For your TANDY Computer

$3.25

AUSTRALIAN RAINBOW

INCORPORATING GOCO

November, 1984

No.41

AUSTRALIAN RAINBOW INCORPORATING GOCO

Dedicated to the Memory of Greg Wilson

AUSTRALIAN EDITOR AND PUBLISHER

Graham Morphett

CO-EDITOR

Kevin Mischewski

EDITOR'S ASSISTANT

Christine Lucas

AND GRATEFUL ASSISTANCE FROM

Brian Dougan
Peggy Annabel
Richard and Judy
Rod Hoskinson
Helga Wilson
Patric Simonis
Annette Morphett
Glen Mischewski
Jim & Sheryl Bentick

COVER DESIGN

Jim Bentick

All Programs in this issue of RAINBOW are available on cassette tape

DEADLINES.

Dec/Jan	7th Nov, 1984.
Feb	7th Jan, 1984.
Mar	7th Feb, 1984.
Apr	7th Mar, 1984.
May	7th Apr, 1984.
Jun	7th may, 1984.

OS-9

The OS-9 Users' Group is set up as a contact by mail type group. They have also recently set up a Bulletin Board, the number of which can be found on the back page of this magazine. Enquiries should be directed to:

Kevin Holmes
39 PEARSON ST.,
NARARA, N.S.W. 2250

Printed by:
Australian Rainbow Magazine
P.O. Box 1742
Southport Qld. 4215

Registered Publication
No QBG4009

PRINT #-2,

The more I talk to you on the phone, the more I realize just what a diverse range of talents are held by the subscribers to this magazine.

There's the Delbourgos in Hobart - ask them about the ROMs of CoCo and they'll tell you how to re-write them. In passing, they'll discuss the Physics of Turbulence with you.

There's Lotars Ginters, in Melbourne, - he turns out designs and provides original thinking par excellence.

There's Wim de Puit in Devonport. Wim is a communicator who can make complex subjects sound dead easy.

There's Dean Hodgson, now married and living in Adelaide. Dean writes in several computer languages with a number of computer brands, mainly on educational software. He is also an original thinker who challanges my senses everytime I talk to him.

There's Iain, John and the boys in Perth - in many ways showing the way for the rest of us - first with a BBS, first with their own modems - first with a strong, regular newsletter.

There's John Brothers in Adelaide - working at soft and hardware distribution. Walking the thin line between providing new and interesting products and trying to keep eating whilst he does it.

There's John Poxon in Brisbane. He lectures at South Brisbane Tech in Physics. His major interests lie in the Forth Language, and the interfacing of implements to computers.

There's Roy Lopez in Nowra, blending retirement with Computers and Amateur Radio to create new methods of communication.

INDEX

RAINBOW

REVIEWS .... P 3
TAROT .... P 5
THE EDUCATION PAGE .... P 15
THE COCO SCHOOL MARM .... P 20
LO RES GRAPHICS .... P 23
CREATE A CALENDAR .... P 28
EVERYTHING YOU ALWAYS WANTED TO KNOW P 30
RAINBOW WARE .... P 32
BLOW UP .... P 37
THE ART OF JOYSTICK PAINTING ..... P 39
PRESERVING THE CLASSICS .... P 43
COOKING WITH COCO .... P 45
MARTHA SAYS .... P 53

GOCO .... P 54
TELECOMMUNICATING WITH POCO ...... P 53
MAKING PIES .... P 53
A SUMMING UP .... P 55
THE POCO AS A SOCIAL
SCIENCE RESEARCH TOOL .... P 57
THE 96K MODEL 100 .... P 57
YOU CAN "PLAY IT" ON YOUR POCO ... P 58
LOGGING ACCOUNT TIME WITH TIME CLOCK P 59
MASTERING MS-DOS .... P 61
STOCKING UP ON REINVESTED DIVIDENDS P 61
DICE TOTAL PROBABILITIES .... P 63

And then there is a bulk of you like Tom Lehane in Penrith, Bruce King in Wagga, Paul Maloney in Junee and many others (whom I hope will forgive me for not mentioning their names because they too do important work) who are quietly getting on with the job of teaching in particular, the youth of this country, about computers.

Around the country there are now many clubs involved in a number of exciting projects. That is really good news, because it means that you are using your knowledge to assist others, and when you do that you learn more yourself.

At Rainbow, we want to encourage you and your group to develop a group project. This is a good way of putting your collective skills to good use.

Experience shows that when you share your knowledge, you grow faster. Whatever you are using CoCo for, why not drop us a line and tell us about it. What you are doing may appear ordinary to you, but to others, what you have to share, may open an entirely new field of enquiry.

Following my recent ramblings re Bankcard, several have suggested that maybe I might like to try American Express. To this I say an emphatic "NO"! They are worse than Bankcard! We are, however, looking at opening Mastercard facilities.

This issue looks at screen graphics and provides some very interesting ideas.

The work being done by folk such as the Delbourgos and Shaun Coyne (see last month's Aust CoCo) in this country, underlines the fact that we can keep up with the rest of the world, if not surpass them in the area of graphics programing!

I would like to thank the Benticks for their review of 'Graphicom' this month. Graphicom is a very clever program, with a great deal of detailed learning to be done before one can expect to obtain good results. Jim & Sheryl, as you are no doubt aware, look after many of the jobs around here that relate to advertising, and also have contributed the cover designs since we started. This keeps them pretty busy, so to also fit in 'Graphicom' over the last two months, has been something of a burden to them, but the results of their labors, both on the cover this month, (as these are screen dumps produced with 'Graphicom'), and in text are appreciated.

We welcome Martha Gritwhistle to these pages. She is really a very person. In fact she has been a little ill and that's why we haven't heard from her lately. In fact, she saw the doctor who told her to take two Pal Meaty Bites and see him next week!

Martha really isn't all that bad, its just a bit embarrassing when we go out in the car - she wants to hang her head out the window all the time! Ah well, if she doesn't look much at least she can catch a Frisbee in her mouth at 200 yards!

REVIEWS

SOFTWARE

GRAPHICOM

GRAPHICOM comes from Cheshire Cat Computer Creations in the US and take it from me if you are into any form of graphics this is going to revolutionize the way you look at your CoCo from now on.

How would you like a disk package that allows you to draw, using a unique 'rubberband technique', any object your heart desires, make a stamp of it mirror it, rotate it, see what it looks like in the 4 modes, then when you feel OK about it stamp it all over the screen, go back to the Main Menu, select one of 4 stamp functions - Opaque, Mask, Clear or Reverse, return to your original drawing and begin stamping again.

When you have finished your masterpiece you can save it to disk and/or screen dump it.

GRAPHICOM gives a choice of printers for screen dumping, Epsom, C-Itoh, Gemini and Okidata also 4 Tandy Dot Matrix printers and Tandy's CGP 115.

Let's return to our Main Menu.

Sixteen icons are presented, apart from the ones mentioned so far there is also Mirror Stamp, Color, Select-Animate, Copy-Format and Send and Receive which is for telecommunications.

The Disk Page Menu.

There are 4 workspaces. These areas are in the computer's memory not the disk, in which you can edit pictures. Workspace 4 is also used for stamp sets which is another article on its own.

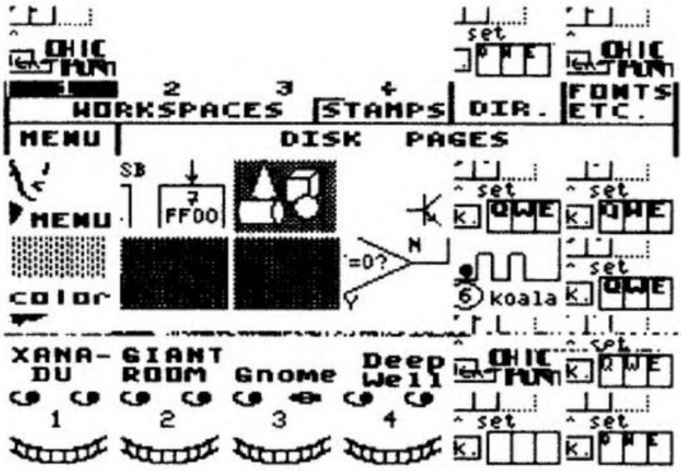
Dir., the 5th area allotted, is where the disk directory is stored.

Fonts etc, the last area on the top row is for the character font and selection of printer and modem.

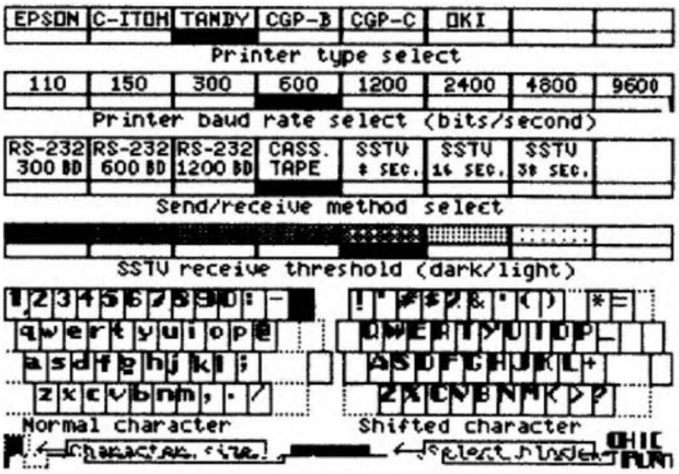
A selection of five type fonts to choose from and the opportunity to redefine the characters produced. Set up a Dvorak Keyboard or create a Hebrew typewriter or Keep a different set for mathmatical and scientific notations.

The rest of the areas are for Disk Pages and are used for your own pictures.

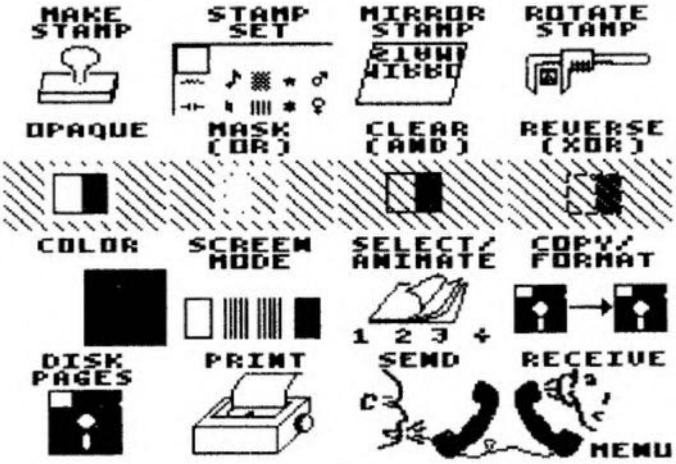
I could go on and on, I've not mentioned the Koala Pad option, Animate which permits animation at the rate of 60



Disk Page Menu.



Five fonts to choose from.



The Main Menu - 16 icons to choose from.

frames per second, or SUXPIX which allows you to access the screen dump graphic pages from your favourite programs, or a host of other features.

GRAPHICOM requires 64k Extended Basic, One Disk Drive System, a pair of joysticks (the keyboard is only used for entering text in the Simple Draw Mode) and of course a TV or monitor.

To enhance and expand GRAPHICOM you might try a printer, a modem, a cassette recorder and/or 'cassette modem' and as the instruction booklet mentions an Amateur Radio Station.

All that can be said is that GRAPHICOM is CoCocolossal.

****

MAYBE U2 OR U4

• *I have a D-board 32K (piggyback) Color Computer with single disk drive. I still have the old BASIC 1.0 version. I have two problems:*

1) My computer quit generating sound. I can't save any programs to cassette (works fine with the disk drive, though). Any "sound" commands or "play" commands do not generate anything. My question is: I'm kind of a do-it-yourself-type of guy and I was wondering if you can tell me which part of the computer is kaput. I can just go ahead and buy the part and put it in myself.

2) My other problem is with Radio Shack's BWDUMP. *I have a feeling I need an eight-bit driver (if it's not built-in) because half the time this program will not work on my computer. And if it does work, the picture is off-center.*

Is there any kind of patch that will fix this? Is there an eight-bit driver in the program?

Noel P. Tomas
Virginia Beach, VA

I like your terminology, Noel. "Kaput" would not be a very appropriate name for a computer, but it's very catchy. That would be a good name for a magazine column like this one.

If you will obtain a copy of the TRS-80 *Color Computer Technical Reference Manual* (Cat. No. 26-3193) Page 66 has a schematic of the cassette circuits that are kaput. I would check your RS-232 port. If it is working, or if your printer is working, try replacing U2, a MC14050B. If your RS-232 port is not working, replace U4, a 68231 PIA chip.

Any graphic information sent to a printer requires an eight-bit printer driver. At one time, Radio Shack offered a patch for your problem. I believe your local rep can order it for you. Ask him for the "eight-bit driver routine" for the 1.0 BASIC ROM. Your best bet is upgrading to the 1.1. or 1.2 BASIC ROM.

Tandy's New Baby

• *Will software written to use printers such as Epson, Gemini, Okidata and the Radio Shack Line Printer VII and DMP-100 work on the new DMP-110?*

Marcus Boyd
Leander, TX

The DMP-110 uses Tandy's standard printer codes, so the features it has are accessed in the same way as on all their dot-matrix printers after the Line Printer VIII (except the DMP-100). The bit-image graphics are the same as on the LP VIII and DMP-100, but enlarged characters are accessed by *CHR$ (27) CHR$(14)* instead of just *CHR$(14)* and canceled by *CHR$(27) CHR$(15)* instead of *CHR$(15)*. The LP VIII and DMP-100 will respond normally to the new codes, though. Programs written to use special features on Epson, Gemini or Okidata printers generally won't work with Radio Shack units

GAME 32K ECB

TAROT

Amir Dimitri

Tarot is a form of magic to predict the future. It is basically a set of playing cards with special pictures for fortune telling. Tarot is surrounded with mystery and legend. The precise origin of these ancient cards is not known with certainty. It is widely believed that these cards originated in Europe, however, there are indications that the earliest use of tarot was in India, China and possibly Egypt in the form of clay tablets. Museums and libraries today possess original tarot decks dating from the 14th century.

The word "tarot" is a French adaptation of "tarocco," a game played in Italy during the 14th century with these ancient cards.

The tarot deck has 78 cards. Fifty-six cards are known as the Lesser Arcana. These are divided into four suits: Spades, Clubs, Hearts and Diamonds with King, Queen, Cavalier and Page. The remaining 22 cards are known as Trump or the Major Arcana cards. These are numbered XXI to I plus an unnumbered card known as "The Fool." The ordinary pack of playing cards today is a direct descendant of the 14th century tarot deck; the Trump cards were dropped, the Cavalier and Page cards were combined into today's jack, and "The Fool" became the joker.

Could fortunes be revealed with tarot cards? In 14th century Italy, a daughter amused her noble family with hand-painted tarot cards; in 16th century Germany a scholar delved into the hidden meaning of the tarot cards; gypsies wandered throughout Europe for centuries interpreting tarot to eager questioners; in the courts of France, cartomancers and diviners foretold catastrophic events to Napoleon.

It is said that some readings are so accurate as to defy rationale, other readings are so inaccurate as to offer little insight to the questioner. Is this due to some ancient wisdom or pure fantasy? Regardless, tarot has enjoyed a history of more than 500 years and is the forerunner of today's modern pack of cards.

The procedure used in spreading the tarot cards requires the questioner (person seeking an answer to a question) to shuffle the cards face down while stating out loud his specific question to the reader (diviner). The reader then lays out the cards in a prescribed sequence and interprets their symbolic meaning. It is found that for fortune telling, the 22 Major Arcana cards suffice and the Lesser Arcana cards could be avoided for simplicity. In this method, the reader lays down the top 10 cards of the shuffled 22-card Major Arcana deck.

Each Major Arcana card has a descriptive title and a symbolic picture for interpretation. These cards represent the physical and spiritual forces that influence people, namely; Strength, Power, Storms, Death and Religion.

Tarot cards cannot be read from opposite directions as conventional cards. If the card is laid down in reverse (upside down), then the interpretation is weakened or reversed. The presence of one card next to the other strengthens or weakens the interpretation. Readings are based not only on the specific interpretation of each card, but the relative proximity in which the cards fall, their frequency and whether a card is upside down (reversed).

The author of this program has merely mechanized this procedure for the enjoyment of the user. Effort has been spent to faithfully represent each of the 22 Major Arcana cards graphically, together with their respective interpretations in text and provide a comprehensive reading based on the methodology stipulated above. The user can now question the computer and find out whether it can really foretell the future with the tarot cards!

After unplugging the disk controller, loading *Tarot* and *RUNning* it, the user is prompted: "WHAT IS YOUR QUESTION?" Upon entering the question, the following prompt appears: "THE CARDS HAVE BEEN SHUFFLED — DO YOU WISH A RE-SHUFFLE (Y OR N)?" Each time the user presses ENTER, a graphic display of each of the 10 top cards is shown in sequence together with the associated text interpretation based on order, symbolic meaning and position (reversed). After the 10th card is displayed, a reading in text format is automatically displayed for the benefit of the user together with an answer to the question posed. Each display is accompanied with a tune to alert the user. Finally, the user is prompted for another reading if desired. Upon termination, a tune is played together with a closing statement for posterity.

This is how the program works: Setup takes place in Lines 10-1410 with calls to subroutines to run the card shuffling randomizing (1670), card display (1730), reversed card logo, etc. Tarot reading routines take place from Lines 1450-1700. Lines 1730-6110 perform the Extended BASIC graphics subroutines for the Major Arcana 22-card deck. The arrays dimensioned in Line 220 hold the following:

C	=	Card number
I$	=	Card interpretation
N	=	Card weighting
R$	=	Reversed card interpretation
R	=	Reversed card weighting
N$	=	Card order interpretation
D$	=	Display interpretation (summary)
D	=	Display weighting

Change the statements and graphics and good fortune shall follow you for the rest of your days!

170	179	2550	161	4460	103
400	243	2760	23	4660	143
670	216	2940	206	4840	120
970	33	3110	101	5010	45
1170	40	3300	169	5300	82
1420	44	3530	9	5440	85
1640	98	3700	150	5620	149
1850	122	3940	44	5760	138
2050	167	4120	234	5890	138
2290	137	4270	127	6050	37
				END	222

