 35   46       05950         PULS    U,D     RESTORE ADDR. OF FFDT
04F7 1F   01       05960         TFR     D,X
04F9 6D   C8 02    05970         TST     <ORFLAG,U       MIXABLE?
04FC 27   16       05980         BEQ     OPADO   NO
04FE A6   C8 14    05990         LDA     <OWID,U
0501 A7   8D 049F  06000         STA     CADX,PCR
0505 A6   C8 0C    06010         LDA     <OLMASK,U       GET ORIG. LMASK
0508 E6   C8 0D    06020         LDB     <ORMASK,U       ORIG. RMASK
050B 6D   C9 0014  06030         TST     OWID,U
050F 26   16       06040         BNE     OPADOA
0511 5F            06050         CLRB
0512 20   13       06060         BRA     OPADOA
0514 A6   C8 0E    06070 OPADO   LDA     <WIDBYT,U
0517 A7   8D 0489  06080         STA     CADX,PCR
051B A6   C8 0A    06090         LDA     <LMASK,U
051E E6   C8 0B    06100         LDB     <RMASK,U
0521 6D   C8 0E    06110         TST     <WIDBYT,U
0524 26   01       06120         BNE     OPADOA
0526 5F            06130         CLRB
0527 43            06140 OPADOA  COMA
0528 A7   8D 0488  06150         STA     AUX2,PCR
052C 53            06160         COMB            INVERT IT
052D E7   8D 047E  06170         STB     AUX3,PCR
0531 17   02FD     06180         LBSR    NORMY   GET #ROWS IN FIG.
0534 1F   12       06190 OPAD1   TFR     X,Y
0536 E6   C8 0F    06200         LDB     <MAXBYT,U       MAX. WIDTH IN BYTES
0539 3A            06210         ABX
053A AF   8D 0472  06220         STX     AUX4,PCR
053E 1F   21       06230         TFR     Y,X     RESTORE ADDR. OF LEFT BYTE OF LINE
0540 A6   84       06240         LDA     ,X
0542 A4   8D 046E  06250         ANDA    AUX2,PCR
0546 A7   84       06260         STA     ,X
0548 E6   8D 0458  06270         LDB     CADX,PCR
054C 3A            06280         ABX             POINT TO RIGHTMOST BYTE
054D A6   84       06290         LDA     ,X      GET THE RIGHTMOST BYTE
054F A4   8D 045C  06300         ANDA    AUX3,PCR        ZEROES PROPAGATION
0553 A7   80       06310         STA     ,X+     AND PUT BACK
0555 AC   8D 0457  06320 OPAD2   CMPX    AUX4,PCR        ADDITIONAL BYTES TO CLEAR?
0559 24   04       06330         BHS     OPAD3   NO
055B 6F   80       06340         CLR     ,X+     CLEAR
055D 20   F6       06350         BRA     OPAD2
055F 6A   8D 0448  06360 OPAD3   DEC     ROWS,PCR        MORE ROWS TO ADJUST?
0563 26   CF       06370         BNE     OPAD1   YES
0565 A6   8D 043F  06380         LDA     ACTION,PCR
0569 81   04       06390         CMPA    #4
056B 1026 0421     06400         LBNE    EXIT
056F 4A            06410         DECA
0570 A7   8D 0434  06420         STA     ACTION,PCR
0574 A6   C8 08    06430         LDA     <AUTOX,U
0577 A7   8D 0429  06440         STA     CADX,PCR
057B A6   C8 09    06450         LDA     <AUTOY,U
057E A7   8D 0424  06460         STA     CADY,PCR
0582 17   0003     06470         LBSR    MOVGEN
0585 16   0408     06480         LBRA    EXIT    NO, TERMINATE
                   06490 *------------------------
                   06500 *SUBROUTINE TO MOVE AN OBJECT IN THE SCREEN
                   06510 *
          0588     06520 MOVGEN  EQU     *
0588 9E   BD       06530         LDX     <$BD    GET X COORD.
058A 109E BF       06540         LDY     <$BF    GET Y COORD.
058D 0F   BD       06550         CLR     <$BD    CLEAR
058F 0F   BF       06560         CLR     <$BF    CLEAR
0591 34   30       06570         PSHS    X,Y
0593 A6   8D 040D  06580         LDA     CADX,PCR        GET COORD X OF DESTINATION
0597 E6   8D 040B  06590         LDB     CADY,PCR        GET COORD. Y OF DESTINATION
059B 17   008A     06600         LBSR    CONVER  GET ADDR. OF TOP CORNER
059E 34   02       06610         PSHS    A
05A0 A6   C8 0A    06620         LDA     <LMASK,U
05A3 6D   C8 0E    06630         TST     <WIDBYT,U
05A6 26   03       06640         BNE     SKO
05A8 A6   C8 0B    06650         LDA     <RMASK,U
05AB A7   8D 0400  06660 SKO     STA     AUX3,PCR
05AF 35   02       06670         PULS    A
05B1 10AE 8D 03F1  06680         LDY     ACTIO2,PCR
05B6 108C 0002     06690         CMPY    #2
05BA 26   0B       06700         BNE     SK1
05BC 34   16       06710         PSHS    X,A,B
05BE 17   013F     06720         LBSR    SWAP2   SWAP FROM BLOCK
```

```
05C1 35   16       06730         PULS    X,A,B   FIRST GET
05C3 108E 0000     06740         LDY     #0
          05C7     06750 SK1     EQU     *
05C7 AF   C8 06    06760         STX     <FIGCAD,U
05CA A7   C8 0A    06770         STA     <LMASK,U
05CD E7   C8 0B    06780         STB     <RMASK,U
05D0 E6   8D 03D8  06790         LDB     AUX,PCR NEW WIDTH
05D4 E7   C8 0E    06800         STB     <WIDBYT,U
05D7 C1   00       06810         CMPB    #0
05D9 26   03       06820         BNE     SKIP2
05DB A6   C8 0B    06830         LDA     <RMASK,U
05DE 108C 0001     06840 SKIP2   CMPY    #1
05E2 27   2A       06850         BEQ     RETGEN
05E4 1F   89       06860         TFR     A,B
05E6 4F            06870         CLRA
05E7 5D            06880         TSTB            SEE IF B=0
05E8 26   04       06890         BNE     COUNT1  NO
05EA 86   F8       06900         LDA     #-8
05EC 20   06       06910         BRA     DONE1   COUNT IS NOT NECESSARY
05EE 54            06920 COUNT1  LSRB            LOOP UNTIL
05EF 25   03       06930         BCS     DONE1   IT FINDS
05F1 4A            06940         DECA            FIRST 1
05F2 20   FA       06950         BRA     COUNT1
05F4 E6   8D 03B7  06960 DONE1   LDB     AUX3,PCR        GET OLD MASK
05F8 5D            06970         TSTB            SEE IF B=0
05F9 26   04       06980         BNE     COUNT2  NO
05FB 8B   08       06990         ADDA    #8
05FD 20   06       07000         BRA     DONE2
05FF 54            07010 COUNT2  LSRB            LOOP UNTIL
0600 25   03       07020         BCS     DONE2   IT FINDS
0602 4C            07030         INCA            FIRST 1
0603 20   FA       07040         BRA     COUNT2
          0605     07050 DONE2   EQU     *
0605 4D            07060         TSTA            IF NO SHIFT REQUIRED
0606 27   03       07070         BEQ     GOSWAP  GO TO SWAP
0608 17   01BE     07080         LBSR    SHIFT   GO TO SHIFT AS REQUIRED
060B 17   0080     07090 GOSWAP  LBSR    SWAP    PLACE FIGURE IN DEST.
060E 35   30       07100 RETGEN  PULS    X,Y     RESTORE
0610 6F   8D 0398  07110         CLR     AUX3,PCR
0614 A6   8D 038C  07120         LDA     CADX,PCR        NEW     X-COORD
0618 A7   C8 08    07130         STA     <AUTOX,U
061B A6   8D 0387  07140         LDA     CADY,PCR        NEW Y-COORD
061F A7   C8 09    07150         STA     <AUTOY,U
0622 9F   BD       07160         STX     <$BD    ORIGINAL COLUMN
0624 109F BF       07170         STY     <$BF    AND ROW
0627 39            07180         RTS             RETURN
                   07190 *** SUBROUTINE TO GET ADDRESS OF AN SPECIFIC XY COORD.
                   07200 *** ENTRY: U= ADDRESS OF FIG. DESCRIPTOR BLOCK
                   07210 ***        A= X COORDINATE OF FIGURE
                   07220 ***        B= Y COORDINATE OF FIGURE
                   07230 *** EXIT:  X= ADDRESS OF UPER/LEFT CORNER
                   07240 ***        A= BITS MASK TO ADJUST LEFT BORDER
                   07250 ***        B= BITS MASK TO ADJUST RIGHT BORDER
                   07260 ***        AUX=WIDTH IN BYTES
0628 34   40       07270 CONVER  PSHS    U
062A 17   00BD     07280         LBSR    NORM    GET ADDRESS
062D 35   40       07290         PULS    U       RESTORE U
062F 5F            07300         CLRB    GET LEFT MASK
0630 8D   31       07310         BSR     SETBND
0632 34   02       07320         PSHS    A
0634 A6   8D 036C  07330         LDA     CADX,PCR        GET COLUMN
0638 AB   C8 04    07340         ADDA    <WIDTH,U        PLUS NUMBER OF COLS
063B 34   52       07350         PSHS    A,X,U
063D E6   8D 0365  07360         LDB     CADY,PCR        GET ROW
0641 17   00A6     07370         LBSR    NORM    GET ADDRESS OF RIGHT COL.
0644 1F   10       07380         TFR     X,D     PREPARE TO SUBTRACT
0646 A3   61       07390         SUBD    1,S     #FYTES BETWEEN RIGHT AND LEFT COLS
0648 35   52       07400         PULS    A,X,U   RESTORE
064A E7   8D 035E  07410         STB     AUX,PCR SAVE WIDTH IN BYTES
064E C6   01       07420         LDB     #1      PREPARE TO OBTAIN RIGHT MASK
0650 8D   11       07430         BSR     SETBND  GET RMASK
0652 6D   8D 0356  07440         TST     AUX,PCR IF WIDTH NO MORE THAN 1 BYTE
0656 26   08       07450         BNE     OUTCON
0658 EA   E4       07460         ORB     ,S      ONLY ONE MASK
065A 1F   98       07470         TFR     B,A
065C 35   04       07480         PULS    B
065E 20   02       07490         BRA     ENDCON
0660 35   02       07500 OUTCON  PULS    A       GET LEFT MASK
0662 39            07510 ENDCON  RTS
                   07520 ***HERE THE BYTE BOUNDARY IS ADJUSTED TO BIT BOUNDARY
          0663     07530 SETBND  EQU     *
0663 96   B6       07540         LDA     <$B6    GET PMODE
0665 84   01       07550         ANDA    #1      IF NOT PMODE 1 OR 3
0667 27   02       07560         BEQ     SET2    DO NOTHING
0669 08   BE       07570         LSL     <$BE    ELSE MULTIPLY END COL. BY 2
066B 96   BE       07580 SET2    LDA     <$BE    GET COLUMN
066D 84   07       07590         ANDA    #$07    NUMBER OF BITS
066F 31   8D 0013  07600         LEAY    >MASTAB,PCR     CONVERSION TABLE
0673 A6   A6       07610         LDA     A,Y     DISPLACEMENT
0675 5D            07620         TSTB            FLAG ON?
0676 26   01       07630         BNE     RISIDE  YES. RIGHT BORDER
0678 39            07640         RTS
          0679     07650 RISIDE  EQU     *
0679 4D            07660         TSTA            IF RMASK=FF
067A 26   06       07670         BNE     RI2     YES
067C 86   FF       07680         LDA     #$FF
067E 6A   8D 032A  07690         DEC     AUX,PCR WIDTH IS ONE BYTE LESS
0682 43            07700 RI2     COMA
0683 1F   89       07710         TFR     A,B
0685 39            07720         RTS
0686      00       07730 MASTAB  FCB     $00
0687      80       07740         FCB     $80
0688      C0       07750         FCB     $C0
0689      E0       07760         FCB     $E0
068A      F0       07770         FCB     $F0
068B      F8       07780         FCB     $F8
068C      FC       07790         FCB     $FC
068D      FE       07800         FCB     $FE
                   07810 * SWAP:
                   07820 *** MOVE A FIGURE FROM SCREEN TO A RESERVED AREA AND
                   07830 *** THE FIGURE FROM THE RESERVED AREA TO SCREEN
                   07840 * ON ENTRY: U= ADDRESS OF FDT
                   07850 * ON EXIT: SWAP PERFORMED
                   07860 * EXCEPT FOR U NO REGS ARE PRESERVED
          068E     07870 SWAP    EQU     *
068E 10AE C8 00    07880         LDY     <ASW,U  ADDR. OF SWAP AREA
0692 AE   C8 06    07890         LDX     <FIGCAD,U       ADDRESS IN SCREEN
0695 17   0199     07900         LBSR    NORMY
```