The listing:

```
10 CLEAR500
20 A3$="T4;O3;L2;C;L4;O2;B;L8;A;
L2.;G;"
30 LA$="C8;U10;R6;D5;L6;R6;D5;C5
;R5"
```

```
40 LB$="C8;U1Ø;R6;D5;L6;D5;R6;U5
;D5;C5;R5"
5Ø LC$="C8;U1Ø;R5;L5;D1Ø;R5;C5;R
5"
6Ø LD$="C8;U1Ø;R4;F2;D6;G2;L4;R4
;C5;R7"
7Ø LE$="C8;U1Ø;R5;L5;D5;R5;L5;D5
;R5;C5;R5"
8Ø LF$="C8;U1Ø;R5;L5;D5;R5;L5;D5
;C5;R1Ø"
9Ø LG$="C8;U1Ø;R5;L5;D1Ø;R5;U5;L
2;R2;D5;C5;R5"
1ØØ LH$="C8;U1Ø;D5;R5;U5;D1Ø;C5;
R5"
11Ø LI$="C5;R3;C8;U1Ø;D1Ø;C5;R7"
12Ø LJ$="C8;U2;D2;R3;U1Ø;D1Ø;C5;
R7"
13Ø LL$="C8;U1Ø;D1Ø;R5;C5;R5"
14Ø LM$="C8;U1Ø;R4;D5;U5;R4;D1Ø;
C5;R5"
15Ø LN$="C8;U1Ø;F5;U5;D1Ø;C5;R5"
16Ø LO$="C8;U1Ø;R6;D1Ø;L6;R6;C5;
R5"
17Ø LP$="C8;U1Ø;R6;D5;L6;R6;C5;D
5;R5"
18Ø LR$="C8;U1Ø;R7;D5;L7;R4;D2;F
3;D1;C5;R5"
19Ø LS$="C8;R5;U5;L5;U5;R5;L5;D5
;R5;D5;C5;R5"
2ØØ LT$="C8;C5;R3;C8;U1Ø;L3;R6;L
3;D1Ø;C5;R8"
21Ø LU$="C8;U1Ø;D1Ø;R5;U1Ø;D1Ø;C
5;R5"
22Ø LV$="C5;R4;C8;H4;U6;D6;F4;E4
;U6;D6;G4;C5;R9"
23Ø LX$="C8;U1;E8;U1;D1;G4;H4;U1
;D1;F8;D1;C5;R5"
24Ø SP$="C5;R5"
25Ø DIM C(22)
26Ø DIM I$(22)
27Ø DIM N(22)
28Ø DIM R$(22)
29Ø DIM R(22)
3ØØ DIM N$(1Ø)
31Ø DIM D$(1Ø)
32Ø DIM D(1Ø)
33Ø N22=1:R22=-1
34Ø N$(1)="***PRESENT POSITION**
*"
35Ø N$(2)="***IMMEDIATE INVOLVEM
ENT***"
36Ø N$(3)="***GOAL OR DESTINY***
"
37Ø N$(4)="***DISTANT PAST***"
38Ø N$(5)="***RECENT PAST***"
39Ø N$(6)="***FUTURE INFLUENCE**
*"
4ØØ N$(7)="***THE QUESTIONER***"
41Ø N$(8)="***ENVIRONMENTAL FACT
ORS***"
42Ø N$(9)="***INNER EMOTIONS***"
43Ø N$(1Ø)="***RESULT***"
44Ø I$(1)="THOUGHTLESSNESS, EXTR
AVAGANCE"
45Ø R$(1)="APATHY, NEGLIGENCE"
46Ø N1=-1 :R1=-1
47Ø I$(2)="SKILL, CREATIVITY"
48Ø R$(2)="INSECURITY, DELAY"
49Ø N2=1 : R2=-1
5ØØ I$(3)="WISDOM, SERENITY"
51Ø I$(17)="MISERY, DECEPTION"
52Ø R$(3)="CONCEIT, SELFISHNESS"
53Ø N3=1 :R3=-1
54Ø I$(4)="ACTION, PROGRESS"
55Ø R$(4)="INDECISION, ANXIETY"
56Ø N4=1 :R4=-1
57Ø I$(5)="AUTHORITY, WEALTH"
58Ø R$(5)="FEEBLENESS"
59Ø N5=1 :R5=-1
6ØØ I$(6)="KINDNESS, HUMILITY"
61Ø R$(6)="SUSCEPTIBILITY, INSEN
SITIVITY"
62Ø N6=1:R6=-1
63Ø I$(7)="LOVE, BEAUTY"
64Ø R$(7)="INRELIABILITY, FICKLE
NESS"
65Ø N7=1:R7=-1
66Ø I$(8)="CHALLENGE, TRIUMPH"
67Ø R$(8)="DEFEAT, RESIGNATION"
68Ø N8=1:R8=-1
69Ø I$(9)="VIRTUE, HONOR"
7ØØ R$(9)="ABUSE, INTOLEREANCE"
71Ø N9=1:R9=-1
72Ø I$(1Ø)="PRUDENCE, CAUTION"
73Ø R$(1Ø)="RASHNESS"
74Ø N1Ø=1:R1Ø=-1
75Ø I$(11)="FORTUNE, LUCK"
76Ø R$(11)="FAILURE, INTERRUPTIO
N"
77Ø N11=1:R11=-1
78Ø I$(12)="COURAGE, ENERGY"
79Ø R$(12)="WEAKNESS, TYRANNY"
8ØØ N12=1:R12=-1
81Ø I$(13)="READJUSTMENT, TRANSI
TION"
82Ø R$(13)="EGOISM"
83Ø N13=1:R13=-1
84Ø I$(14)="LOSS, FAILURE"
85Ø R$(14)="RECOVERY"
86Ø N14=-1:R14=-1
87Ø I$(15)="PATIENCE, MODERATION
"
88Ø R$(15)="DISCORD, HOSTILITY"
89Ø N15=1:R15=-1
9ØØ I$(16)="VIOLENCE, DISASTER"
91Ø R$(16)="FREEDOM, ENLIGHTENME
NT"
92Ø N16=-1:R16=1
93Ø R$(17)="ENTRAPMENT, OPPRESSI
ON"
```

```
94Ø N17=-1:R17=-1
95Ø I$(18)="HOPE, SATISFACTION"
96Ø R$(18)="PESSIMISM, DISAPPOIN
TMENT"
97Ø N18=1:R18=-1
98Ø I$(19)="CAUTION, DECEPTION"
99Ø R$(19)="DECEPTIONS, MISTAKES
"
1ØØØ N19=-1:R19=1
1Ø1Ø I$(2Ø)="HAPPINESS, SUCCESS"
1Ø2Ø R$(2Ø)="UNHAPPINESS, LONELI
NESS"
1Ø3Ø N2Ø=1:R2Ø=-1
1Ø4Ø I$(21)="DEVELOPMENT, PROMOT
ION"
1Ø5Ø R$(21)="DELAY, DISILLUSION"
1Ø6Ø N21=1:R21=-1
1Ø7Ø I$(22)="PERFECTION, SUCCESS
"
1Ø8Ø R$(22)="IMPERFECTION"
1Ø9Ø Y=Ø
11ØØ GOSUB612Ø:PRINT@132,"BY AMI
R DIMITRI":'3352 BREARD, BROSSAR
D J4Z 2E2 QUEBEC-CANADA 1981
111Ø FORTM=1TO5ØØ:NEXTTM:PRINT@1
32,"";:INPUT"WHAT IS YOUR QUESTI
ON";Q$
112Ø IFLEN(Q$)>25THEN166Ø
113Ø FOR J1=1 TO 1Ø
114Ø GOSUB 167Ø
115Ø NEXT J1
116Ø GOSUB612Ø:PRINT@224,"THE CA
RDS HAVE BEEN SHUFFLED-DO YOU WI
SH A RE-SHUFFLE ":INPUT" (Y OR N
)";G$
117Ø IF G$="N"  THEN 123Ø
118Ø GOSUB612Ø:PRINT@97,"THE CAR
DS ARE BEING RE-SHUFFLED NOW"
119Ø FOR I=1 TO 22
12ØØ C(I)=Ø
121Ø NEXT I
122Ø GOTO 113Ø
123Ø FOR J=1 TO 1Ø
124Ø PRINT@48Ø,"";:GOSUB611Ø:INP
UT"PRESS<ENTER> TO SEE EACH CARD
";H$:CLS
125Ø FOR U=1 TO 22
126Ø IF C(U)<>J  THEN 14ØØ
127Ø M$=""
128Ø RX=RND(Y):IFRX>.5 AND RX<.5
5 THEN129Ø ELSE13ØØ
129Ø M$="(REVERSED)"
13ØØ GOSUB173Ø
131Ø PRINT"TAROT CARD #"J""M$
132Ø PRINT@1ØØ, "INTERPRETATION:
"
133Ø PRINT@132,"---------------
"
134Ø PRINT@225, N$(J)
135Ø IFM$<>"(REVERSED)"  THEN 13
8Ø
136Ø D$(J)=R$(U):D(J)=R(U)
137Ø PRINT@29Ø,D$(J):PRINT:GOTO1
39Ø
138Ø D$(J)=I$(U):D(J)=N(U): GOTO
 137Ø
139Ø GOTO 141Ø
14ØØ NEXT U
141Ø NEXTJ
142Ø GOSUB172Ø
143Ø PRINT@13Ø,"PRESENTLY THERE
IS"
144Ø IF D(7)+D(1)=Ø  THEN 147Ø
145Ø PRINT@162,D$(1)" "D$(7)"."
146Ø GOTO 148Ø
147Ø PRINT@162,D$(7)"."
148Ø PRINT@194,"TO OTHERS YOU AR
E":PRINT@226,D$(8)"."
149Ø PRINT@258,"YOUR THOUGHTS TE
ND TOWARD":PRINT@29Ø,D$(9)"."
15ØØ GOSUB172Ø
151Ø PRINT@162,"ALTHOUGH IN THE
PAST YOU HAD"
152Ø IF D(4)+D(5)=Ø  THEN 155Ø
153Ø PRINT@194,D$(4)" "D$(5)","
154Ø GOTO 156Ø
155Ø IFD(4)=-1 THEN PRINT@194,D$
(4)","ELSE PRINT@194,D$(5)","
156Ø PRINT@226,"YOU WILL BE IMME
DIATELY":PRINT@258,"INVOLVED WIT
H":PRINT@29Ø,D$(2)"."
157Ø PRINT@322,"IN THE FUTURE TH
ERE WILL BE"
158Ø IF D(6)+D(3)=Ø  THEN 161Ø
159Ø PRINT@354,D$(6)" "D$(3)"."
16ØØ GOTO 2Ø7Ø
161Ø IF D(6)=1 THEN PRINT@354,D$
(6)"."ELSE PRINT@354,D$(3)"."
162Ø GOSUB172Ø
163Ø PRINT@13Ø,"REGARDING YOUR Q
UESTION,":PRINT@162,CHR$(34)Q$CH
R$(63)CHR$(34):PRINT@194,"THE CA
RDS REVEAL...":PRINT@226,D$(1Ø)"
."
164Ø GOSUB172Ø:PRINT@418,"";:INP
UT"ANOTHER READING? (Y OR N)";Q$
:IFQ$="N"THEN614Ø
165Ø CLS:GOTO11ØØ
166Ø GOSUB612Ø:PRINT@1ØØ,"IN FOU
R WORDS OR LESS,":GOTO111Ø
167Ø Z=(RND(Y)*22+1)
168Ø X=INT(Z)
169Ø IF C(X)<>Ø  THEN 167Ø
17ØØ C(X)=C(X)+J1
171Ø RETURN
172Ø GOSUB611Ø:FORTM=1TO3ØØØ:NEX
TTM:CLS:GOSUB612Ø:GOSUB613Ø:PRIN
T@194,"";:RETURN
173Ø PMODE 3,1
174Ø PCLS
```

```
1750 SCREEN 1,1
1760 ON U GOSUB599Ø,179Ø,197Ø,23
2Ø,28ØØ,299Ø,319Ø,347Ø,366Ø,389Ø
,4Ø7Ø,428Ø,449Ø,469Ø,49ØØ,5Ø5Ø,5
22Ø,542Ø,553Ø,57ØØ,578Ø,59ØØ
1770 FORTM=1TO1ØØØ:NEXT TM
1780 RETURN
1790 DRAW"BM5Ø,19Ø;R17Ø;U2Ø;L17Ø
;U15Ø;R17Ø;U2Ø;L17Ø;D19Ø;R17Ø;U1
9Ø"
18ØØ DRAW"BM135,16;"+LI$
181Ø DRAW"BM8Ø,185;"+LL$+LE$+SP$
182Ø DRAWLB$+LA$+LT$
183Ø DRAWLE$+LL$
184Ø DRAW LE$+LU$+LR$
185Ø DRAW"BM5Ø,116;C8;E39;R32;U8
;L8;R44;L8;D36;L28;U28"
186Ø DRAW"BM22Ø,116;H39;L32"
187Ø DRAW"BM176,116;U5;L84;D5;R8
4"
188Ø DRAW"BM124,152;U24;L36;D6;R
16;L22;D6;R22;L16;D6;R16;L1Ø;D6;
R3Ø"
189Ø DRAW"BM144,152;U24;R36;D6;L
16;R22;D6;L22;R16;D6;L16;R1Ø;D6;
L3Ø"
19ØØ CIRCLE(12Ø,56),5,8,1,Ø,1
191Ø CIRCLE (148,56),5,8,1,Ø,1
192Ø CIRCLE(135,36),5,8,1,Ø,1
193Ø PAINT(12Ø,56),8,8:PAINT(148
,56),8,8:PAINT(135,36),8,8
194Ø PAINT(13Ø,76),7,8:PAINT(124
,113),7,8
195Ø PAINT(175,8Ø),6,8
196Ø RETURN
197Ø LINE(5Ø,19Ø)-(22Ø,1),PSET,B
198Ø LINE(5Ø,1ØØ)-(22Ø,1ØØ),PSET
199Ø LINE(5Ø,2Ø)-(22Ø,2Ø),PSET
2ØØØ LINE(5Ø,17Ø)-(22Ø,17Ø),PSET
2Ø1Ø DRAW"BM13Ø,16;"+LI$+LI$
2Ø2Ø DRAW"BM112,185;"+LJ$+LU$
2Ø3Ø DRAWLN$+LO$+LN$
2Ø4Ø DRAW"BM124,52;C8;U2Ø;F4;D4;
R4;E4;F4;D4;R4;E4;F4;D16;N;F8;H8
;L2Ø;U4;R2Ø;F8"
2Ø5Ø DRAW"BM124,52;D2;R12;F6;D12
;F12;U8;H4;U4;E2;U8"
2Ø6Ø DRAW"BM124,64;L4;E4"
2Ø7Ø DRAW"BM132,8Ø;U9"
2Ø8Ø DRAW"BM136,96;N;E15;N;H15"
2Ø9Ø DRAW"BM136,116;H4;R8;G4"
21ØØ CIRCLE(136,54),15,8,(24/15)
,.25,.5
211Ø CIRCLE(136,56),24,8,(32/24)
,.6,1:CIRCLE(136,56),24,8,(32/24
),Ø,.Ø625
212Ø CIRCLE(132,1ØØ),2Ø,8,1,.5,.
75:CIRCLE(14Ø,1ØØ),2Ø,8,1,.75,1
213Ø LINE(16Ø,1ØØ)-(184,2Ø),PSET
:LINE(164,1ØØ)-(188,2Ø),PSET
214Ø CIRCLE(136,12Ø),4:CIRCLE(13
6,12Ø),16,8,1,.5,1:CIRCLE(136,17
2),16,8,(48/16),.5,1
215Ø CIRCLE(96,158),2Ø,8,1,.375,
.75:CIRCLE(1Ø8,132),2Ø,8,1,.375,
.875
216Ø CIRCLE(176,158),2Ø,8,1,.75,
1:CIRCLE(164,132),2Ø,8,1,.625,1
217Ø CIRCLE(176,158),2Ø,8,1,Ø,.1
25:CIRCLE(164,132),2Ø,8,1,Ø,.125
218Ø PAINT(136,9Ø),5,8
219Ø PAINT(136,98),8,8
22ØØ PAINT(154,98),8,8
221Ø PAINT(14Ø,56),7,8
222Ø PAINT(154,5Ø),8,8
223Ø PAINT(136,5Ø),8,8
224Ø PAINT(14Ø,35),8,8
225Ø PAINT(6Ø,98),6,8
226Ø PAINT(184,98),6,8
227Ø PAINT(2ØØ,168),5,8
228Ø PAINT(153,72),6,8
229Ø PAINT(136,168),8,8
23ØØ PAINT(166,168),7,8
231Ø RETURN
232Ø LINE(5Ø,19Ø)-(22Ø,1),PSET,B
233Ø LINE(5Ø,2Ø)-(22Ø,2Ø),PSET
234Ø LINE(5Ø,17Ø)-(22Ø,17Ø),PSET
235Ø DRAW"BM122,16;"+LI$+LI$+LI$
236Ø DRAW"BM7Ø,185;"+LL$+SP$+LI$
237Ø DRAWLM$+LP$:DRAWLE$+LR$:DRA
WLA$+LT$:DRAWLR$+LI$:DRAWLC$+LE$
238Ø CIRCLE(132,32),4,8,1,Ø,1
239Ø CIRCLE(132,44),8,8,1,.375,1
24ØØ CIRCLE(132,44),8,8,1,Ø,.175
241Ø CIRCLE(112,44),4,8,1,.5,1
242Ø CIRCLE(112,44),4,8,1,Ø,.175
243Ø CIRCLE(12Ø,44),4,8,1,Ø,.5
244Ø CIRCLE(144,44),4,8,1,Ø,.5
245Ø CIRCLE(152,44),4,8,1,.375,1
246Ø DRAW"BM126,52;C8;N;U8;L6;N;
H6;L4;N;H8;D8;R32"
247Ø DRAW"BM148,6Ø;U8;N;E8;L4;N;
E6;L6;U4"
248Ø CIRCLE(116,64),4,8,1,.25,.7
5
249Ø CIRCLE(148,64),4,8,1,.75,1
25ØØ CIRCLE(148,64),4,8,1,Ø,.25
251Ø CIRCLE(132,76),12,8,(16/12)
,Ø,1
252Ø CIRCLE(116,8Ø),12,8,1,.25,.
75
253Ø CIRCLE(148,8Ø),12,8,1,.75,1
254Ø CIRCLE(148,8Ø),12,8,1,Ø,.25
255Ø DRAW"C8;D6;U8":DRAW"BM148,9
2;C8;D4"
256Ø CIRCLE(12Ø,96),6,8,1,.125,.
625
257Ø CIRCLE(144,96),6,8,1,.875,1
258Ø CIRCLE(144,96),6,8,1,Ø,.375
259Ø DRAW"BM124,1ØØ;N;U14;N;G12;
```

```
D4"
2600 DRAW"BM140,100;N;U14;F12;D4
"
2610 CIRCLE(120,108),2,8,1,0,1
2620 CIRCLE(144,108),2,8,1,0,1
2630 CIRCLE(128,112),2,8,1,0,1
2640 CIRCLE(136,112),2,8,1,0,1
2650 CIRCLE(104,120),12,8,1,.25,
.925
2660 CIRCLE(160,120),12,8,1,.625
,1
2670 CIRCLE(160,120),12,8,1,0,.2
5
2680 CIRCLE(104,136),4,8,1,.75,1
2690 CIRCLE(112,136),20,8,1,.75,
1
2700 CIRCLE(120,136),12,8,1,.25,
.5
2710 CIRCLE(144,136),12,8,1,0,.2
5
2720 CIRCLE(152,136),20,8,1,.5,.
75
2730 CIRCLE(160,136),4,8,1,.5,.7
5
2740 CIRCLE(120,160),12,8,1,.75,
1
2750 CIRCLE(144,160),12,8,1,.5,.
75
2760 PAINT(122,50),8,8:PAINT(142
,50),8,8
2770 PAINT(116,80),7,8:PAINT(148
,80),7,8
2780 PAINT(104,120),8,8:PAINT(20
0,160),6,8
2790 RETURN
2800 LINE(50,190)-(220,1),PSET,B
2810 LINE(50,20)-(220,20),PSET
2820 LINE(50,170)-(220,170),PSET
2830 DRAW"BM127,16;"+LI$+LV$
2840 DRAW"BM90,185;"+LL$+SP$
2850 DRAWLE$+LM$:DRAW LP$+LE$:DR
AW LR$+LE$:DRAWLU$+LR$
2860 CIRCLE(132,48),4,8,1,0,1
2870 CIRCLE(132,52),20,8,1,.5,1
2880 DRAW"BM132,40;C8;N;R4;N;D4;
N;L4;N;U4"
2890 DRAW"BM112,52;D2":DRAW"BM15
2,52;D2"
2900 CIRCLE(116,60),8,8,1,.5,1:C
IRCLE(132,60),8,8,1,.5,1:CIRCLE(
148,60),8,8,1,.5,1
2910 DRAW"BM108,60;D4;F8;R8;N;U1
2;R16;N;U12;R8":DRAW"E8;U4;D4;G8
;D4;L32;U4"
2920 CIRCLE(88,72),28,8,1,0,.25:
CIRCLE(176,72),28,8,1,.25,.5
2930 DRAW"BM132,76;G8;L4;D20;E8;
R8;F8;U20;L4;H8"
2940 CIRCLE(104,108),12,8,1,.25,
.75:CIRCLE(160,108),12,8,1,.75,1
:CIRCLE(160,108),12,8,1,0,.25
2950 CIRCLE(104,148),28,8,1,.75,
1:CIRCLE(160,148),28,8,1,.5,.75
2960 CIRCLE(132,124),48,8,(36/48
),0,.675:CIRCLE(132,124),48,8,(3
6/48),.925,1
2970 PAINT(140,48),8,8:PAINT(132
,100),7,8:PAINT(160,140),8,8:PAI
NT(200,140),6,8
2980 RETURN
2990 LINE(50,190)-(220,1),PSET,B
3000 LINE(50,20)-(220,20),PSET
3010 LINE(50,170)-(220,170),PSET
3020 DRAW"BM128,16;"+LV$
3030 DRAW"BM100,185;"+LJ$
3040 DRAWLU$+LP$:DRAWLI$+LT$:DRA
WLE$+LR$
3050 DRAW"BM132,36;C8;G8;H8;G4;L
4;H4;L4;F12;N;R40;D12"
3060 DRAW"R40;U12;E12;L4;G4;L4;H
4;G8;H8"
3070 CIRCLE(132,76),20,8,(24/20)
,.875,1:CIRCLE(132,76),20,8,(24/
20),0,.125:CIRCLE(132,76),20,8,(
24/20),.375,.625
3080 CIRCLE(132,76),32,8,(24/32)
,.375,.625:CIRCLE(132,76),32,8,(
24/32),.875,1:CIRCLE(132,76),32,
8,(24/32),0,.125
3090 CIRCLE(132,104),20,8,1,.625
,.875
3100 DRAW"BM112,92;C8;N;L20;G8;D
12;G4"
3110 DRAW"BM152,92;N;R20;F8;D12;
F4"
3120 CIRCLE(92,112),20,8,1,.5,.7
5
3130 CIRCLE(172,112),20,8,1,.75,
1
3140 CIRCLE(116,116),16,8,1,0,.5
:CIRCLE(148,116),16,8,1,0,.5
3150 CIRCLE(172,92),100,8,(56/10
0),.22,.471
3160 DRAW"BM72,112;C8;D58;R120;U
58"
3170 PAINT(180,168),8,8:PAINT(15
0,112),7,8:PAINT(200,168),6,8
3180 RETURN
3190 LINE(50,190)-(220,1),PSET,B
3200 LINE(50,20)-(220,20),PSET
3210 LINE(50,170)-(220,170),PSET
3220 DRAW"BM127,16;"+LV$+LI$
3230 DRAW"BM90,185;"+LL$+SP$
3240 DRAWLA$+LM$:DRAWLO$+LU$:DRA
WLR$+LE$:DRAWLU$+LX$
3250 DRAW"BM132,28;C8;N;G48;N;D6
4;N;F48"
3260 DRAW"BM131,30;N;D62;R2;D62;
R3;G4;H4;R8"
3270 CIRCLE(96;76),12,8,(8/12),.
```

```
375,1
3280 CIRCLE(96,8Ø),12,8,(8/12),.
5,1
329Ø CIRCLE(12Ø,76),12,8,(8/12),
Ø,.5
33ØØ CIRCLE(12Ø,8Ø),12,8,(8/12),
Ø,.5
331Ø CIRCLE(144,76),12,8,(8/12),
Ø,.5
332Ø CIRCLE(144,8Ø),12,8,(8/12),
Ø,.5
333Ø CIRCLE(168,76),12,8,(8/12),
.5,1
334Ø CIRCLE(168,76),12,8,(8/12),
Ø,.175
335Ø CIRCLE(168,8Ø),12,8,(8/12),
.5,1
336Ø CIRCLE(88,12Ø),12,8,1,.25,1
337Ø CIRCLE(112,12Ø),12,8,1,.5,1
338Ø CIRCLE(112,12Ø),12,8,1,Ø,.2
5
339Ø CIRCLE(152,12Ø),12,8,1,.25,
1
34ØØ CIRCLE(176,12Ø),12,8,1,.5,1
341Ø CIRCLE(176,12Ø),12,8,1,Ø,.2
5
342Ø DRAW"BM1ØØ,148;N;H2Ø;N;E2Ø"
343Ø DRAW"BM164,148;N;H2Ø;N;E2Ø"
344Ø PAINT(112,12Ø),8,8:PAINT(17
6,12Ø),8,8
345Ø PAINT(2ØØ,168),7,8:PAINT(14
4,76),7,8:PAINT(12Ø,76),7,8
346Ø RETURN
347Ø LINE(5Ø,19Ø)-(22Ø,1),PSET,B
348Ø LINE(5Ø,2Ø)-(22Ø,2Ø),PSET
349Ø LINE(5Ø,17Ø)-(22Ø,17Ø),PSET
35ØØ DRAW"BM127,16;"+LV$+LI$+LI$
351Ø DRAW"BM9Ø,185;"+LL$+LE$+SP$
352Ø DRAWLC$+LH$:DRAW LA$+LR$:DR
AW LI$+LO$+LT$
353Ø CIRCLE(124,36),2,8,(4/2),.7
5,1:CIRCLE(124,36),2,8,(4/2),Ø,.
25
354Ø CIRCLE(124,4Ø),8,8,1,.75,1:
CIRCLE(96,44),2Ø,8,(16/2Ø),Ø,.25
355Ø CIRCLE(124,52),12,8,1,.125,
.375:CIRCLE(112,68),8,8,1,.5,1
356Ø CIRCLE(124,52),12,8,1,.625,
.75
357Ø DRAW "BM1Ø4,68;C8;H4;U8"
358Ø CIRCLE(132,92),28,8,1,.5,.7
5:CIRCLE(132,92),28,8,(52/28),.7
5,1
359Ø DRAW"BM132,92;C8;N;L28;R28"
36ØØ CIRCLE(132,132),28,8,1,Ø,1:
CIRCLE(132,132),2Ø,8,1,Ø,1
361Ø CIRCLE(132,132),8,8,1,Ø,1
362Ø DRAW"BM128,128;C8;N;L16;N;U
16;C5;R8;C8;N;U16;N;R16"
363Ø DRAW"C5;D8;C8;N;R16;N;D16;C
5;L8;C8;N;L16;D16"
364Ø PAINT(132,9Ø),7,8:PAINT(144
,12Ø),8,8:PAINT(12Ø,12Ø),8,8:PAI
NT(12Ø,14Ø),8,8:PAINT(144,14Ø),8
,8:PAINT(2ØØ,168),8,8
365Ø RETURN
366Ø LINE(5Ø,19Ø)-(22Ø,1),PSET,B
367Ø LINE(5Ø,2Ø)-(22Ø,2Ø),PSET
368Ø LINE(5Ø,17Ø)-(22Ø,17Ø),PSET
369Ø DRAW"BM117,16;"+LV$+LI$:DRA
WLI$+LI$
37ØØ DRAW"BM9Ø,185;"+LL$+LA$+SP$
:DRAWLJ$+LU$:DRAW LS$+LT$:DRAW L
I$+LC$+LE$
371Ø CIRCLE(132,36),4,8,1,Ø,1:DR
AW"BM132,4Ø;C8;G4;D16;L28;D4;R32
;N;H4;R32;U4"
372Ø DRAW"L28;N;G4;U16;H4"
373Ø LINE(1ØØ,64)-(92,88),PSET
374Ø LINE(1ØØ,64)-(1Ø8,88),PSET
375Ø LINE(164,64)-(156,88),PSET
376Ø LINE(164,64)-(172,88),PSET
377Ø CIRCLE(1ØØ,88),8,8,1,Ø,.5
378Ø DRAW"BM92,88;C8;R16"
379Ø CIRCLE(164,88),8,8,1,Ø,.5
38ØØ DRAW"BM156,88;C8;R16"
381Ø DRAW"BM76,116;C8;D4;F4;L4;D
32;R4;G4;D4;R8":DRAW"U4;H4;R4;U3
2;L4;E4;U4;L8"
382Ø DRAW"BM76,136;L12;D4;H4;L4;
D8;R4;E4;D4;R12"
383Ø LINE(212,14Ø)-(84,136),PSET
384Ø LINE(212,14Ø)-(84,144),PSET
385Ø PAINT(132,44),7,8:PAINT(68,
14Ø),7,8:PAINT(2ØØ,168),6,8
386Ø PAINT(1ØØ,8Ø),6,8:PAINT(164
,8Ø),6,8
387Ø PAINT(1ØØ,9Ø),8,8:PAINT(164
,9Ø),8,8
388Ø RETURN
389Ø LINE(5Ø,19Ø)-(22Ø,1),PSET,B
39ØØ LINE(5Ø,2Ø)-(22Ø,2Ø),PSET
391Ø LINE(5Ø,17Ø)-(22Ø,17Ø),PSET
392Ø DRAW"BM127,16;"+LI$+LX$
393Ø DRAW"BM1ØØ,185;"+LL$+SP$
394Ø DRAW LE$+LR$:DRAW LM$+LI$:D
RAW LT$+LE$
395Ø CIRCLE(176,52),8,8,1,.5,1:C
IRCLE(176,52),12,8,1,.5,1
396Ø DRAW"BM188,52;C8;G2;H2"
397Ø DRAW"BM167,52;D92;R4;U92"
398Ø CIRCLE(92,72),8,8,(12/8),.5
,1
399Ø DRAW"BM99,72;L16"
4ØØØ CIRCLE(92,72),12,8,(16/12),
.5,1
4Ø1Ø DRAW"BM8Ø,72;L8;D52;R4;U4;R
32;D4;R4;U52;L8"
4Ø2Ø DRAW"BM76,76;D4Ø;R12;N;R8;U
8;R8;D8;R12;U4Ø;L32"
```

```
4030 DRAW"BM92,96;N;E8;N;R8;N;F8
;N;D8;N;G8;N;L8;N;H8;U8"
4040 PAINT(110,114),7,8:PAINT(16
9,140),7,8:PAINT(200,168),6,8:PA
INT(100,70),7,8
4050 PAINT(92,70),6,8
4060 RETURN
4070 LINE(50,190)-(230,1),PSET,B
4080 LINE(50,20)-(230,20),PSET:L
INE(50,170)-(230,170),PSET
4090 DRAW"BM128,16;"+LX$:DRAW"BM
55,185;"+LL$+LA$+SP$
4100 DRAW LR$+LO$:DRAW LU$+LE$+S
P$:DRAW LD$+LE$+SP$
4110 DRAW LF$+LO$:DRAWLR$+LT$:DR
AW LU$+LN$+LE$
4120 CIRCLE(132,60),4,8,1,0,1:CI
RCLE(132,60),8,8,1,0,1:CIRCLE(13
2,60),24,8,1,0,1:CIRCLE(132,60),
32,8,1,0,1
4130 DRAW"BM132,60;C8;N;E18;N;F1
8;N;G18;N;H18"
4140 DRAW"BM130,57;U25;L6;E8;F8;
L6;D25"
4150 CIRCLE(144,104),38,8,(20/38
),0,.25
4160 CIRCLE(184,104),4,8,1,.5,1:
CIRCLE(108,104),80,8,(64/80),0,.
25
4170 CIRCLE(144,132),8,8,1,.5,1:
CIRCLE(142,132),6,8,1,.5,1
4180 CIRCLE(108,132),44,8,(36/44
),0,.25:CIRCLE(108,132),40,8,(32
/40),0,.25
4190 CIRCLE(108,132),28,8,1,0,.2
5:CIRCLE(108,164),4,8,1,.25,.75:
CIRCLE(108,162),4,8,1,.25,.75
4200 CIRCLE(104,116),12,8,1,.5,1
:CIRCLE(104,116),8,8,1,.5,1
4210 DRAW"BM112,116;D12;R4;U12":
DRAW"BM92,116;D12;R4;U12"
4220 CIRCLE(80,144),8,8,1,.5,1:C
IRCLE(80,144),4,8,1,.5,1
4230 DRAW"BM72,144;D8;R4;U8":DRA
W"BM84,144;D8;R4;U8"
4240 PAINT(132,26),8,8:PAINT(132
,30),8,8:PAINT(132,34),8,8
4250 PAINT(132,44),8,8:PAINT(132
,60),8,8:PAINT(116,60);8,8
4260 PAINT(148,60),7,8:PAINT(142
,140),7,8:PAINT(200,168),6,8
4270 RETURN
4280 LINE(50,190)-(220,1),PSET,B
4290 LINE(50,20)-(220,20),PSET:L
INE(50,170)-(220,170),PSET
4300 DRAW"BM126,16;"+LX$+LI$
4310 DRAW"BM100,185;"+LL$+LA$+SP
$
4320 DRAW LF$+LO$: DRAW LR$+LC$+
LE$
4330 CIRCLE(96,54),16,8,1,0,1:CI
RCLE(168,54),16,8,1,0,1
4340 DRAW"BM156,52;C8;L12;D4;N;R
12;D4;G4;D4"
4350 LINE(140,68)-(148,100),PSET
4360 DRAW"BM144,52;H4;G4;L4;D4;R
8;G4;D4"
4370 DRAW"BM140,48;L4;N;D4;L4;N;
D6;L4;N;D6;L4;D6;N;R8"
4380 DRAW"BM124,56;D4;F4;D4"
4390 DRAW"BM128,116;R4":DRAW"BM1
24,52;L12":DRAW"BM124,56;L12"
4400 DRAW"BM124,56;R16"
4410 DRAW"BM208,92;E12"
4420 CIRCLE(128,92),8,8,(24/8),.
25,.75
4430 CIRCLE(164,100),16,8,1,.5,1
:CIRCLE(200,100),20,8,(12/20),.5
,.875
4440 CIRCLE(156,116),24,8,(8/24)
,0,.5:CIRCLE(220,116),40,8,(32/4
0),.25,.5
4450 CIRCLE(164,144),20,8,1,.75,
1:CIRCLE(200,128),16,8,(42/16),.
25,.5
4460 PAINT(96,54),7,8:PAINT(120,
54),7,8:PAINT(128,55),7,8
4470 PAINT(168,54),7,8:PAINT(148
,54),7,8:PAINT(200,54),8,8
4480 RETURN
4490 LINE(50,190)-(220,1),PSET,B
4500 LINE(50,20)-(220,20),PSET:L
INE(50,170)-(220,170),PSET
4510 DRAW"BM121,16;"+LX$+LI$+LI$
4520 DRAW"BM100,185;"+LL$+LE$+SP
$
4530 DRAWLP$+LE$:DRAWLN$+LD$+LU$
4540 DRAW"BM104,36;C8;D12;R60;D1
22;R12;U134;L72"
4550 DRAW"BM120,36;D24;R2;U24;D2
4;R6;U4;L16;D24;N;D16;L4"
4560 DRAW"G4;D20;R16;U16;N;U8;R1
2;U16;L4;D8;L8;U20"
4570 DRAW"BM104,104;L4;D8;L4;G4;
D20;F4;R32;E4;U12;H4"
4580 DRAW"L4;U16;L4"
4590 DRAW"BM124,120;L16;D8;R16;D
4"
4600 DRAW"BM124,120;L24;D12"
4610 DRAW"BM112,120;N;D8;N;U8;R2
;N;U8;N;D8;R4;U8;L18"
4620 CIRCLE(112,148),8,8,1,0,1
4630 DRAW"BM104,148;D12;F4;N;U10
;R4;N;U8;R4;N;U10;E4;U12"
4640 PAINT(116,88),8,8:PAINT(12+
,58),8,8
4650 PAINT(112,148),8,8:PAINT(12
4,84),8,8
4660 PAINT(106,158),7,8:PAINT(11
0,158),7,8:PAINT(114,158),7,8:PA
```

```
INT(118,158),7,8
467Ø PAINT(2ØØ,168),6,8
468Ø RETURN
469Ø LINE(5Ø,19Ø)-(22Ø,1),PSET,B
47ØØ LINE(5Ø,2Ø)-(22Ø,2Ø),PSET:L
INE(5Ø,17Ø)-(22Ø,17Ø),PSET
471Ø DRAW"BM119,16;"+LX$+LI$:DRA
W LI$+LI$
472Ø DRAW"BM1Ø5,185;"+LL$+LA$+SP
$
473Ø DRAW LM$+LO$:DRAW LR$+LT$
474Ø CIRCLE(132,64),28,8,(24/28)
,.375,1:CIRCLE(132,64),28,8,(24/
28),Ø,.125
475Ø CIRCLE(132,8Ø),16,8,1,Ø,.5
476Ø LINE(132,64)-(136,76),PSET
477Ø LINE(136,76)-(128,76),PSET:
LINE(128,76)-(132,64),PSET
478Ø DRAW"BM128,64;C8;H4;L8;G4;D
4;F4;R8;E4;U4"
479Ø DRAW"BM136,64;E4;R8;F4;D4;G
4;L8;H4;U4"
48ØØ CIRCLE(1Ø4,1Ø6),4,8,(6/4),Ø
,1
481Ø CIRCLE(16Ø,1Ø6),4,8,(6/4),Ø
,1
482Ø CIRCLE(1Ø4,142),4,8,(6/4),Ø
,1
483Ø CIRCLE(16Ø,142),4,8,(6/4),Ø
,1
484Ø L.NE(1Ø4,1Ø4)-(16Ø,14Ø),PSE
T
485Ø LINE(1Ø4,1Ø8)-(16Ø,144),PSE
T
486Ø LINE(1Ø4,14Ø)-(16Ø,1Ø4),PSE
T
487Ø LINE(1Ø4,144)-(16Ø,1Ø8),PSE
T
488Ø PAINT(12Ø,64),8,8:PAINT(144
,64),8,8:PAINT(2ØØ,168),8,8
489Ø RETURN
49ØØ LINE(5Ø,19Ø)-(22Ø,1),PSET,B
491Ø LINE(5Ø,2Ø)-(22Ø,2Ø),PSET:L
INE(5Ø,17Ø)-(22Ø,17Ø),PSET
492Ø DRAW"BM12Ø,16;"+LX$+LI$:DRA
WLV$
493Ø DRAW"BM87,185;"+LT$+LE$:DRA
W LM$+LP$:DRAW LE$+LR$:DRAWLA$+L
N$:DRAWLC$+LE$
494Ø CIRCLE(132,28),12,8,(6/12),
Ø,1:CIRCLE(132,48),12,8,1,Ø,1
495Ø DRAW"BM132,4Ø;C8;N;L8;R8"
496Ø CIRCLE(172,56),12,8,1,.625,
1:CIRCLE(172,56),12,8,1,Ø,.125
497Ø CIRCLE(92,56),12,8,1,.325,.
875:CIRCLE(132,96),2Ø,8,(36/2Ø),
.5,1
498Ø DRAW"BM1ØØ,48;C8;F2Ø;G2Ø;N;
F12;G4;D4;R4;E4"
499Ø DRAW"BM164,48;G2Ø;F2Ø;N;G12
;F4;D4;L4;H4"
5ØØØ CIRCLE(84,96),8,8,(32/8),.7
5,1:CIRCLE(84,96),8,8,(32/8),Ø,.
25
5Ø1Ø CIRCLE(18Ø,96),8,8,(32/8),.
25,.75:CIRCLE(98,128),14,8,(8/14
),.5,1:CIRCLE(166,128),14,8,(8/1
4),.5,1
5Ø2Ø DRAW"BM112,96;D48;G8;R56;H8
;U48"
5Ø3Ø PAINT(132,28),7,8:PAINT(2ØØ
,168),7,8
5Ø4Ø RETURN
5Ø5Ø LINE(5Ø,19Ø)-(22Ø,1),PSET,B
5Ø6Ø LINE(5Ø,2Ø)-(22Ø,2Ø),PSET:L
INE(5Ø,17Ø)-(22Ø,17Ø),PSET
5Ø7Ø DRAW"BM125,16;"+LX$+LV$
5Ø8Ø DRAW"BM95,185;"+LL$+LE$+SP$
:DRAWLD$+LI$:DRAWLA$+LB$:DRAWLL$
+LE$
5Ø9Ø CIRCLE(152,36),8,8,1,.875,1
:CIRCLE(152,36),8,8,1,Ø,.25
51ØØ CIRCLE(148,4Ø),12,8,1,.875,
1:CIRCLE(148,4Ø),12,8,1,Ø,.125
511Ø CIRCLE(112,36),8,8,1,.25,.6
25:CIRCLE(116,4Ø),12,8,1,.375,.6
25
512Ø CIRCLE(132,64),32,8,(24/32)
,.5,1:CIRCLE(132,56),16,8,(8/16)
,Ø,.5
513Ø CIRCLE(116,64),28,8,(4Ø/28)
,.375,.625:CIRCLE(148,64),28,8,(
4Ø/28),.875,1:CIRCLE(148,64),28,
8,(4Ø/28),Ø,.125
514Ø DRAW"BM1Ø4,52;C8;H12":DRAW"
BM16Ø,52;E12"
515Ø DRAW"BM116,56;G4;D8;F4;R8;E
8;F8;R8;E4;U8;H4"
516Ø DRAW"BM96,84;E4;F32;E32;F4"
517Ø CIRCLE(132,128),26,8,(36/26
),Ø,.5:CIRCLE(132,128),22,8,(32/
22),Ø,.5
518Ø DRAW"BM1Ø8,12Ø;C8;G8;R16;H8
":DRAW"BM132,12Ø;G8;R16;H8":DRAW
"BM156,12Ø;G8;R16;H8"
519Ø DRAW"BM13Ø,128;D42;R4;U42"
52ØØ PAINT(132,52),8,8:PAINT(1Ø8
,126),7,8:PAINT(132,126),7,8:PAI
NT(156,126),7,8:PAINT(11Ø,13Ø),7
,8:PAINT(132,13Ø),7,8:PAINT(154,
13Ø),7,8:PAINT(132,168),7,8
521Ø RETURN

522Ø LINE(5Ø,19Ø)-(22Ø,1),PSET,B
523Ø LINE(5Ø,2Ø)-(22Ø,2Ø),PSET:L
INE(5Ø,17Ø)-(22Ø,17Ø),PSET
524Ø DRAW"BM12Ø,16;"+LX$+LV$:DRA
WLI$
525Ø DRAW"BM55,185;"+LL$+LA$+SP$
:DRAWLM$+LA$:DRAWLI$+LS$:DRAWLO$
+LN$+SP$:DRAWLD$+LE$+SP$:DRAWLD$
```

```
+LI$:DRAWLE$+LU$
5260 CIRCLE(112,100),12,8,1,.5,1
5270 DRAW"BM50,124;C8;R52;N;U24;
R24;N;U24;R44;N;R52;U60;L8"
5280 DRAW"BM160,64;N;D12;U12;L8;
N;L16;U8;N;L8":DRAW"BM152,44;R4;
U8;L4;N;L4;U8;L8;D8;L8;D8;L16;D1
6;L12"
5290 LINE(180,44)-(184,56),PSET,
B:LINE(208,48)-(200,60),PSET,B:L
INE(184,64)-(176,80),PSET,B:LINE
(186,80)-(188,92),PSET,B
5300 LINE(188,108)-(196,116),PSE
T,B
5310 DRAW"BM184,100;L8;U8;F8":LI
NE(144,64)-(128,84),PSET,B
5320 DRAW"BM92,36;G12;H8;G8;H8;G
8;D16;F12;E8":DRAW"BM68,68;F8;G4
;F4;E8;F4;G4;F4;E8;F8"
5330 DRAW"BM104,88;E8;H4;E8;H16;
G4;N;G4;H4;E8;H8"
5340 LINE(88,64)-(72,64),PSET,B:
LINE(144,84)-(128,64),PSET,B
5350 LINE(120,152)-(208,132),PSE
T:LINE(108,156)-(208,136),PSET
5360 LINE(120,152)-(132,140),PSE
T:LINE(108,156)-(120,144),PSET
5370 LINE(132,140)-(72,152),PSET
:LINE(120,144)-(72,156),PSET
5380 DRAW"BM208,132;D4"
5390 LINE(72,148)-(76,160),PSET:
DRAW"BM76,160;L16;E12"
5400 PAINT(112,100),7,8:PAINT(13
2,80),7,8:PAINT(200,168),8,8:PAI
NT(190,100),7,8
5410 RETURN
5420 LINE(50,190)-(220,1),PSET,B
5430 LINE(50,20)-(220,20),PSET:L
INE(50,170)-(220,170),PSET
5440 DRAW"BM110,16;"+LX$+LV$:DRA
W LI$+LI$
5450 DRAW"BM105,188;"+LL$+SP$:DR
AW LE$+LT$:DRAW LO$+LI$:DRAW LL$
+LE$
5460 DRAW"BM132,36;C8;D48"
5470 LINE(132,36)-(160,124),PSET
:LINE(132,36)-(104,124),PSET
5480 LINE(180,68)-(132,84),PSET:
LINE(180,68)-(104,124),PSET:LINE
(180,68)-(84,68),PSET
5490 LINE(160,124)-(132,36),PSET
:LINE(160,124)-(132,84),PSET:LIN
E(160,124)-(84,68),PSET
5500 LINE(132,84)-(104,124),PSET
:LINE(132,84)-(84,68),PSET
5510 PAINT(200,168),7,8
5520 RETURN
5530 LINE(50,190)-(220,1),PSET,B
5540 LINE(50,20)-(220,20),PSET:L
INE(50,170)-(220,170),PSET
```

```
5550 DRAW"BM110,16;"+LX$+LV$:DRA
W LI$+LI$+LI$
5560 DRAW"BM105,185;"+LL$+LA$+SP
$:DRAW LL$+LU$:DRAW LN$+LE$
5570 CIRCLE(132,54),26,8,1,0,1:C
IRCLE(132,60),32,8,1,0,1
5580 CIRCLE(120,104),8,8,(12/8),
0,.75:CIRCLE(120,104),8,8,(12/8)
,.875,1
5590 CIRCLE(144,104),8,8,(12/8),
0,.625:CIRCLE(144,104),8,8,(12/8
),.75,1
5600 DRAW"BM120,104;C8;N;U12;E8"
5610 DRAW"BM144,104;N;U12;H8"
5620 CIRCLE(132,116),4:CIRCLE(13
2,116),12,8,1,.3,.5:CIRCLE(132,1
16),12,8,1,0,.2
5630 CIRCLE(132,116),16,8,1,.31,
.56:CIRCLE(132,116),16,8,1,0,.18
:CIRCLE(132,116),16,8,1,.94,1
5640 CIRCLE(132,140),8,8,(24/8),
0,1
5650 CIRCLE(124,152),8,8,1,.125,
.75
5660 CIRCLE(140,152),8,8,1,.75,1
:CIRCLE(140,152),8,8,1,0,.375
5670 PAINT(132,54),7,8:PAINT(120
,108),8,8:PAINT(144,108),8,8:PAI
NT(132,116),8,8:PAINT(120,122),8
,8:PAINT(144,122),8,8:PAINT(132,
140),8,8:PAINT(120,152),8,8:PAIN
T(144,152),8,8
5680 PAINT(200,168),7,8
5690 RETURN
5700 LINE(50,190)-(220,1),PSET,B
5710 LINE(50,20)-(220,20),PSET:L
INE(50,170)-(220,170),PSET
5720 DRAW"BM120,16;"+LX$+LI$+LX$
5730 DRAW"BM95,185;"+LL$+LE$+SP$
5740 DRAW LS$+LO$:DRAW LL$+LE$:D
RAW LI$+LL$
5750 CIRCLE(132,80),40,8,1,0,1:C
IRCLE(132,80),44,8,1,0,1:CIRCLE(
132,80),52,8,1,0,1:CIRCLE(132,80
),64,8,1,0,1:CIRCLE(132,80),80,8
,1,.875,1:CIRCLE(132,80),80,8,1,
0,.625
5760 PAINT(132,80),8,8:PAINT(174
,80),7,8:PAINT(180,80),7,8:PAINT
(188,80),7,8:PAINT(200,80),7,8:P
AINT(216,80),7,8
5770 RETURN
5780 LINE(50,190)-(220,1),PSET,B
5790 LINE(50,20)-(220,20),PSET:L
INE(50,170)-(220,170),PSET
5800 DRAW"BM120,16;"+LX$+LX$:DRA
WLI$
5810 DRAW"BM85,185;"+LL$+LE$+SP$
:DRAW LJ$+LU$:DRAW LG$+LE$:DRAW
LM$+LE$:DRAW LN$+LT$
```

```
5820 CIRCLE(120,52),20,8,1,.5,1:
CIRCLE(84,68),28,8,1,.0625,.875:
CIRCLE(156,72),24,8,1,0,.5:CIRCL
E(176,60),20,8,1,.75,1:CIRCLE(17
6,60),20,8,1,0,.25:CIRCLE(168,44
),12,8,1,.5,1
5830 DRAW"BM168,20;C8;G28;F4;E32
"
5840 CIRCLE(104,4),60,8,1,.125,.
25:CIRCLE(182,82),52,8,1,.5,.625
5850 CIRCLE(104,76),24,8,(12/24)
,.75,1:CIRCLE(104,70),2,8,(6/2),
.25,.75:CIRCLE(128,76),24,8,(12/
24),.25,.5:CIRCLE(128,82),2,8,(6
/2),.75,1:CIRCLE(128,82),2,8,(6/
2),0,.25
5860 DRAW"BM132,128;C8;L32;D4;R5
2;D8;N;R12;L4;D4;R20;U4;L4"
5870 DRAW"BM132,128;R20;U8;L4;U4
;R20;D4;L4;N;L12;D20"
5880 PAINT(200,168),7,8
5890 RETURN
5900 LINE(50,190)-(220,1),PSET,B
5910 LINE(50,20)-(220,20),PSET:L
INE(50,170)-(220,170),PSET
5920 DRAW"BM120,16;"+LX$+LX$:DRA
WLI$
5930 DRAW"BM100,185;"+LL$+LE$+SP
$:DRAW LM$+LO$:DRAW LN$+LD$+LE$
5940 CIRCLE(132,88),48,8,1,0,1
5950 DRAW"BM120,40;C8;D12;R8;E4;
R16;D8;L28;G8;D16;F4;R12;D24"
5960 DRAW"BM128,108;F12;E16;U16;
E8;H16;R4;F12;E12"
5970 PAINT(132,88),6,8:PAINT(100
,88),7,8
5980 RETURN
5990 LINE(50,190)-(220,1),PSET,B
6000 LINE(50,170)-(220,170),PSET
6010 DRAW"BM115,185;"+LL$+LE$+SP
$:DRAW LM$+LA$+LT$
6020 CIRCLE(152,68),28,8,1,.5,1:
CIRCLE(160,68),20,8,1,.5,1:CIRCL
E(152,72),12,8,1,.5,1:CIRCLE(156
,72),8,8,1,.5,1:CIRCLE(112,72),1
2,8,1,.5,1:CIRCLE(108,72),8,8,1,
.5,1
6030 CIRCLE(132,88),24,8,1,.625,
.875:CIRCLE(132,88),20,8,1,0,1
6040 DRAW"BM116,72;C8;D4;R32;U4"
6050 CIRCLE(132,76),16,8,1,.125,
.375:CIRCLE(132,76),16,8,(24/16)
,.1,.42
6060 CIRCLE(116,88),12,8,1,.25,.
75:CIRCLE(148,88),12,8,1,.75,1:C
IRCLE(148,88),12,8,1,0,.25
6070 DRAW"BM116,100;C8;G20;R12;D
12;E12;F12;E12;F12;U12;R12;H20"
6080 PAINT(132,60),8,8:PAINT(152
,60),7,8:PAINT(132,72),8,8:PAINT
(132,95),8,8:PAINT(108,88),7,8:P
AINT(156,88),7,8
6090 PAINT(200,168),6,8
6100 RETURN
6110 PLAY"XA3$;":RETURN
6120 CLS:PRINT@38,"***TAROT CARD
S***":RETURN
6130 PRINT@71,"****READING****":
RETURN
6140 CLS:GOSUB6110
6150 B$="L4;F;L2;E;D;L2.;C;P32;L
4;G;L2;A;L4;P32;A;L2.;B;P32;L4;B
;O3;L1.;C"
6160 PLAY"XB$;":END
```

PHONE NUMBERS

Way back in July when this thing started, I organized with Telecom to have my existing number moved to the office. The idea was that they would then put an extention through to the house.

They put in a temporary number at home - they said for two weeks.

Well it's still on, and we're not going to wait any longer!(Terry G. why don't you work on the Gold Coast?)

So my home number for calls between 7.00pm and 9.00pm Queensland time is:

075—32 9463

Graham.

THE EDUCATION PAGE

ROSS SPICER

Ross Spicer is one of three Computer Consultants working for the Education Department in the Wide Bay area of Queensland.

The work of the three consultants in that area is divided such that one works with Primary Schools, one works with students in Secondary Schools, and Ross's role is that of In Service Liason with Secondary School teachers. Ross travels between Noosa, Gympie, Maryborough and Bundaberg.

All High Schools in the Wide Bay area have computers and Ross's aim is to give each High School teacher in his area, a working knowledge of the computers in that teacher's school.

Many schools have seen the need to have a number of different brands of computers available to their students. It is felt that such a policy gives students a broader perspective and it also allows for easy divisions of utilization of computers within the school.

In the special teacher training course, Ross covers:

Keyboard Familiarity
Use of Disks
Curriculum Use
Public Domain Software
Commercial Programs

He also has "Goodies Sessions" where he demonstrates the use of light pens, graphics tablets, and screen graphics.

Next year will be a "crash" year when the consultants of the Wide Bay Area attempt to get computer instruction to every student.

Ross sees a need for a 'starter pack' of software for schools which purchase new computers. His department has in fact, just completed such a package for schools with Apples. This package of 5 disks covers a redesigned Logo for schools, Spreadsheets, word processors plus other software found to be essential in the classroom.

I asked Ross for his "top 3" programs - the programs that really work for both teacher and student. His reply:

1. First Fleet - Elizabeth Computer Centre. TAS.
2. One World.
3. Word processors in general.

I am of the opinion that with the programs already on CoCoOz, we are not that far away from being in a position to produce a good quality starter pack for the CoCo, so we will be working towards having something ready for January. As usual - if you have a contribution for the project - send it ... please!!

****

MAZE

This program has just been released by Deloitte, Haskins and Sells for use on the Model 2000.

Maze is different to every other educational package that we've mentioned hitherto in that it is for use by the administrators of the school, rather than the teachers.

Maze provides word processing, full student administration, TIMETABLING, general ledger, accounts payable and receivable, payroll and fixed assets.

Designed in W.A., we believe that Maze is the only program that is specifically designed to perform these tasks.

And, to me, the most exciting thing about Maze is that it is sold by Deloitte, Haskins and Sells - one of the oldest and largest international Accounting Firms.

Deloitte's have offices in 21 Australian cities and offer a breadth of financial services ranging from:

1. Accounting Services

BREAKTHROUGH FOR SCHOOL ADMINISTRATORS

Now! A software package designed exclusively for schools!

Imagine this – a software package that provides general ledger, accounts payable and receivable, payroll, fixed assets...along with full student administration, **timetabling,** enrolment planning, old students, fund raising...plus extensions such as report generator and word processing.

'MAZE' is the name of this new package. It's ease of operation will amaze you. Specially developed for primary and secondary school use. The package is proven, economical and easy to install.

For full details and a demonstration of 'MAZE' in action, contact Mr. R.J. Gibson.

Deloitte
Haskins+Sells
Chartered Accountants

307 Queen Street,
BRISBANE QLD. 4000.
Telephone 221 9122.
And in all States.

MAZE
SCHOOL ADMINISTRATION SOFTWARE

2. Auditing Services
3. Computer Applications
4. Computer Accounting
5. Compute Auditing
6. Financial Services
7. Government - assistance with Economic Planning
8. Management Consultancy
9. Assistance in Receivership and Insolvency
10. Share Registry
to 11. Taxation.

Anyway that's enough of the free ad for now......

We will be looking at Maze in place in a school and reporting to you upon it's operation in the near future. Expect to see a lot of schools using Maze from now on!

***

SPEECH/SOUND PACK

Four years ago I knew nothing about computers.

I purchased a Tandy PC 1 and then a Color Computer. I have specialized in Data handling in Basic and feel reasonably comfortable in that field - step me outside that and I'm still a babe in the woods.

What I know, I've learnt from Tandy. I've already said it and it bears repeating. Tandy produce better back-up than anyone - and nowhere in this more evident than when you purchase something extremely new from them. Their documentation is just EXCELLENT.

Perhaps because I got overly charitable in the last issue and said a few unnecessarily nice things about them, Greg and Leo at Fortitude Valley Computer Centre in Brisbane went out of their way to ensure that we received one of Tandy's new Speech/Sound packs in adequate time to give it a proper try-out.

Speech is going to come - in fact speech recognition can't be that far away either, and this pack paves the way beautifully.

The speech produced can at times sound "tinny"; fiddling with words can make them sound right, eg: "Deeveeit" for "deviate" (and don't ask which store we found it necessary to try that and other similar words - of course it HAD to be Pacific Fair!). (The classic sentance from the computer as the customer enters the store - "I must warn you that I am a deeveeit" - all done in measured staccato computereese - we had to stop it - we cleared the store!) Crudities aside, it was the kids who responded immediately - from my own 2 year old, to high school students at a recent computer night held at Miami High.

We set up a simple maths quiz program and everyone wanted to try it.

It wasn't just the novelty, it was the feedback - the computer becomes a friend - it talks to you - it helps you, it now tells you how to do things.

In order that you get the best from this pack, some work with the manual is necessary. Some of the information is just hard slog - particularly if you are an assembly language, binary number muggins like me - but the rewards are there - not only in speech, but also music (3 channels) and sound effects.

We were sufficiently impressed with this pack and it's possibilities in education to immediately commission a syllabus relevant spelling program and a maths program for use in Queensland primary schools with the pack.

These will be available for start of school next year.

The Speech/Sound Pack at around $160.00 is an exciting addition to the Tandy range.

We were asked to work on a demonstration program for the pack. The idea is to have something simple for the stores to use and at the same time be useful to someone just starting out.

We were unable to fit this program into Rainbow this month but it can be found on Rainbow on Tape (Nov).

We would absolutely love to receive programs which utilize this pack for publication in Australian CoCo.

'MAZE' x 1 = TIME/4

School Marm continued:

```
20 'COPYRIGHT 1982 BY JUDY M. AN
D DAVID M. DACUS, 206 CAPRI, LAS
 CRUCES, NM 88001
30 CLEAR 500:DIM N$(50),G$(50)
40 CLS:PRINT@0,STRING$(64,134);"
WE ARE NOW READY TO LIST GRADES.
"
50 PRINT@128,STRING$(32,134);"BE
 SURE THAT THE - GRADES - TAPEIS
 IN THE RECORDER, THE TAPE IS RE
WOUND, AND THE RECORDER IS ON PL
AY.":PRINT@288,STRING$(32,134);
60 INPUT"WHEN YOU ARE READY PUSH
 <ENTER>";A$:CLS:PRINT@0,STRING$
(224,134);"LOADING GRADES."
70 I = 0:N$ = "":G$ = ""
80 OPEN "I", #-1, "GRADES"
90 IF EOF (-1) THEN 140
100 I = I + 1
110 INPUT #-1, N1$,G1$
120 N$(I) = N1$:G$(I) = G1$
130 GOTO 90
140 CLOSE #-1
150 INPUT"DO YOU WANT 1.CRT OR 2
.HARDCOPY LISTING";A:IF A = 2 TH
EN 190
160 CLS:PRINT"LIST OF GRADES"
170 FOR X = 1 TO I:PRINT N$(X),G
$(X):NEXT X
180 INPUT "PRESS <ENTER> TO END
PROGRAM.";A$:GOTO 220
190 CLS:PRINT@0,STRING$(224,134)
;"PRINTING GRADES."
200 PRINT#-2, CHR$(14);"LIST OF
GRADES";CHR$(10);CHR$(10)
210 FOR X = 1 TO I:PRINT#-2, N$(
X),G$(X):NEXT X
220 END
```

16K ECB

THE CoCo SCHOOL MARM

Part 2

Judy & David Dacus

In this second of two articles we will present the programs to produce an integrated and automated spelling practice and examination system. The programs presented here require the support of the programs presented last article. The last article provided student practice and preparation. These programs provide individual evaluation of student progress, and recording and display of grades. Both the spelling practice program AUDIO SPELL presented last article, and SPELLTEST presented this article, use the tape recorder to produce the necessary pronunciation and use of words in sentences. The sound tape is controlled by the computer so that the tape pauses for student response after each word. We have used color block graphics to facilitate understanding, particularly for younger children.

The systems approach we have taken to automatic practice and examination sessions of the weekly spelling lessons for elementary and secondary schools is designed to work as follows. The student receives preparatory practice on the week's words with the program, AUDIOSPELL. the student is tested for ability to spell the week's words using a format identical to the practice sessions by SPELLING TEST. After the student completes the weekly spelling test, his or her grade is recorded to tape to be retrieved later and placed in the class gradebook. The WORD LOAD program provides an easy means of placing the week's spelling words on a data tape for either the AUDIOSPELL or the SPELLING TEST programs.

Data tapes created by *Word Load* are interchangeable between the practice and testing programs. The *Grade List* program lists student grades recorded by *Spelling Test*. *AudioSpell* and *Word Load* were presented last month. This month we present and explain the functions of *Spelling Test* and *GradeList*.

The Audio Spelling System is designed to operate on the 16K Extended Color BASIC Radio Shack Color Computer with nothing more than a tape recorder and color television.

Routines are provided for the use of a line printer if it is available. Modifications are presented later in the article for modifying the programs for use on a level one BASIC CoCo.

Spelling Test

The materials needed are:

Program Tape or Disk – Program Name *SpelTest*
Spelling Words Tape (to be made using *Word Load*)
Grades Tape
Color Computer, television, and tape recorder
Line printer or student-provided pencil and paper

Here are the instructions: *Spelling Test* is self-instructing, and works nearly identically to the *AudioSpell* program. If the student is familiar with operation of the Color Computer and loading programs from tape, he or she should be able to operate the program with no assistance. Otherwise, load and run the program for the student. Then place the Spelling Words tape in the recorder and push the play button. When the student completes the exam, a hard copy list of the exam will be printed for the student if you have a line printer. The exam listing will print the student's spelling of each word. When a word has been misspelled, an X is placed beside the word, and the correct spelling printed beside the incorrect word. The percentage score is printed at the top of the listing under the student's name. If a printer is not available, the student should be prepared to copy misspelled words from the screen. When the exam listing is completed, the student will be instructed to get the teacher. The keyboard will accept only the code word "KEYWORD" to continue the program. Insert the Grades tape into the tape recorder, type in "KEYWORD," ENTER, and follow the instructions for recording the student's grade to the Grades tape.