```
0698 4F            07910 MORE1    CLRA
0699 E6    86      07920          LDB     A,X      FIRST BYTE FROM ROW
069B 34    04      07930          PSHS    B
069D E4    C8 0A   07940          ANDB    <LMASK,U          PREPARE IT TO MIX
06A0 EA    A6      07950          ORB     A,Y
06A2 20    06      07960          BRA     MORCOM  CONTINUE SWAPING ROW
06A4 E6    86      07970 MORE2    LDB     A,X     INTERMIDIATE ROW BYTE
06A6 34    04      07980          PSHS    B
06A8 E6    A6      07990          LDB     A,Y     BTE FROM SWAP
           06AA    08000 MORCOM   EQU     *
06AA 34    02      08010          PSHS    A
06AC 6D    C8 02   08020          TST     <ORFLAG,U       MIXABLE?
06AF 27    0F      08030          BEQ     NOOR    NO
06B1 96    B6      08040          LDA     <$B6    IS PMODE 1 OR 3?
06B3 84    01      08050          ANDA    #1
06B5 27    07      08060          BEQ     ORTWO   NO
06B7 A6    61      08070          LDA     1,S     BYTE FROM SCREEN
06B9 17    00CE    08080          LBSR    ORFIG   PERFORM "OR"
06BC 20    02      08090          BRA     NOOR    RESUME
06BE EA    61      08100 ORTWO    ORB     1,S     NORMAL OR OPERATION
06C0 35    02      08110 NOOR     PULS    A       RESTORE A
06C2 E7    86      08120          STB     A,X     STORE ON SCREEN
06C4 35    04      08130          PULS    B       GET BYTE FROM SCREEN
06C6 E7    A6      08140          STB     A,Y     STORE IT IN SWAP
06C8 4C            08150          INCA            A=A+1
06C9 A1    C8 0E   08160          CMPA    <WIDBYT,U       A=WIDTH IN BYTES?
06CC 22    0D      08170          BHI     ENDCOL
06CE 25    D4      08180          BLO     MORE2
06D0 E6    86      08190          LDB     A,X     PROCESS RIGHTMOST BYTE
06D2 34    04      08200          PSHS B
06D4 E4    C8 0B   08210          ANDB    <RMASK,U        MIX IT WITH BYTE FROM SWAP
06D7 EA    A6      08220          ORB     A,Y
06D9 20    CF      08230          BRA     MORCOM  TO NORMAL PROCESS
06DB D6    B9      08240 ENDCOL   LDB     <$B9    NUMBER OF BYTE FOR ROW
06DD 3A            08250          ABX             ADD TO X
06DE E6    C8 0F   08260          LDB     <MAXBYT,U       BYTES PER ROW
06E1 31    A5      08270          LEAY    B,Y     ADD TO Y
06E3 6A    8D 02C4 08280          DEC     ROWS,PCR        #ROWS-1
06E7 22    AF      08290          BHI     MORE1   IF NOT ZERO CONTINUE
06E9 39            08300          RTS
                   08310 *** NORMALIZE X,Y COORDINATES
                   08320 *** ON ENTRY: A=X COORD. B=Y COORD.
06EA 97    BE      08330 NORM     STA     <$BE    X-COORD.
06EC D7    C0      08340          STB     <$C0    Y-COORD.
06EE 96    B6      08350          LDA     <$B6    GET PMODE
06F0 81    04      08360          CMPA    #4      PMODE 4 OUT
06F2 27    08      08370          BEQ     ENNORM
06F4 04    BE      08380          LSR     <$BE    DIVIDE X BY 2
06F6 81    01      08390          CMPA    #1      PMODE >1
06F8 22    02      08400          BHI     ENNORM
06FA 04    C0      08410          LSR     <$C0    DIVIDE Y BY 2
06FC BD    9298    08420 ENNORM   JSR     $9298   TO ROM GET ADDRESS
06FF 39            08430          RTS
                   08440 * SWAP2.
                   08450 *** MOVE A FIGURE FROM SCREEN TO A RESERVED AREA AND
                   08460 *** THE FIGURE FROM THE RESERVED AREA TO SCREEN
                   08470 * DOES NOT CHECK FOR OR OPERATIONS. (FAST MODE)
                   08480 * ON ENTRY: U= ADDRESS OF FDT
                   08490 * ON EXIT: SWAP PERFORMED
                   08500 * EXCEPT FOR U NO REGS ARE PRESERVED
           0700    08510 SWAP2    EQU     *
0700 10AE C8 00    08520          LDY     <ASW,U  ADDR. OF SWAP AREA
0704 AE    C8 06   08530          LDX     <FIGCAD,U       ADDRESS IN SCREEN
0707 17    0127    08540          LBSR    NORMY
070A 63    C8 0A   08550          COM     <LMASK,U
070D 63    C8 0B   08560          COM     <RMASK,U
0710 4F            08570 MORE21   CLRA
0711 E6    86      08580          LDB     A,X     FIRST BYTE FROM ROW
0713 E4    C8 0A   08590          ANDB    <LMASK,U        CUM BITS AT LEFT
0716 20    02      08600          BRA     MORCO2  CONTINUE SWAPING ROW
0718 E6    86      08610 MORE20   LDB     A,X     INTERMIDIATE ROW BYTE
071A 34    04      08620 MORCO2   PSHS    B
071C E6    8D 0288 08630          LDB     ACTION,PCR      CREATE FIG?
0720 C1    01      08640          CMPB    #1      YES
0722 27    04      08650          BEQ     COMX
0724 E6    A6      08660          LDB     A,Y     GET BYTE IN SWAP
0726 E7    86      08670          STB     A,X     PUT IN SCREEN
0728 35    04      08680 COMX     PULS    B
072A E7    A6      08690          STB     A,Y     SAVE IN SWAP
072C 4C            08700          INCA            A=A+1
072D A1    C8 0E   08710          CMPA    <WIDBYT,U       A=WIDTH IN BYTES?
0730 22    09      08720          BHI     ENDCO2
0732 25    E4      08730          BLO     MORE20
0734 E6    86      08740          LDB     A,X     PROCESS RIGHTMOST BYTE
0736 E4    C8 0B   08750          ANDB    <RMASK,U        MIX IT WITH BYTE FROM SWAP
0739 20    DF      08760          BRA     MORCO2  TO NORMAL PROCESS
073B D6    B9      08770 ENDCO2   LDB     <$B9    NUMBER OF BYTE FOR ROW
073D 30    85      08780          LEAX    B,X     ADD TO X
073F E6    C8 0F   08790          LDB     <MAXBYT,U       BYTES PER ROW
0742 31    A5      08800          LEAY    B,Y     ADD TO Y
0744 6A    8D 0263 08810          DEC     ROWS,PCR        #ROWS-1
0748 22    C6      08820          BHI     MORE21  IF NOT ZERO CONTINUE
074A 63    C8 0A   08830          COM     <LMASK,U
074D 63    C8 0B   08840          COM     <RMASK,U
0750 6D    C8 02   08850          TST     <ORFLAG,U       OR-ABLE?
0753 27    34      08860          BEQ     ENSWX   ...NO GET OUT
0755 AE    C8 02   08870          LDX     <ORFLAG,U
0758 A6    8D 024C 08880          LDA     ACTION,PCR      CREATE FIGURE?
075C 81    01      08890          CMPA    #1      IF YES GET-OUT
075E 27    10      08900          BEQ     ENSW2
0760 A6    C8 0C   08910          LDA     <OLMASK,U       RESTORE ORIG. LMASK
0763 6D    C9 0014 08920          TST     OWID,U
0767 26    03      08930          BNE     NORK
0769 A6    C8 0D   08940          LDA     <ORMASK,U
076C A7    8D 023F 08950 NORK     STA     AUX3,PCR
           0770    08960 ENSW2    EQU     *
0770 10AE C9 0012  08970          LDY     FIGBYT,U
0775 34    40      08980          PSHS    U
0777 EE    C8 00   08990          LDU     <ASW,U  GET TO-ADDRESS
077A A6    8D 022A 09000          LDA     ACTION,PCR      CURRENT OPTION
077E 81    01      09010          CMPA    #1      IF NOT CREATE
0780 26    02      09020          BNE     ENSW3   GO AHEAD
0782 1E    31      09030          EXG     U,X     ELSE COPY FROM FIRST AREA TO SECOND
0784 17    0141    09040 ENSW3    LBSR    COPYSW  COPY AREAS
0787 35    40      09050          PULS    U       RESTORE U
           0789    09060 ENSWX    EQU     *
0789 39            09070          RTS
                   09080 * ORFIG. DETERMINES IF PIECE OF FIGURE ON
```

```
                   09090 * SCREEN SHOULD BE LEFT
                   09100 * ON ENTRY: A= BYTE FROM SCREEN
                   09110 *           B= BYTE FROM SWAP
                   09120 *           U= ADDR. OF FDT
                   09130 * ON EXIT:  B= ADJUSTED SWAP BYTE
                   09140 * REGS A AND B ARE NOT PRESERVED
           078A    09150 ORFIG    EQU     *
078A 34    06      09160          PSHS    A,B
078C 86    C0      09170          LDA     #$C0    FIRST TWO BITS TO CHECK
078E 6F    8D 0217 09180          CLR     RESB,PCR        CLEAR RESULT BYTE
0792 A7    8D 0214 09190          STA     BITAN,PCR
0796 A6    61      09200 OR1      LDA     1,S     GET BYTE FROM SWAP
0798 A4    8D 020E 09210          ANDA    BITAN,PCR       SUPRESS UNWANTED BITS
079C E6    8D 0212 09220          LDB     BKCOLO,PCR      GET COLOR TO BE 'ORED'
07A0 E4    8D 0206 09230          ANDB    BITAN,PCR       REMOVE UNWANTED BITS
07A4 34    04      09240          PSHS    B
07A6 A1    E0      09250          CMPA    ,S+     BAKGROUND COLOR IN SWAP?
07A8 26    06      09260          BNE     OR2     NO
07AA A6    E4      09270          LDA     ,S      GET BYTE FROM SCREEN
07AC A4    8D 01FA 09280          ANDA    BITAN,PCR       SUPRESS UNWANTED BITS
07B0 AA    8D 01F5 09290 OR2      ORA     RESB,PCR        PUT SELECTED COLOR
07B4 A7    8D 01F1 09300          STA     RESB,PCR
07B8 64    8D 01EE 09310          LSR     BITAN,PCR       ANALIZE NEXT TWO BITS
07BC 64    8D 01EA 09320          LSR     BITAN,PCR
07C0 26    D4      09330          BNE     OR1     NOT FINISHED YET?
07C2 32    62      09340          LEAS    2,S     ADJUST STACK
07C4 E6    8D 01E1 09350          LDB     RESB,PCR        BYTE TO PUT IN SCREEN
07C8 39            09360          RTS
                   09370 * SHIFT. SUBROUTINE TO SHIFT A MATRIX AN SPECIFIED
                   09380 * NUMBER OF BITS
                   09390 * ON ENTRY:
                   09400 *           U= ADDRESS OF FIG. DESC. TABLE
                   09410 *           A=#BITS TO SHIFT.
                   09420 *             IF A<0 SHIFT LEFT
                   09430 *             IF A>0 SHIFT RIGHT
                   09440 * ON EXIT:  THE SWAP AREA FOR FIGURE IS SHIFTED.
                   09450 *           EXCEPT FOR U, NO REGISTERS ARE PRESERVED
           07C9    09460 SHIFT    EQU     *
                   09470 *PUT #BITS TO SHIFT IN X
07C9 1F    89      09480          TFR     A,B
07CB 4D            09490          TSTA
07CC 2A    01      09500          BPL     POSIT
07CE 50            09510          NEGB
           07CF    09520 POSIT    EQU     *
07CF 34    02      09530          PSHS    A
07D1 4F            09540          CLRA
07D2 1F    01      09550          TFR     D,X     PUT IN X
07D4 17    005A    09560          LBSR    NORMY   TO NORMALIZE Y
07D7 35    02      09570          PULS    A
07D9 10AE C8 00    09580          LDY     <ASW,U
07DD 34    30      09590          PSHS    X,Y
07DF E6    C8 0F   09600 SH0      LDB     <MAXBYT,U
07E2 10AE 62       09610          LDY     2,S
07E5 4D            09620 SHOR     TSTA    SHIFT RIGHT?
07E6 2B    09      09630          BMI     SH2A    NO, LEFT
07E8 1C    FE      09640          ANDCC   #$FE
07EA 66    A0      09650 SH1      ROR     ,Y+
07EC 5A            09660          DECB
07ED 26    FB      09670          BNE     SH1
07EF 20    09      09680          BRA     NEXTST
07F1 31    A5      09690 SH2A     LEAY    B,Y
07F3 1C    FE      09700          ANDCC   #$FE
07F5 69    A2      09710 SH2B     ROL     ,-Y
07F7 5A            09720          DECB
07F8 26    FB      09730          BNE     SH2B
07FA 30    1F      09740 NEXTST   LEAX    -1,X    MORE BITS TO SHIFT?
07FC 26    E1      09750          BNE     SH0     YES
07FE 6A    8D 01A9 09760          DEC     ROWS,PCR        MORE ROWS?
0802 27    0F      09770          BEQ     ENSHIF  NO
0804 E6    C8 0F   09780          LDB     <MAXBYT,U       ADJUST TO FIRST COL.
0807 10AE 62       09790          LDY     2,S
080A 31    A5      09800          LEAY    B,Y
080C 10AF 62       09810          STY     2,S
080F AE    E4      09820          LDX     ,S
0811 20    D2      09830          BRA     SHOR
0813 35    B0      09840 ENSHIF   PULS    Y,X,PC
                   09850 * CALCULATES MAXIMUM NUMBER OF BYTES PER ROW
                   09860 * FOR A FIGURE.
                   09870 * ON ENTRY: U= ADDRESS OF FIGURE DESCRIPTOR TABLE
                   09880 *           B= WIDTH IN PIXELS
                   09890 * ON EXIT:  MAXBYT WITH VALUE
                   09900 * U,X,Y ARE PRESERVED
           0815    09910 CMAXBY   EQU     *
0815 96    B6      09920          LDA     <$B6    GET PMODE
0817 81    04      09930          CMPA    #4      IS PMODE 4?
0819 27    05      09940          BEQ     CMAX1   YES
081B 84    01      09950          ANDA    #1      IF PMODE 1 OR 3
081D 26    01      09960          BNE     CMAX1   DO NOTHING
081F 54            09970          LSRB            NO, DIVIDE BY 2
           0820    09980 CMAX1    EQU     *
0820 1F    98      09990          TFR     B,A     SAVE TO OBTAIN
0822 84    07      10000          ANDA    #$07    REMAINDER OF D/8
0824 54            10010          LSRB            DIVIDE BY 8
0825 54            10020          LSRB
0826 54            10030          LSRB
0827 5C            10040          INCB            ADD 1 TO THE RESULT
0828 81    02      10050          CMPA    #2      IF REMAINDER<2
082A 2D    01      10060          BLT     CMAX2   GO OUT
082C 5C            10070          INCB            IF NOT ADD 1 OF RESULT
082D E7    C8 0F   10080 CMAX2    STB     <MAXBYT,U
0830 39            10090          RTS
                   10100 * FIND REQUIRED NUMBER OF
                   10110 * BYTES FOR A GIVEN NUMBER
                   10120 * OF ROWS.
                   10130 * ON ENTRY: U= ADDR, OF FDT
                   10140 * ON EXIT: A AND ROWS WITH VALUE
                   10150 * EXCEPT FOR A ALL REGS. ARE PRESERVED
0831 A6    C8 05   10160 NORMY    LDA     <HEIGHT,U       GET ROWS
0834 A7    8D 0173 10170          STA     ROWS,PCR
0838 96    B6      10180          LDA     <$B6    GET PMODE
083A 81    01      10190          CMPA    #1      NORMALIZE?
083C 22    04      10200          BHI     RETNY   NO
083E 64    8D 0169 10210          LSR     ROWS,PCR        YES, DIVIDE
0842 A6    8D 0165 10220 RETNY    LDA     ROWS,PCR
0846 39            10230          RTS
                   10240 * GET ADDRESS OF FIGURE
                   10250 * DESCRIPTOR TABLE (FDT)
                   10260 * ON ENTRY: X= ADDR. OF PARMLIST
```