Grade List

The materials needed are:

Program Tape or Disk - Program Name *GradList*
Grades Tape or Data File on Disk
Line Printer or Pencil and Paper

The instructions are: The *GradeList* program is self-instructing. If you have a printer, each student's grade will be printed for you. If you do not have a printer it will be necessary for you to copy each student's grades from the television screen.

Modification Of The Programs

It is good practice to make a copy of the program and store away the original program before making modifications. This allows you to fall back to the original program if something should happen to the copy while you are making the modifications.

Five seconds recording time was selected as optimum for the average user. This time can be easily modified to suit the individual teacher's needs. To change recording time, it is necessary to change only one value in each of three programs. The Color Computer requires one second to count to 460 in a *FOR... NEXT* loop such as the one found in Line 280 of the *Word Load* program. To change the length of time allowed to pronounce the word and use it in a sentence, you must multiply the number of seconds desired times 460 and place the resulting value in Line 280 of the *Word Load* program, Line 290 of the *AudioSpell* program, and Line 280 of the *Spelling Test* program in place of the value 2300. All three programs must contain the same value in the timing loop for the tapes to be read correctly.

Using Programs When No Printer Is Available

If you do not have access to a printer, you may want to eliminate the student input regarding the printer. If you have a printer and always intend to have misspelled words printed rather than displayed on the TV screen, you may wish to eliminate the choice of TV display. To eliminate choice of the printer, change the expression "Do you have a printer (yes or no)" in Line 360 of *Spelling Test* to "Press ENTER to continue," and eliminate everything after the variable A$ in Line 360. Next, eliminate program Lines 400 through 460. To eliminate the choice of printing the list to the TV, modify Line 360 exactly as above, and eliminate program Lines 370 through 390.

Changing Printer Codes

The printing algorithms of these programs are written using ASCII codes for an Epson MX-80 printer. This printer uses *CHR$(14)* to print double width characters and *CHR$(10)* as a line feed command. If your printer does not use these two codes you must substitute your printer code for *CHR$(14)* in Lines 410,420, and 710 in *Spelling Test*, and Line 200 of *Grade List*, and substitute your printer's equivalent of *CHR$ (10)* in Lines 410, 420, and 710 of *Spelling Test*, and Line 200 of *Grade List*.

Modification To Run On A Non-Extended CoCo

As the programs are listed, they are for use on an Extended Color BASIC machine. To use these programs on a level one machine requires only removal or replacement of one reserved Extended word. The screens are formatted with the reserved word *STRING$*. This command prints a string of N copies of the ASCII character X as in *PRINT @ 0, STRING$ (N,X)*. To substitute for the *STRING$* command using level one BASIC you can substitute the algorithm

15 FOR I = 1 TO 32, : SC$ = SC$ + CHR$(169): NEXT I

You will need one line and one variable for each different color band you wish to print. After you have set up the variable at the front of the program, you may then substitute the command — *PRINT SC$* — in place of each *PRINT STRING$* command in the program.

Explanation Of ASCII Symbol

In Line 110 of *SpellTest* the listing shows an underscore character in parentheses in the instructions to be printed to the screen. On the computer screen this shows as a left arrow, and is made by entering a shift-up arrow on the keyboard.

Disk Version Availability

We have developed versions of the spelling programs modified for the Disk Extended Color Computer. Using the system on disk will allow automated recording of grades in a grade file without teacher intervention. It will also expedite and facilitate student use of the spelling programs. For those who use disk systems, the authors will provide a copy of the complete set of programs for disk on a tape for $10. Our address is 206 Capri Road, Las Cruces, NM 88005. If you have problems with your entered version of these programs, be sure you have thoroughly proof-read your code before writing or calling. Please send a self-addressed stamped envelope with your request for help or you can call us at 505-524-3389. Please call between 5 and 10 p.m. Mountain Time.

100	243
200	237
360	200
500	98
660	19
END	238

Listing 1:

```
10 ' SPELLING EXAM
20 'COPYRIGHT 1982 BY JUDY M. AN
D DAVID M. DACUS, 206 CAPRI, LAS
 CRUCES, NM 88001
30 CLEAR 2000:NW = 50:DIM WRD$(N
W),W$(NW),N$(30),G$(30)
40 CLS:PRINT@0,STRING$(32,175);:
PRINT@42,"SPELLING EXAM"
50 PRINT@64,STRING$(32,175);"HI!
  MY NAME IS COCO THE COLOR  COM
PUTER.  CALL ME COCO.  THAT'SWHA
T ALL MY FRIENDS CALL ME."
60 PRINT@192,STRING$(32,175);:IN
PUT "WHAT IS YOUR NAME";NAM$
70 PRINT@256,STRING$(32,175);"TH
AT'S A NICE NAME - "NAM$".":PRIN
T@320,STRING$(32,175);
80 PRINT@352,"DO YOU KNOW HOW I
WORK? (TYPE NOAND I WILL TELL YO
U WHAT TO DO) PLEASE TYPE YES OR
 NO AND          PUSH <ENTER>"
90 INPUT A$:IF LEFT$(A$,1) = "
 THEN 140
100 CLS:PRINT@0,STRING$(32,175);
"I WILL SAY THE WORD, I WILL USE
 THE WORD IN A SENTENCE, AND THE
NSAY THE WORD AGAIN.  WHEN I
 FINISH THE WORD I WILL ASK YOU
 TO SPELL IT.":PRINT@192,STRING$
(32,175);
110 PRINT"IF YOU MAKE A MISTAKE
USE THE   BACK ARROW (_) TO ERAS
E.":PRINT@288,STRING$(32,175);:I
NPUT"PUSH <ENTER> TO CONTINUE";A
$
120 CLS:PRINT@0,STRING$(32,175);
"WHEN YOU THINK YOU HAVE SPELLED
 THE WORD CORRECTLY,PRESS <ENTER
>.":PRINT@128,STRING$(32,175);
130 INPUT"PUSH <ENTER> TO CONTIN
UE";A$
140 CLS:PRINT@0,STRING$(32,175);
"NOW WE ARE READY TO START
 SPELLING THIS WEEK'S WORDS.":PR
INT@96,STRING$(32,175);
150 PRINT"PLEASE CHECK TO SEE TH
AT THE      TAPE MARKED - SPELLING
 WORDS -    IS IN THE TAPE RECORDE
R, THE      TAPE IS REWOUND, AND T
HE          RECORDER IS ON PLAY.":
PRINT@288,STRING$(32,175);
160 INPUT"WHEN YOU HAVE CHECKED
ALL THIS, PUSH MY <ENTER> BUTTON
 AND I'LL MOVE THE TAPE TO GET R
EADY.";A$:CLS:PRINTSTRING$(32,17
5);"I AM LOADING THE WORDS FROM
TAPE"
170 I = 0:W = 0:W$ = ""
180 OPEN"I", #-1,"WORDS"
190 IF EOF (-1) THEN 240
200 I = I + 1
210 INPUT #-1, W$
220 WRD$(I) = W$
230 GOTO 190
240 CLOSE #-1
250 NW = I
260 CLS:PRINT@0,STRING$(32,175);
:INPUT"WHEN YOU ARE READY FOR YO
UR      FIRST WORD PUSH MY <ENTER
>       BUTTON.";A$:CLS:PRINT@0,S
TRING$(224,175);"          LISTEN
CAREFULLY."
270 FOR I = 1 TO NW
```

```
28Ø AUDIO ON:MOTORON:FOR V = 1 T
O 23ØØ:NEXT V:MOTOROFF:AUDIO OFF
29Ø SKIPF "MARKER":CLS:PRINT@Ø,S
TRING$(224,175);
3ØØ INPUT"PLEASE SPELL THE WORD
YOU JUST  HEARD.";W$(I)
31Ø IF W$(I) = WRD$(I) THEN R =
R + 1 ELSE W = W + 1
32Ø PRINT@288,STRING$(32,175);:I
NPUT"READY FOR THE NEXT WORD";A$
:CLS:PRINT@Ø,STRING$(224,175);"
       LISTEN CAREFULLY"
33Ø NEXT I
34Ø CLS:AUDIO OFF:PRINT@Ø,STRING
$(32,175);"YOU ATTEMPTED TO SPEL
L              ";NW;" WORDS.":G = (R
/(W+R))*1ØØ
35Ø PRINT"YOU MISSPELLED ";W;" W
ORDS OUT OF ";W+R;" ATTEMPTS FOR
 A SCORE OF      ";G;"%."
36Ø PRINT@288,STRING$(32,175);:I
NPUT"DO YOU HAVE A PRINTER(YES O
R NO)";A$:IF LEFT$(A$,1) = "Y" T
HEN 4ØØ
37Ø CLS:PRINT "THESE ARE THE WOR
DS MISSED ":FOR I = 1 TO NW
38Ø IF W$(I) <> WRD$(I) THEN PRI
NT WRD$(I),:NEXT I ELSE NEXT I
39Ø PRINT:INPUT"WHEN YOU HAVE CO
PIED THESE WORDSON A PIECE OF PA
PER PUSH <ENTER>";A$:GOTO 47Ø
4ØØ CLS:PRINT@Ø,STRING$(224,175)
;"PRINTING SPELLING TEST"
41Ø PRINT#-2,CHR$(14);"SPELLING
TEST FOR ";NAM$;CHR$(1Ø);CHR$(1Ø
)
42Ø PRINT#-2,CHR$(14);"YOUR GRAD
E = ";G;"%";CHR$(1Ø);CHR$(1Ø)
43Ø PRINT#-2,"YOUR SPELLING", "M
ISSED","CORRECT SPELLING"
44Ø FOR I = 1 TO NW
45Ø PRINT#-2,I;".";W$(I),:IF W$(
I) <> WRD$(I) THEN PRINT#-2, "
 X",WRD$(I) ELSE PRINT#-2, ""
46Ø NEXT I
47Ø CLS:PRINT@Ø,STRING$(224,175)
;"GOOD LUCK WITH YOUR SPELLING.
  COME PRACTICE WITH ME FOR NEXT
  WEEK'S TEST."
48Ø PRINT@32Ø,STRING$(32,246);:I
NPUT"PLEASE GET YOUR TEACHER";A$
49Ø IF A$ <> "KEYWORD" THEN 48Ø
5ØØ CLS:PRINT@Ø,STRING$(64,134);
"WE ARE NOW READY TO RECORD
 GRADES."
51Ø INPUT "IS THIS THE FIRST GRA
DE TO BE   RECORDED THIS WEEK";A
$
52Ø IF LEFT$(A$,1) = "Y" THEN X
= 1:N$(1) = NAM$:G$(1) = STR$(G)
:GOTO 73Ø
53Ø CLS:PRINT@Ø,STRING$(64,134);
"BE SURE THAT THE - GRADES - TAP
EIS IN THE RECORDER, THE TAPE IS
 REWOUND, AND THE RECORDER IS ON
 PLAY."
54Ø INPUT"WHEN YOU ARE READY PUS
H <ENTER>";A$:CLS:PRINT@Ø,STRING
$(224,134);"LOADING GRADES."
55Ø I = Ø:N$ = "":G$ = ""
56Ø OPEN "I", #-1, "GRADES"
57Ø IF EOF (-1) THEN 62Ø
58Ø I = I + 1
59Ø INPUT #-1, N1$,G1$
6ØØ N$(I) = N1$:G$(I) = G1$
61Ø GOTO 57Ø
62Ø CLOSE #-1
63Ø N$(I+1) = NAM$:G$(I+1) = STR
$(G):X = I + 1
64Ø CLS:PRINT@Ø,STRING$(224,134)
;:INPUT"DO YOU WANT A LIST OF AL
L GRADES";A$
65Ø IF LEFT$(A$,1) <> "Y" THEN 7
3Ø
66Ø INPUT"DO YOU WANT 1.CRT OR 2
.HARDCOPY LISTING";A:IF A = 2 TH
EN 7ØØ
67Ø CLS:PRINT"LIST OF GRADES"
68Ø FOR I = 1 TO X:PRINT N$(I),G
$(I):NEXT I
69Ø INPUT "PRESS <ENTER> TO CONT
INUE.";A$:GOTO 73Ø
7ØØ CLS:PRINT@Ø,STRING$(224,134)
;"PRINTING GRADES."
71Ø PRINT#-2, CHR$(14);"LIST OF
GRADES";CHR$(1Ø);CHR$(1Ø)
72Ø FOR I = 1 TO X:PRINT#-2, N$(
I),G$(I):NEXT I
73Ø CLS:PRINT@Ø,STRING$(64,134);
"NOW WE WILL RECORD GRADES TO
 TAPE.  REWIND THE TAPE,AND PUSH
 PLAY  AND RECORD."
74Ø INPUT"WHEN YOU ARE READY PUS
H <ENTER>.";A$
75Ø MOTORON:FOR V = 1 TO 25ØØ:NE
XT V:MOTOROFF
76Ø OPEN "O", #-1, "GRADES"
77Ø FOR I = 1 TO X
78Ø N1$ = N$(I):G1$ = G$(I)
79Ø PRINT#-1, N1$,G1$
8ØØ NEXT I
81Ø CLOSE #-1
82Ø CLS:PRINT@Ø,STRING$(224,169)
;"THE RECORDING OF GRADES IS
  FINISHED.":END
```

Listing 2:

```
1Ø REM THIS PROGRAM LISTS GRADES
 RECORDED BY THE SPELLING TEST P
ROGRAM
```

GRAPHICS

16K ECB 32K ECB

Lo-Res Graphics For The 'ASCII-ing'

Jim Schmidt

Hi-Res graphics are all the rage. And CoCo is certainly no slouch in that area. But let's not forget that Lo-Res graphics are very useful, too. In fact, there are some very distinct advantages to using them for certain applications. Particularly useful are the Lo-Res ASCII block graphics characters. The what?

For instance, try this. *POKE 1234, 255*. That orange rectangle that has magically appeared on your screen is ASCII graphics character 255 (ASCII Code 255). Since it has appeared on your text screen which resides from 1024 to 1535 (decimal), then obviously here is one kind of graphics that can be mixed with text. And since it uses ASCII Codes, it can be "massaged" in a program arithmetically.

ASCII (American Standard Code for Information Interchange) Codes are an industry attempt to put some standardization into data recognition between different computers and among data handling peripherals, like printers. Between ASCII Code 32 and 127 things are pretty much standard. Below 32 and above 127, all bets are off. Our little gem, the CoCo, uses the codes above 127 (128-255) to afford us with a plethora of colorful little graphics block characters. These "characters" have no direct keyboard access as do the letters and numerals. We can *POKE* them or *PRINT* them to the screen as with any other character using their ASCII Code in a *POKE* or by using *CHR$* in a *PRINT*. We can use the ASCII Codes in arithmetic expressions to derive other ASCII Codes. Because the ASCII Code is numeric, we can randomize it and display the result (as I did in the title screen in the program that accompanies this article).

"OK," you ask, "what do these guys buy me?" I'm glad you asked! Here are a few of the uses I've put them to:

1) Title and trailer screens
2) Game play boards
3) Maps for games/Adventures
4) Borders around text screens
5) Emphasis or eye-catchers
6) Graphs or charts

Since block graphics are so easy to use, I (being devoutly lazy) use them. However, a text screen full of block graphics is, in reality, a matrix of 32 characters by 16 lines. Trying to figure out where to *POKE* what character in a full screen can be a bit much. I decided to let CoCo help me design and create these screens. I wanted a program that would:

1) Create screens a line (32 characters) at a time
2) Create each line a byte at a time
3) Create each line a group of bytes at a time or a line of all one byte
4) Copy the previous line with one keystroke
5) Copy any previously created line
6) Modify any previously created line(s)
7) Display the screen at any point in its creation
8) Save finished screens to tape or disk for later load/modification
9) Generate a screen and driver program and save to tape or disk
10) View the ASCII blocks to aid in selecting appropriate blocks for the screen construction

I like to rough out a screen first by sketching it on a grid of 32 x 16 squares. I hate drawing grids, so if you have a printer, then Listing 1 is a grid drawing program that should work with any printer at 10 characters per inch.

The Screen Generator — How To Use It

The following refers to Listing 2, the Block Graphics Generator (BGG). After you have roughed out your screen on a grid or otherwise, the next step is to select the graphics characters for each line. You can then note them on the grid, although you will have to write small. From the menu, select *HELP*. You will then be prompted to enter a number from one to 255. This is then the number of the ASCII character from which the display will begin showing you what the various ASCII characters look like. The display will continue until it reaches character number 255 or you press 'Q' for quit. 'P' will pause the display, 'R' will resume after a pause. A single graphic character will be shown next to the ASCII Code for that character and a line of the characters will display to give you some perspective. All 255 ASCII characters can be displayed, but the color characters begin at 128.

At this point you should know what you want to create and the ASCII values that go into each line to make up your screen. It is now a matter of using the BGG to actually create the screen line by line.

The Functions

(A)LL — Creates a line of 32 (ALL) of the same code

(B)YTE/BYTE — Creates a line one byte at a time keeping you informed where you are in the line (used for detailed areas)

(G)ROUP/GROUP — Creates a line in 'hunks' of the same code and tracks your progress

(R)EPEAT — Copies the immediately previous line

(C)OPY — Copies the requested previously created line (by line number)

(M)ODIFY — Lets you change any existing line

(D)ISPLAY — Lets you see how your screen looks at any point in its creation

(S)AVE — Actually save or load, lets you store a completed screen for later retrieval/modification. Build a library of screens for later customization.

(P)ROGRAM — Will generate a driver program for your completed screen and save the program with your screen in Data statements. This program is saved in ASCII format just as if you had done so with the *,A* option of *SAVE* and *CSAVE*. *LOAD* or *CLOAD* and run normally. The line numbers generated are very high so that this program can be appended or merged behind your program within which you wish to use this screen, Two versions are available, with or without "visibility" as the screen is drawn. Here is an area where the more astute of you might wish to make some changes to my program. For instance, you might prefer to have your screen loaded into string variables and *PRINT* them. This is the fastest way to display your screen, but you will have to fight the text scroll routine to do it. (Hint: Print the first 15 lines and *POKE* the 16th. Be sure to use a ';' at the end of the *PRINTed* lines.)

(H)ELP — This will display the characters and their ASCII Codes.

Some Final Notes

Let your imagination go. Put color and style in your programs. Be user-friendly to your favorite user — you.

After you have saved your completed screen with or without driver logic, you will be given the option of clearing memory or not. You may want several versions of the same basic screen saved. And by replaying 'N' to the above prompt, you can continue to modify and save the same screen as often as you wish. Actually, the BGG is one big loop. It never ends until you hit BREAK or Reset. But, if you respond 'Y' to the prompt, you begin again at the menu with cleared memory (the computer's, not yours).

Attention 16K/tape users: Because of the large string space requirement, to make BGG fit in 16K, delete Lines 100-760, 1000, and 30000-30800. Also, you must PCLEAR1. I'm afraid that 16K *and* disk won't fly because of the disk buffers. Maybe one of you can chop it down enough to run on 16K disk, but I frankly haven't had much luck doing so.

I am greatly interested in seeing any screens of unusual interest or usage that you may create. I'd appreciate hearing from you and seeing such screens. Also, like all programs, BGG is never finished. Let me hear from you if you enhance BGG in some nifty way.

A word on using these screens in your programs. Please don't think that the screens created by BGG can only be static. There are several ways of "updating" a screen dynamically while your program is running. *POKE* and/or *PRINT* to it. Use several screens in data statements and bounce around among them. Animation it is not. Eye-appeal enhancer, interest raiser it can be. Perhaps more on these techniques in a later RAINBOW.

Have fun!

Listing 1:

1400	47
3300	209
END	95

```
100 '--LISTING ONE---
200 '
300 CLS:PRINT@100,"          M A T R
 I X":PRINT
400 PRINT"            COPYRIGHT (C)
 1983":PRINT
500 PRINT"             J. J. SCHMID
T":PRINT
600 PRINT"          ALL RIGHTS RESE
RVED"
700 FORX=1TO900:NEXT:CLS
800 CLS:PRINT@40,"MATRIX PRINT":
PRINT
900 PRINT"THIS PROGRAM REQUIRES:
"
1000 PRINT"       6 LINES PER INC
H"
1100 PRINT"       10 CHARACTERS P
ER INCH"
1200 PRINT"       PRINT HEAD POSI
TIONED JUST";
1300 PRINT"       UNDER PERFORATI
ON"
1400 PRINT"       9 1/2 X 11 INCH
 PAPER"
1500 PRINT"        (INCLUDING TEAR
 STRIPS)
1600 PRINT"       no VERTICAL TAB
S SET"
1700 PRINT:PRINT"        <ENTER> W
HEN READY"
1800 LINEINPUTQ$
1900 A=0
2000 FORZ=1TO33
2100 ST$=ST$+":  "
2200 NEXTZ
2300 FORX=1TO8
2400 PRINT#-2,""
2500 NEXT
2600 PRINT#-2,"                  GRA
PHICS SCREEN GENERATOR DESIGN MA
TRIX":PRINT#-2,"":PRINT#-2,""
2700 PRINT#-2,"POSITION==========
=>1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
2 2 2 2 2 3 3 3"
2800 PRINT#-2," 1 2 3 4 5 6 7 8
9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
5 6 7 8 9 0 1 2     LINE NO."
2900 PRINT#-2,""
3000 FORY=1TO16
3100 A=A+1
3200 PRINT#-2,STRING$(65,"-")
3300 PRINT#-2,ST$;:PRINT#-2,"   <
===    ";A
3400 NEXT
3500 A=0
3600 PRINT#-2,STRING$(65,"-")
3700 PRINT#-2,""
3800 FORX=1TO18
3900 PRINT#-2,""
4000 NEXT
4100 CLS:PRINT@204,"AGAIN???"
4200 R$=INKEY$:IFR$=""THEN4200
4300 CLS
4400 IFLEFT$(R$,1)="Y"THENRUNELS
EEND
```