```
                10270 * ON EXIT: U=ADDR. OF FDT
                10280 * X AND Y ARE PRESERVED
        0847    10290 GETFDT  EQU     *
0847 86 AA      10300         LDA     #$AA    SYSTEM INITIALIZED?
0849 A1 8D 01DF 10310         CMPA    2+FCTAB,PCR
084D 27 05      10320         BEQ     GE2     YES, OK
084F C6 07      10330         LDB     #XNI    ELSE ERROR
0851 16 00D9    10340         LBRA    ERROR
0854 33 8D 01D8 10350 GE2     LEAU    FFDT,PCR        ADDR. OF FIRST FDT
0858 34 40      10360         PSHS    U
085A A6 01      10370         LDA     1,X     FIGURE NUMBER
085C 27 06      10380         BEQ     GE3     FIGURE CAN'T BE ZERO
085E A1 8D 01C8 10390         CMPA    FCTAB,PCR       EXCEEDS MAX. NBR. OF FIG
0862 23 05      10400         BLS     GE4     NO, OK
0864 C6 02      10410 GE3     LDB     #XOF    ELSE ERROR
0866 16 00C4    10420         LBRA    ERROR
0869 A1 8D 01BE 10430 GE4     CMPA    1+FCTAB,PCR     GREATER THAN CREATED FIGS
086D 23 05      10440         BLS     GE5     NO, OK
086F C6 06      10450         LDB     #XNC    ELSE ERROR
0871 16 00B9    10460         LBRA    ERROR
0874 C6 18      10470 GE5     LDB     #24     SIZE IN BYTES OF A FDT
0876 4A         10480         DECA            TO OFFSET
0877 3D         10490         MUL             DISPLACEMENT
0878 E3 E1      10500         ADDD    ,S++    REAL ADDRESS
087A 1F 03      10510         TFR     D,U     LEAVE IN U
087C 39         10520         RTS
        |       10530 * RANDOM. FIND A RANDOM NUMBER
                10540 * ON ENTRY: A= MAX. VALUE OF NUMBER TO GENERATE
                10550 * ON EXIT: B=RANDOM NUMBER. ALL REGS. PRESERVED BUT B
        087D    10560 RANDOM  EQU     *
087D 34 02      10570         PSHS    A       SAVE MAX. VALUE OF RANDOM NUMBER
087F 6D 8D 0134 10580         TST     PERIOD,PCR      SEQUENCE EXHAUSTED?
0883 26 07      10590         BNE     RA2     NO
0885 FC 0112    10600         LDD     $112    GET VALUE OF TIMER
0888 ED 8D 0129 10610         STD     SEED,PCR        AND USE IT AS NEW SEED
        088C    10620 RA2     EQU     *
088C EC 8D 0125 10630         LDD     SEED,PCR        GET SEED NUMBER
0890 34 06      10640         PSHS    D
0892 86 02      10650         LDA     #2      WILL MULTIPLY SEED BY 4
0894 68 8D 011E 10660 RA3     LSL     1+SEED,PCR      TWO TIMES 2
0898 69 8D 0119 10670         ROL     SEED,PCR
089C 4A         10680         DECA
089D 26 F5      10690         BNE     RA3     IF NOT DONE CONTINUE
089F 35 06      10700         PULS    D       GET OLD SEED
08A1 E3 8D 0110 10710         ADDD    SEED,PCR        THUS: OLD SEED BY 5
08A5 C3 0035    10720         ADDD    #53     PLUS 53
08A8 ED 8D 0109 10730         STD     SEED,PCR        NEW SEED
08AC C6 FF      10740         LDB     #$FF    MASK TO REDUCE RANDOM
08AE E7 8D 0102 10750         STB     AUX2,PCR
08B2 A1 E4      10760 RMAX    CMPA    ,S      IS NUMBER LESS OR EQUAL THAN MAX?
08B4 23 0A      10770         BLS     ENRA    YES, GET OUT
08B6 64 8D 00FA 10780         LSR     AUX2,PCR        GET RID OF LEFT BIT
08BA A4 8D 00F6 10790         ANDA    AUX2,PCR
08BE 20 F2      10800         BRA     RMAX    AND COMPARE AGAIN
        08C0    10810 ENRA    EQU     *       NUMBER FOUND
08C0 1F 89      10820         TFR     A,B     LEAVE IN B
08C2 6C 8D 00F1 10830         INC     PERIOD,PCR      PERIOD COUNTER
08C6 35 82      10840         PULS    A,PC    ADJUST STACK AND RETURN
                10850 * COPY ONE SWAP AREA INTO ANOTHER
                10860 * ON ENTRY: X= ADDR. OF FROM AREA
                10870 *           U= ADDR. OF TO AREA
                10880 *           Y=# OF BYTES TO COPY
                10890 * ON EXIT: AREA COPIED. ONLY A IS PRESERVED
        08C8    10900 COPYSW  EQU     *
08C8 E6 80      10910         LDB     ,X+     GET BYTE
08CA E7 C0      10920         STB     ,U+     STORE IN TO-AREA
08CC 31 3F      10930         LEAY    -1,Y    DECREMENT COUNTER
08CE 26 F8      10940         BNE     COPYSW  MORE TO COPY?
08D0 39         10950         RTS
                10960 * VERIFY IF ACTION ON OUT
                10970 * OF SCREEN IS REQUIRED
                10980 * ON ENTRY: Y= ADDR. OF VALUE FOR X/Y
                10990 *           FLG WITH 0 FOR X OR 64 FOR Y
                11000 * ON EXIT: CACX/CACY WITH X/Y DESTINATIONS
                11010 * EXCEPT FOR D ALL REGISTERS ARE PRESERVED
        08D1    11020 ACTOSC  EQU     *
08D1 6D 8D 00D9 11030         TST     1+FLG,PCR       TESTING X-COORD?
08D5 27 05      11040         BEQ     L0
08D7 E6 C8 05   11050         LDB     <HEIGHT,U       GET VERTICAL SIZE
08DA 20 03      11060         BRA     L1
08DC E6 C8 04   11070 L0      LDB     <WIDTH,U        GET HORIZONTAL SIZE
08DF E7 8D 00D1 11080 L1      STB     AUX2,PCR
08E3 EC A4      11090         LDD     ,Y      GET X/Y VALUE
08E5 4D         11100         TSTA            NEGATIVE?
08E6 2D 0F      11110         BLT     L1B
08E8 E3 8D 00C1 11120         ADDD    FLG,PCR ADD IT WITH FLAG (64 IF Y)
08EC E3 8D 00C3 11130         ADDD    AUX2H,PCR       ADD TO SUBTOTAL
08F0 4D         11140         TSTA            GREATER THAN 255?
08F1 27 2F      11150         BEQ     LXX     ...NO GET-OUT
08F3 C6 02      11160         LDB     #2
08F5 20 02      11170         BRA     L1A
08F7 C6 01      11180 L1B     LDB     #1
08F9 EA 8D 00BC 11190 L1A     ORB     1+STATUS,PCR
08FD E7 8D 00B8 11200         STB     1+STATUS,PCR
0901 E6 C8 10   11210         LDB     <OUTSCR,U       WHAT TO DO?
0904 27 1D      11220         BEQ.    LERX    MARK ERROR
0906 C0 02      11230         SUBB    #2
0908 27 1E      11240         BEQ     LREX    REMOVE FIGURE
090A 4D         11250         TSTA            OUT-OF-SCREEN LEFT OR UP?
090B 2A 01      11260         BPL     L2      ...NO
090D 53         11270         COMB
090E 5D         11280 L2      TSTB            WHAT TO DO?
090F 2C 0D      11290         BGE     L4      ADJUST TO ZERO
0911 4F         11300         CLRA            FREEZE IN BORDER
0912 C6 FF      11310         LDB     #255
0914 E0 8D 0096 11320         SUBB    1+FLG,PCR       191 IF Y
0918 E0 8D 0098 11330         SUBB    AUX2,PCR
091C 20 02      11340         BRA     LPX     GET OUT
091E 4F         11350 L4      CLRA            ADJUST TO ZERO
091F 5F         11360         CLRB
0920 ED A4      11370 LPX     STD     ,Y      STORE NEW DEST. COORD.
0922 39         11380 LXX     RTS             END ROUTINE
0923 C6 00      11390 LERX    LDB     #XOS    ERROR INDICATOR
0925 16 0005    11400         LBRA    ERROR
0928 32 62      11410 LREX    LEAS    2,S     ADJUST STACK
092A 16 FABE    11420         LBRA    REMOV2  TO REMOVE FIGURE
                11430 * ERROR. THIS ROUTINE WILL SIGNAL ERRORS AND EXIT
                11440 * ON ENTRY: B= CODE OF ERROR TO BE ISSUED
                11450 * ON EXIT: ERROR MESSAGE PLUS PROGRAM TERMINATION
```

```
        092D    11460 ERROR   EQU     *
092D 31 8D 0089 11470         LEAY    ERMASK,PCR      ADDR. OF PRINT MASK
0931 30 8D 00A7 11480         LEAX    ERMSGT,PCR      TABLE OF ERRORS
0935 58         11490         ASLB    MULTIPLY CODE BY 2
0936 3A         11500         ABX             OFFSET
0937 A6 80      11510         LDA     ,X+     GET FIRST CHAR OF ERROR TYPE
0939 A7 22      11520         STA     2,Y     ON MASK
093B A6 84      11530         LDA     ,X      SECOND CHAR
093D A7 23      11540         STA     3,Y
093F A6 8D 0065 11550         LDA     ACTION,PCR      ACTION
0943 8A 30      11560         ORA     #$30    TO ASCII FORMAT
0945 A7 A8 20   11570         STA     32,Y    PUT IN MASK
0948 4F         11580         CLRA
0949 E6 8D 00D6 11590         LDB     1+PARMS,PCR     FIGURE NUMBER
094D 31 A8 15   11600         LEAY    21,Y    POINT TO MASK AREA FOR FIG. NUMBR.
0950 C1 64      11610 ERRO1   CMPB    #100    HERE WE WILL CONVERT
0952 2D 05      11620         BLT     ERRO2   FIG. NMBR. TO ASCII
0954 C0 64      11630         SUBB    #100    NUMBER OF HUNDREDS
0956 4C         11640         INCA
0957 20 F7      11650         BRA     ERRO1
0959 8A 30      11660 ERRO2   ORA     #$30    HUNDREDS IN ASCII
095B A7 A0      11670         STA     ,Y+     PUT IN PRINT MASK
095D 4F         11680         CLRA
095E C1 0A      11690 ERRO2B  CMPB    #10     TENS
0960 2D 05      11700         BLT     ERRO3
0962 C0 0A      11710         SUBB    #10
0964 4C         11720         INCA
0965 20 F7      11730         BRA     ERRO2B
0967 8A 30      11740 ERRO3   ORA     #$30 TO ASCII
0969 A7 A0      11750         STA     ,Y+
096B CA 30      11760         ORB     #$30    AND UNITS IN ASCII TOO
096D E7 A4      11770         STB     ,Y
096F C6 21      11780         LDB     #33     NUMBER OF CHARS IN MASK
0971 31 8D 0045 11790         LEAY    ERMASK,PCR      TO BEGINING OF MASK
0975 A6 A0      11800 ERRPRI  LDA     ,Y+     GET CHAR FROM BYTE
0977 AD 9F A002 11810         JSR     [$A002] WRITE CHAR ON SCREEN
097B 5A         11820         DECB            DECREMENT COUNTER
097C 26 F7      11830         BNE     ERRPRI  PRINT MORE CHARS
097E 6C 8D 0036 11840         INC     STATUS,PCR      FLAG STATUS WITH ERROR
0982 6D 8D 00A9 11850         TST     5+FCTAB,PCR     IF CALLED FROM ASSEMBLER
0986 27 08      11860         BEQ     EXIT    USE NORMAL EXIT
0988 10EE 8D 0061 11870       LDS     SAVSTK,PCR      RESTORE STACK POINTER
098D 7E B44A    11880         JMP     $B44A   MAKE A FC ERROR
                11890 *
0990 10EE 8D 0059 11900 EXIT  LDS     SAVSTK,PCR      RESTORE STACK ADDR.
0995 EC 8D 001F 11910         LDD     STATUS,PCR      TO PRESENT STATUS
0999 6D 8D 0092 11920         TST     5+FCTAB,PCR     IF CALLED FROM ASSEMBLER
099D 27 03      11930         BEQ     EXITA   GET-OUT
099F 7E B4F4    11940         JMP     $B4F4   OTHERWISE RETURN TO BASIC
09A2 39         11950 EXITA   RTS
                11960 *** DATA REFERENCES
09A3    00      11970 CACX    FCB     0
09A4    00      11980 CADX    FCB     0       TOP-LEFT COLUMN DEST. CORNER
09A5    00      11990 CACY    FCB     0
09A6    00      12000 CADY    FCB     0       TOP-LEFT ROW    DEST. CORNER
09A7    00      12010 ACTIO2  FCB     0
09A8    00      12020 ACTION  FCB     0       CURRENT OPTION
09A9    00      12030 RESB    FCB     0
09AA    00      12040 BITAN   FCB     0
09AB    00      12050 ROWS    FCB     0
09AC    00      12060 AUX     FCB     0
09AD    0000    12070 FLG     FDB     0
09AF    00      12080 AUX3    FCB     0
09B0    0000    12090 AUX4    FDB     0
09B2    00      12100 BKCOLO  FCB     0
09B3    00      12110 AUX2H   FCB     0
09B4    00      12120 AUX2    FCB     0
09B5    0000    12130 SEED    FDB     0       SEED VALUE FOR RANDOM ROUTINE
09B7    00      12140 PERIOD  FCB     0       PERIOD COUNTER OF RANDOM SEQUENCE
09B8    0000    12150 STATUS  FDB     0       STATUS
09BA    0D      12160 ERMASK  FCB     $0D     SKIP LINE
09BB    58      12170         FCC     /X## ERROR /
09C5    4F      12180         FCC     /ON FIGURE ### /
09D3    41      12190         FCC     /ACTION #/
09DB    0D      12200         FCB     $0D     SKIP LINE
09DC    4F      12210 ERMSGT  FCC     /OSOMOFIOEXEYNCNIIC/
09EE    0000    12220 SAVSTK  FDB     0
09F0            12230 STACK   RMB     $32
        0000    12240 XOS     EQU     0       OUT OF SCREEN
        0001    12250 XOM     EQU     1       OUT OF MEMORY
        0002    12260 XOF     EQU     2       INVALID FIG. NUMBER
        0003    12270 XIO     EQU     3       INVALID OPTION
        0004    12280 XEX     EQU     4       EXCEEDS MAX. X PIXELS
        0005    12290 XEY     EQU     5       EXCEEDS MAX. Y PIXELS
        0006    12300 XNC     EQU     6       FIG. NOT CREATED
        0007    12310 XNI     EQU     7       ANIMATIC NOT INITIALIZED
        0008    12320 XIC     EQU     8       CAN'T COPY FIGS.
                12330 *
0A22            12340 PARMS   RMB     8
                12350 *
0A2A    0000    12360 FCTAB   FDB     0       FIGURES CONROL TABLE
0A2C    0000    12370         FDB     0
0A2E    0000    12380         FDB     0
        0000    12390 NFIGS   EQU     0
        0001    12400 ADFFDT  EQU     1
        0003    12410 NEXTSW  EQU     3
0A30            12420 FFDT    RMB     1       DISPLACEMENT TO FIRST FDT
        0000    12430 ASW     EQU     0       ADDR. OF KEEP AREA
        0002    12440 ORFLAG  EQU     2       MIX. FLAG/SWAP FOR MIX. FIGS
        0004    12450 WIDTH   EQU     4       WIDTH IN PIXELS
        0005    12460 HEIGHT  EQU     5       HEIGHT IN PIXELS
        0006    12470 FIGCAD  EQU     6       ADDR. OF FIG. ON SCREEN
        0008    12480 AUTOX   EQU     8       COLUMN POSITION ON SCREEN (PIXELS)
        0009    12490 AUTOY   EQU     9       ROW POSITION ON SCREEN (PIXELS)
        000A    12500 LMASK   EQU     10      MASK WITH VALID BITS OF LEFTMOST BYTE
        000B    12510 RMASK   EQU     11      MASK WITH VALID BITS OF RIGHTMOST BYTE
        000C    12520 OLMASK  EQU     12      ORIGINAL LMASK
        000D    12530 ORMASK  EQU     13      ORIGINAL RMASK
        000E    12540 WIDBYT  EQU     14      ACTUAL WIDHT IN BYTES-1
        000F    12550 MAXBYT  EQU     15      MAXIMUM POSSIBLE WIDTH IN BYTES
        0010    12560 OUTSCR  EQU     16      ACTION IF OUT OF SCREEN
        0011    12570 FLAGCR  EQU     17      FLAG FOR NEWLY CREATED FIG.
        0012    12580 FIGBYT  EQU     18      MAX NUMBER OF BYTES FOR FIGURE
        0014    12590 OWID    EQU     20      ORIG. WIDTH
                12600 * BYTES 21-23 RESERVED FOR FUTURE *
0A31    00      12610 ENDGRA  FCB     0
        0000    12620         END
00000 TOTAL ERRORS
```

DOUBLE YOUR DISK STORAGE

BY Mark Rothwell

Well you have finally upgraded to a Disk System, but have found that floppy disks are not cheap to buy. Well by doing the following you will be able to halve the cost of each box of disks you buy.

What we are going to do, is use the other side of the disk, by turning each of your Floppies into Flippies.

The materials you need are :

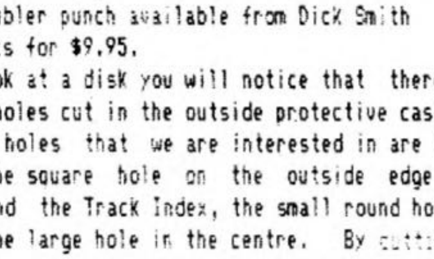
A piece of cardboard or an old Diskette Casing;

A paper hole punch, these can be purchased for $1.95 in Big W Supermarkets;

A Disk Doubler punch available from Dick Smith Electronics for $9.95.

If you look at a disk you will notice that there are a number of holes cut in the outside protective casing.

The two holes that we are interested in are the Write Protect, the square hole on the outside edge of the casing, and the Track Index, the small round hole to the right of the large hole in the centre. By cutting these holes in a mirror image, we will be able to use the reverse side of the disk by flipping the disk over.



Using the cardboard, make a pattern of the disk casing, making sure the position of the two holes is accurate. If you have a old disk that gives I/O errors use the outer casing for the pattern. After doing this, place the pattern on the disk you wish to flip and mark the position of the holes with a Texta on both sides, do not use a ball point pen as this could damage the media. After doing this carefully cut the hole in the casing on both sides.

When cutting the small round hole in the centre, make sure that you do not damage the magnetic media as this is very fragile. Place a piece of paper between the disk media and the casing before cutting the round hole, as this will help protect the media from scratching. Remember only cut the outer casing not the actual disk media.

After cutting the holes in the casing try to format the reverse side of the disk. If an I/O error occurs, check that the round holes are correctly positioned, or if a WP error occurs check the position of the write protect notch. If no errors occur, you have successfully halved the cost of buying your disks.

I personally have not experienced any problems, although a member of my Users' Group, has one disk drive that will not format the disk if it has the extra holes cut. Therefore try one first before doubling all your disks.

BYTE MASTER 16K ECB

At Your Request: Readers' Most Frequent Questions Answered

By R. Bartly Betts

Programs By Chris Bone

This column is being written the same month that the 51-column screen program appeared in the RAINBOW. I feel like a celebrity; never in my life have I had so many phone calls and letters. I was really pleased to get them, but I would be even more pleased if they hadn't resulted from a mistake in the 51-column screen article.

Part of the loading instructions was left out of the column, causing needless trouble, time and anxiety for many of you. In an effort to make amends, I have written the following BASIC program to load and execute the 51-screen program. So that I won't have to say and type "51-column screen" so often, I have now named the program *Bytescreen*, and my BASIC loader is saved under the name *BYTESCRN*. To install *Bytescreen*, I just type RUN "BYTESCRN" and ENTER.

CONVENTIONS

You may have noticed the growing use of several lines in the majority of the programs used over the last few months in this magazine.

The lines are:

```
1 'Program Name  and    Author
2 GOTO10
3 SAVE"Program Name":'or CSAVE
4 I$=INKEY$:IFI$=""THEN4
5-9 Space for additional coments.
```

The advantage of these lines is that when a program is being modified, the name is unimportant, you can make a quick SAVE (or CSAVE) just by RUNning 3.

The routine could be anywhere in your program, but if we all use the same line numbers, then it will be easier for all in the longer term.

We would encourage the use of these lines in programs submitted to Australian Rainbow and CoCo. We usually end up inserting them in any case - it really does help, especially when you are moving quickly from one program to another!

Listing 1: *BYTESCRN*

(To load 51-column screen on disk-based systems)

```
10 CLEAR1,&H7CC1 :REM ...change
this value to &h3cc1 for 16k mac
hines
20 CLEAR 300
30 LOADM"BYTESCRN
32 REM change LOADM to CLOADM fo
r tape systems
40 PMODE 4,1:SCREEN 1,0
50 EXEC &H7CC2:REM ...change thi
s value to &h3cc2 for 16k machin
es
60 CLS
70 S$=STRING$(28,"*")
80 PRINT TAB(14) S$
90 PRINT TAB(14) "* 51 column Sc
reen program *"
100 PRINT TAB(14) "*  Written by
: Chris Bone  *"
110 PRINT TAB(14) "*        for BY
TEMASTER        *"
120 PRINT TAB(14) S$
130 NEW
```

It was the PMODE 4,1: SCREEN 1,0 that was left out of the loading instructions in the February listing. After you did everything the column said, you still only had a regular text screen, a flashing cursor and an OK prompt. I apologize for the trouble this has caused.

The Silver Lining

There is one good thing that came out of the problem, however. I had the chance to talk to, and hear from, a great many of you. I believe I have a better idea of the questions you have and the difficulties you are facing. Because of the many questions I received, I decided to use part of this column to answer a few of the most common ones. Here are some answers:

1) The program listing in THE RAINBOW is correct. If entered correctly, and loaded as shown at the beginning of this column, it works as advertised.

2) The program does work with graphics. You can write or modify a BASIC program to do such things as draw a graph, label the points and provide explanations, all on the same screen. As well as draw, you can use the CIRCLE, LINE, PAINT, etc., all combined with your new text characters.

3) While the CLEAR key does not clear the screen, you can accomplish this function with either the CLS command or a PRINT CHR$(12) command.

4) The majority of machine language programs you have, or would like to have, probably do not work with *Bytescreen*. If the machine language programs perform character output by using the ROM call at &HA002, you might be in luck. If it writes to the screen by loading the text screen memory locations from a register, nothing appears on the graphic screen.

In any case, *Bytescreen* does not work with a machine language program that loads or uses memory anywhere above &H7CC1. Machine language programs cannot overlap. (My personal experience is that very few machine language programs work together unless they were specifically designed to do so.)

5) It would be extremely difficult to patch programs, like a word processor or *EDTASM+*, to work with *Bytescreen*. Chris cannot take on the job of trying to do so.

6) The *Bytescreen* program is for you to use in any way you like for a noncommercial purpose. If you wish to use any of the code in a program you are creating for commercial purposes, you must obtain permission from Chris to do so.

7) It does not matter if you load *Bytescreen* before or after you load a BASIC program, as long as you do not use a BASIC loader, such as the one at the beginning of this article. Loading one BASIC program when another is in memory destroys the program in memory.

You can, however, load *Bytescreen* by following the CLEAR 1,&H7CC1: CLEAR 300: LOADM (or CLOADM) "BYTESCRN": EXEC &H7CC2 routine. Remember, if you have a 16K machine, exchange &H7CC1 and &H7CC2 with &H3CC1 and &H3CC2.

8) I don't know what to do for those of you who wish to use the listings in our articles without having an editor/assembler. The reason I am writing these articles is to teach you how to write assembly language. If you are reading the articles, you should be interested in assembly language. On the other hand, I, too, was interested in what machine language programs could do before I felt ready to tackle assembly language programming.

For now, I have written a BASIC program that lets you poke machine language code directly into your computer's memory (see Listing 2). Along with the program, I have included instructions on how to use it and how to save the results as a machine language program.

I know from experience that using such a program is hard to do. Making errors is easy, but finding them is difficult. With great amounts of care and checking, you can succeed. My program, called *Bytemaster Coder*, also helps you examine and change memory locations.

Chris and I have discussed writing a monitor program to give you options of examining and changing memory, examining registers, single stepping and the like. However, there have already been good monitor programs in THE RAINBOW. (In fact, there was one in the February issue, along with *Bytescreen*.) I have written the *Bytemaster Coder* because it is simple and easy to use and takes no special knowledge or techniques. At the same time, what it does is simple — it lets you examine and change RAM memory.

Because of the many who have expressed a wish to get started in machine language without an assembler, I am interrupting the planned flow of my columns for a bit to spend some time on the relationship between assembly language and machine language.

9) Because of the many letters and phone calls I receive, I will try and answer more of your questions in this column. This means there will be less room for instructions, but the topics should be of more immediate concern to you. I find that many of your questions are about some very basic assembly language problems and, as this column is for beginners, I think it would be best to clear up more of these points.

Also, I just don't have the time or money to answer so many letters. Many letters still don't have return postage, but I hate to be miserable and not answer them just because of a 20 cent stamp. I will still try to answer letters directly as much as possible, but you can expect some of my answers to appear in print.

10) No, I won't provide BASIC programs that automatically

poke the assembly language programs we present into memory and execute them. We have to draw the line somewhere.

11) Yes, I do like Texas and I love the winters where temperatures never get below zero. But I can't understand why they don't install plumbing so pipes don't freeze in *above* zero weather!

12) The only way to learn assembly language is to do it. Start at a level you can handle and keep plugging away until you have it beat, then move up. There is no easy way. If you are dedicated, tutorial programs can help. If you are not dedicated, they waste your money. Assembly language is a prime example of the old saying, "You get what you pay for." In this case, the payment is time and determination.

All about Machine Code

Now, on to the matters at hand. As mentioned, I am going to spend some time on machine language code. I get the very distinct impression that there are many who do not really understand what machine language is, what assembly language is, or what to do with either one. I am gbing to give you the information you need to actually write a machine language program without the aid of an editor/assembler, and to know what you are doing.

Those of you who have an editor/assembler can follow along, too. This information is important to your understanding of assembly language as well. You can do assembly language programming without knowing about machine language code, but it definitely rounds out your education.

First, machine language is code that tells your computer what to do. Your computer would be a nice looking box of plastic and metal without it. Originally, computers had to be fed information using physical switches. A switch turned on created a value of one; a switch turned off created a value of zero.

It may help you to understand the concept if you think of bits and bytes as a type of Morse code. They are very close to the same thing. Instead of long and short beeps, machine language code uses ones and zeros, which are created by a high and low voltage level. Although home computers do not need you to flip toggle switches, they still work in the same way.

The switches that get your computer up and running are built into ROM chips and are permanently set either off or on. The central processing unit (CPU) in your computer reads the values of these switches (either zero or one). It is programmed (it also has switches) to perform certain acts, according to the message it gets from ROM.

The CPU receives data sequentially, that is, one instruction after another, just as Morse code is sent. However, unlike Morse code, it can accept eight bits (one byte) at a time, where Morse code has to be sent serially, one bit of the code at a time. Some computers have CPUs that can receive 16 bits or 32 bits at a time.

Talking in Machine Code

Now, with that background, let me say that you also can talk to the CPU — machine language is the way you do it. While ROM contains messages that cannot be changed, RAM is just a bunch of blank pages waiting to be filled. Your CPU can be told to read its messages from RAM just as well as from ROM. Thus, all you need to know is how to write its language — machine language. Machine language and machine code are the same thing.

When you write a program in machine language, it is called hand assembling. That is, you must either have memorized the codes to perform a certain function or you must look it up. For instance, the code to tell the CPU to load the 'A' register with a number is $86. That is 86 Hex or 134 decimal or 10000110 in binary (remember those switches?). This causes an immediate load of the 'A' register of the byte following this code.

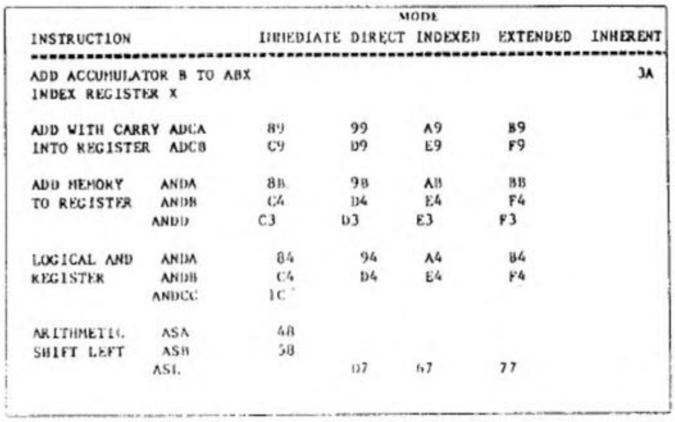
For instance, if you wanted to load the 'A' register with the decimal number 10, you would write two bytes of code: 86 0A. If you have an assembler, it does the dirty work for you. It translates your commands into machine code (assembles them). The same instruction in assembly language is: LDA $0A.

The code to load the 'B' register with a specified number is $C6 or 198 decimal or 11000110. Feeding this code to the CPU is the same as if you had a bank of eight switches and turned on switches '7,' '6,' '2' and '1' (remember that the eight bits of a byte are numbered from zero to seven). Table 2 shows some more examples of machine language code as it relates to assembly language code.

As you can see, the list could go on and on. I won't use up more of this column's space on it, but any good assembly language book contains a similar and complete list of "op" codes.

Assembling in BASIC

How does all this help those of you who do not have assemblers? Well, your CPU doesn't really care how you create the codes that tell it what to do. These codes can come from a BASIC program just as well as from an assembler. In fact, you could write a BASIC assembler, if you wished, but it would be slow.



INSTRUCTION		MODE IMMEDIATE	DIRECT	INDEXED	EXTENDED	INHERENT
ADD ACCUMULATOR B TO ABX INDEX REGISTER X						3A
ADD WITH CARRY	ADCA	89	99	A9	B9	
INTO REGISTER	ADCB	C9	D9	E9	F9	
ADD MEMORY	ANDA	8B	9B	AB	BB	
TO REGISTER	ANDB	C4	D4	E4	F4	
	ANDD	C3	D3	E3	F3	
LOGICAL AND	ANDA	84	94	A4	B4	
REGISTER	ANDB	C4	D4	E4	F4	
	ANDCC	1C				
ARITHMETIC	ASA	48				
SHIFT LEFT	ASB	58				
	ASL		07	67	77	

All an assembler does is look at a mnemonic, such as LDA, and convert it to the proper numeric code, such as 86. It is the 86 that tells the CPU that it is to load the next byte presented to it into the 'A' register.

To see how this works, let's write a machine language program without an editor, assembler or anything but good old BASIC. The program uses the 'D' register to add two 16-bit numbers together. The numbers are $300 and $400.

Here are the codes you need to do the job:

1) The code to load a load a number in to the 'D' register is CC.
2) The code to add a number to the 'D' register is C3.
3) The code to store a number in memory from the 'D' register is FD.
4) The code to return to BASIC from fhe add routine is 39.

Now, from BASIC, type the following and ENTER after each line:

```
POKE &H3000,&HCC
POKE &H3001,&H03
POKE &H3002,&H00
POKE &H3003,&HC3
POKE &H3004,&H04
POKE &H3005,&H00
POKE &H3006,&HFD
POKE &H3007,&H04
POKE &H3008,&H00
POKE &H3009,&H39
```

And you have just written a machine language program. First, memory location $3000 was chosen so the program would work in any 16K and up computer. Then, the code to load the 'D' register with a number was poked into the first memory location. The next two numbers poked are the most significant byte and least significant byte of the number to be loaded: $03 and $00. The code to add a number to the 'D' register, C3, was then poked into the next memory location, &H3003. This process was carried through to the end of the program.

To see if the program works, type EXEC &H3000 and ENTER. If all of your codes are right, a reverse '@' and an asterisk appears in the top-left corner of the video screen. If you are using *Bytescreen*, nothing appears (but it puts some unwanted values in *Bytescreen* if you are operating a 16K machine). You have to be in the regular text screen to see the results of the machine language program.

Clearing the Mystery

I hope that clears up the mystery of machine language code. Now let's deal with how to find the machine language code in an assembler program source listing, then how to use my BASIC program to make it much easier to enter code into memory. Below is an assembler listing of the previous program:

```
3000                  00000    ORG     $3000
3000  CC   0300       00010    LDD     #$300
3003  C3   0400       00020    ADDD    #$400
3006  FD   0400       00030    STD     $400
3009  39              00040    RTS
           0000       00050    END
```

If you look closely, you see that this listing contains all of the numbers you previously poked into memory from BASIC. They are found in the second and third columns. The first column is the memory location where the code goes, the second column is the machine language code and the third column is the values the code acts upon, or the operands.

What you do is put all of the values starting at CC into successive memory locations. Numbers with four digits require two memory locations. If you can do this without making mistakes, you have accomplished everything that an assembler does.

To enter the above program, you start at memory location &H3000 and enter the Hex values CC 03 00 C3 04 00 FD 04 00 39 into $3000 through $3009. The following BASIC program is designed to make that task much easier. This gives those of you who do not have assemblers a chance to try out our codes.

Also, note that the previous assembly language listing has a beginning line using ORG. This tells you where the program is to begin in memory, in this case $3000. You can also tell that the execution address is also at $3000. The beginning and execution addresses are not always the same, but you are usually told if they are different. The end of the program is where the last program code ends ($3009 in the sample program).

Listing 2: A BASIC Program to enter Machine Language Code

Enter and run the program: You are prompted to enter the starting address; type in and enter the address where you wish your machine language code to begin.

Twenty-four memory location values are to be printed to the screen, beginning at the starting address. Use the arrow keys to move anywhere in these 24 bytes and make any changes you wish. If you try to go beyond the memory locations displayed on the screen, the display automatically increments or decrements by 24 bytes. Any changes you make are to be poked into the memory location displayed to the right of the 24 bytes.

To enter a machine language program, look for the proper values in the assembled listing, choose the memory location indicated by the program and begin typing in the values.

When all of the code is entered, use the CLEAR key to escape to the saving procedure. You are asked for a beginning address, an end address and the execution address. Enter these values as indicated by the assembly language listing and as explained in this article. The code is saved as a machine language program and can be placed into memory with CLOADM or LOADM.

100	194
360	42
500	95
660	206
END	41

Listing 2: *BYTECODR*

```
1 '***************************
2 '*      BYTEMASTER CODER     *
3 '*     BY R. BARTLY BETTS    *
4 '*       2251  LIPSCOMB      *
5 '*     FORT WORTH,  TEXAS    *
6 '*           76110           *
7 '***************************
8 'USE THIS PROGRAM TO INPUT
9 'MACHINE LANGUAGE CODE INTO
10 'MEMORY.  THE FOLLOWING KEYS
11 'ARE ACTIVE:
12 'RIGHT ARROW = AHEAD 1 BYTE
13 'LEFT ARROW = BACK 1 BYTE
14 'UP ARROW = BACK 8 BYTES
15 'DOWN ARROW = AHEAD 8 BYTES
16 '<-> OR <=> = BACK 24 BYTES
17 '<+> OR <;> = AHEAD 24 BYTES
18 '<CLEAR> = PREPARE TO SAVE
19 'THE NUMBER KEYS AND THE
20 'ALPHABET CHARACTERS "A - "F
21 'CAUSE A VALUE TO BE PUT
22 'INTO MEMORY
24 '
25 '
100 CLS
110 V=32
```

```
120 DIM M(24)
130 A$(1)="BYTEMASTER CODER"
140 A$(2)="BY R. BARTLY BETTS"
150 A$(3)="JANUARY :: 1985"
155 '       SET UP SCREEN AND
156 '       GET ADDRESS
160 FOR T=1 TO 3
170 PRINT TAB(16-LEN(A$(T))/2) A
$(T)
180 NEXT
190 GOSUB 790
200 PRINT@V*12," START ADDRESS I
N HEX";
210 INPUT BG$
220 B=VAL("&H"+BG$)
230 BB=B
240 FOR T=0 TO 23
250 PRINT@M(T),HEX$(PEEK(BB))
260 BB=BB+1
270 NEXT T
290 A$=CHR$(128):B$=CHR$(32)
300 P=0
305 '       KEYBOARD INPUT
306 '       TO EXAMINE AND CHANGE
310 IF P>23 THEN P=0:B=B+24:GOTO
 230
320 IF P<0 THEN P=0:B=B-24:GOTO
230
330 M=M(P):C=PEEK(M+1024):H$=""
340 PRINT@187,HEX$(B+P);
350 IF C>63 THEN G=C-64 ELSE G=C
+64
355 '       WAIT FOR KEYPRESS
356 '       AND PRODUCE CURSOR
360 K$=INKEY$:POKE M+1024,G: IF
K$="" GOTO 360
370 POKE M+1024,C
375 '       LOOK FOR VALID
376 '       KEYPRESS
380 IF K$=CHR$(94) THEN P=P-8:GO
TO 310
390 IF K$=CHR$(10) THEN P=P+8:GO
TO 310
400 IF K$=CHR$(8) THEN P=P-1:GOT
O 310
410 IF K$=CHR$(9) THEN P=P+1:GOT
O 310
420 IF K$=CHR$(12) THEN 660
430 IF K$="+" OR K$=";" THEN B=B
+P+24:GOTO 230
440 IF K$="-" OR K$="=" THEN B=9
+P-24:GOTO 230
450 IF K$="N" THEN RUN
455 '       LOOK FOR INVALID
456 '       KEYPRESS
460 IF ASC(K$)<48 OR ASC(K$)>70
THEN 360
470 IF ASC(K$)>57 AND ASC(K$)<65
 THEN 360
475 '       INCREMENT MEMORY
```

```
476 '       IF END OF DISPLAY
480 IF P<0 THEN B=B-24:P=1:GOTO
230
490 IF P>24 THEN B=B+24:P=1:GOTO
 230
495 '       PRINT TO SCREEN
496 '       AND GOT TO NEXT CHAR
500 PRINT@M,K$;
510 H$=H$+K$
520 M=M+1
525 '       ROUTINE FOR SECOND
526 '       CHARACTER INPUT
530 C=PEEK(M+1024)
540 IFC>63 THEN G=C-64 ELSE G=C+
64
550 POKE M+1024,G
560 K$=INKEY$:IF K$="" THEN 560
570 IF ASC(K$)<48 OR ASC(K$)>70
THEN 560
580 IF ASC(K$)>57 AND ASC(K$)<65
 THEN 560
590 POKE M+1024,ASC(K$)+64
595 '       ADD UP INPUT VALUES
596 '       AND POKE IN MEMORY
600 H$=H$+K$
610 PK=VAL("&H"+H$)
620 POKE B+P,PK
630 PRINT@M-1,HEX$(PK);
640 P=P+1
650 GOTO 310
655 '       SAVE PROGRAM TO
656 '       TAPE OR DISK ROUTINE
660 CLS
670 A$(1)="BYTEMASTER CODER"
680 A$(2)="================"
690 FOR T=1 TO 3
700 PRINT TAB(16-LEN(A$(T))/2) A
$(T)
710 NEXT T
720 PRINT@V*4+2,"* START (HEX)..
.";:INPUT BM$:BM=VAL("&H"+BM$)
730 PRINT@V*5+2,"* END (HEX)...."
;:INPUT EM$:EM=VAL("&H"+EM$)
740 PRINT@V*6+2,"* EXECUTION (HE
X)...";:INPUT EA$:EA=VAL("&H"+EA
$)
750 PRINT "NAME OF PROGRAM...";:
INPUT NP$
760 REM USE THIS LINE FOR DISK:
SAVEM NP$,BM,EM,EA
770 REM USE THIS LINE FOR CASETT
E:CSAVEM NP$,BM,EM,EA
780 END
785 '       DATA FOR POSITION
786 '       OF SCREEN DISPLAY
790 FOR X=160 TO 224 STEP 32
800 FOR T=0 TO 21 STEP 3
810 M(X/4-40+T/3)=X+T
820 NEXT T,X
830 RETURN
```

KISSable OS-9

News, Hints And Answers

By Dale L. Puckett

We don't have a lot of news this month, but we have more questions to answer. We'll start with a load of hints and we'll wrap up the column with a number of interesting BASIC09 procedures from several readers.

First, I stumbled upon a long thread where members were discussing the merits of several alternatives to Tandy's CCDISK module and learned about a new driver we haven't mentioned before. MJS Software (3121 Sea Lane, Bremen, IN 46506, (219) 546-4009) offers a CCDISK that reportedly does an excellent job handling 80-track, double-sided drives.

A lot of the coding was done by an OS-9 pioneer, Carl Kreider. Carl is one of the leading contributors to the OS-9 Users Group's software library and is very knowledgeable. If you call MJS, tell them they should have let us know about it sooner! That goes for anyone producing OS-9 software . . . tell us and we'll tell the world in "KISSable OS-9."

We mentioned recently that several readers were interested in running OS-9 on the Dragon computer; while reading the SIG, we noticed that Jim Omura had left the company's address: Dragon Data Ltd., Kenfig Industrial Estate, Margam, Port Talbot, West Glamorgan, SA13 2PE. Should be a good place to write for Dragon information.

Speaking of addresses, Jonathan C. Keatley left the following for the Dragon's 6551 ACIA:

$ff04 — Receive/Transmit Data
$ff05 — Status Register
$ff06 — Command Register
$ff07 — Control Register

Jonathan also left a four-line BASIC program that emulates a dumb terminal. If you have one of the new RS-232 Paks and the new version of OS-9 with the *ACIAPAK* drivers, you should be able to emulate it nicely in BASIC09. When you do, you'll need to use the corresponding addresses for the RS-232 Pak's registers. See the *SysType* listing later in this column or look in the device descriptor for /T2 to find the base address of the RS-232 Pak's ACIA. Here goes!

```
10 POKE &HFF06, &H6B : POKE
   &HFF07, &H36
20 Y$=INKEY$ : IF Y$< > "" THEN
   POKE &HFF04, ASC(Y$)
30 IF PEEK (&HFF05) AND 8 THEN
   PRINT CHR$(PEEK(&HFF04));
40 GOTO 20
```

Software Library News

You've probably had a chance to peruse the complete listing of the OS-9 Users Group's Software Exchange Library in the May RAINBOW. Here's some more good news. The list you read was complete as of February 1, 1985. I've learned that 10 more disks have already been added to the list. We'll try to get it compiled for you in a future RAINBOW. Dave Kaleita, the group's software librarian, has sure been busy.

MOTD, the group's newsletter, has

continued from Page 26

```
730 IFSC>SC(1) THENIFGP<>1THENCO
LOR5,0:FORCF=1TO2:Q1=127:Q2=104:
Q3=104:FORT=127TO0STEP-3:Q1=Q1+3
:Q2=Q2+2:Q3=Q3-2:LINE(T,Q3)-(Q1,
Q2),PSET,B:NEXTT:COLOR0,0:NEXT:C
QLOR5,0:GP=1
740 DRAW"C5BM72,3D4ND4R4NU4D4BR3
NU8BR4U8NL2R2BR3D5BD2D"
750 GOTO450
760 PLAY"L20V31O1;6;5;4;3;2;1;L3
0;6;5;4;3;2;1;L50;6;5;4;3;2;1;L7
0;6;5;4;3;2;1;L90;6;5;4;3;2;1;L1
30;6;5;4;3;2;1;L200;6;5;4;3;2;1"
:FORT=1TO50:LINE(RND(255),RND(19
1))-(AS,SD),PSET:NEXT
770 GOTO910
780 CLS
790 PRINT@11,"CHOPPER":PRINT@32+
11,"ASSAULT":PRINT@32*2+7,"BY JE
NS PETERSEN"
800 PRINT@32*3,"PRESS LEVEL OF D
IFFICULTY"
810 PRINT@32*4,"1- BEGINNER":PRI
NT@32*5,"2- EXPERT":PRINT@32*6,"
3- PRO"
820 PLAY"L255V31O1":FORT=1024TO1
535:Z=PEEK(T):IFZ>63THENPOKET,Z-
64:PLAY"1"
830 NEXT
840 A$=INKEY$:IFA$=""THEN840ELSE
IFVAL(A$)<1ORVAL(A$)>3THEN840
850 PLAY"L255V31O1;1;2;3;4;5;6;7
;8;9;10;11;12"
860 PRINT@256,"name";
870 POKE282,0
880 INPUTNA$:IFNA$=""THEN860
890 POKE282,1
900 U7=VAL(A$)+2:F3=VAL(A$):RETU
RN
910 CLS
920 PRINT@64+11,"GAME OVER"
930 PRINT@0,"";:PRINTTAB(8)"CHOP
PER ASSAULT"
940 IFSC>SC(1)THENSC(3)=SC(2):NA
$(3)=NA$(2):SC(2)=SC(1):NA$(2)=N
A$(1):SC(1)=SC:NA$(1)=NA$
950 IFSC<SC(1)ANDSC>SC(2)THENSC(
3)=SC(2):NA$(3)=NA$(2):SC(2)=SC:
NA$(2)=NA$
960 IFSC<SC(2)ANDSC>SC(3)THENSC(
3)=SC:NA$(3)=NA$
970 PRINT@128+11,SC;NA$;
980 PRINT@192+10,"HIGH SCORES"
990 PRINT@256+10,SC(1);NA$(1);
1000 PRINT@288+10,SC(2);NA$(2);
1010 PRINT@320+10,SC(3);NA$(3);
1020 FORT=1024TO1535:Z=PEEK(T):I
FZ>63THENPOKET,Z-64
1030 PLAY"L255V31O4;D":NEXT
1040 P=PEEK(65280):IFP=126ORP=25
4THEN80ELSEIFINKEY$="Q"THEN1050E
LSE1040
1050 CLS:CLEAR:POKE65494,0
```

picked up a new contributing editor. Hubert "Bert" Schneider in Omaha, Neb., has signed on to write a regular column about the OS-9 Users Group's Software Exchange Library. He'll be highlighting software in the library and reviewing it for you. I received *MOTD* number five recently. It looked great and featured an excellent overview of OS-9 from Greg Morse, plus at least a dozen other good articles.

But, the group's new editor, Tim Grovac, is already preparing another issue. We quote: "I need some more articles for *MOTD*. Become famous instantly! Help support your Users Group! Certainly there must be something you all are doing with your computers that others would like to hear about." Send disk or printed copy to:

MOTD Publishing
25825 104th Ave. SE
Suite 344
Kent, WA 98042

We still keep getting letters here at THE RAINBOW asking how to join the OS-9 Users Group. Once again, here's the address.

OS-9 Users Group
P. O. Box 7586
Des Moines, IA 50322

You may use this address either for information or to join. To join, simply enclose a check for $25 — one year's dues — and state the name of the computer you own and the type of disk drives you use so you will receive your copy of Users Group Disk #0 on a disk of the right format. Make sure you include your correct address and include your CompuServe number if you have one.

Joe Dubuc, chairman of the Membership Committee, has received many requests for information about local users groups. People want to know where they meet and how to form one. Please send Joe information about any groups you know about. Give him the group's name, its main interest, the name of a contact person, the group's BBS number, its meeting place and the date and time of its monthly meetings. Here's the address.

OS-9 Users Group Membership
  Committee
13229 Blue Quail Rd.
Yukon, OK 73099

New OS-9 Machines

Since the rumor mill has decided that Tandy won't release a new Color Computer until 1986, I found this note from Steve Sampson interesting.

If you are interested in a high performance OS-9 machine, please contact Jack Gerblick, 1945 Gallows Road, Suite 305, Vienna, VA 22180. It seems Fujitsu is thinking about selling its 68XX(X)-based machines here in the states.

In Steve's words, "The FM-11 is a simply astonishing dual 6809-based machine with very good color graphics and OS-9/6809 Level Two. In my opinion, it blows away a Macintosh, even without the 68000 board that can be installed in it. The FM-77 is another dual 6809 machine that starts smaller than the FM-11 but is expandable. The FM-16, in its Japanese incarnation at least, is a 68000-based computer."

Anyone who attended either of the last two Microware Seminars in Des Moines can vouch for Steve's description. They were pretty slick. The bottom line? I guess it's up to us — today's OS-9 pioneers — to convince Fujitsu they need to invest in the U.S. market. You've got the address; go to it.

Here's a tip from John Schira that may help solve your problems with *ACIAPAK* and /T2 in OS-9 Version 1.01. John believes the people who aren't having any trouble are using smart modems. Conversely, he feels if you are using a dumb modem — a Radio Shack Modem I, for example — you're probably having trouble making this combination work right. The reason?

"Smart modems leave the carrier detect signal between the computer and the modem high — or on — so the computer can send commands to the modem while it is offline. I've found that /T2 and *ACIAPAK* work consistently well as long as this signal is present," he said. "Without this signal, they won't work. The solution is to try jumpering pin 8. If this doesn't work, try pin 6."

Invasion of the Hard Drives? Maybe!

I've received about a half-dozen calls about hard disk drives during the past month. It seems like everyone has noticed the price dive the bare drives are taking and are hoping some enterprising entrepreneur will come out with a system for their CoCo. I saw one working at Irvine, but the company hadn't announced it yet — they believe in announcing a product when it is ready, not before. When they tell us it's ready, we'll let you know.

The real problem here is the cost of the cables, controller, power supply and everything else it takes to build a complete hard drive system. Another firm designed a system for the Color Computer recently, but will it ever go into production? I doubt it. Why? Even though a manufacturer can buy a five megabyte hard drive for around $100 now, it is still going to cost them around $600 (final selling price) to build a tacky system . . . or $1400 to build one that discriminating computer owners would be proud to own. The question then remains: Is a person who paid $200 for his computer going to spend $600 — or $1400 — for a hard disk system? Probably not!

For the same reason you aren't seeing software houses rush to invest the talent and time necessary to develop new applications software, you probably won't see much new hardware either. Because of the unique marketing strategy used by Tandy (i.e., they only sell their computers in their own stores), a manufacturer can only sell peripheral equipment by mail order. When he does, he may reach 10 percent of the market. To succeed he needs a higher percentage. The software houses are in the same boat, so we all lose. That's life!

This scenario was played out again in a letter I received from Cliff Davis (12714 Burson Drive, Manchaca, TX 78652). It seems that he and Jim Smith have designed a CoCo RAM Disk. The ttl prototype uses 55 chips in addition to four banks of dynamic memory chips and bank select logic to support four additional banks — a total of 512K. It plugs into the bus expander and uses the Color Computer's 'E' and 'Q' clocks for timing. To transfer data, you send a two-byte logical sector address and a function code to the controller. The software includes an OS-9 device driver, device descriptor and a "prep" utility.

So what's the problem? Well, Cliff and Jim have gone to three companies so far. All have said that it looked like a great enhancement for the CoCo, but, they don't believe the market will bear the cost of the finished product. Cliff's alternative is to offer the board as a construction project in RAINBOW. He would like to create enough interest in the design to justify making a PC board. Let him know what you think.

The listing:

```
procedure elapsed
 REM by Thomas Alan Ring
 REM 75 Market, Apt. #4
 REM 315-265-2808
```

New Software Newsletter

I talked with Frank Hogg at FHI and learned that his company plans to publish a newsletter for software developers. It's for you if you are developing software for any computer. Regular columns will feature columns for programmers, engineers and yes, even the marketing types. The new publication will be named *SoftNews*.

Frank has been in the software business for several years so he has plenty of experience to share. The price: $24 for 12 issues. The first issue was scheduled to hit the stands in May. After that, it will be published every other month until August when it goes monthly. If you are developing software for any computer or are interested in the software development business, call FHL. By the way, if you are in the business and have a few tips to share, Frank is also looking for writers.

Everyone's Talking about OS-9 68K

Frank couldn't contain his enthusiasm for OS-9 — 68K that is. He had been working with Microware's C compiler on his K System."QT," and successfully ported many C programs from the OS-9 Users Group Library to the QT.

"All of the C programs that were written in 6809 Microware C compiled the first time in 68000 C and most ran immediately," Frank said. "The only ones that wouldn't run were the ones that were written specifically for the 6809 microprocessor using in-line assembly language code. Some of the C programs written in Introl C would not compile, but this is true for the 6809 C compiler, also."

Frank reported that most of the BASIC09 programs loaded and ran immediately. The only one that wouldn't run was a modem program that used a lot of direct pokes to memory. "It is quite a kick to type 'Basic09 #375375k and receive a report that you have 388,106 bytes free for programs (in a 512K QT)."

And Frank wasn't the only one excited about OS-9 68K. Rodger Snyder at Great Plains Computing — now named Stylo Software, Inc. — reports that you can edit a file 150 pages long and have it all in memory at one time. Wow! Also, Brian Lantz, author of Computerware's *Databank Manager*, reported that BASIC09 appeared to be almost 100 percent compatible — at the source level — with BASIC09 on the Color Computer. He noticed that a new function, "INKEY #filenumber" has

```
REM To start typing "run elapsed("S",et)
REM To Finish typing "run elpased("F",et)

PARAM sf:STRING[1]; et:INTEGER
DIM f,s:STRING[17]
DIM sc,fs,sm,fm:INTEGER
DIM sh,fh,sd,fd:INTEGER
DIM es,em,eh,ed:INTEGER

ON ERROR GOTO 1

 IF sf="S" THEN
   f=DATE$
   END
 ELSE
     PRINT "Wrong Input Parameter: "
     PRINT "Use S(tart) or F(inish)"
     END
 ENDIF

 sc =VAL(MID$(s,16,2))
 fs = VAL(MID$(f,16,2))
 sm=VAL(MID$(s,13,2))
 fm=VAL(MID$(f,13,2))
 sh=VAL(MID$(s,10,2))
 fh=VAL(MID$(f,10,2))
 sd=VAL(MID$(s,7,2))
 fd=VAL(MID$(f,7,2))

 IF fs-sc < 0 THEN
    fh=fm-1
    fs=fs+60
 ENDIF

 IF fm-sm < 0 THEN
    fh=fh-1
    fm=fm+60
  ENDIF

 IF fh-sh<0 THEN
   fd=fd-1
   fh=fh+24
 ENDIF

  es=fs-sc
  em=fm-sm
  eh=fh-sh
  ed=fd-sd
  et=es+60*(em+60*(eh+24*ed))
 ENDIF
 END

1 PRINT "Probable date/time error: "
   PRINT "Elapsed Time will be wrong. "
   PRINT "Check date, t."
   END

PROCEDURE cursor_positioning
 0000       DIM test_string:STRING[80]
 000C       DIM blank:STRING[1]
 0018       DIM data_inputs,count,data_lines,remainder,zilch:INTEGER
 002F       blank:=" "
 0037       data_lines:=14
 003E       data_inputs:=1
 0045       zilch:=0
 004C       BASE 0
 004E       PRINT CHR$($0C)
 0054       PRINT USING "S80^","Contributed to RAINBOW by Mark W. Smith"
 0087       PRINT USING "S80^","Routine to show one way the MOD function can be
used in cursor positioning"
                                          ;
 00DE       PRINT USING "S79^","to replace the POS function that WORDPAKII does
not support."
 0126       PRINT
 0128       PRINT CHR$($02); CHR$($20); CHR$($36);
 0139       PRINT USING "S80^","Hold down the <RETURN> key for demo";
 0169       PRINT CHR$($02); CHR$($20); CHR$($26);
 017A       REPEAT
 017C         remainder:=MOD(data_inputs,data_lines)
 0188         INPUT "P R O G R A M  D A T A  M A Y  B E  P L A C E D  I N
T H I S  A R E A !"
                                          ,test_string
 01DF         PRINT CHR$($08); "!";
 01EA         IF remainder=zilch THEN
 01F7           PRINT CHR$($02); CHR$($20); CHR$($26);
 0208           PRINT CHR$($1B); CHR$($42);
```

been added along with a "DIGITS" statement that lets you control the number of digits printed from a real number.

More Version 1.01 Notes

John Carter of Smyrna, Ga., who showed you how to personalize your OS-9 prompt several months ago, has been studying the differences between OS-9 Version 1.0 and Version 1.01, and he was good enough to share them with us.

His first tip is that the *OS9Boot* file that comes with the new version is $3607 bytes long compared with $3032 bytes in the original. This means if you use the trick we passed along in the February column to place the CMDS directory at the same location on each disk, you must make sure all of those disks are using the same version of OS-9. If you have different versions the trick will not work, so beware.

Here is a table that shows you a few more of the size differences.

In the CMDS directory:

File	Original Size	New Size
dcheck	$28A0	$27C6
free	$2C1	$2D1
ident	$6CE	$6E7
tmode	$2CF	$2DE
xmode	$380	$38F

In the DEFS directory:

OS9Defs	$4A7A	$54B4
RBFDefs	$11FF	$154E
SCFDefs	$A0F	$E94
SysType	$42	$81

The DEFS directory in the new version has a new file named *defsfile*. This file has been in non-Color Computer versions of OS-9 for several years and simply tells the assembler to use all of the other "defs" files.

If you are a person who delights in trying to stay on top of what Tandy is up to with the Color Computer, Carter suggests you browse through the files in the new DEFS directory. You'll learn that plans really did exist for a "Deluxe CoCo" at one time — study these lines from the new *SysType* file.

```
  ifeq COCOType-Delux
 ACIAType set ACIA6551
 A.T2 set $FF3C 6551 Acia
Internal
 A.T3 set $FF68 6551 AciaPak
  else
```

```
 0214            FOR count=1 TO 16
 0224              PRINT blank
 0229            NEXT count
 0234            PRINT USING "S80^","Hold down the <RETURN> key for demo"


 0264            PRINT CHR$($02); CHR$($20); CHR$($26);
 0275          ENDIF
 0277          data_inputs=data_inputs+1
 0282        UNTIL data_inputs=500
 028E        END
PROCEDURE gfxtest
 0000        DIM f,g:INTEGER
 000B
 000C        g=0
 0013 10     PRINT CHR$(15); CHR$(1); CHR$(g)
 0024        PRINT CHR$(20)
 0029        PRINT CHR$(21); CHR$(4); CHR$(0)
 0036        PRINT CHR$(22); CHR$(4); CHR$(g)
 0044        PRINT CHR$(21); CHR$(122); CHR$(95)
 0051
 0052        FOR f=5 TO 95 STEP 5
 0067          PRINT CHR$(26); CHR$(f)
 0071        NEXT f
 007C
 007D        g=g+1
 0088
 0089        IF LAND(g,3)=0 THEN g=g+1
 00A2        ENDIF
 00A4
 00A5        FOR f=1 TO 3000
 00B6        NEXT f
 00C1
 00C2        PRINT CHR$(19)
 00C7        GOTO 10
 00CB
 00CC 100    PRINT CHR$(18)
 00D4
 00D5        REM chr$(18) clears the graphics memory
 00FB        REM RUN, BREAK to exit, RUN 100 to clear gfx memory
 012D
PROCEDURE ftype
 0000        PARAM name:STRING[60]
 000C        DIM path:INTEGER
 0013        DIM f:REAL
 001A        DIM char:STRING[1]
 0026        DIM tst:BOOLEAN
 002D
 002E        ON ERROR GOTO 100
 0034        tst=TRUE
 003A        OPEN #path,name:READ
 0046        f=0
 004E
 004F        WHILE tst DO
 0058          SEEK #path,f
 0062
 0063          IF EOF(#path) THEN PRINT
 006E            CLOSE #path
 0074            END
 0076          ENDIF
 0078
 0079          GET #path,char
 0083          f=f+1
 008F
 0090          IF char=CHR$(7) OR char=CHR$(10) OR char=CHR$(13) OR char

    >=CHR$(31) AND char<=CHR$(127) THEN PRINT char;
 00C2          ELSE
 00C6            PRINT "\";
 00CC            PRINT USING "h2",char;
 00D8          ENDIF
 00DA
 00DB        ENDWHILE
 00DF
 00E0        CLOSE #path
 00E6        PRINT
 00E8
 00E9        REM if you want a character count, add the next line
 011C        PRINT "character count="; f
 0134 100    IF ERR=216 THEN PRINT name; " not found"
 0152        ENDIF
 0154        BYE
 0156


PROCEDURE screentest
```

```
A.T2 set $FF68 6551 ACIA
external
  endc
```

If you browse deeper into the "defs" files you'll also find a hint of OS-9's popularity in Japan — it's second in popularity there only to UNIX. There are references to "kata" and "kanji" and "Hoshi." Think about it — these Japanese characters can be drawn on a high resolution screen just as easily as English letters. Interesting!

BASIC09 Graphics Programs

Carter donated several BASIC09 listings that should really help you learn some of the language's fundamentals. We've had a lot of requests for information about using graphics under BASIC09. Two of John's procedures will really get you started. I was impressed when I ran them.

Gfxtest is a simple routine that draws a line and a series of concentric circles in several background/foreground combinations using print statements. *Screentest* uses BASIC09's "gfx" module to dazzle you with circles and lines in several colors and prints big letters. It also shows you how you can use several of the cursor positioning commands on an alpha screen from within OS-9. Enjoy!

Carter wrote a BASIC09 procedure that emulates the CP/M and MS-DOS Type command. It simply lets you display the printable characters in any file on the terminal. Itype, on the other hand, displays printable characters but, also displays the other characters in the file as a two-digit hexadecimal number. It works a lot like the standard "dump" utility it's just in a different format.

And finally, his *CoCoDir* lets you read the directory of a Radio Shack DOS disk from within BASIC09. It shows you how you to use OS-9's '@' operator along with BASIC09's SEEK and GET statements to look at any disk.

We received another BASIC09 procedure — *cursor_position* — that demonstrates yet another function from Mark W. Smith of Latonia, Ky. He uses the MOD function to create a window on PBJ's *Word-Pak II* since it does not recognize cursor positions greater than 512 when using the POS function.

Smith also had a question. He mentioned that he was unable to install the *Word-Pak II* drivers properly with Version 1.01 of OS-9. He mentioned that *ACIAPAK* and a few other modules didn't appear in memory after he created a new boot file.

Here's the answer, Mark. Most likely the "install" procedure and the *bootlist*

```
0000        (* demonstrates screen controls under coco os9
002E        (* John Carter - WB4HLZ - Feb. 1985
0052        DIM f,g:INTEGER
005D        DIM a,b,c,d:INTEGER
0070        DIM tst:BOOLEAN
0077
0078        tst=TRUE
007E        (* 12 clears screen, 1 homes cursor without clearing screen
00B9        PRINT CHR$(12);
00BF
00C0        FOR f=1 TO 12
00D0          PRINT "line "; f
00DD        NEXT f
00E8
00E9        FOR f=1 TO 3000
00FA        NEXT f
0105
0106        FOR f=1 TO 4
0116          (* 9 is "up one line"
012B          PRINT CHR$(9);
0131        NEXT f
013C
013D        PRINT "up from 12"
014B
014C        FOR f=1 TO 3000
015D        NEXT f
0168
0169        PRINT CHR$(1); "top line";
017A
017B        (* 10 is LF
0186        FOR f=1 TO 12
0196          PRINT CHR$(10);
019C        NEXT f
01A7
01A8        PRINT "press enter for graphics"
01C4        INPUT x$
01C9
01CA        (* this is the fun part
01E1        (* set 4 color mode [1] - (green background) yellow foreground [1]
0223
0224        RUN gfx("mode",1,1)
0236
0237        (* clear the graphics screen
0253        (* just in case there's something there
027A
027B        RUN gfx("clear")
0288        (* wait a bit
0295
0296        FOR f=1 TO 1000
02A7        NEXT f
02B2
02B3        FOR g=5 TO 35 STEP 5
02C8          RUN gfx("circle",45,95,g)
02E1          RUN gfx("circle",210,95,g)
02FA        NEXT g
0305
0306        (* wait a bit
0313        FOR f=1 TO 4000
0324        NEXT f
032F        (* "alpha" takes you back to the alpha screen
035C
035D        RUN gfx("alpha")
036A        INPUT "press enter to add blue [illegible]es",x$
038F        (* set blue foreground [2]
03A9
03AA        RUN gfx("mode",1,2)
03BC        RUN gfx("line",0,0,255,191)
03D4        RUN gfx("line",0,191,255,0)
03EC        RUN gfx("line",180,12,95,97)
0404        RUN gfx("line",180,181,95,96)
041C        RUN gfx("line",76,180,161,95)
0434        RUN gfx("line",76,12,161,97)
044C
044D        (* wait
0454        FOR f=1 TO 3000
0465        NEXT f
0470
0471        (* 14 is also back to text
048B        PRINT CHR$(14)
0490        PRINT "press enter to add red lines and text"
04B9        PRINT "then press enter again to exit"
04DB        INPUT x$
04E0
04E1        (* set red foreground [3]
04FA        RUN gfx("mode",1,3)
050C        RUN gfx("line",76,181,180,181)
```

file on the *Word-Pak II* disk probably were written for Version 1.0 which didn't contain those modules. Just edit "install" and *bootlist* to include the missing modules and I'm pretty sure they will appear. Good luck!

Another gung-ho BASIC09 programmer in the CoCo crowd is Tom Ring in Potsdam, N.Y. Tom sent two tips and a procedure that will give you accurate execution timings. It's called *elapsed*.

Ring passed along this tip which you may not have tried before. Use the global editing capabilities of BASIC09 to your advantage. It can save a lot of wear and tear on your fingers. Imagine that you want to use a long variable name like ElapsedSeconds in a BASIC09 procedure. Why not simply type ES and then use BASIC09's global change command. Give it a try.

```
E: c* .ES.ElapsedSeconds.  ENTER
```

Also Ring advised that if you are a little tight on memory, you can save 768 bytes when you run BASIC09 by using OS-9's built-in ex command. You'll have to use the Chd and Chx commands after you return from BASIC09, however, because when you run ex, OS-9 throws away the Shell that called it. Here's the command line:

```
OS-9: ex basic09
```

Don't Forget the Null Cable

If you're looking for a public domain communication protocol that gives you error checking and can be used on your Color Computer, Mark E. Sunderlin, a.k.a. Dr. Megabyte, suggests *Kermit*. It runs on more than 200 different machines ranging from the IBM 370 down to the CoCo and lets any two computers transfer text or binary files. Mark uses it to transfer data between his CoCo and a Zilog Z-8000 UNIX system at work. The CoCo version is written in C. You can get all versions from Columbia University in New York City but Mark didn't give us the address, so here's his: 1430 Greystone Terrace, Winchester, VA 22601.

And speaking of communications, Richard Cambell of Havelock, N.C., wrote to ask why he couldn't get his two Color Computers to communicate with OS-9. He uses OS-9 and an RS-232 Pak on one CoCo and wants to use the other as a terminal via its built-in RS-232 port. He says they both can talk to local bulletin boards, but when he connects one to the other — using

```
0524        RUN gfx("line",76,12,180,12)
053C        RUN gfx("line",95,96,161,96)
0554        RUN gfx("line",0,0,255,0)
056C        RUN gfx("line",1,191,255,191)
0584        RUN gfx("line",0,0,0,191)
059C        RUN gfx("line",255,0,255,191)
05B4
05B5        (* set yellow foreground for letters
05D9        (* if mode is '0,1' you get green letters on black background
0616        RUN gfx("mode",1,1)
0628
0629        WHILE tst DO
0632          READ a,b,c,d
0643
0644          IF a=999 THEN COTO 100
0654          ENDIF
0656
0657          RUN gfx("line",a,b,c,d)
0677        ENDWHILE
067B
067C 100    INPUT x$
0684        (* "quit" de-allocates the graphics memory
06AE        RUN gfx("quit")
06BA        END
06BC
06BD        (* data for the letters
06D4        DATA 4,160,4,188
06E4        DATA 4,160,12,174,12,174,18,160
0700        DATA 20,160,20,188
0710        DATA 24,160,24,188,24,160,36,160
072C        DATA 24,174,30,174,24,188,36,188
0748        DATA 42,160,54,160,42,160,42,188
0764        DATA 60,160,60,188,60,160,72,160
0780        DATA 60,188,72,188
0790        DATA 78,160,78,188,78,160,90,160
07AC        DATA 78,188,90,188,90,160,90,188
07C8        DATA 96,160,96,188,96,188,108,176
07E4        DATA 108,176,120,188,120,160,120,188
0800        DATA 126,160,126,188,126,160,138,160
081C        DATA 126,188,138,188,126,174,132,174
0838        DATA 150,188,162,188,156,160,156,188
0854        DATA 168,160,180,160,168,188,180,188
0870        DATA 168,160,168,188,180,160,180,188


PROCEDURE type
 0000       PARAM name:STRING[60]
 000C       DIM path:INTECER
 0013       DIM f:REAL
 001A       DIM char:STRING[1]
 0026       DIM tst:BOOLEAN
 002D
 002E       ON ERROR GOTO 100
 0034
 0035       tst=TRUE
 003B
 003C       OPEN #path,name:READ
 0048       f=0
 0050
 0051       WHILE tst DO
 005A         SEEK #path,f
 0064
 0065         IF EOF(#path) THEN PRINT
 0070           CLOSE #path
 0076           END
 0078         ENDIF
 007A
 007B         CET #path,char
 0085         f=f+1
 0091
 0092         IF char=CHR$(7) OR char=CHR$(10) OR char=CHR$(13) OR char

   >=CHR$(31) AND char<=CHR$(127) THEN PRINT char;
 00C4         ENDIF
 00C6
 00C7       ENDWHILE
 00CB
 00CC       CLOSE #path
 00D2       PRINT
 00D4
 00D5       REM if you want a character count, add the next line
 0108       PRINT "character count="; f
 0120 100   IF ERR=216 THEN PRINT name; " not found"
 013E       ENDIF
 0140       BYE
 0142
```

the same cables — they just sit there.

Here's the problem: Both computers are talking and both are listening, but they aren't talking to each other. Since you mentioned that both machines can talk to local bulletin boards through your modem, we know that the RS-232 ports on both of the Color Computers are working.

The answer: When you connect two computers together, you need to use a null modem cable — a cable that connects the transmit or output line of one to the receive or input line of the other. You can build one by reversing those two wires on the cable you're using with your modem. Or, if you would rather not attack the cable with a soldering iron, Bob Rosen at Spectrum Projects will sell you one.

Communications was also the topic of concern for John Kresin of Port Huron, Mich. He's in a local TRS-80 computer club where his Color Computer is outnumbered by Model IIIs and Model 4s. He really wants to find a bulletin board program for his CoCo. John, see if you can reach Saturn Electronics Company, 62 Commerce Drive, Farmingdale, NY 11735, (516) 249-3388. They advertised an "OS-9 BBS" for $89.95 last summer. If they are out of business, I suggest you put the question on the CompuServe OS-9 SIG as there were several threads discussing bulletin boards for the Color Computer and OS-9 last summer.

Finally, as we wrap up the file named */d/RAINBOW/KISS.June*, here's a note about another new product that hit the stands this month. Computerware is now shipping *Look and Listen* for OS-9. Inside, you'll find the high resolution screen that Brian Lantz developed for their stand-alone *Databank Manager*, a font editor to create characters for it, several sound commands, as well as a device driver and descriptor that lets you use Tandy's Speech/Sound cartridge.

The Speak command in this package is like the standard OS-9 Echo utility, except it sends its output to the Speech Cartridge, i.e., "Speak Hey turkey, you better not delete that file!" On the other hand "Talk" and "Talker," the device descriptor and driver, act just like any other OS-9 device.

For example, if you want your CoCo to read a listing of the files in your current data directory you need only type this command line:

```
OS-9:dir>/talk ENTER
```

How can the Fourth of July compete?

```
PROCEDURE cocodir
0000       REM
0003       REM programs by John E. Carter - WB4HLZ
0029       REM written as a learning exercise
004B       REM maybe they can be of use to others
0070       REM
0073       REM uses ideas from Mike Dziedzic's "put_dos" program,
00A8       REM which makes an OS9 disk bootable from RS dos under
00DD       REM Disk Basic 1.0 - check the CoCo SIG for the original program
011C       REM
011F
0120       BASE 1
0122       ON ERROR GOTO 1
0128
0129       DIM path:INTEGER
0130       DIM i,k:INTEGER
013B       DIM j,firstchar,file_type,ascii_flag,first_gran:BYTE
0152       DIM number_of_bytes:INTEGER
0159       DIM fat(64):BYTE
0165
0166       PRINT
0168       INPUT "drive = /",disk$
0179
017A 1     REM "dir" to put head on track 0
019C       SHELL "dir /+disk$"
01AB       OPEN #path,"/"+disk$+"@":UPDATE
01BF       REM reading FAT - for future use - to copy rs to os9
01F2
01F3       FOR k=0 TO 63
0203         SEEK #path,307.*256+k
021A         GET #path,fat(k+1)
0228       NEXT k
0236
0237       REM reading directory sectors
0253
0254       FOR k=256 TO 1279 STEP 32
0268         SEEK #path,307.*256+k
0282         GET #path,j
028C         firstchar=j
0294
0295         REM if firstchar=255 we're past the active directory
02C8       EXITIF firstchar=255 THEN
02D4       ENDEXIT
02D8
02D9         REM print the filename.ext
02F2
02F3         IF j>31 AND j<127 THEN
0306           PRINT CHR$(j);
030D
030E           FOR i=1 TO 10
031E             SEEK #path,307.*256+k+i
033A             GET #path,j
0344             PRINT CHR$(j);
034B
034C             IF i=7 THEN
0358               PRINT ".";
035E             ENDIF
0360
0361           NEXT i
036C
036D           REM get the file type - 0,1,2,3
038B           SEEK #path,307.*256+k+11
03A6           GET #path,file_type
03B0           PRINT " "; CHR$(file_type+48);
03BE
03BF           REM get the ascii flag
03D4           SEEK #path,307.*256+k+12
03EF           GET #path,ascii_flag
03F9
03FA           IF ascii_flag=255 THEN
0406             PRINT " A";
040D           ELSE PRINT " B";
0417           ENDIF
0419
041A           IF firstchar>31 AND firstchar<127 THEN
042D             PRINT
042F           ENDIF
0431
0432         ENDIF
0434
0435 10      REM
043B       NEXT k
0446       CLOSE #path
044C       PRINT
044E       END
0450
```

TEXTPRINT

by Tony Ceneviva

With the increasing number of people now migrating to disk and beginning to experiment with OS-9, the time appears right to publish this contribution from Tony Ceneviva. Tony is a member of the Perth Users' Group and this series has been appearing in CoCoPug. We thought it too good not to be shared with you all. It will be presented in five parts - one for each procedure. The first is the TEXTPRINT procedure included below. Please feel free to contact Tony if you have any questions.

This suite of programs allows the creation and maintenance of very large disk files containing defined mail list type fields. The size of the file is only limited by the capacity of the disk storage device.

It is designed for a task we had recently when we were given an electoral roll with 10000 names, addresses and electoral roll numbers arranged in alphabetical order. We had to provide individually addressed letters to members of each household advising them their electoral roll number and sorted in street and house number order so that they could be delivered by hand. We also sorted the streets into different suburbs so that residents could be given a different letter text for each of the suburbs.

The system can be used as a mailing list, mail sort, mail merge facility.

The sort routine will sort 1200 clients in 130 minutes, 200 clients take 10 minutes.

With the 128K expansion it is possible to arrange the sort as background task and with the additional RAM space the block group size can be set as high as 1000 by amending two lines of the source code and the sort time can be cut in half. This compares with the time taken by some machine code sort routines.

The sort facility will allow files to be split into different categories and transferred into many subsiduary files thus giving almost limitless capacity.

It's usefulness can be better understood by looking at each module of the system.

PREPARE THE DISK SYSTEM

The system disk should contain:-

(a) OS9 boot
(b) Edit command
(c) Tmode command
(d) Runb
(e) Pmaillistfileman
(f) Psort
(g) Pdatafileload
(h) Ptextprint
(i) Pprinttransfer
(j) Check [edit macro]

Files (a) to (d) can be copied from the system disk.

Files (e) to (i) are Basic09 packed procedures which are entered from Basic09 edit mode as per listing for procedures Maillistfileman, sort, Datafleload, Textprint, Printtransfer.

Each of these procedures will be presented in this and the next four installments, with instructions for running the system to accompany the final article. Each of the procedures is capable of being used independantly.

After entering the code for each procedure and test running it under Basic09, rename each procedure to save confusion by adding file (j) is an edit macro which is created as follows:-

1. Call the edit command
2. At the 'E' prompt, type .mac//
3. The 'M' prompt will display
4. Type 'space'checkmacro'enter'

```
        'space'v0 +2 s"$"(.str"$"(-2 s
          "$"(+)5L.str"$":-5(L+)5))*'enter'
        Q 'enter'
```

5. When back in edit mode save the macro on the system disk.

The purpose of the macro is to check to ensure that each block record of data entered from edit mode contains five lines.

The advantage of this system is that procedures can be written in Basic09 to perform specific tasks to suit individual requirements.

The textprint procedure for instance can be modified to print a letter to all or specified clients on the data file.

The text of the message or letter file can be prepared using the editor or any word processing package and saving on disk in ASCII format.

These instructions were prepared using Scripsit and transferred to OS9 formatted disk with Opack xcopy utility and the file was then printed by Ptextprint.

Here is the first procedure - a short one to start with:

```
PROCEDURE TEXTPRINT
 0000  REM BY A. CENIVIVA 30 HATFIELD WAY BOORAGOON
        W.A. 5154
 0035  DIM CHR(6000):STRING[1]
 0046  DIM TEXTFILE:STRING[20]
 0052  DIM PRINTPATH:STRING[20]
 005E  DIM PPA[H:BYTE
 0065  PRINTERPATH="/P"
 006E  DIM PATH:BYTE
 0075  PRINTCHR$(12)
 007A  INPUT"TEXT FILE TO BE PRINTED ",TEXTFILE
 009A  OPEN #PATH,TEXTFILE:READ
 00A6  OPEN #PPATH,PRINTERPATH:WRITE
 00B2  PRINTUSING "S25 ","PRINTING TEXT FILE"
 00D0  PRINTUSING "S25 ",TEXTFILE
 00DD  X=1
 00E5  SEEK #PATH,0
 00EE  LOOP
 00F0    GET #PATH,CHR(X)
 00FF  EXITIF EOF(#PATH)=TRUE THEN CHR(X)="%"
```

REVIEW

THE OS9 SOLUTION

by Brian Dougan

Did you ever feel somebody has gone to a lot of trouble to prove a point to you?? A certain editor of a popular computer magazine has been known to make the odd remark about my reservations on the OS9 operating system.

Having had a small exposure to the FLEX system, my first reaction (and second) was that OS9 was about as "user friendly" as a bear with a sore tooth. Whenever I expressed some minor doubt on how beginners would cope with it's rigid command structure, complicated directories, and frustrating "ERROR #XXX" messages, I was accused of being less than open minded.

These doubts obviously had stung the finer feelings of the author of OS8, and to show me the error of my ways he has gone to the extraordinary length of getting a program called "The OS9 Solution" released on the Australian market. Then to ensure I was aware how wrong I was, he asked me to review the program.

OK!! I may have to withdraw one or two of my previous remarks! (Maybe all but the one about the error messages!) The OS9 solution is well named, it DOES solve most of the problems and will allow the beginner to enter OS9 with much less hair tearing. The program instructions give a step by step explanation on how to install "The Solution" on your system disk with the added bonus that you can delete many of the existing files that it replaces thus reducing the number of commands you have to master.

Once up and running the screen display gives a list of all files on the current directory. Using the up and down arrow keys, you can step through the menu to select the file on which you wish to work. Then by pressing any of the single keys tabulated on the screen you can (L)oad, (C)opy (K)ill etc. the file very simply, no long involved pathnames or command structure to get 99% right.

This easy handling of all of the common file handling requirements should remove most of the complexity and all of the confusion for new (and many not so new OS9 users). Once this initial confusion is removed I believe the hidden pleasures of OS9 will more be easily discovered.

But don't get the wrong idea! This utility is not only for the novice, its quick and easy handling will be a boon to any user of this operating system who is not a touch typist trying to develop repetitive strain injury. It deserves serious consideration by all users.

SPECTRUM PROJECTS.
Released in Australia through Paris Radio Electronics.
(See Ad this issue.)