2000	148	15510	85
4300	20	16900	206
6000	5	18700	203
7900	108	19800	215
9900	229	30000	209
11900	58	END	131
13700	236		

Listing 2:

```
100 '--LISTING TWO---
```

```
200 '--BLOCK GRAPHICS--
300 '--SCREEN BUILDER--
400 '
500 '--A UTILITY PROGRAM--
600 '
700 '--BY JIM SCHMIDT
710 '      196A ARLENE CT.
720 '      WHEELING, IL.
730 '              60090
733 '
740 '--COPYRIGHT (C) 1983--
750 '--ALL RIGHTS RESERVED--
760 '
1000 CLEAR4200
1050 CLSØ:GOSUB30000
1500 DIML$(16):DIMP$(32)
1600 CLS:L=1:GOTO1800
1700 L=L+1:CLS:IFL=17 THEN 7400
1800 PRINT@5,"":PRINT@33,"aLL (A
SCII #)":PRINT@48,"bYTE/BYTE"
1900 PRINT@97,"gROUP/GROUP":PRIN
T@112,"rEPEAT LAST LINE"
2000 PRINT@161,"cOPY A LINE":PRI
NT@176,"mODIFY A LINE"
2100 PRINT@225,"dISPLAY":PRINT@2
40,"sAVE/LOAD SCREEN"
2200 PRINT@289,"pROGRAM SAVE":PR
INT@304,"hELP (ASCII)"
2300 PRINT@359,"ENTER YOUR CHOIC
E"
2400 IF L<17 THENPRINT@424,"NEXT
 LINE IS ";L
2500 A$=INKEY$:IF A$="" THEN 250
Ø
2600 IF A$ ="A" OR A$="B" OR A$=
"C" OR A$="G" OR A$="H" OR A$="S
" OR A$="D" OR A$="M" OR A$="R"
OR A$="P" THEN SOUND169,1:SOUND1
69,1
2700 IF L<17 THEN IF A$="B" THEN
 3800
2800 IF L<17 THEN IF A$="R" THEN
 5900
2900 IF L<17 THEN IF A$="A" THEN
 6400
3000 IF L<17 THEN IF A$="C" THEN
 8500
3100 IF A$="D" THEN 7400
3200 IF A$="M" THEN 12900
3300 IF A$="S" THEN 15450
3400 IF L<17 THEN IF A$="G" THEN
 10900
3500 IF L=17 THEN IF A$="P" THEN
 17200
3600 IF A$="H" THEN 9700
3700 GOTO2500
3800 P=1:CLS
3900 CLS:PRINT@1,"":PRINT"LINE="
;L;"COLUMN=";P
4000 PRINT:PRINTL$(L):PRINT
4300 PRINT"ENTER ASCII VALUE  WA
NTED OR    <ENTER> TO REPEAT LAS
T ONE."
4400 PRINT
4500 INPUT"CHARACTER ASCII VALUE
";CH$
4600 SOUND222,1:IF P=32 THEN SOU
ND222,3
4700 IF CH$="" THEN 5100
4800 IF LEN(CH$) <3 THEN CH$=STR
ING$(3-LEN(CH$),"Ø")+CH$
4900 NN=VAL(CH$)
5000 IF NN <1 OR NN>255 THEN PRI
NT"INVALID ENTRY":SOUND7,7:FORH=
1TO300:NEXT:GOTO3900
5100 IF CH$="" AND LEFT$(L$(L),3
)="" THEN PRINT@490,"NO ENTRY YE
T":SOUND7,7:FORH=1TO400:NEXT:GOT
O3900
5200 IF CH$="" THEN CH$=SA$:GOTO
 5400
5300 SA$=CH$
5400 L$(L)=L$(L)+CH$
5500 IF P=32 AND BR$="Y" THEN BR
$="":L=SL:GOTO1700
5600 IF P=32 THEN 1700
5700 P=P+1
5800 GOTO3900
5900 L$(L)=L$(L-1)
6000 IF L$(1)="" THEN PRINT@359,
"NO VALID LINE YET":SOUND7,7:FOR
 H=1TO300:NEXT:CLS:GOTO1800
6100 PRINT@359,"  LINE REPEATED
         ":SOUND222,1
6200 FORH=1TO400:NEXT
6300 GOTO 1700
6400 CLS:PRINT@128,"":INPUT"ENTE
R ASCII CODE FOR ENTIRE LINE";CH
$
6500 IF LEN(CH$)<3 THEN CH$=STRI
NG$(3-LEN(CH$),"Ø")+CH$
6600 IF LEN(CH$)>3 THEN PRINT"
  TOO LONG":SOUND7,7:FOR H=1TO40
Ø:NEXT:GOTO 6400
6700 IF VAL(CH$)>255 OR VAL(CH$)
<1 THEN PRINT"      INVALID ENTRY
":SOUND7,7:FORH=1TO400:NEXT:GOTO
6400
6800 FOR TY=1 TO 32
6900 L$(L)=L$(L)+CH$
7000 NEXT
7100 PRINT"           LINE BUILT":SO
UND222,6
7200 FORH=1TO400:NEXT
7300 GOTO1700
7400 CLS:PS=1:L=1:BB=1024:EB=105
5
7500 FOROL=1TO16:FORIL=BB TO EB:
IFL$(L)="" THEN 9600:BY$=MID$(L$
(L),PS,3):POKEIL,VAL(BY$):PS=PS+
3:NEXT:EB=EB+32:BB=BB+32:PS=1:L=
L+1:NEXT
7600 FORIL=BB TO EB
7700 IFL$(L)="" THEN 9600
7800 BY$=MID$(L$(L),PS,3)
7900 POKEIL,VAL(BY$)
8000 PS=PS+3:NEXT
8200 EB=EB+32:BB=BB+32:PS=1:L=L+
1:NEXT
8400 NM$="Y":FORH=1TO2500:NEXT:P
RINT@224,"   THE SCREEN IS READY
 TO SAVE":FOR H=1TO50:SOUND222,1
:NEXT:CLS:GOTO1800
8500 CLS:PRINT:PRINT:PRINT"
     NEXT LINE IS ";L
8600 PRINT
8700 IF L=1 THEN PRINT"         N
O LINES EXIST YET":SOUND7,7:FORH
=1TO400:NEXT:CLS:GOTO1800
8800 INPUT"LINE # OF LINE TO BE
COPIED";LN
8900 PRINT
9000 IF LN=>L OR LN>16  THEN PRI
NT" THAT LINE DOES NOT EXIST":SO
UND7,7:FORH=1TO400:NEXT:CLS:GOTO
1800
9100 L$(L)=L$(LN)
9300 PRINT:PRINT"          LINE CO
PIED":SOUND222,6
9400 FORH=1TO400:NEXT
9500 GOTO1700
9600 FORH=1TO2500:NEXT:CLS:GOTO1
800
9700 CLS:PRINT:PRINT"ENTER Ø TO
RETURN TO MENU"
9900 INPUT"START AT NNN";ST
9950 IFST=Ø THENCLS:GOTO1800
10000 IFST >255 THEN PRINT"255 M
AXIMUM":SOUND7,7:FORDE=1TO400:NE
XT:CLS:GOTO9700
10050 CLS
10100 FOR H=ST TO 255
10300 PRINT@264,"PRESS Q TO QUIT
"
10302 PRINT@101,"PRESS P TO PAUS
E DISPLAY"
10303 PRINT@133,"PRESS R TO RESU
ME"
10320 Q$=INKEY$
10500 IFQ$="Q" THEN CLS:SOUND222
,6:GOTO1800
10510 IFQ$="P"THENGOSUB 40000
10520 PRINT@200,"ASCII ";ST;"= "
;:POKE1235,H
10530 FORLL=1344TO1375:POKELL,H:
NEXT
10600 FOR HH=1TO1000:NEXT
10700 ST=ST+1
10800 NEXT:CLS:GOTO1800
10900 P=1
```

```
11000 CLS:PRINT:PRINT:PRINTL$(L)
11200 PRINT"BUILDING LINE NO ";L
:PRINT
11300 PRINT"NEXT COLUMN IS ";P:P
RINT
11400 PRINT"POSITIONS REMAINING=
";(32-P)+1
11500 PRINT
11600 INPUT"ASCII #";CH$:SOUND22
2,1:IF P=32 THEN SOUND222,1
11700 PRINT
11800 IFCH$=""THEN 11000
11900 IF LEN(CH$)<3 THENCH$=STRI
NG$(3-LEN(CH$),"Ø")+CH$
12000 NN=VAL(CH$)
12100 IFNN<1 OR NN>255 THENPRINT
"INVALID ENTRY":SOUND7,7:FORH=1T
O400:NEXT:GOTO11000
12200 INPUT"HOW MANY";HM
12250 SOUND222,1
12300 IF HM> (32-P)+1 THEN PRINT
"TOO MANY":SOUND7,7:FORH=1TO400:
NEXT:GOTO12200
12400 FORLO=1TOHM:L$(L)=L$(L)+CH
$:P=P+1:NEXT
12800 IF P=33 THEN 1700 ELSE 110
ØØ
12900 CLS:PRINT:PRINT:P=1
13000 INPUT"NUMBER OF LINE TO MO
DIFY";NL
13100 IF NL=Ø OR NL>16 THEN CLS:
GOTO1800
13150 IF L$(NL)="" THEN PRINT@13
Ø,"NO SUCH LINE EXISTS":SOUND7,7
:FORH=1TO400:NEXT:CLS:GOTO1800
13200 PRINT
13300 PRINT"rEVIEW OR dELETE AND
 DO OVER?"
13400 RD$=INKEY$:IF RD$="" THEN
13400 ELSE SOUND222,6
13500 IF RD$<>"R" AND RD$<>"D" T
HEN 13400
13600 IF RD$="D" THEN SL=L-1:L=N
L:BR$="Y":L$(L)="":GOTO3800
13700 SL=L:ML$=L$(NL)
13800 FOR LO=1 TO 96 STEP 3
13900 OB$=MID$(L$(NL),LO,3)
14000 CLS:PRINT
14100 PRINT"COLUMN NUMBER= ";P
14200 PRINT"ENTER Q TO QUIT CHAN
GES"
14300 PRINT"THIS BYTE IS NOW =";
OB$
14400 INPUT"<ENTER> IF OK OR ENT
ER NEW VALUE";NB$
14500 SOUND222,6
14600 IF NB$="" THEN P=P+1:GOTO
15300
14700 IF NB$="Q" THEN 15400
14800 IF LEN(NB$)<3 THEN NB$=STR
```

```
ING$(3-LEN(NB$),"Ø")+NB$
14900 NN=VAL(NB$)
15000 IF NN<1 OR NN>255 THEN PRI
NT"INVALID ENTRY":SOUND7,7:FORH=
1TO400:NEXT:GOTO13800
15100 MID$(ML$,LO,3)=NB$
15200 P=P+1
15300 NEXT
15400 L$(NL)=ML$:L=SL:CLS:GOTO18
ØØ
15450 CLS:PRINT:PRINT:PRINT"
      dISK OR tAPE??":PRINT:PRINT
15460 QU$=INKEY$:IFQU$=""THEN154
6Ø
15470 IFQU$<>"D" AND QU$<>"T"THE
N15450
15480 IFQU$="D"THENDV=1ELSEDV=-1
15500 PRINT:PRINT"             READY
YOUR DRIVE":PRINT
15510 FORDE=1TO1000:NEXT
15600 CLS:PRINT@137,"sAVE OR 1OA
D??"
15700 SL$=INKEY$:IF SL$="" THEN
15700
15800 SOUND222,2:IF SL$<>"S" AND
 SL$<>"L" THEN 15600
15848 PRINT
15850 IF SL$="S" AND L<>17 THENP
RINT"NOT A COMPLETE SCREEN":SOUN
D7,7:FORH=1TO400:NEXT:CLS:GOTO18
ØØ
15900 IF SL$="S" THEN SL$="O" EL
SE SL$="I"
16000 PRINT:INPUT"FILENAME";FI$
16100 FI$=LEFT$(FI$,8)
16200 PRINT@294,"  OPENING FILE"
16300 OPEN SL$,#DV,FI$
16400 PRINT@293," <ANY KEY TO BE
GIN>"
16500 Q$=INKEY$:IF Q$="" THEN 16
500 ELSE SOUND 222,2
16600 FORL=1TO16
16700 IF SL$="I" THEN INPUT #DV,
 L$(L):PRINT@294,"      READING "
:FOR RT=1TO100:NEXT
16800 IF SL$="O" THEN PRINT#DV,L
$(L):PRINT@294,"     WRITING ":FO
RWT=1TO100:NEXT
16900 PRINT@296,"               "
17000 NEXT
17100 CLOSE#DV
17105 IF SL$="I" THEN17140
17110 PRINT"WANT STORAGE CLEARED
 <Y/N>?"
17120 Q$=INKEY$:IFQ$=""THEN 1712
Ø
17130 IFQ$="Y" THEN RUN
17140 L=17:CLS:GOTO1800
17200 IF ND$="" THEN DIM DA$(32)
17210 ND$="Y"
```

```
17300 CLS:PRINT@200,"hIDDEN OR v
ISIBLE?"
17310 OP$=INKEY$:IFOP$=""THEN173
1Ø
17320 IFOP$<>"H"ANDOP$<>"V"THEN1
7310
17330 CLS:PRINT@200,"PLEASE WAIT
..."
17400 NP=1:D=1:L=1
17500 FORHH=63000 TO 63015
17600 FORH=1TO32
17700 NB$=MID$(L$(L),NP,3)
17800 IF SW$="Y" THEN 18200
17900 FU$=STR$(HH)
18000 UF$=RIGHT$(FU$,5)
18100 DA$(D)=UF$+" DATA":SW$="Y"
18200 IF H<32 THENDA$(D)=DA$(D)+
NB$+CHR$(44) ELSE DA$(D)=DA$(D)+
NB$
18300 NP=NP+3
18400 NEXT
18500 NP=1:L=L+1:D=D+1:SW$="N"
18600 NEXT
18650 IFOP$="H"THENI1$="63016 PC
LS:SCREEN1"ELSEI1$=""
18700 I2$="63017 BB=1024:EB=1055
"
18800 I3$="63018 FOR OL=1TO16"
18900 I4$="63019 FOR IL=BB TO EB
"
19000 I5$="63020 READ BY$"
19100 I6$="63021 POKEIL,VAL(BY$)
"
19300 I7$="63023 NEXT"
19400 I8$="63024 EB=EB+32:BB=BB+
32"
19500 I9$="63025 NEXT"
19550 IFOP$="H"THENIA$="63026 SC
REENØ"ELSEIA$=""
19575 IB$="63027 FORTI=1TO3000:N
EXT"
19580 SOUND234,1:FORDE=1TO50:NEX
T:SOUND234,1
19600 CLS:PRINT@72,"SUBROUTINE C
REATE":PRINT:PRINT
19610 PRINT"          tAPE OR dIS
K ?":PRINT:PRINT
19620 QU$=INKEY$:IFQU$=""THEN196
2Ø
19630 IFQU$<>"T"ANDQU$<>"D"THEN1
9600
19640 IFQU$="T"THENDV=-1ELSEDV=1
19700 INPUT"FILENAME";FI$
19800 IF QU$="T"THENPRINT@233,"R
EADY RECORDER"ELSEPRINT@233,"REA
DY DISK DRIVE"
19900 PRINT@294,"<ANY KEY WHEN R
EADY>"
20000 Q$=INKEY$:IF Q$="" THEN 20
000
```

```
20100 OPEN"O",#DV,FI$
20200 FORH=1TO16
20300 PRINT#DV,DA$(H)
20400 IF QU$="T"THENPRINT@294,"
   WRITING TAPE     ":FORWT=1TO10
0:NEXT
20410 IFQU$="D"THENPRINT@294,"
  WRITING DISK     ":FORWT=1TO100
:NEXT
20500 PRINT@294,"
       "
20550 FORWT=1TO100:NEXT
20600 IFQU$="T"THENPRINT@294,"
  WRITING TAPE     "ELSEPRINT@294
,"    WRITING DISK      "
20700 NEXT
20800 PRINT#DV,I1$:PRINT#DV,I2$:
PRINT#DV,I3$:PRINT#DV,I4$:PRINT#
DV,I5$:PRINT#DV,I6$:PRINT#DV,I7$
:PRINT#DV,I8$:PRINT#DV,I9$:PRINT
#DV,IA$:PRINT#DV,IB$
20810 CLOSE#DV
20820 PRINT"WANT STORAGE CLEARED
 <Y/N>?"
20822 Q$=INKEY$:IF Q$="" THEN 20
822 ELSE SOUND222,1
20824 IFQ$="Y" THEN RUN
20900 CLS:GOTO1800
30000 FOR KK=1024TO1535
30100 VV=RND(125)
30200 POKE KK,VV+130
30300 NEXT
30400 FORDE=1TO1500:NEXT
30600 SOUND169,2:SOUND169,1
30710 SOUND169,1:PRINT@64,"    BL
OCK"
30720 SOUND169,1:PRINT@160," ":P
RINT@171,"GRAPHICS"
30730 SOUND169,2:PRINT@256," ":P
RINT@276,"GENERATOR"
30740 SOUND 169,1:PRINT@352," ":
PRINT@358,"BY - JIM SCHMIDT"
30750 PRINT@448,"         COPYRIGH
T (C) 1983 "
30795 FORDE=1TO1500:NEXT
30798 CLS
30800 RETURN
40000 QQ$=INKEY$:IFQQ$="R"THENRE
TURNELSE40000
```

EDUCATION NOTES

16K
ECB

Create A Calendar

By Steve Blyn

September is back to school time. Unlike the usually sluggish end of the school year, children are quite motivated to learn at this time. Although few will admit it, children often get bored by the end of the summer vacation and are glad to return to school.

It's a good idea to seize this moment of enthusiasm before it fades. One way is to explain some of the exciting upcoming events that you have planned for your children or students.

To help you accomplish this, we will illustrate a September calendar on your CoCo. We also will show a way to print a blank calendar form for any month on a printer.

Our program will draw an outline of the calendar for September on Lines 30 to 250. When you run this program you will notice several space size limitations on this calendar. These are due to the limitations of the screen size of our computer. There are, of course, no space limitations on the printer portion of the program.

Having only 32 spaces across horizontally and 16 vertical lines presents some space problems. Although some months have parts of six weeks, we could only fit five weeks on the screen. We had to include Sept. 1 above Sept. 8 in the line with the name of the month. The names of the days of the week also were compromised by being placed at the bottom of the screen. Please keep in mind that the other popular competitive computers currently sold would all present similar screen limitations.

Here are some of the ways that we have used the screen calendar portion of this program with children:

1) Review the September holidays. What are their dates? What days of the week are these?

2) Review any student birthdays or any class trips for the month.

3) How many Mondays or Wednesdays are there in this month? Are there more of a certain day than another? For example, are there more Wednesdays or Saturdays?

4) Which date is 10 days after Sept. 8? Which is 17 days after Sept. 4? Which date is 12 days before Sept. 29?

5) What day of the week was the last day in August? What day of the week will the first day in October be?

6) Which date is the fourth Thursday in September? Which is the third Monday?

7) How many more days is it until Freddy's birthday on Sept. 12 or Cheryl's on Sept. 26?

After the calendar appears on the screen, you may press 'E' to end the program or 'P' to printout a blank calendar form on your printer. This choice appears on Lines 260-270.

The remainder of the program is the printer routine. This appears on Lines 280-390. If you do not have a printer, the program need not be keyed in beyond Line 250.

We have included no printer control codes. This is because there is such a wide variety of printers that are used with CoCos. Each printer has its own accompanying control codes. We suggest that you first select the elongated mode on whichever printer you use. You will then get the largest calendar possible. This will fill up most of an 8 by 11 sheet of paper.

The blank form has no month indicated nor has it any days indicated. This is done to enable you to use it for any or all months. Filling in the dates is, of course, one of the child's activities.

Each month, a new form could be handed out to your child or students. If you have a large amount of students, it is best to have copies of an original made rather than to run off too many copies on your printer. Most of our printers are not really made for the heavy use of multiple copies.

Once the dates are filled in on the calendar, it is time to indicate the special events and birthdays of that month. September is a good example of a month with many special days. Among others, they include Labor Day, the beginning of Autumn, Rosh Hashana, and RAINBOWfest, Princeton, New Jersey!

We hope you and your children enjoy using these calendars. We, at Computer Island, would of course enjoy hearing of any other ways you can think of to use the calendar.

210 45
END 211

The listing:

```
10 REM"STEVE BLYN"
20 REM"COMPUTER ISLAND,NY,1984
30 CLS
40 A$=" "+STRING$(30,207)
50 B$=" *   *   *   *   *   *
*   *"
60 FOR T= 1 TO 5
70 PRINTA$:PRINTB$:PRINTB$
80 NEXTT
90 PRINT@27,"1";
100 PRINT@11,"september";
110 FOR T=2 TO 8:PRINT@34+N,T;:N
=N+4:NEXT
120 N=Ø
130 FOR T=10 TO 15:PRINT@133+N,"
*";T;:N=N+4:NEXT
140 PRINT@131,"9";
150 PRINT@157,"*";
160 N=Ø
170 FOR T=17 TO 22:PRINT@229+N,"
*";T;:N=N+4:NEXT
180 PRINT@227,"16";
190 PRINT@253,"*";
200 N=Ø
210 FOR T=24 TO 29:PRINT@325+N,"
*";T:N=N+4:NEXT T
220 PRINT@323,"23";
230 PRINT@349,"*";
240 PRINT@419,"30";
250 PRINT@480,"  SUN MON TUE WED
 THU FRI SAT ";
260 EN$=INKEY$
270 IF EN$="E" THEN 390 ELSE IF
EN$="P" THEN 280 ELSE 260
280 CLS:PRINT"WHAT IS THE NAME O
F THIS MONTH":INPUT M$
290 PRINT#-2.TAB(5) "CALENDAR FO
R ";M$
300 A$=" "+STRING$(29,"#")
310 PRINT#-2," "
320 PRINT#-2,A$
330 PRINT#-2,"  SUN MON TUE WED
THU FRI SAT"
340 FOR A= 1 TO 6
350 PRINT#-2,A$
360 FOR B= 1 TO 4:PRINT#-2,B$:NE
XT B
370 NEXT A
380 PRINT#-2,A$
390 CLS:END
```

CoCoConf JUNE '85 GOLD COAST QUEENSLAND

Plans for CoCo Conf continue.

Currently it appears that we will be able to hold Conference fees to about $40.00

There would be an additional cost for the Saturday night meal.

We plan to be able to announce full details next month.

In the mean time, you must have decided that you can think of no one that deserves recognition, because we haven't yet received any nominations for the Greg Wilson Award.

PLAN TO BE THERE!

The second in a five part series on . . .

Everything You Always Wanted To Know About The Color Computer

But Radio Shack Didn't Tell You

By Andy Kluck

Here is the second installment of my compilation of old and new information and techniques involving the Color Computer. This month's article features a method of speeding up tape I/O to about 2700 Baud and one more way to merge cassette programs.

The Memory Almost Full Condition

When BASIC's available free memory is almost used up, a strange condition sometimes occurs. It can occur accidently, such as inside a program, while entering program lines, or on purpose, by:

```
CLEAR 0:CLEAR MEM-50
```

When this happens, there is not enough stack space for BASIC to think straight, and any statement that requires evaluation of a numeric or string expression gives an OM Error. This prevents *SAVE, CSAVE, CLEAR0, PCLEAR 1,* and just about anything else that could restore control of the system short of *NEW*, from working. If it happens inside a program which uses one too many variables, a simple *CLEAR* may free up enough space to allow a *CLEAR 10:(C)SAVE "HELP!"*. Otherwise, if you don't have the program *CSAVEd*, just about the only way out is to *LIST* or *LLIST* one line (or more if necessary), delete it, make more space (i.e. *CLEAR 10*) and retype the line. To prevent this problem in the first place, it is helpful to monitor the value of *MEM* during a test run of the program and do whatever is necessary (reserve less string space in *CLEAR, PCLEAR* fewer pages, crunch the program, etc.) to keep it above 200.

RENUM

When using *RENUM*, there are several good reasons for saving the program on tape or disk first in case of problems during renumbering, as there are at least two different possible sources of trouble. First of all, *RENUM* without a liberal amount of free memory can cause a wrecked program or system crash, so a *CLEAR 10:PMODE 0,1:PCLEAR 1* is recommended first with long programs. There is also the case of illegal line numbers as in this example:

```
1 GOTO 2
2 GOTO 3
3 GOTO 64000
```

Running this program results in an SN Error in 30, since line numbers greater than 63999 are not accepted by BASIC. Attempting to *RENUM*ber with this program also causes an SN Error (without printing the line number since the error occurs in direct mode) and makes the program disappear — just list it. But there may be a way to recover. Saving and reloading the program, or just fixing its line pointers and doing a *CLEAR*, by:

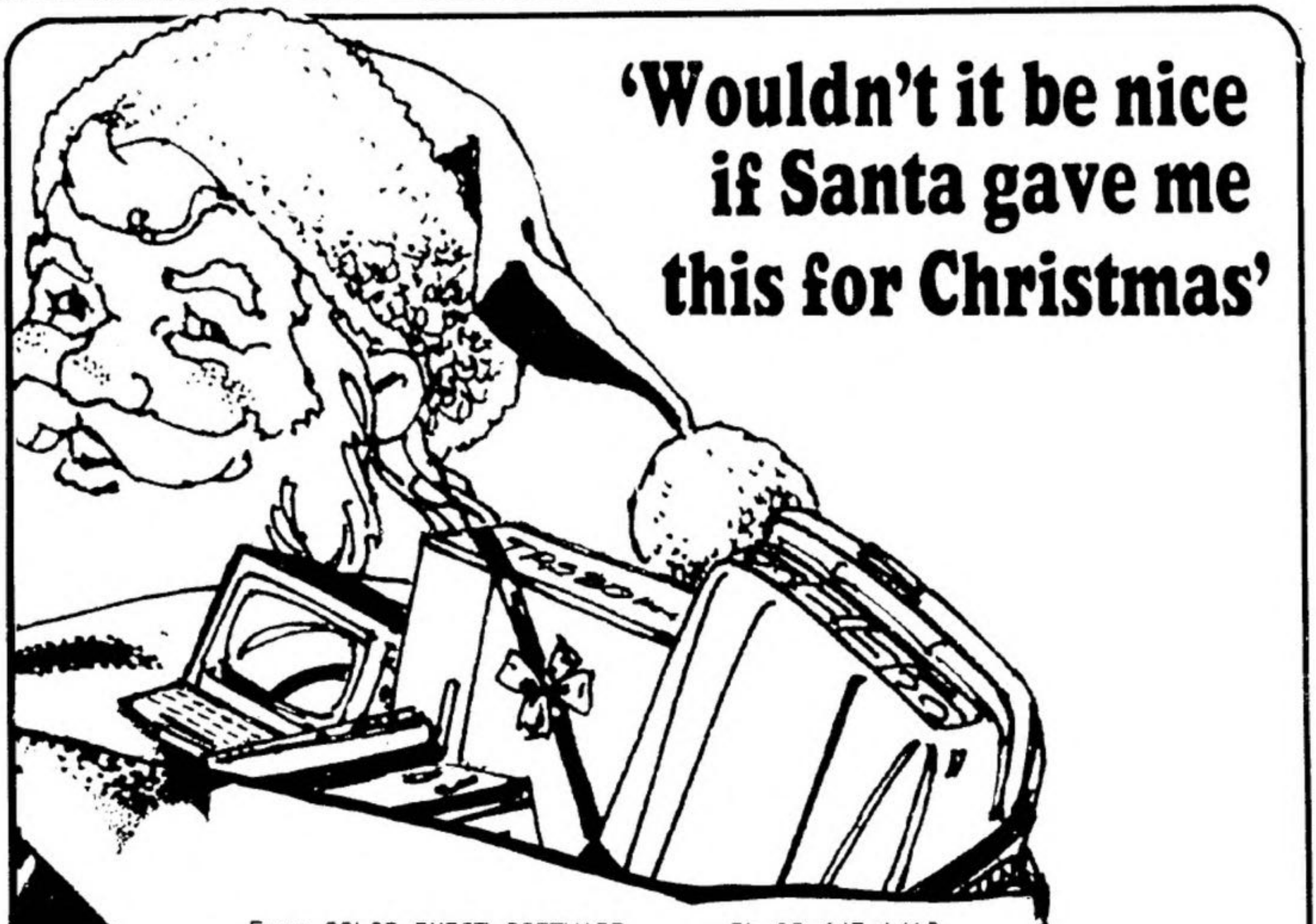
```
EXEC &HACEF:CLEAR
```

at this point will often bring all of it back except some of the line numbers, which are still replaced by internal codes. Now, if you fix the illegal line number and *RENUM* again, the program may be completely restored, if you're lucky.

Adjustments For High Speed Mode

To speed up execution of BASIC programs, it has been suggested that the SAM chip may be set to its Address Dependent mode by *POKE65495,0* and set back to normal speed by *POKE 65494,0* or pressing Reset. Note that not all Color Computers will work in this mode, so it should not be used (or at

'Wouldn't it be nice if Santa gave me this for Christmas'



From COLOR BURST SOFTWARE	Ph.02 467-1619	
FROG TREK	$17.00	□
SEA QUEST	$29.00	□
From BAYNE AND TREMBATH	Ph.057 86-8288	
16K BASIC COLOR COMPUTER	$169.00	□
From PARIS RADIO	Ph.02 344-9111	
HACKERS KIT FOR OS-9	$34.95	□
WORD PAK 80 COLUMN CARD	$200.00	□
From CASULA HOBBIES	Ph.02 602-8640	
81 CLASS HAND CRAFTED BRASS LOCO	P.O.A.	□
From NORTH QUEENSLAND COLOR SOFTWARE	Ph.077 73-2064	
VIP WRITER	$62.95	□
WHIRLYBIRD	$28.95	□
From SOFTWARE SPECTRUM	Ph.08 211-8763	
MUSICA	$41.95	□
MINI DISK DRIVES	$529.95	□
From RAINBOW MAGAZINES	Ph.075 51-0015	
SUBSCRIPTION TO AUSTRALIAN COCO/MICO	$29.00	□
SUBSCRIPTION TO RAINBOW/GOCO	$29.00	□

MARK THIS PAGE ACCORDINGLY AND LEAVE WHERE IT'LL BE SEEN.

Introducing RAINBOW WARE

This range of clothing is made here on the Gold Coast.

DESCRIPTION	SIZES	PRICE
V-Neck Crew Neck	12. 14. S. M. L. XL	14.95
Tank Tops	10. 12. 14. S. M. L. XL	13.85
Tennis Dress	S. M. L.	26.20
Hooded Zip-up Jacket	S. M. L.	31.95
Ladies Hooded Pullover Top	S. M. L.	28.50
Ladies Shorts	S. M. L.	17.95
Children's Tank Tops	6. 8. 10. 12. 14	12.80
Children's T-Shirts	8. 10. 12. 14	14.20

COLORS AVAILABLE white, pink, aqua sky blue yellow, hot pink, blac'

PLEASE SEND

ORDER FORM

________________ $ ________

________________ $ ________

________________ $ ________

________________ $ ________ AMOUNT ________

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

Mr □ Mrs □ Miss □ Initials □□ Surname □□□□□□□□□□□□

Subscription No. □□□□□□□

Address

□□□□□□□□□□□□□

□□□□□□□□□□□□□

□□□□□□□□□□□□□

□□□□□□□□□□ PC □□□

□ CASH

□ CHEQUE

□ POSTAL ORDER

□□□□□□□□□□□□□□□□

□ VISA BANKCARD □

________ Cardholder

UNDERSTAND THAT I CAN RETURN THESE GOODS WITHIN 7 DAYS FOR A FULL REFUND FOR ANY REASON

Now is the time to subscribe to Australian Rainbow/GoCo

Copies of monthly 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 $3.25 including postage and packing. Please enclose your cheque/postal order made payable to: Graham Morphett, PO Box 1742, Southport, 4215.

Send off the slip below to ensure that you get on the mailing list for DECEMBER

CUT OUT AND MAIL or PHONE BANKCARD NO. (075 51 0015)

Subscription Rates

AUSTRALIAN CoCo/MiCo			AUSTRALIAN RAINBOW/GoCo
□	Latest	3.25 per copy	□
□	6 months	$18	□
□	12 months	$29	□

Back Issues

□ 1st Issue '83	$3 per copy	□ 1st Issue '81

Tapes

Rainbow on Tape (program listings) $12 for month of □

Please note that RAINBOW on TAPE is issued irregularly

Tape Monthly

CoCoOz			MiCoOz
□	Latest	$6	□
□	6 months	$32	□
□	12 months	$58	□

Blank tapes 12 for $18 or $1.70 ea. □

Cassette Cases 10 for $5 □

BOOKS

Byte Elementary	$5.95	□
Help Medium	$9.95	□
Facts Advanced	$11.95	□
MiCoHelp Medium	$9.95	□
MiCo Exposed V. Advanced	$11.50	□

□□□□□□□□□□□□□□

□ VISA BANKCARD □

________________ Cardholder

□ CASH

□ CHEQUE

□ POSTAL ORDER

VISA

New Subscription □ Renewal □

Date ________________

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.

Mr □ Mrs □ Miss □ Initials □□ Surname □□□□□□□□□□□□

Address

□□ PC □□□□

Telephone Number STD Code □□□□ Local Number □□□□□□□□□□□

CUT OUT AND MAIL or PHONE BANKCARD NO. (075 51 0015)

DID YOU READ?

'RAINBOW July '83

- Move over Walt Disney
- The Snails Strike Back.
- Las Vegas CoCo?
- NsNs Nyqcus
- That Key Does What?
- Plotting 3-D Graphs

RAINBOW August '83

- Visual Presentation
- Circles and Arcs
- Regular Polygons
- Screen Displays
- MiniDOS
- Ring It Up.

RAINBOW Sept. '83

- Autodesigner
- Disk Directory
- Audio Tutorials
- Home Budget
- Icosahedron
- Computer Concept

Get 'em while they last — there'still afew copies left

Every issue is NEW until you 've READ IT!

MICOOZ THIS MONTH

SPELLIT	Graham Pollock
ALIEN	Grahame Burke
MOONSCAN	Vic Bodero
A.B.C.	Allan Bridges
PEGLEG	Tom Lehane
ALPHABET SORT	Chris Thomas

CoCoOz This Month

OS8	Paul Humphries
MAZE	Max Betteridge
KALEIDOSCOPE	John Curtis
FRENCH POLISH	Tino & Bob Delbourgo
COMICAL CHARACTERS	Glen Kentwell
SATURDAY NIGHT FEVER	G. Adamczewski
TIME CALCULATION	Robb Webb
DISK LOAD	Morrie Singer
THREE CIRCLE PUZZLE	Daniel Delbourgo
KARBAR	Keith Roach

least be a user option) in programs to be distributed to others. It appears, however, that most of them will work at the high speed if certain modifications are made as described on Page 78 of the January 1983 RAINBOW. Some modifications are almost always necessary to get systems with the disk interface installed to work at the high speed. Also, normal low speed should always be selected during disk access to prevent strange problems that can otherwise occur. In the high speed mode, the processor runs at twice its normal speed when accessing ROM. Since BASIC accesses RAM as it runs, the actual measured speed is less than twice normal, depending on the program. The printer output routine runs mostly in ROM, so the time constants for the Baud rate and carriage return delay generally need to be doubled. Cassette tapes made at the high speed have a data rate of about 2,700 Baud, almost twice the normal rate and five times as fast as the Model I: however, they usually do not load normally even at the high speed because the tape read routine accesses RAM more often than the write routine and, therefore, runs slower and gets out of sync with the tape. But by changing the bytes that control the reading of tapes, it is possible to read tapes made at both speeds at either speed:

To read normal tapes at low speed
POKE 143,18:POKE 144,24:POKE 145,10
(These are the normal values.)

To read normal tapes at high speed
POKE 143,29:POKE 144,30:POKE 145,15

To read 2,700 Baud tapes at low speed
POKE 143,8:POKE 144,24:POKE 145,4

To read 2,700 Baud tapes at high speed
POKE 143,13:POKE 144,24:POKE 145,6

Tapes made at the high speed may not be readable on all systems, but I have had good luck at a volume level of about eight with the standard recorder. Recently it was suggested that tapes made in the high speed mode could be loaded by using *POKE 65497,0* and no adjustments to locations 143 to 145, but I have found this method less reliable; besides, this disables the dynamic RAM's refresh cycles, sometimes resulting in RAM cells "forgetting" at random. Remember that pressing Reset sets the SAM to its normal speed but does not reset the tape read parameters; this can cause confusion when you reset the computer and forget to adjust these values and all the tapes stop working.

Tape Filenames

Whenever an empty string ("") is used as the tape filename for an input operation, BASIC acts as if no filename was specified and uses the next file on the tape. This is useful with the *OPEN* statement when the name of the file is unknown, and also with *CLOADM* to load a file using an offset without typing the filename.

CSAVEM, CLOADM And Offsets

The index of at least some versions of the CBASIC manual lists a command called *CSAVEM*, which supposedly will "write out a machine language file" and has the syntax: *CSAVEM X,4E,6F,5F*.

But don't believe them. First of all, Color BASIC doesn't have a *CSAVEM;* it's an Extended BASIC command. The Extended BASIC manual says the same thing, but the command as given still doesn't work because all four of the arguments given are incorrect. For a while it was assumed that Extended BASIC didn't have a *CSAVEM* either, but eventually the correct syntax was discovered, either through experimentation or disassembly of the ROM:

CSAVEM "filename", start address, end address, exec address

The Disk BASIC manual carries on the great tradition by giving an incorrect example for the *SAVEM* command. At least now they use a string for the filename. The arguments for *CSAVEM* are not hexadecimal numbers; they are standard numerical expressions. Of course, if you only know the addresses in Hex, you can use the &H prefix, which evaluates to such an expression. To load one of these files at a different address than it was made at, an offset is used, and the file is loaded at its original address plus the offset. To load a file at an address greater than the address it was made at:

(C)LOADM "filename", new address old address

To load a file at an address lower than the original, a wrap-around effect is used:

(C)LOADM "filename",new address-old address+65536

and $10000 is subtracted from the address; i.e., an offset of $F000 causes the file to be loaded $1000 below its original address.

EXEC

When a file is *CLOADMed* or *LOADMed*, the exec address from the file plus the offset is stored in the exec pointer at $9D. When *EXEC* is used without an argument, the routine addressed by the pointer is called as a subroutine. If *EXEC* is used with an argument, the argument is stored in the exec pointer for use by the next *EXEC*. When making a machine language file which is not to be executed, such as a block of data or a saved picture, an exec address of $B44A may be used, since this is the address that the pointer is set to when BASIC is started and is the address of BASIC's FC Error routine.

ASCII Files And The Cassette Merge

The *SAVE* and *CSAVE* commands support two formats for the output file. The tokenized or compressed form is the most common. It consists of an exact dump of BASIC's program area, and since command words and functions are replaced by one or two byte tokens, it usually produces shorter files. The ASCII or listed format is invoked by commands of the form: *(C)SAVE "filename",A*.

Since ASCII files are made by simply opening the output file and listing the program into it, they can be accessed from BASIC as data files or read directly into any text editor that doesn't use its own file format. The *LOAD* and *CLOAD* routines test the input file for which type it is and act accordingly. Tokenized files are read back into the program area, the proper pointers are set, and the program's line pointers are fixed according to its new position in memory. The ASCII file loader does a *NEW*, opens the file for input, and jumps to the same "idle loop" that normally inputs lines from the keyboard. Often it is helpful to be able to combine lines from two programs. Disk BASIC provides this utility with the *MERGE* command, which operates similarly to *LOAD* except it only accepts ASCII files and doesn't call *NEW* first. Several methods have been suggested for merging two cassette programs together, often by setting the "start of program" pointer to the end of the first program to load the second; however, most of these require several *POKEs* and *PEEKs* or a machine language routine, and part of the procedure has been omitted in some accounts so that if the end of the first program happens to fall on a page boundary, the user is required to *POKE* a 2 into the "start of program" pointer. Besides, this process requires that the line numbers of the first program be lower than those of the second. By emulating the *MERGE* command in cassette BASIC, these problems are solved. The process of opening the file, setting the device number to -1, and calling the idle loop can be accomplished in one line: *OPEN "I",-1,"filename":POKE 111,255:EXEC 44156* for the program "filename", or *OPEN "I", -1,"":POKE 111,255:EXEC 44156* for the next file on the tape. Remember that this only works with an ASCII file. If any line numbers exist in both programs, the lines in the file will replace those in RAM.

SKIPF

According to the Color BASIC manual, *SKIPF* is used to position the tape to the end of the last program. I have found it just as useful as a method of verifying that a file has been written without errors. Since the tape read routines used by *SKIPF* test the checksums of each data block, attempting to *SKIPF* a bad file will cause an I/O Error. In the case of a BASIC or machine language program, the user can then rewrite the file. *SKIPF* is more versatile than the Level II equivalent *"CLOAD?"*, which only works with BASIC files of which an identical copy is still in RAM.

READ and INPUT

Data to be read or inputted may not be a variable or arithmetic expression, but it may be a Hex or Octal constant in Extended BASIC. Strings to be read or input may be enclosed in quotation marks, which allow leading and trailing blanks, commas, and colons to be included in the string:

```
10 READ A,A$:DATA &H3FF,"
COMMA, COLON: "
```

There are several standard methods of speeding up Microsoft BASIC programs. First of all, *GOTOs* and *GOSUBs* work faster if the line referenced is either near the beginning of the program or immediately after the line with the *GOTO* or *GOSUB*. Hex and Octal constants are evaluated much faster than decimal ones. Programs will run faster if the most often used variables are created first. Also, every time a simple variable is created, all the arrays are moved to make room for it, so if large arrays are used, all simple variables should be declared before the arrays are dimensioned. Finally, Color BASIC 1.2 and Disk BASIC 1.1 have a new interpret loop that only scans the keyboard before each statement if at least one key is down. With either of these ROMs or the equivalent in RAM, execution is speeded up by varying amounts depending on program content.

DIM

Besides dimensioning arrays, *DIM* may be used to create a list of simple variables. A program that declares all of its variables and then dimensions its arrays with a statement like *DIM A,B,C,I,X,Y,A$,B$,A(1000),B$(50)* will run faster than one that doesn't.

Relational and Logical Operators and IF/THEN

The Color BASIC manual gives a list of BASIC operators on Page 306 but doesn't define most of them. The relational operators ("=", ">", "<", ">=", etc.) with numeric operands give a value of -1 if the expression is true, or 0 if it is false. For example, *PRINT B>=C* gives -1 if B is greater than or equal to C, or 0 if B is less than C. Relational operators used with string operands compare them alphabetically. AND and OR convert each expression to a 16-bit integer and do the correct logical operation to get the result. For example, a binary 0111 *ANDed* with 1110 equals binary 0110:

```
PRINT 7 AND 14
```

gives 6= 0110 binary. The *NOT* operator has one operand and simply complements each bit. This has the effect of turning a -1 into a 0 or a 0 into a -1. According to the Color BASIC manual, IF/THEN "tests the relationship" and acts accordingly. Actually, IF simply evaluates a numerical expression and takes 0 as false and anything else as true. Therefore, *X= A=1 AND B>6: IF X THEN PRINT B* is the same as *IF A=1 AND B>6 THEN PRINT B* and *IF Y THEN 300* may be substituted for *IF Y<>0 THEN 300.*

NEXT

Like most Microsoft BASICs, Color BASIC allows *NEXT* without a variable to close the last loop entered. Also, statements of the form *NEXT X,Y,Z* may be used to close multiple loops.

INKEY$

INKEY$ does not simply return the key being pressed at the instant it is executed. Before each BASIC statement is executed, the keyboard is tested and if a new key is pressed (other than *SHIFT* @ or BREAK) its value is stored at $87. *INKEY$* tests this address, and if a key has been pressed, it returns a string with that character and stores a 0 in $87. Otherwise, *INKEY$* scans the keyboard again and if a new key is pressed, it uses it for the string. This sometimes causes *INKEY$* to eat a BREAK character and return a *CHR$(3)*. If you want to have a program stop and wait for the user to press a key, it is best to use a routine like:

```
60000 IN$=INKEY$
60010 IN$=INKEY$: IF IN$=""
THEN 60010
60020 IF IN$=CHR$(3)THEN STOP
60030 RETURN
```

Where the first *INKEY$* clears out any key that may have been previously pressed, and Line 60020 tests for the BREAK key.

Joystick Buttons

According to the manual, *PEEK (65280)* returns 255 or 127 if neither joystick button is pressed, 126 or 254 if the right button is pressed, or 125 or 253 if the left button is pressed. Obviously this cannot be correct when both buttons are pressed at once. To separate the button bits from each other as well as from the keyboard scan inputs which appear in the same byte, it is much better to use the *AND* operator with lines like:

```
10 IF (PEEK(65280) AND 1)=0
THEN ? "RIGHT BUTTON"
20 IF (PEEK(65280) AND 2)=0
THEN ? "LEFT BUTTON"
30 GOTO 10
```

RND

According to the Color BASIC manual, *RND* returns a random integer between one and its argument, which is supposed to be greater than one. This works fine; however, it is not the only way to use *RND*. For arguments in the range between zero and one, *RND* returns one. But *RND(0)* returns a number in the range of *0<=X<1*. This is the way "standard" BASIC defines *RND(0)*. For arguments less than zero, *RND* returns a value which is not random but actually is dependent only on the argument. More importantly, using *RND* with a negative argument sets Color BASIC's random seed value at $116-$119 according to the argument. A statement like *X=RND(-TIMER)* in Extended BASIC randomizes the random number generator much as the Level II RANDOM statement does. Note that Radio Shack's newsletter once recommended *A=RND(TIMER)* to do this, but this positive argument does not randomize anything. This feature can also be used to "unrandomize" the seed: *X=RND(-6)* 'or any negative constant executed at the beginning of a program or routine using *RND* will cause the same "random" number sequence each time the program or routine is run.

GRAPHICS UTILITY

32K
ECB

For *PMODE 4* Screen Enlargement, There's . . .

Joseph Kohn

Blowup

The graphics capability of the CoCo continues to be an essential and useful tool. The ability to create and manipulate graphics has been the subject of many articles and much software. The *BLOWUP* program described here should be a useful adjunct to most of these for the purpose of enlarging portions of the *PMODE 4* screen.

The program allows the user to load the *PMODE 4* screen from either disk or tape. Then by using the right joystick, a portion of this "source" screen can be selected for "blowing up," or enlarging. The enlarging ratio is 2 to 1, vertically and horizontally. The section of the source screen selected is 128 pixels wide by 96 pixels high, which is the size of one-quarter of the *PMODE 4* screen. Pressing the fire button will enlarge this section to a full screen.

The resulting blowup can be examined or copied to the source screen, where it can be saved to tape or disk, or enlarged again. This provides for interfacing *BLOWUP* to other graphics programs such as a screen printer or *Graphicom*, and allows repeated enlargements which can create some unusual effects.

The *BLOWUP* program is shown in Listing 1. The program is menu-oriented and contains the necessary instructions. The only additional note is that when viewing a graphics screen, pressing any key will return to the menu. Several safeguards are provided so that the user has the option of aborting an operation and returning to the menu.

BLOWUP uses all eight graphics pages. The source screen resides on pages one to five. The blowup is generated on pages five to eight. By using eight pages, the source screen is preserved and can be examined at any time. The source screen will only be destroyed by the *COPY BLOWUP* command, which copies the blowup screen to the source screen. The *PCLEAR 8* statement in Line 40 sets up the graphics pages. If the program does not run as written, enter *PCLEAR 8* before loading and running *BLOWUP*.

The BASIC program is quite straightforward, and requires little explanation, except for the graphics cursor routine and embedded machine language (ML) subroutine. The program structure, by lines, is:

40-50	Initialize
60-80	ML subroutine
90-120	Main menu
130-180	Load source
190-240	Save source
250-260	Blowup instructions
270-340	Graphics cursor
350-360	Call to ML subroutine
370	See source
380	See blowup
390-400	Copy blowup
410-420	Utilities

The program adapts automatically to use of disk or tape for determining where the graphics reside. Remember that page one starts at *$600* without disk and *$E00* with disk. This information is conveniently stored at *$BA*, which is the most significant byte (MSB) of the start of the page selected by the *PMODE* command.

The sequence of generating the graphics cursor begins by establishing the start address of page one. This is accomplished by SB in Line 270. The joystick inputs are weighted and then added to SB. The address AD is the location of the upper left (UL) cursor byte. This address is offset by 3055 bytes for the lower right (LR) cursor byte.

To provide a non-destructive cursor that is visible regardless of screen content, the value in the UL and LR bytes is first *PEEKed*. The same bytes are *POKEd* with their numerical complement (Line 300). The original values are finally restored in Line 320.

The cycle of reading the joysticks and blinking the cursor is repeated if neither the fire button (Line 340) or space bar (Line 330) are pressed. When the fire button is pressed, the program branches to Line 350 where the USR call is made to the ML subroutine.

This subroutine is actually contained in Line 60 as the string *ML$*. Each pair of characters are a byte of ML code. The assembly listing is provided in Listing 2 for illustration. The ML code is *POKEd* into memory by Line 70. Line 80 is provided as a check on typing skills. If *ML$* is not entered correctly, the program will end before the main menu appears.

The ML subroutine begins by receiving and processing the address of the UL cursor byte, AD, via the USR call and BASIC's INTCNV subroutine. The X register (Line 110) is used as a pointer to the source screen byte being processed. The start and end of graphics pages five and eight are established by Lines 130 through 170. The Y register is used to point to the destination, or blowup, bytes being generated.

Three loops are used. *LOOP1* for eight bits of each source byte, *LOOP2* for 16 horizontal bytes of source screen, and *LOOP3* for vertical increments until the end of page eight is reached. Note that each source byte generates two vertical and two horizontal destination bytes.

The source screen is preserved by using the *ROL ,X* instruction to examine each source bit. If a bit is 'on', the weight corresponding to that bit is selected from the *TABLE* values. This value is then *ORed* with the destination bytes. After completing *LOOP1*, a final *ROL* brings the source byte back to its original state.

It is hoped that you find this program a welcome addition to your graphics repertoire.

Listing 1:

```
10 'BLOWUP
20 'JOSEPH KOHN
30 '22MAR84
40 PCLEAR8:GOTO50
50 CLEAR50,&H4FFF:DEFUSR0=&H5000
60 ML$="BDB3ED1F0196BA5F1F028B18
ED8C478610A78C44338C2D6984240BEC
C4AAA4EA21EDA4EDA8203342ECC426EB
6984300131226A8C232EDD31A8203088
```

```
1Ø1ØAC8C1525CC39CØØØ3ØØØØCØØØ3ØØ
ØØCØØØ3ØØØØCØØØ3ØØØØ"
7Ø FORI=ØTO(LEN(ML$)/2)-1:POKE&H
5ØØØ+I,VAL("&H"+MID$(ML$,(I*2)+1
,2)):NEXT
8Ø CK=Ø:FORI=&H5ØØØ TO &H5Ø55:CK
=CK+PEEK(I):NEXT:IFCK<>7973THEN
END
9Ø X$(Ø)="LOAD SOURCE":X$(1)="SA
VE SOURCE":X$(2)="BLOWUP SOURCE"
:X$(3)="SEE SOURCE":X$(4)="SEE B
LOWUP":X$(5)="COPY BLOWUP"
1ØØ X$="BLOWUP":GOSUB42Ø:FORI=ØT
O5:PRINTI+1". "X$(I):NEXT
11Ø PRINT@48Ø,"CHOICE?";:GOSUB41
Ø:K=VAL(K$):IFK<1 OR K>6 THEN11Ø
12Ø ON K GOTO 13Ø,19Ø,25Ø,37Ø,38
Ø,39Ø
13Ø X$=X$(Ø):GOSUB42Ø
14Ø INPUT"dISK OR tAPE";I$
15Ø LINEINPUT"FILE NAME? ";FI$
16Ø INPUT"READY";K$:IFK$="N"THEN
1ØØELSEPMODE4,1:PCLS:SCREEN1,Ø
17Ø IFI$="D"THEN LOADM FI$ ELSE
CLOADM FI$
18Ø GOTO1ØØ
19Ø X$=X$(1):GOSUB42Ø
2ØØ INPUT"dISK OR tAPE";I$
21Ø LINEINPUT"FILE NAME? ";FI$
22Ø INPUT"READY";K$:IFK$="N"THEN
1ØØELSEPMODE4,1:SCREEN1,Ø
23Ø IFI$="D"THEN SAVEM FI$,SB,SB
+6144,SB ELSE CSAVEM FI$,SB,SB+6
144,SB
24Ø GOTO1ØØ
25Ø X$=X$(2):GOSUB42Ø
26Ø PRINT"USE THE RIGHT JOYSTICK
 TO        SELECT THE SOURCE SECT
ION.":PRINT:PRINT"PRESS THE FIRE
 BUTTON TO BLOWUP.";:PRINT:PRINT
"USE THE space bar TO ABORT.":PR
INT
27Ø INPUT"READY";K$:IFK$="N"THEN
1ØØELSEPMODE4,1:SCREEN1,Ø:SB=PEE
K(&HBA)*256
28Ø JX=JOYSTK(Ø):JY=JOYSTK(1)
29Ø AD=SB+INT(JX/3.937)+32*INT(1
.52381*JY)
3ØØ V1=PEEK(AD):POKEAD,255-V1:V2
=PEEK(AD+3Ø55):POKEAD+3Ø55,255-V
2
31Ø FB=PEEK(6528Ø)
32Ø POKEAD,V1:POKEAD+3Ø55,V2
33Ø IFINKEY$=" "THEN1ØØ
34Ø IF FB=127 OR FB=255 THEN28Ø
35Ø PMODE4,5:PCLSØ:SCREEN1,Ø:A=U
SRØ(AD)
36Ø GOSUB41Ø:GOTO1ØØ
37Ø PMODE4,1:SCREEN1,Ø:GOSUB41Ø:
GOTO1ØØ
38Ø PMODE4,5:SCREEN1,Ø:GOSUB41Ø:
GOTO1ØØ
39Ø X$=X$(5):GOSUB42Ø:INPUT"ARE
YOU SURE";K$:IFK$="N"THEN1ØØ
4ØØ PMODE4,1:SCREEN1,Ø:FORI=5TO8
:PCOPY I TO I-4:NEXT:GOSUB41Ø:GO
TO1ØØ
41Ø K$=INKEY$:IFK$=""THEN41ØELSE
RETURN
42Ø CLS:X=LEN(X$):Y=INT((32-X)/2
):PRINTSTRING$(Y,"*")X$STRING$(3
2-Y-X,"*"):RETURN
```

Listing 2:

```
                   ØØØ1Ø *BLOWUP
                   ØØØ2Ø *JOSEPH KOHN
                   ØØØ3Ø *23MAR84
                   ØØØ4Ø *
                   ØØØ5Ø *ENTRY:
                   ØØØ6Ø *  PASS SOURCE START BYTE WITH USR
                   ØØØ7Ø *  PMODE4,5 SELECTED
5ØØØ               ØØØ8Ø         ORG $5ØØØ
                   ØØØ9Ø *
5ØØØ BD   B3ED     ØØ1ØØ START   JSR $B3ED       INTCNV
5ØØ3 1F   Ø1       ØØ11Ø         TFR D,X         X=START BYTE
                   ØØ12Ø *GET START OF PAGE 5 AND END OF PAGE 8
5ØØ5 96   BA       ØØ13Ø         LDA $BA
5ØØ7 5F            ØØ14Ø         CLRB
5ØØ8 1F   Ø2       ØØ15Ø         TFR D,Y         Y=START OF PAGE 5
5ØØA 8B   18       ØØ16Ø         ADDA #$18
5ØØC ED   8C 47    ØØ17Ø         STD <ENDBYT,PCR
5ØØF 86   1Ø       ØØ18Ø LOOP3   LDA #16
5Ø11 A7   8C 44    ØØ19Ø         STA <COUNT,PCR
5Ø14 33   8C 2D    ØØ2ØØ LOOP2   LEAU <TABLE,PCR START OF WEIGHT TABLE
5Ø17 69   84       ØØ21Ø LOOP1   ROL ,X          GET SOURCE BIT
5Ø19 24   ØB       ØØ22Ø         BCC NOT1        GO IF Ø
5Ø1B EC   C4       ØØ23Ø         LDD ,U          GET WEIGHT
5Ø1D AA   A4       ØØ24Ø         ORA ,Y
5Ø1F EA   21       ØØ25Ø         ORB 1,Y
5Ø21 ED   A4       ØØ26Ø         STD ,Y          PUT NEW VALUE IN DEST.
5Ø23 ED   A8 2Ø    ØØ27Ø         STD 32,Y
5Ø26 33   42       ØØ28Ø NOT1    LEAU 2,U        GET NEXT WEIGHT
5Ø28 EC   C4       ØØ29Ø         LDD ,U
5Ø2A 26   EB       ØØ3ØØ         BNE LOOP1       GO AT END
5Ø2C 69   84       ØØ31Ø         ROL ,X
5Ø2E 3Ø   Ø1       ØØ32Ø         LEAX 1,X        GET NEXT SOURCE BYTE
5Ø3Ø 31   22       ØØ33Ø         LEAY 2,Y        GET NEXT DEST. BYTE
5Ø32 6A   8C 23    ØØ34Ø         DEC <COUNT,PCR
5Ø35 2E   DD       ØØ35Ø         BGT LOOP2
5Ø37 31   A8 2Ø    ØØ36Ø         LEAY 32,Y       SKIP DEST. ROW
5Ø3A 3Ø   88 1Ø    ØØ37Ø         LEAX 16,X       SKIP UNUSED BYTES
5Ø3D 1ØAC 8C 15    ØØ38Ø         CMPY <ENDBYT,PCR
5Ø41 25   CC       ØØ39Ø         BLO LOOP3       GO IF NOT AT END
5Ø43 39            ØØ4ØØ         RTS
                   ØØ41Ø *
5Ø44      CØØØ     ØØ42Ø TABLE   FDB 49152       16384+32768
5Ø46      3ØØØ     ØØ43Ø         FDB 12288       4Ø96+8192
5Ø48      ØCØØ     ØØ44Ø         FDB 3Ø72        1Ø24+2Ø48
5Ø4A      Ø3ØØ     ØØ45Ø         FDB 768         256+512
5Ø4C      ØØCØ     ØØ46Ø         FDB 192         64+128
5Ø4E      ØØ3Ø     ØØ47Ø         FDB 48          16+32
5Ø5Ø      ØØØC     ØØ48Ø         FDB 12          4+8
5Ø52      ØØØ3     ØØ49Ø         FDB 3           1+2
5Ø54      ØØØØ     ØØ5ØØ         FDB Ø
                   ØØ51Ø *
5Ø56               ØØ52Ø ENDBYT  RMB 2
5Ø58               ØØ53Ø COUNT   RMB 1
          ØØØØ     ØØ54Ø         END
ØØØØØ TOTAL ERRORS
```

GRAPHICS

32K
DISK
ECB

The Art of Joystick Painting

Brian Preble

Sketch is a color graphics editor for a 32K Color Computer with Extended BASIC and at least one joystick. It works in *PMODEs 3* and *4*.

Sketch contains all the normal commands used in most graphics editors such as *CIRCLE, LINE, BOX, DRAW, PAINT*, etc. It also has a feature that sets it apart from all other BASIC graphics editors I've seen. In most editors, if you choose a command you don't like, you must painstakingly erase the results and then redraw anything that was destroyed.

With *Sketch*, if you don't like something, press 'X' and the screen is restored to its original display. If you decide you like it, press the space bar and the screen will be updated. You *must* press the space bar to save a command! If you don't, the screen will be restored if you move the joystick or change modes.

How to use *Sketch*

Sketch is a simple program to use. To move the cursor, simply move the right joystick in the direction you want.

To choose a command, press the key for the command desired. These commands may be listed by pressing 'H' for help.