```
0117  ENDEXIT
011B  EXITIF CHR(X)="%"THEN ENDEXIT
012B    X=X+1
013B  ENDLOOP
013F  X=1
0147  LOOP
0149  EXITIF CHR(X)="%"THEN ENDEXIT
015D    PRINT #PPATH,CHR(X);
016C    X=X+1
0178   ENDLOOP
017C    CLOSE #PATH,#PPATH
```

NOW! is the time to subscribe to Australian RAINBOW

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

RATES

AUSTRALIAN CoCo / MiCo / softgold		AUSTRALIAN RAINBOW
□ $3 45	Latest per copy	$4.50 □
□ $19	6 months	$27.75 □
□ $31	12 months.	$39.95 □

BACK ISSUES

MiCo — First Issue Oct '83 □ $3/copy

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

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

DISKS & TAPES

Rainbow on Tape (programme listings) $12 for month of □
Please note that RAINBOW on TAPE is issued irregularly

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

Tape Monthlies

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

or Debit my Credit Card Monthly

Blank tapes 12 for $18 or $1.70 ea □

Cassette Cases 10 for $5 □

Disks — $3.50 ea □ 10 for $29.99 □

BOOKS

Byte
Elementary $5.95 □

Help
Medium $9.95 □

Facts
Advanced $11.95 □

MiCo Help
Medium $9.95 □

MiCo Exposed
SORRY
OUT OF STOCK □

BULLETIN BOARD

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

□□□□□□□□□□□□□□□

□ VISA BANKCARD □
MASTERCARD

________________ Signature

□ CASH
□ CHEQUE
□ MONEY ORDER

Authorised:
Amount $ ________________

BLOCK CAPITALS PLEASE

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

□□□□□□□

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

FIRST NAME SECOND NAME

Address

PC

Telephone Number STD Code Local Number

Subscription □ Renewal □

COMPUTERWARE FOR MICROS.

bankcard welcome here

VISA

PO Box 104,
MAGILL, SA, 5072
Ph (08) 336 6588

BACK IT UP!

$59.95
(Supplied on Disk)

SUPER BACK-UP UTILITY©

SUPER BACK-UP UTILITY WILL PERFORM ALL OF THE FOLLOWING FUNCTIONS

1 **TAPE TO TAPE** (Regardless of most protection schemes)
2 **TAPE TO DISK** (Move Cassette programs to Disk)
3 **AUTO RELOCATE** (For those Cassette programs that conflict WITH Disk operating systems)
4 DISK TO TAPE (Place Disk programs onto Cassette)
5 DISK TO DiSK (Our powerful Spit-N-Image Program. Regardless of protection schèmes)

- **MENU DRIVEN!**
- **REQUIRES 32K EXTENDED COCO**
- **REQUIRES 1 OR 2 DRIVES** (For Disk Functions)
- **ALL MACHINE LANGUAGE!!!**

COMPARE WITH OTHER INDIVIDUAL PROGRAMS COSTING IN EXCESS OF $100.00 OR MORE!!!

COLORPACK $39.95

ROM/RAM pack (specify configuration)

COLOR QUAVER $29.95

Software Music Synthesizer on tape (requires 32-64K)

COLOR BURNER $99.95

EPROM Programmer (2716/32/32A/64/64A/128, 68764/66) with software

LOWERKIT III $89.95

- Full-time upper and lowercase installs in 15 minutes
- Normal and reverse video standard
- Fully compatible with all Alpha and Graphic modes
- Assembled and tested

Important! Specify Color Computer or Color Computer II

The amazing...
...CoCo MAX

CoCo MAX has these powerful features:
Mirror images, rubber banding, edge tracing, zoom, lasso, sixteen colours, thirty patterns, thirty-two paint brush shapes and textures, undo, rubber stamping, icons pull down menus, pencil, spray can, eraser toolbox and so on and so on...

CoCo MAX is simply the most incredible graphic and text creation 'system' you've ever seen. You will generating images in minutes.

$149.95

TAPE VERSION NOW AVAILABLE!!!

DRIVE O PACKAGES

$649
$7 shpg

More storage, Less cost!

.. **Our single-sided disk package gives 23,040 bytes more for a dollar less!**

179, 712 Bytes DD-1 SSDD Drive
DC-1 40 Track Controller
CA-1 Cable

GRAPHICOM PART II Introductory pack $39.95

Graphicom Part II is a video processing package that provides many functions that are missing in Graphicom. Here are just a few of the features provided by Graphicom Part II:

ENLARGE/REDUCE/ROTATE

Enlarge or reduce any portion of a screen by any amount, just like a photographic enlarger! Independent of the enlargement or reduction, rotate by any degree or fraction of a degree about any point on the screen.

PAN & ZOOM

"Zoom in" x2, x4, or x8 on any portion of the screen to do fine pixel work. Allows editing of Graphicom character sets with ease!

TYPESETTER & FONT EDITOR

Add text in 16 different sizes with several display modes to choose from including COLORED FOREGROUND & BACKGROUND text! Edit 8x8 characters for use in the typesetter. Over 30 character sets supplied on disk. "GRAB" function allows transfer of some Graphicom character sets to Graphicom Part II format.

PIXEL BLASTER

Allows the user to easily substitute or remove colors. Widen lines, swap BLUE & RED without effecting BLACK & WHITE, etc.

GRAPHICOM PART II DOES NOT REQUIRE GRAPHICOM TO RUN!

Graphicom Part II requires a 64K extended disk basic system, it will load and save both standard BIN files and Graphicom screens, and supports 1 to 4 disk drives with keyboard or joystick (analog or switch type). All functions support color or Hi-Res operation, as well as 4 screen display modes.

Green Mountain Learning Lab

Not everyone can program. Writing a few lines in Basic now and then doesn't make you a programmer. But if you can program, then our MICRO LANGUAGE LAB will teach you — the right stuff, the right way.

We'll teach you in 24 half-hour lessons on 12 audio cassettes, with a 220-page textbook, with data booklets, with 35 sample programs, and with a programming reference card. You'll spend 50 hours or more with our course, listening, watching and working. And when you're done, you'll be programming your Color Computer in the 6809's language.

$179.95

Nrth Qld Colour Software, 9 Durham Court, KIRWAN, TOWNSVILLE, QLD. Phone (077) 73-2064
Rainbow Valley Comuters, RMB 6680, MAFFRA, VIC. Phone (051) 454-315
Blaxland Computer Service, P.O. BOX 2774, BLAXLAND, N.S.W. Phone (047) 39-3903
Paris Radio Electronics, 161 Bunnerong Road, KINGSFORD, N.S.W. Phone (02) 344-9111
Geoff Tolputt, P.O. Box 140, WOOLOONGABBA, QLD. Phone (07) 446084.
Crystal Blade Software, P.O. Box 256, ROSEVILLE, N.S.W. Phone (02) 467-1619
The Computer Hut, 21 William Street, BOWEN, QLD. Phone (077) 86-2220

user group CONTACTS

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

			GRAFTON	DAVID HULME	066.42.0627	ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
			GREENACRES	BETTY LITTLE	08 261 4083	ROSEVILLE	KEN UZZELL	02 467 1619
ADELAIDE	JOHN HAINES	08 278 3560	HASTINGS	MICHEAL MONK	059.79.1513	SALE	BRYAN McHUGH	051 44 4792
ADELAIDE NTH	STVN EISENBERG	08 250 6214	HERVEY BAY	LESLEY HORWOOD	071 22 4989	SANDGATE	MARK MIGHELL	07 269 5090
ALBURY	RON DUNCAN	060 43 1031	HILLS DIST	DENNIS CONROY	02 671 4065	SEACOMBE HTS	GLENN DAVIS	08 296 7477
ARMIDALE	TOM STUART	067 72 8162	HOBART	BOB DELBOURGO	002 25 3896	SMYTHESDALE	TONY PATTERSON	053 42 8815
BAIRNSDALE	COLIN LEHMANN	051 57 1545	IPSWICH	MILTON ROWE	07 281 4059	SPRINGWOOD	DAVID SEAMONS	047 51 2107
BALLARAT	MARK BEVELANDER	053 32 6733	JUNEE	PAUL MALONEY	069 24 1860	STURT	MARY DAVIS	08 296 7477
BANKSTOWN	KEN HAYWARD	02 759 2227	KALGOOLIE	TERRY BURNETT	090-21-5212	SUNBURY	JACK SMIT	03.744.1355
BLACKTOWN	KEITH GALLAGHER	02-627-4627	KENMORE	GRAHAM BUTCHER	07 376 3400	SUTHERLAND	IAN ANNABEL	02 528 3391
BLACKWATER	ANNIE MEIJER	079.82.6931	LEETON	CHRIS NAGLE	069 53 2969	SWAN HILL	BARRIE GERRAND	050.32.2838
BLAXLAND	BRUCE SULLIVAN	047 39 3903	LITHGOW	STUART RAYNER	063 51 4214	SYDNEY EAST	BOB JONES	02-331-4621
BOWEN	TONY EVANS	077 86 2220	LIVERPOOL	LEONIE DUGGAN	02-607-3791	SYDNEY TEENS	ROD HOSKINSON	02 48 5948
BRASSALL	BOB UNSWORTH	07 201 8659	MACKAY	LEN MALONEY	079511333x782	TAMWORTH	ROBERT WEBB	067 65 7256
BRIGHTON	GLENN DAVIES	08 296 7477	MACLEOD	ROBIN ZIUKELIS	03 450211x465	TAHMOOR	GARY SYLVESTER	046 81 9318
BRISBANE EAST	ROB THOMPSON	07 848 5512	MacQUARIEFIELDS	KIETH ROACH	02 618 2858	TONGALLA	TONY HILLIS	058 59 2251
BRISBANE SH	PATRICK SIMONIS	07 209 3177	MAFFRA	MAX HUCKERBY	051 45 4315	TOOWOOMBA		
BRISBANE SW	GRAHAM BUTCHER	07 376 3400	MAITLAND	LYN DAWSON	049 49 8144	" BEGIN NTH	DAVID PROUT	076.32.7533
BRISBANE WEST	BRIAN DOUGAN	07 30 2072	MARYBOROUGH	NORM WINN	071 21 6638	" BEGIN STH	LEW GERSEKOWSKI	076 35 8264
BUNDABERG	JIM McPHERSON	071 72 8329	MELBOURNE	JEFF SHEEN	03 528 3724	" ADVANCED	GRAHAM BURGESS	076 30 4254
CAMBERWELL	TONY BALDWIN	03 728 3676	MELTON	MARIO GERADA	03 743 1323	TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064
CAMPBELLTOWN	LEO GINLEY	02 605 4572	MILDURA	SCOTT HEWISON	050 23 6016	TRARALGON	MORRIS GRADY	051 66 1331
CANBERRA	SHAUN WILSON	062 51 2339	MOE	STEPHEN SEMPLE	051 27 6841	UPPER HUNTER	TERRY GRAVOLIN	065 45 1698
CAULFIELD	JEFF SHEEN	03 528 3724	MORPHETTVALE	KEN RICHARDS	08 384 4503	WAGGA WAGGA	BRUCE KING	069 25 3091
CHATSWOOD	BILL O'DONNELL	02 411 3336	MOREE	ALF BATE	067 52 2465	WESTLEIGH	ATHALIE SMART	02 848 8830
CHURCHILL	GEOFF SPOWART	051 22 1389	MORWELL	GEORGE FRANCIS	051 34 5175	WHYALLA NORRIE	CHRIS HUNTER	086 45 3395
COLYTON TEENS	DWAYNE MANSON	02 623 5805	MT ISA	PAUL BOUCKLEY-SIMONS	077 43 6280	WOLLONGONG	BRIAN McCAULEY	042 71 4265
COOMA	ROSS PRATT	0648 23 065	MUDGEE	BRIAN STONE	063-72-1958	WONTHAGGI	PAT KERMODE	056 74 4583
DANDENONG	DAVID HORROCKS	03 793 5157	MURGON	PETER ANGEL	071 68 1628			
DARWIN	BRENTON PRIOR	089.81.7766	NAMBUCCA HDS	WENDY PETERSON	065 68 6723	SPECIAL INTEREST GROUPS		
DENILIQUIN	WAYNE PATTERSON	058 81 3014	NEWCASTLE	LYN DAWSON	049 49 8144	BRIZBIZ	BRIAN BERE-STREETER	07 349 4696
DUBBO	GRAEME CLARKE	068 89 2095	NOARLUNGA	ROBBIE DALZELL	08 386 1647	BRISBANE OS9	JACK FRICKER	07 262 8869
EMERALD	LEIGH EAMES	059 68 3392	NOWRA	ROY LOPEZ	044 48 7031	CARLISLE MiCo	STUART HALL	08 361 1922
FORSTER	GARY BAILEY	065 54 5029	PARKES	DAVID SMALL	068 62 2682	MONARO OS9	FRED BISSELING	0648 23263
FRANKSTON	BOB HAYTER	03.783.9748	PENRITH	ALEX SCHOFIELD	047-31-5303	BLAXLAND OS9	BOB THOMSON	047 30 2468
GIPPSLAND STH	PAT KERMODE	056 74 4583	PERTH	IAN MACLEOD	09 448 2136	BLAXLAND 128K	BOB THOMSON	047 30 2468
GLADSTONE	ALBERT VAN GORKUM	079 72 2353	PORT MacQUARIE	RON LALOR	065 83 8223	ROCKHAMPTON MiCo	TIM SHANK	079 28 1846
GOLD COAST	SHERYL BENTICK	075-39-2003	PORT NOARLUNGA	ROB DALZELL	08 386 1647	SYDNEY MiCo	RAJA VIJAY	02 519 4106
GOSFORD	PETER SEIFERT	043 32 7874	PORT PIRIE	KEVIN GOWAN	086 32 1368	SYDNEY MS DOS	ROGER	047-39-3903
GOULBURN VALLEY	TONY HILLIS	058 59 2251	RINGWOOD	ANDREW RAWLINGS	03 726 6521	BRISBANE MS DOS	BRIAN DOUGAN	07 30 2072

AUSTRALIAN CoCo THIS MONTH

- * SABRE by Andrew Simpson winner of this year's GAMES competion
- * WORD PUZZLE by Keith Wray
- * QWERL by Darrell Berry
- * ALIEN by Stuart Sanders
- * MOVE ABOUT by Kevin Gowan runner up in the GAMES competion
- * LABYRINTH by James Redmond runner up in the GAMES competion

PLUS OS8, and lots of articles that will be of interest to you!

CoCoOz tapes are produced monthly and contain all the programs from CoCo magazine, already typed up and ready for you to RUN!

Australian Rainbow Magazine
Registered by Australia Post
Publication No. QBG 4009.

PO Box 1742,
Southport. QLD. 4215.