For a circle, move the cursor to the center of the circle and press 'C'. You may then move the joystick and a circle will be drawn with the cursor as a point on the circle. If you like it, press the space bar and it will be saved. If you don't, press 'X' to cancel it.

The other commands work in a similar manner. Move the cursor to the origin of the line, box, etc. Then press a key and move the joystick.

One exception to this rule is the WRITE command. This command allows you to type letters and other characters directly on the graphics screen. To use this command, move the cursor to the position of the first letter and press 'W'. You are now in the WRITE mode. Anything you type will be shown on the screen except lowercase. To erase what you wrote, press the DELETE key (left arrow). To save what you wrote, press ENTER. If your letters aren't in the right position, move the joystick and type again; there is no need to press DELETE.

Two especially useful commands are GET 'G' and PUT 'P'. Use GET to store an image in a buffer for use somewhere else in the picture or if it wasn't placed quite right. For example: You drew a picture of a house and later decided you wanted it somewhere else. You would have to move to one corner of the house or other object (give yourself some clear space) and press 'G'. Then move to the opposite corner. A box will form indicating the area you will be storing. When you have it all, press the space bar and it will be stored.

To put it somewhere else, press 'P' and a house will appear near the cursor. Move the joystick and the house or other object will move with it. Press the space bar when it is properly positioned or 'X' to erase it.

Bear in mind that the GET command only stores an object, it *does not* erase it. You must do that manually if you desire.

The LINE and RAY commands don't end until 'X' is pressed. These commands are similar to each other, but LINE continues from where the previous line left off and RAY always starts from the point where it was chosen.

The Menu

By far the most powerful command in *Sketch* is 'M'. This command brings up a menu screen from which all other commands are controlled. Displayed on the menu are a number of commands followed by various numbers. The meaning of the numbers will become clear when you use that particular command. To use a command, press the key shown in inverse video (green on black) for that command.

For example: If you want to change colors, press 'D' (for *Draw*). The screen will clear and the prompt *"FOREGROUND COLOR?"* will be displayed. Type in the desired color (0-3) and press ENTER. The prompt *"BACKGROUND COLOR?"* will appear. Answer that in the same manner.

MOVE is the rate of movement of the cursor. Its default is 1,1. The first number is the number of dots moved each time the joystick is moved left or right, and the second is the number of dots moved up or down. Thus, if MOVE was 2,3 the cursor would move two left or right and three up and down. In the draw or erase modes ("+", "-") this would result in dotted lines.

Pattern is a command that allows patterns of colors to be used instead of solid colors. For example, if you had a foreground pattern of 48 and drew a filled-in box (F) the box would show up as a series of vertical bars in *PMODE 4*. If the foreground pattern was 1, the same box would show up in blue or red instead of the usual black or white. The range of patterns allowed is 0-255. Background pattern is used for clearing the screen.

Pattern is turned on by pressing 'A' and selecting your foreground and background patterns. It is turned off by setting the normal foreground and background colors as described above.

The SAVE and LOAD commands will save or load a picture from disk. To change them to tape, change Line 69 to:

```
69 INPUT"PRESS ENTER TO CON-
TINUE"X$:CSAVEMF$,1536,7679,
1536
```

And change Line 61 to:

```
61 SCREEN1:CLOADMF$
```

WRITE determines the size of the letters; 4 is normal, 8 is double, 12 is triple, and so on by fours.

Hi Speed Poke Problems

If your computer can't handle *POKE 65495,0* you will have to remove it from Lines 1, 43, 57, 61 and 69. By "can't handle," I mean the screen goes crazy. I *don't* mean that some keys don't respond. If you are one of the latter then simply press SHIFT with the offending key. This should do the trick.

How It Works

The main body of *Sketch* is contained in Lines 2 to 16. These lines read the joystick and keyboard. If the joystick is moved, the screen is restored and control is passed to the appropriate subroutine for the command chosen. If the joystick wasn't moved then the keyboard is read for a mode change, character to be typed, or a cancel/save command ('X' or space). If a key wasn't pressed then the cursor is blinked if necessary and control returns to the joystick checker; otherwise, control is passed to the appropriate subroutine as above.

The subroutine at Line 85 copies the display screen to the backup screen when the space bar is pressed, the program is first run, or SHIFT/CLEAR is pressed.

The subroutine at Line 84 copies the backup graphics screen to the display screen when 'X' is pressed, the joystick is moved, or the cursor blinks.

These routines are a little faster than a machine language routine would be due to the time that would be needed to call such a routine from BASIC and its lack of response to the high-speed poke.

Here's one last hint. The GET/PUT option "NOT" in the menu will not put the contents of the GET buffer on the screen; instead, it will reverse an area of the screen the same *size* as the GET buffer. That is, black becomes white, white becomes black, red becomes blue, etc.

I hope you enjoy this program.

14	222
30	107
43	84
58	78
72	248
94	184
115	31
140	217
END	216

The listing:

```
1 POKE65495,Ø:CLS:PCLEAR8:PMODE4
,1:SCREEN1,1:GOSUB85:CX=128:CY=9
6:M=1:CH=.9:CS=1:CE=1:SX=1:SY=1:
PF=5:PB=5:S=4:DIML$(58):FORX=ØTO
58:READL$(X):NEXT:C$="Ø+MCBFLRZW
-DGPJ\X;OT":O$="CDLMPQRSWGXA":CF
=5:CB=Ø:DIMG(15ØØ):PO=4:COLORCF,
CB
2 X=JOYSTK(Ø):Y=JOYSTK(1)
3 A$=INKEY$:IFM=1ØTHEN4ELSEIFA$=
" "THEN41ELSEIFA$="H"GOSUB84:GOT
O86ELSEIFA$="J"GOSUB38ELSEIFA$="
M"GOSUB84:GOTO46ELSEIFINSTR(C$,A
$)=ØORA$=""THEN4ELSEM=INSTR(C$,A
$):A$="":FL=Ø:GOSUB84
4 IFM<>1THENONM-1GOSUB17,84,18,2
Ø,22,24,24,26,43,27,28,3Ø,32,38,
39,4Ø,42,74,79
5 IFX>ØANDX<63ANDY>ØANDY<63ANDM<
>1THEN2
6 IFM<>2ANDM<>11GOSUB84
7 IFX=ØTHENCX=CX-SX
8 IFCX<ØTHENCX=255
9 IFX=63THENCX=CX+SX
1Ø IFCX>255THENCX=Ø
11 IFY=ØTHENCY=CY-SY
12 IFCY<ØTHENCY=191
13 IFY=63THENCY=CY+SY
14 IFCY>191THENCY=Ø
15 DRAW"BM"+STR$(CX)+","+STR$(CY
):IFM<>11ANDM<>1ØANDM<>2THENIFPA
=1THENDRAW"BRRH2NDG2NHRF2NUE2"EL
SEIFPPOINT(CX+1,CY)=CF ANDPPOINT
(CX,CY+1)=CF THENDRAW"C=CB;BRRH2
NDG2NRF2NUE2C=CF;"ELSEDRAW"C=CF;
BRRH2NDG2NRF2NUE2"
16 GOTO2
17 IFFL=ØGOSUB84:FL=1:PSET(CX,CY
):RETURNELSEPSET(CX,CY):RETURN
18 IFFL=ØTHENXE=CX:YE=CY:FL=1
19 CIRCLE(XE,YE),SQR((CX-XE)^2+(
CY-YE)^2),,CH,CS,CE:RETURN
2Ø IFFL=ØTHENXE=CX:YE=CY:FL=1
21 LINE(XE,YE)-(CX,CY),PSET,B:RE
TURN
22 IFFL=ØTHENXE=CX:YE=CY:FL=1
23 LINE(XE,YE)-(CX,CY),PSET,BF:R
ETURN
24 IFFL=ØTHENXE=CX:YE=CY:FL=1
25 LINE(XE,YE)-(CX,CY),PSET:RETU
RN
26 PAINT(CX,CY),,PB:RETURN
27 IFFL=ØGOSUB84:FL=1:PRESET(CX,
CY):RETURNELSEPRESET(CX,CY):RETU
RN
28 IFFL=ØTHENXE=CX:YE=CY:FL=1
29 CR=SQR((CX-XE)^2+(CY-YE)^2):F
ORX9=ØTOCR:CIRCLE(XE,YE),X9,CB,C
H,CS,CE:NEXT:CIRCLE(XE,YE),CR,CF
,CH,CS,CE:PAINT(XE,YE),PF,PB:RET
URN
3Ø IFFL=ØTHENXE=CX:YE=CY:FL=1
31 GET(XE,YE)-(CX,CY),G,G:LINE(X
E,YE)-(CX,CY),PSET,B:GX=ABS(XE-C
X):GY=ABS(YE-CY):RETURN
32 ONPO GOTO33,34,35,36,37:RETUR
N
33 PUT(CX,CY)-(CX+GX,CY+GY),G,PS
ET:RETURN
34 PUT(CX,CY)-(CX+GX,CY+GY),G,PR
ESET:RETURN
35 PUT(CX,CY)-(CX+GX,CY+GY),G,AN
D:RETURN
36 PUT(CX,CY)-(CX+GX,CY+GY),G,OR
:RETURN
37 PUT(CX,CY)-(CX+GX,CY+GY),G,NO
T:RETURN
38 POKE65494,Ø:CLS:GOSUB84:PRINT
"CURSOR IS AT"CX","CY:PRINT:INPU
T"JUMP CURSOR TO";CX,CY:POKE6549
5,Ø:SCREEN1:RETURN
39 PCLS:GOSUB85:CX=128:CY=96:M=1
:RETURN
4Ø GOSUB84:M=1:RETURN
41 IFM=1THEN2ELSEIFM=2ORM=11GOSU
B85:M=1:GOTO2ELSEGOSUB84:ONM-1GO
SUB17,1,18,2Ø,22,24,24,26,43,27,
28,3Ø,32,38,39,4Ø,42,74,79:IFM=1
3GOSUB84:M=1:GOTO2ELSEGOSUB85:IF
M=7THENFL=Ø:GOTO2ELSEIFM=8THEN2E
LSEM=1:GOTO2
42 M=2:RETURN
43 POKE65494,Ø:DRAW"S=S;":IFA$="
"THENRETURNELSEIF(A$>"Z"ORA$<CHR
$(13)ORA$=CHR$(21))ANDA$<>CHR$(8
)THENRETURNELSEIFA$=CHR$(13)THEN
```

```
M=1:GOSUB85:POKE65495,Ø:RETURNEL
SEIFA$=CHR$(8)GOSUB84:M=1:POKE65
495,Ø:RETURNELSEDRAWL$(ASC(A$)-3
2)
44 IFS<>4THENDRAW"S4":RETURNELSE
DRAW"BL6"+L$(ASC(A$)-32)+"S4":RE
TURN
45 M$=STR$(SC):DRAW"BMØ,12C1S8":
GOSUB43:SC=SC+P:M$=STR$(SC):DRAW
"BMØ,12C3S8":GOSUB43:DRAW"C1":LI
NE(208,Ø)-(255,12),PSET,BF:DRAW"
BM21Ø,12C4":M$=STR$(F):GOSUB43:R
ETURN
46 POKE65494,Ø:CLS:PRINT@13,"OPT
IONS":PRINT@45,"-------"
47 PRINT"  cIRCLE ="CH","CS","CE
48 PRINT"  dRAW    ="CF","CB
49 PRINT"  mOVE    ="SX","SY
50 PRINT"  pAINT   ="PF","PB
51 PRINT" PaTTERN =";:IFPA=ØTHEN
PRINT" OFF"ELSEPRINTTF","TB
52 PRINT"  wRITE  ="S
53 PRINT"  sAVE":PRINT"  1OAD":P
RINT"  qUIT"
54 PRINT"  rESOLUTION"
55 PRINT"  gET/PUT OPTION ="PO:P
RINT:PRINT"  ExIT TO SCREEN"
56 I$=INKEY$:IFINSTR(O$,I$)=ØORI
$=""THEN56ELSEI=INSTR(O$,I$)
57 IFI$="X"THENPOKE65495,Ø:SCREE
N1:GOTO2ELSECLS:ONI GOSUB58,59,6
Ø,63,64,65,66,68,71,72,,73:GOTO4
6
58 INPUT"CIRCLE HEIGHT";CH:INPUT
"CIRCLE START";CS:INPUT"CIRCLE E
ND";CE:RETURN
59 INPUT"FOREGROUND COLOR";CF:IN
PUT"BACKGROUND COLOR";CB:PA=Ø:CO
LORCF,CB:RETURN
6Ø CLS:PRINT@74,"LOAD PICTURE":P
RINT@224,;:LINEINPUT"FILENAME: "
;F$
61 SCREEN1:POKE65495,Ø:LOADMF$
62 GOSUB85:RETURN
63 INPUT"SPEED _,^";SX,SY:RETURN
64 INPUT"PAINT COLOR";PF:INPUT"B
ORDER COLOR";PB:RETURN
65 CLS:END
66 INPUT"RESOLUTION (3 OR 4)";R:
IFR=3THENPMODE3,1:SCREEN1,ØELSEI
FR=4THENPMODE4,1:SCREEN1,1ELSECL
S:GOTO66
67 GOTO46
68 CLS:PRINT@74,"SAVE PICTURE":P
RINT@224,;:LINEINPUT"FILENAME: "
;F$
69 POKE65495,Ø:SAVEMF$,3584,9727
,Ø
7Ø RETURN
71 INPUT"SIZE (4 IS NORMAL)";S:R
ETURN
72 INPUT"OPTION (1=PSET,2=PRESET
,3=AND,  4=OR,5=NOT)";PO:RETURN
73 INPUT"PATTERN (FOREGROUND, BA
CKGROUND)";TF,TB:PA=1:POKE178,TF
:POKE179,TB:RETURN
74 IFFL=ØTHENXE=CX:YE=CY:FL=1
75 RO=INT(SOR((CX-XE)^2+(CY-YE)^
2))
76 IFRO/3<>INT(RO/3)THENRO=RO+1:
GOTO76
77 RO=RO/3:IFRO>32THENRO=32
78 DRAW"BM"+STR$(XE)+","+STR$(YE
)+"C=CF;S=RO;BM+Ø,-6R2F4D4G4L4H4
U4E4R2BM+Ø.6S4":RETURN
79 IFFL=ØTHENXE=CX:YE=CY:FL=1
8Ø RO=INT(SQR((CX-XE)^2+(CY-YE)^
2))
81 IFRO/4<>INT(RO/4)THENRO=RO+1:
GOTO81
82 RO=RO/4:IFRO>32THENRO=32
83 DRAW"BM"+STR$(XE)+","+STR$(YE
)+"C=CF;S=RO;BM+Ø,-4F8L16E8BM+Ø,
4S4":RETURN
84 DRAW"S4":FORSC=1TO4:PCOPYSC+4
TOSC:NEXT:RETURN
85 DRAW"S4":FORSC=1TO4:PCOPYSC T
OSC+4:NEXT:RETURN
86 CLS:PRINTTAB(13)"HELP":PRINTS
TRING$(32,131);
87 PRINT"c=CIRCLE","+=DRAW"
88 PRINT"d=DISK","Ø=MOVE"
89 PRINT"b=BOX","-=ERASE"
9Ø PRINT"f=FILLED BOX","m=MENU"
91 PRINT"1=LINE","h=THIS HELP"
92 PRINT"r=RAY","j=JUMP"
93 PRINT"g=GET      <SHIFT-CLEAR
>=CLEAR"
94 PRINT"p=PUT","x=CANCEL"
95 PRINT"z=PAINT","<SPACE>=STORE
"
96 PRINT"o=OCTAGON","t=TRIANGLE"
97 PRINT"w=WRITE",CHR$(127)"=UNW
RITE"
98 PRINT"<ENTER>=STORE WRITE"
99 PRINT:PRINT"    USE THE JOYST
ICK TO MOVE";
1ØØ IFINKEY$=""THEN1ØØELSESCREEN
1:GOTO2
1Ø1 FORI=ØTO58:READL$(I):NEXT
1Ø2 DATA"BM+7,Ø
1Ø3 DATA"BM+2,1UBM+Ø,-2U5BM+5,7
1Ø4 DATA"BM+1,-4U2BM+2,ØD2BM+4,4
1Ø5 DATA"BM+1,ØU6BM+2,ØD6BM-3,-4
R4BM-4,2R4BM+3,2
1Ø6 DATA"BM+4,-5L2NUND5L2D2R4D2L
4BM+7,1
1Ø7 DATA"UE4UBM-4,ØDBM+4,4DBM+3,
Ø
1Ø8 DATA"BM+5,ØNEH4UERFDGL2GDFR2
```

```
E2BM+2,2
109 DATA"BM+2,-5EBM+4,6
110 DATA"BM+3,ØH2U2E2BM+4,6
111 DATA"BM+1,ØE2U2H2BM+6,6
112 DATA"BM+3,-3NU2NR2ND2NL2NHNE
NFNGBM+4,3
113 DATA"BM+2,-1U2NU2NL2R2BM+3,3
114 DATA"BM+2,ØNUGBM+6,-1
115 DATA"BM+Ø,-3R4BM+3,3
116 DATA"BM+2,ØUBM+5,1
117 DATA"UE4UBM+3,6
118 DATA"BM+1,ØHU4ER2FD4GL2BM+6,
Ø
119 DATA"BM+1.ØRNRU6GBM+6.5
120 DATA"NR4UERE2UHL2GBM+7,5
121 DATA"BM+Ø,-1FR2EH2E2HL3BM+7,
6
122 DATA"BM+3,ØU2NRL3UE3D3BM+4,3
123 DATA"BM+Ø,-1FR2EU2HL3U2R4BM+
3,6
124 DATA"BM+4,-5HL2GD4FR2EUHL3BM
+7,3
125 DATA"UE4UL4BM+7,6
126 DATA"BM+1,ØHUEHUER2FDGNL2FDG
L2BM+6,Ø
127 DATA"BM+Ø,-1FR2EU4HL2GDFR2BM
+4,3
128 DATA"BM+2,-1UBM+Ø,-2UBM+5,5
129 DATA"BM+1,1EUBM+Ø,-2UBM+5,4
130 DATA"BM+4,ØH3E3BM+3,6
131 DATA"BM+1,-2R3BM-3,-2R3BM+3,
4
132 DATA"BM+2,ØE3H3BM+5,6
133 DATA"BM+1,-6ER2FDG2BM+Ø,2DBM
+4,Ø
134 DATA"BM+Ø,-3UER2D4LNH2R3EU3H
2L5G2D4F2R3BM+3,-1
135 DATA"U4E2F2D2NL4D2BM+3,Ø
136 DATA"U6R3FDGNL3FDGL3BM+7,Ø
137 DATA"BM+1,ØHU4ER2FBM+Ø,4GL2B
M+6,Ø
138 DATA"U6R3FD4GL3BM+7,Ø
139 DATA"NR4U3NR2U3R4BM+3,6
140 DATA"U3NR2U3R4BM+3,6
141 DATA"BM+1,ØHU4ER2FBM+Ø,2NL1D
2GL2BM+6,Ø
142 DATA"U3NU3R4NU3D3BM+3,Ø
143 DATA"BM+1,ØRNRU6NLRBM+4,6
144 DATA"BM+Ø,-1FREU5NLRBM+3,6
145 DATA"U3NU3RNE3F3BM+3,Ø
146 DATA"NU6R4UBM+3,1
147 DATA"U6F2NDE2D6BM+3,Ø
148 DATA"U6FDF2DFNU6BM+3,Ø
149 DATA"BM+1,ØHU4ER2FD4GL2BM+6,
Ø
150 DATA"U6R3FDGL3BM+7,3
151 DATA"BM+1,ØHU4ER2FD3GNHNFGLB
M+6,Ø
152 DATA"U6R3FDGL2NLF3BM+3,Ø
153 DATA"BM+Ø,-1FR2EUHL2HUER2FBM
+3,5
154 DATA"BM+2,ØU6NL2R2BM+3,6
155 DATA"BM+Ø,-1NU5FR2EU5BM+3,6
156 DATA"BM+Ø,-6D2FDFNDEUEU2BM+3
,6
157 DATA"NU6E2NUF2U6BM+3,6
158 DATA"UE4UBM-4,ØDF4DBM+3,Ø
159 DATA"BM+Ø,-6D2F2ND2E2U2BM+3,
6
160 DATA"NR4UE4UL4BM+7,6
```

Hint . . .

How To Be A Printer Artist In One Easy Lesson

Anyone not having "Printer Artist" from the November 1983 issue of THE RAINBOW may find the following program useful.

```
10 READ A$
20 FOR X=1 TO LEN(A$) STEP 3
30 B$=MID$(A$,X,3)
40 C$=RIGHT$(B$,1)
50 PRINT#-2,STRING$(VAL(B$),C$);
60 NEXT X:PRINT#-2:GOTO 10
```

With this program you may enter each line of the printer mysteries as *DATA* statements. For example:

Line
1. 23SP,1X,12N
2. 19SP,1X,6SP,8$

would be entered as:

```
101 DATA 23 01X12N
102 DATA 19 01X06 08$
```

Start *DATA* statements at Line 101 (old line number plus 100) and change *INFO* to a two-digit number followed by the character you want printed. You must also drop the commas. The *23SP,1X,12N* becomes *'23 01X12N'*.

After you have any or all the lines changed, just type *RUN*. The program will print out the picture to your printer until it runs out of data. An OD Error will appear on the screen but this will not hurt anything. You may then go back and edit any errors until you have your picture correct. Then *SAVE* each program for future use.

Michael B. Kromeke

GRAPHICS

64K ECB

Preserving The Classics By Patching Art Gallery

Paul S. Hoffman

Here's another graphics program modification, prompted by a letter to THE RAINBOW way, way back in April 1982. Mr. Gary Burkhardt of Coldwater, Mich., asked for help in getting picture tapes from Radio Shack's *Art Gallery* dumped to printer. Not only is there no printer dump in *Art Gallery*, but *Art Gallery* picture tapes won't even load in using other programs or BASIC! Wouldn't it be nice if an *Art Gallery* picture could be saved like a standard machine language tape *CSAVEM*?

Having played around with adapting *Micropainter* to operate from disk (THE RAINBOW, March '84), and having newly acquired my 64K upgrade (which makes modifying ROM Pak programs simpler), I decided to tackle *Art Gallery's* tape save routine. I found two problems: *Art Gallery* creates tapes with a slightly different coding at the beginning, and the loading address is the same as the start of BASIC's text page memory, *$400*. I have replaced the 'tape save' portion of *Art Gallery* with a routine which creates a 'standard' machine language tape readable by BASIC's *CLOADM* routine. It turns out that *Art Gallery* will read these 'standard' tapes without any change in the tape input portion of the program.

The tapes created by this revision to *Art Gallery* must be *OFFSET LOADED* to be used by Extended or Disk Extended BASIC. If you are loading a tape into Extended BASIC (without disk), the offset is *$200 (CLOADM "", &H200)*. For Disk Extended BASIC, it's *$A00 (CLOADM "", &HA00)*.

Note: *Art Gallery* does not put a filename or title on a tape, so make sure to use two quotation marks to indicate a blank filename.

The tapes will also load automatically into my disk version of *Micropainter* without worrying about the offset — then they can be saved directly to disk. For those without *Micropainter*, Listing 3 will load files from modified *Art Gallery* tapes, display them, then save them to tape or disk at the revised addresses. The tapes will not load into The MicroWorks' *Magigraph* because of the loading address; first transfer the pictures using *Micropainter* or Listing 3, then you can make use of them with *Magigraph*.

To modify your version of *Art Gallery*, use Listing 1. Make sure to start-up in non-disk Extended BASIC, and disable the cartridge auto-start by entering *"POKE &HFF23,36."* With a Multi-Pak Interface or other selectable-port interface, select the slot with the *Art Gallery* cartridge. Otherwise, insert the *Art Gallery* cartridge very carefully. Note: Plugging or unplugging cartridges with the power on can cause serious damage. This is *not* recommended. Now run Listing 1, which will copy the cartridge contents to lower memory, alter the tape save routine, and add a short routine to move the whole program back up to its proper memory addresses. You will end up with a machine language program called *ARTGAL* saved on cassette tape. Turn off your computer and then power up with the disk system engaged. Save the taped *ARTGAL* program to disk by typing the following:

```
CLOADM "ARTGAL" ENTER
SAVEM "ARTGAL/BIN",&H4000,
&H5014, &H5000
ENTER
```

On the same disk, save Listing 2 as *ARTGAL/BAS*. Now when you *RUN ARTGAL*, the computer will be changed to 64K RAM operation with the BASIC ROMs copied to RAM. This is so that Disk BASIC can load your program, but then be replaced by *Art Gallery* operating in RAM but at its 'correct' address location; starting at *$C000*, the beginning of cartridge memory. The *ARTGAL/BAS* program will immediately execute the modified *Art Gallery*. You will not be able to return to BASIC because hitting Reset sends the computer off into oblivion, never to return without turning it off and back on.

Now, I can make modifications to any graphics in my files, using *Micropainter, Magigraph, Art Gallery, Graphicom*, or the *X-PAD* — the files are almost fully interchangeable. Figure 1 is a drawing from the *Art Gallery* side of Radio Shack's *Fantasy Images* tape (Cat. No. 26-3304). Figure 2 is the same picture modified by adding a mirror-image rubber stamp using *Graphicom* and shifting to the *PMODE4* artifacted colors. Both images were printed on the Transtar-315 Color Printer. Radio Shack's *CODUMP* software for the Tandy CGP-220 Color Ink-Jet Printer will not print a *PMODE ONE* picture which is the mode used by *Art Gallery* in the proper ratio.

160...... 247
END 128

Listing 1:

```
10 '****************************
   *  CONVERTS <ART GALLERY> *
   *     TO WRITE STANDARD      *
   *  MACHINE LANGUAGE TAPES *
   *       P. HOFFMAN, 1984       *
   ****************************
15 'NOTE:  64K NEEDED TO RUN
   FINAL PROGRAM!
20 'REMEMBER TO START WITH ROM-
   PACK AUTO-START DISABLED
   (POKE &HFF23, 36), THEN
30 'SWITCH TO THE INTERFACE SLOT
   CONTAINING ART GALLERY (IF
   YOU'RE USING MULTIPAK INTER-
   FACE)
```

```
40 'SOFTWARE SWITCHING WITH A
    POKE IS PREFERABLE TO USING
    THE FRONT SWITCH ON THE
    INTERFACE.
50 CLS:PRINT:PRINT"  READY TO MO
VE <ART GALLERY>     TO RAM AND
ALTER IT?            <PRESS ANY
KEY TO CONTINUE>
6Ø IF INKEY$ = "" THEN 6Ø
65 PRINT:PRINT"  MOVING.....
7Ø FORM=&HCØØØ TO&HCFFF
8Ø POKEM-&H8ØØØ,PEEK(M)
9Ø NEXTM
1ØØ FORX=1TO1ØØ
11Ø READ A$,B$
12Ø A$="&H"+A$:B$="&H"+B$
13Ø PRINTA$;" = ";B$
14Ø POKEVAL(A$),VAL(B$)
15Ø NEXT
16Ø DATA 4529,8E,452A,Ø1,452B,E2
,452C,86,452D,Ø2,452E,A7,452F,8Ø
17Ø DATA 453Ø,6F,4531,8Ø,4532,6F
,4533,8Ø,4534,CC,4535,Ø4,4536,ØØ
18Ø DATA 4537,ED,4538,81,4539,ED
,453A,81,453B,86,453C,Ø2,453D,97
19Ø DATA 453E,92,453F,8E,454Ø,ØØ
,4541,ØØ,4542,86,4543,Ø2,4544,BD
2ØØ DATA 4545,A6,4546,63,4547,ØF
,4548,92,4549,BD,454A,A7,454B,D8
21Ø DATA 454C,8E,454D,Ø4,454E,ØØ
,454F,9F,455Ø,7E,4551,86,4552,FF
22Ø DATA 4553,97,4554,7D,4555,CC
,4556,1C,4557,ØØ,4558,93,4559,7E
23Ø DATA 455A,27,455B,11,455C,1Ø
,455D,83,455E,ØØ,455F,FF,456Ø,24
24Ø DATA 4561,Ø2,4562,D7,4563,7D
,4564,86,4565,Ø1,4566,97,4567,7C
25Ø DATA 4568,BD,4569,A7,456A,F4
,456B,2Ø,456C,E2,456D,86,456E,FF
26Ø DATA 456F,97,457Ø,7C,4571,ØF
,4572,7D,4573,BD,4574,A7,4575,F4
27Ø DATA 4576,2Ø,4577,Ø8,5ØØØ,8E
,5ØØ1,4Ø,5ØØ2,ØØ,5ØØ3,1Ø,5ØØ4,8E
28Ø DATA 5ØØ5,CØ,5ØØ6,ØØ,5ØØ7,EC
,5ØØ8,81,5ØØ9,8C,5ØØA,5Ø,5ØØB,ØØ
29Ø DATA 5ØØC,27,5ØØD,Ø4,5ØØE,ED
,5ØØF,A1,5Ø1Ø,2Ø,5Ø11,F5,5Ø12,7E
3ØØ DATA 5Ø13,CØ,5Ø14,ØØ
31Ø PRINT:PRINT" READY TO SAVE A
LTERED PROGRAM   TO TAPE? (ANY K
EY TO CONINUE)
32Ø IFINKEY$=""THEN32Ø
33Ø CSAVEM"ARTGAL",&H4ØØØ,&H5Ø14
,&H5ØØØ
```

Listing 2:

```
Ø '*****************************
   *    "ARTGAL/BAS" -- LOADS    *
   *   MODIFIED <ART GALLERY>    *
   *     FROM DISK & RUNS IT     *
   *****************************
1 '********* P. HOFFMAN **********
   ************ 1984 *************
5 '64K ROM-TO-RAM ROUTINE THANKS
   TO FRANK HOGG.
1Ø CLEAR999
2Ø DATA 26,8Ø,19Ø,128,Ø,183,255,
222,166,128
3Ø DATA 183,255,223,167,31,14Ø,2
24,Ø,37,241,57
4Ø FORI=1TO21:READA:A$=A$+CHR$(A
):NEXTI
5Ø P=VARPTR(A$)+1
6Ø POKEP,126
7Ø EXECP
8Ø CLS:PRINT:PRINT"  NOW IN RAM!
"
9Ø LOADM"ARTGAL/BIN":POKE&HFF4Ø,
Ø:EXEC&H5ØØØ
```

Listing 3:

```
Ø '****************************
  * LOADS MODIFIED ARTGAL PIX *
  * TO EITHER EXTENDED OR DISK*
  * BASIC                     *
1 '****** P. HOFFMAN ***********
2 '********** 1984 *************
1Ø PMODE1,1:PCLS
2Ø CLS:PRINT:PRINT" POSITION REC
ORDER FOR PLAYBACK   - THEN PRES
S ANY KEY:"
3Ø IFINKEY$=""THEN3Ø ELSESCREEN1
,Ø
4Ø IFPEEK(&HBA)=6 THEN5Ø ELSEIFP
EEK(&HBA)=&HE THEN8Ø ELSEGOTO14Ø
5Ø CLOADM"",&H2ØØ:GOSUB1ØØ
6Ø CSAVEMF$,&H6ØØ,&H12ØØ,413
7Ø INPUT" ANOTHER PICTURE (Y/N)"
;I$:IFASC(I$)=89 THEN2ØELSEEND
8Ø CLOADM"",&HAØØ:GOSUB1ØØ
9Ø SAVEMF$,&HEØØ,&H1AØØ,413:GOTO
7Ø
1ØØ IFINKEY$=""THEN1ØØELSEINPUT"
 NAME FOR SAVED FILE";F$:'******
*IF SAVING FOR DISK <MAGIGRAPH>,
 MAKE SURE TO USE "/MGF" EXTEN-
 SION *
11Ø PRINT" PRESS ANY KEY WHEN RE
ADY TO      RE-RECORD THE PICTU
RE.
12Ø IF INKEY$=""THEN12Ø
13Ø RETURN
14Ø PRINT" GRAPHIC PAGES NOT SET
 PROPERLY":CLEAR:PCLEAR4
15Ø PRINT" PRESS <RESET> AND TRY
 RUNNING   AGAIN -- IF NO LUCK,
TURN THE   COMPUTER OFF AND RELO
AD THE     PROGRAM.":END
```

EXPANDING BASIC

DISK BASIC

the RAINBOW

COOKING WITH CoCo

Colin J. Stearman

PART III

Having built the utensils, we now start on the recipe to enhance CoCo's Disk Operating System.

Editor's Note:

Due to the considerable interest in this article from users of the new Disk BASIC 1.1, Colin Stearman has done some more "cooking" and has come up with the patch addresses needed. You will find this month's listing indicates the lines which are unique to each revision. The actual assembly shown is for version 1.0, so if you have 1.1 your assembly will look a little different. Next month, the author will explain the differences for you 1.1 owners. (This month's RAINBOW ON TAPE has the patch programs for both 1.0 and 1.1.)

Also, the patched "DIR" command as it stands at the end of this month's revision will give some "garbage" on the screen. This is normal and the real file creation date will appear after Part 5 of this series.

We are now at the point where we can start in earnest modifying CoCo's disk operating system (DOS). We have the capability of saving to disk and reloading a modified DOS (on a 64K CoCo) and we can also save it in an EPROM. Starting this month and for the remainder of this series, I will be presenting an assembly language program to modify or "patch" the DOS to add the desired features described earlier.

The Ground Rules

Before I start on this month's details I think we had better discuss the rules for building each layer of the assembly language "cake." This may be a little tedious but if we all understand the approach now, it'll stop problems from cropping up later.

At the end of the series you will have a complete patch program called *DOSPATCH* which will add all the commands and functions. This program generates a binary file which overlays Disk BASIC, modifying what is already there and adding new code. This month we will develop the foundation of this program and each month add a new section until it is complete. Each month you will be able to assemble the composition so far and use it to patch the DOS to check the functions implemented.

However, it is inevitable that each month we will add some code which is not fully functional because it requires code not destined to be added until a future installment. When this happens we will use a technique called "commenting out," which makes a "comment" of the line of code which cannot yet be made functional. Then later, when the required code is there, we can remove the comment and reassemble to fully activate the feature. In assembly language an asterisk at the start of the line signifies a comment line and the assembler simply ignores the entire line, no matter what its contents.

As you look through Listing 1 you will see lines marked with a reference number in square brackets (for example, [REF 12]). Later in the series we will make some modification to the associated line (most likely remove the asterisk) and I will refer to it by the reference number.

So the best approach is to use your editor to enter the listing exactly as shown. Then each month add the new listing to it, modify the reference lines as described in the text of the article, and reassemble.

The Parallel Port

A final "housekeeping" note before we begin. In a later installment I will be describing a "Centronics" parallel printer port. This month's code contains lines for this purpose. My assembler (*MACRO* by Computerware) allows conditional assembly. This simply means that I can control which lines get assembled and which do not. I use this feature to control the assembly of all the code associated with the parallel port. You will notice a section of code bounded by the following assembler directive lines:

```
IFDF PARPRT
     .
     .
(lines of code)
     .
     .
ENDC
```

This simply means that if a label called *PARPRT* has been defined, then assemble all the bounded lines; otherwise, do not. At the very beginning of the listing the variable *PARPRT* is equated to one, thus defining it and causing the lines to be assembled. If this line were "commented out," the label would not be defined and the lines would not be assembled. If your assembler does not have this feature and you will be building the parallel port, type in the bounded lines of code and leave out the "IFDF" and "ENDC" lines. If you do not intend building it, leave the whole lot out.

Enough of all this mundane detail and on to the assembly language program.

A Strong Foundation

Listing 1 is the base we will build on over the months. It consists of these primary parts:

1) Equates to memory locations and BASIC routines
2) Overlay lines to "hook in" the new code
3) Revisions to existing commands
4) New commands and functions look-up table
5) Installation code for the new commands
6) Parallel port initialization
7) Automatic file startup
8) Dummy commands and functions

Overlays

By using the *ORG* (origin) statement in this section of the code I have patched in various jumps and subroutine calls right into the existing DOS code. This is one of the main techniques for modifying existing commands. The call jumps to our new code and this usually completes the operation replaced by the jump code, then performs the revisions and returns to the original code.

You will also notice two small patches to *DSKI$* and *DSKO$*. These allow a track value up to 40 instead of 35, for use with the revised functions below.

Revisions to Existing Commands

I am sure you have encountered the "bug" in the *PCLEAR* command when used in a program. Maybe you have not come across a similar one in the *FILES* command. Each stem from the same type of error. Both commands have to relocate the BASIC program in memory but they forget to update the parse pointer so that BASIC can continue interpreting your program. The parse pointer points to the next item in your program to be interpreted by BASIC.

The revised code for these functions partly replaces the original code, duplicating much of it. At the crucial point the new pointer is calculated and stored at $A6. Then the old code is used to complete the command. As an added bonus, the revisions to *PCLEAR* allow values of up to 16 instead of the customary eight. No changes have been made to the operation of *FILES* command.

OPEN

The five lines at the label *FILDAT* complete what was happening before the jump and then add the value in the *DATE$* variable to the directory entry. This results in a creation date being stored in the directory every time a new file is created. The date is stored in the first two bytes of the directory entry reserved for future use by Radio Shack. These are bytes 16 and 17, counting from zero. The date is compressed into two bytes by a particular coding method as follows:

```
! FIRST BYTE ! SECOND BYTE !
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
!<---YEAR--->!<MONTH>!<--DAY-->!
```

The year value is stored as the last two digits only. Besides the obvious advantage of saving storage space, this compression technique allows the resulting 16-bit word to be sorted correctly, if this is desired.

When the directory command revisions are complete, the directory will show the creation date along with the usual information. It is very useful to know when a file was created, especially if you have the same file on another disk. Which is the most recent? This modification will tell you.

DIR

There are two revisions to this command. First, the creation date of each file is now displayed and second, the listing pauses after each screen is full, giving time to read it.

The date is displayed as *MM/DD/YY* as part of the directory line. At this time the date will not be displayed correctly because of a missing subroutine called *DATOUT*. The call to it has been commented out in line [REF 5].

When the screen is full the display will halt and wait for any key press. All keys will continue the display, except BREAK, which will terminate the command immediately. The pause will only occur if the output is to the screen. The new *LDIR* command (described in a future installment) uses the *DIR* command but redirects it to the printer. As a result, no pause occurs.

DSKINI

Many of you have disk drives capable of accessing 40 tracks. Even the 35-track Radio Shack drives can usually access 37 tracks. Although the DOS cannot use the tracks above 35, BASIC could make use of them via the *DSKI$* and *DSKO$* commands (suitably modified, of course).

However, to do this, the extra tracks must be formatted and thus the revisions to *DSKINI*. The syntax of the command is now:

DSKINI drive, number of tracks, skip factor

"Drive" is the drive number as usual. "Number of tracks" is any value from 35 to 40. If no value is given, 35 is assumed. "Skip factor" is as described in the DOS manual. If omitted, a skip factor of four is used. Because of the slight revision to this command, if you specify a skip factor you must also specify the number of tracks.

Some acceptable calls include:

DSKINI1	— A normal initialization
DSKINI0,37	— Initialize 37 tracks with skip = 4
DSKINI3,40,2	— Initialize 40 tracks with skip = 2

BACKUP

Similarly, the *BACKUP* command has been modified to include any of the additional tracks from 36 to 40. The new syntax is:

BACKUP source drive [TO destination drive],[tracks]

Therefore, acceptable commands include:

BACKUP0	— backup to a second disk in 0, 35 tracks
BACKUP0,40	— ditto, but all 40 tracks
BACKUP1TO0,37	— backup disk in 1 to disk in 0, 37 tracks

The only requirement for backing up more than 35 tracks is that both disks be previously initialized for at least the number of tracks specified in the command.

KILL

The final command revision is to the file *KILL* command. If this is issued as a direct command then CoCo will check that you are sure you wish to erase it. An uppercase 'Y' is the only response which will result in the file being deleted. All others will cancel the kill. If the disk should have a write protect tab on it, this command will indicate the file was deleted and then return a "Write Protected" error (?WP). The file will still be there.

If the *KILL* command is used from within a BASIC program then no verification is performed. The assumption is that you have thoroughly debugged your program first!

New Commands and Functions

Next comes the command table and its dispatch address table. You will find all the new commands here. These tables are in standard BASIC format with the last character of each command having bit seven set to indicate its end. It is important that the order of the command words and the dispatch table be the same, otherwise you will issue one command and get another! The first command *(COLD)* is tokenized as $E1 with the remainder sequentially from there. The *PARALLEL* command is last because some of you will not need it and this keeps the tokens for all other commands consistent.

Immediately following the command tables are those for the new functions. These start at $A8 and when tokenized are preceded by $FF.

Because all the new functions and commands are established here but the code has not yet been implemented, I have put dummy calls at the end of the listing for each. As a result, BASIC will accept the new words but do nothing. This way you can check the operation of the tables and installing code. When each function is added, these dummy calls will be deleted.

Installation Code

The section of code starting at the label *ADDCOM* is run whenever the CoCo does a cold start (described in a future installment). This code sets up a table in low memory which is used to search for each BASIC command and function as the interpreter encounters them. Microsoft (who wrote this BASIC) kindly set things up so one more table can be added above and beyond the Disk BASIC commands.

At the end of this section is a revision to the "hook" in memory which gets taken when an error is encountered. For now this revision has been "commented out," but later it will allow us to both trap errors and prevent BASIC from halting program execution and also return more meaningful error messages.

Parallel Port Initialization

Continuing the code, which is executed during a cold start, we encounter the parallel port "hook" patch and the initialization routine for the new peripheral interface adapter (PIA) which will run it. If you are not going to use the parallel port, leave this entire section out.

Auto File Execution

Just prior to this, I have put a small reminder indicating who brought you these useful revisions. Then comes a feature which is more powerful than you might at first imagine. Before completing start-up and giving you the OK prompt, the revised BASIC tries to find and run a BASIC file called *AUTOEXEC.BAS* on drive 0. If successful, this program is automatically run. If a disk is present but with no such file on it, then an NF Error is returned. If no disk is in the drive then an I/O Error results.

The power of this feature lies in the fact that you write the *AUTOEXEC.BAS* file and you can put in it anything you want. For example, it could simply be line calling for the running of some other program on the disk. Or perhaps an automatic backup scheme. Listing 2 is designed to request the date and store it in the new memory location for this purpose. I suggest that at the very least you have such a file on your disks.

The power up sequence I have used successfully is:

1) Power up the video monitor
2) Power the Multi-Pak Interface, if you have one
3) Then switch on each disk drive
4) Load the disk with the *AUTOEXEC.BAS* file in drive 0
5) Power up CoCo

I have used this hundreds of times with no problem. After a few seconds the banner will display and drive 0 will turn on. If the file exists it will automatically run.

Now you can get your favorite program running without even touching the keyboard!

The Final Odds and Ends

The code at *COMCOD* and *FUNCOD* is executed during BASIC interpretation to get the address of the code needed to execute the command or function. Then immediately following you will see the dummy calls mentioned earlier.

Testing The Program

64K COMPUTER OWNERS

Testing is very easy for these people. If you did as I suggested last month, you should have a bootable disk with unmodified disk BASIC on it. If so, load it and start.

Once you have BASIC running in the all RAM mode, the procedure is to disable the interrupts, then overlay the patch file and cold start the new BASIC. As all interrupts are generated through one of the PIAs, they can be simply disabled by disabling the PIA. The steps are as follows, once the all RAM BASIC is running.

```
1) POKE &HFF03,&H34:'stop interrupts
2) LOADM"DOSPATCH":POKE&H71,0:
   EXEC&HA027
```

These two lines should be entered as direct commands to BASIC. When complete, a new start-up banner with the revisions copyright notice should be displayed. You should now be able to test all the revised commands implemented so far. Also, all the new commands and functions should be acceptable to BASIC (no SN Error), but of course, they will do nothing.

You could save the revised DOS back to disk, but I recommend you save this until all revisions are completed.

NON-64K COMPUTER OWNERS

For you the testing is a little more difficult. We really do not want to go replacing the DOS ROM (Read only memory) in the disk controller cartridge quite yet. We can however, put the revised code in an EPROM and load it into the socket on the EPROM programmer addressed at $C000.

If you have the Multi-Pak Interface you can fully test the result; if not, then basic functionality can be tested by plugging the EPROM programmer with programmed EPROM inserted into the expansion socket and then trying the commands. Of course, those accessing the disk drives will not work because the controller is not plugged in.

Without the Multi-Pak

From last month, you should already have Disk BASIC saved on tape under filename *DBASIC*. With the disk system operational, do this:

```
CLEAR 200,&H3FFF
CLOADM"DBASIC",&H4000-&HC000+65536
LOADM"DOSPATCH",&H4000-&HC000+65536
CSAVEM"DBASIC#1",&H4000,&H5FFF,&HA027
```

Then power down, plug in the EPROM programmer, and do this:

```
CLEAR 200,&H3FFF
CLOADM"DBASIC#1"
CLOADM"EPROM"
EXEC
```

Then transfer the memory contents from $4000 to $5FFF to a completely erased EPROM.

With Multi-Pak

Program the EPROM following the steps given last month under the subtitle "Using the Programmer with the Disk," but just before doing the *EXEC*, enter:

```
LOADM"DOSPATCH",&H4000-&HC000+65536
```

To test, use the procedure in the same section. But after doing the *POKE65407,3* also enter *POKE&H71,0* and *EXEC&HA027*. This will cold start the new system and allow you to see the automatic file execution feature.

Next Month

We will fill in some of the code for those commands and functions we just added. Also we will add *FLEXIKEY*. This is a keyboard utility which is so useful (even though I say it myself!) that you'll wonder how you ever survived without it!

Finally, if you would like the entire *DOSPATCH* program source (with all future installments), along with binary files with and without the parallel port driver, just send me a disk (no cassettes please) along with $6 and a stamped, addressed disk mailer. I will load the disk and return it to you promptly.

Address this request or any questions to Colin Stearman, 143 Ash Street, Hopkinton, MA 01748.

Looking forward to your company next month.

Listing 1:

```
DOSPATCH - PATCH0001         OPT   COMPUTERWARE MACRO ASSEMBLER  PAGE 1
Patch to RSDOS P0002****************** 3. (C)1984 C.J.STEARMAN
0000          0003 REV    EQU   0
              0004 * Set REV = 0 for DOS 1.0, =1 for DOS 1.1
              0005 **********************************************
              0006 *        RADIO SHACK COLOR COMPUTER DOS      *
              0007 *        IMPROVEMENTS AND MODIFICATIONS      *
              0008 *                                            *
              0009 *         (C)1984 COLIN J. STEARMAN          *
              0010 **********************************************
              0011 *  Patch #1
              0012        OPT   NOG
              0013 ********************
              0014 * COMMENT OUT THE NEXT LINE FOR A SERIAL PORT VERSION
              0015 *(Controls conditional assembly)
0001          0016 PARPRT EQU   1
              0017 ********************
              0018 ****DOS 1.0 PATCH ADDRESSES*******
0000          0019        IFEQ  REV
C028          0020 A001   EQU   $C028
C0D1          0021 A0001  EQU   $C0D1
C106          0022 A0002  EQU   $C10B
C118          0023 A0003  EQU   $C118
C124          0024 A0004  EQU   $C124
C17D          0025 A0005  EQU   $C17D
C570          0026 A0006  EQU   $C570
C575          0027 A0007  EQU   $C575
C58F          0028 A0008  EQU   $C58F
C65F          0029 A0009  EQU   $C65F
C6B8          0030 A0010  EQU   $C6B8
C6CB          0031 A0011  EQU   $C6CB
C6CF          0032 A0012  EQU   $C6CF
C990          0033 A0013  EQU   $C990
CA3B          0034 A0014  EQU   $CA3B
CB4A          0035 A0015  EQU   $CB4A
CBCF          0036 A0016  EQU   $CBCF
CBD5          0037 A0017  EQU   $CBD5
CC26          0038 A0018  EQU   $CC26
CC41          0039 A0019  EQU   $CC41
CC44          0040 A00195 EQU   $CC44
CE2E          0041 A0020  EQU   $CE2E
CEE5          0042 A0021  EQU   $CEE5
D169          0043 A0022  EQU   $D169
D182          0044 A0023  EQU   $D182
D18E          0045 A0024  EQU   $D18E
D1AF          0046 A0025  EQU   $D1AF
D1E5          0047 A0026  EQU   $D1E5
D446          0048 A0027  EQU   $D446
D4AB          0049 A0028  EQU   $D4AB
D4B2          0050 A0029  EQU   $D4B2
D571          0051 A0030  EQU   $D571
D594          0052 A0031  EQU   $D594
D670          0053 A0032  EQU   $D670
D6CD          0054 A0033  EQU   $D6CD
D723          0055 A0034  EQU   $D723
D7DD          0056 A0035  EQU   $D7DD
00E0          0057 HITOKN EQU   $E0
              0058 *  Highest command token in DOS 1.0
              0059        ENDC
              0060 **************
              0061 ****DOS 1.1 PATCH ADDRESSES******
              0062        IFGT  REV
              0063 A001   EQU   $C02C
              0064 A0001  EQU   $C0E4
              0065 A0002  EQU   $C11B
              0066 A0003  EQU   $C12B
              0067 A0004  EQU   $C137
              0068 A0005  EQU   $C190
              0069 A0006  EQU   $C59D
              0070 A0007  EQU   $C5A2
              0071 A0008  EQU   $C5BC
              0072 A0009  EQU   $C6BC
              0073 A0010  EQU   $C6E5
              0074 A0011  EQU   $C6FB
              0075 A0012  EQU   $C6FC
              0076 A0013  EQU   $CA3E
              0077 A0014  EQU   $CAE9
              0078 A0015  EQU   $CC1C
              0079 A0016  EQU   $CCA9
              0080 A0017  EQU   $CCAF
              0081 A0018  EQU   $CD00
              0082 A0019  EQU   $CD1B
              0083 A00195 EQU   $CD1E
              0084 A0020  EQU   $CF0A
              0085 A0021  EQU   $CFC1
              0086 A0022  EQU   $D256
              0087 A0023  EQU   $D26F
              0088 A0024  EQU   $D27B
              0089 A0025  EQU   $D29C
              0090 A0026  EQU   $D2D2
              0091 A0027  EQU   $D534
              0092 A0028  EQU   $D599
              0093 A0029  EQU   $D5A0
              0094 A0030  EQU   $D65E
              0095 A0031  EQU   $D681
              0096 A0032  EQU   $D761
              0097 A0033  EQU   $D7C0
              0098 A0034  EQU   $D816
              0099 A0035  EQU   $D8D0
              0100 HITOKN EQU   $E1
              0101 *   Highest command token in DOS 1.1
```

```
                      ENDC
               *******************
               *
C58F           CHRVCT EQU   A0008      OLD VECTOR JUMP
0080           NTRACK EQU   $80        USE CASSETTE TEMP STORE
               **********************
               * USES UNUSED(?) LOW RAM LOCATIONS
0076           ELINE  EQU   $76        LINE # CAUSING ERROR
00DC           JLINE  EQU   $DC        LINE TO JUMP TO ON ERROR
005A           ECODE  EQU   $5A        ERROR CODE
               **********************
008A           ZERO   EQU   $8A        ZERO CONSTANT 16 BITS
FF26           DATA   EQU   $FF26      PIA DATA REGISTER
A1B1           GETKEY EQU   $A1B1      BASIC'S CURSOR/KEY ROUTINE
B958           RETURN EQU   $B958      OUTPUTS A CARRIAGE RETURN
CC41           SPACE  EQU   A0019      OUTPUT A SPACE
A282           CHROUT EQU   $A282      OUTPUTS CHARACTER IN A
B9A2           STROUT EQU   $B9A2      BASICS STRING OUTPUT X POINTS
               *                     TO STRING, B HAS CHAR COUNT
006F           DEVNUM EQU   $6F        OUTPUT DEVICE NUMBER
01DA           HLDBFR EQU   $1DA       CASSETTE BUFFER FOR HOLD
02DD           BASBFR EQU   $2DD       BASIC BUFFER
01D7           HLDPTR EQU   $1D7       IN CASSETTE FILE NAME BFR
01D8           INSERT EQU   $1D8       DITTO
01D9           WHLINE EQU   $1D9       DITTO
0095           BDFLAG EQU   $95        BAUD RATE LOCATION USED AS
               *                     SERIAL/PARALLEL FLAG
0096           BAUDRT EQU   $96        NORMAL SERIAL BAUD RATE LSB
               * NEXT 3 WORDS ARE IN CASSETTE FILE NAME
01D1           LINNUM EQU   '$1D1      AUTO CURRENT LINE NUMBER
01D3           INCNUM EQU   $1D3       AUTO LINE INCREMENT
01D5           LCOUNT EQU   $1D5       USED IN DIR DELAY
               * there are 4 empty ram locations in the command
               *dispatch table terminator. they are $149/4A and
               *  $14E/F.
0149           AUTOFG EQU   $149
014A           INTFLG EQU   $14A       RAM FLAG FOR REISSUED LINE
014E           DATUM  EQU   $14E       USES TWO BYTES TO STORE DATE
               *
               * This section contains the overlays to patch in
               * the new commands, functions and revisions
               *
               * REMOVE <CR> AFTER BANNER
C17D                  ORG   A0005
C17D 00               FCB   0
               *
               **** PCLEAR PATCH ****
C028                  ORG   A001       SETS TABLE TO A0020 ORIGINALLY
C028 CCD7DD           LDD   #PCLEAR    REPOINT TO NEW ROUTINE
               *
0000                  IFEQ  REV        DOS 1.0
               **** FILES PATCH ****
D0E4                  ORG   $D0E4      PATCH OVER EXISTING CODE
D0E4 7ED824           JMP   FILES      DO EXTRA CODE
                      ENDC
               *
               *** PATCH FOR NEW KEYBOARD ROUTINE ****
C108                  ORG   A0002      SETUP FOR JMP AT $16A
               *         FDB KEYBRD GOES TO NEW KEYBOARD RTN [REF 1]
               *  [REF 1: Uncomment when FLEXIKEY code is installed]
               * DID HAVE A0008, JUMP TO THIS IF DEV CODE<>0
               *
               **** ADD COMMANDS PATCH ****
C0D1                  ORG   A0001
C0D1 7ED991           JMP   ADDCOM
               *
C124                  ORG   A0004
               *   FDB ERCNCL          [REF 2]
               * [REF 2: Uncomment when ERRORS code is installed]
               *PATCH INTO RUN COMMAND TO CANCEL ERROR JUMP
               *A0004 ORIGINALLY HAD A0013
               *******
               *PATCH IN FOR AUTO INPUT
C118                  ORG   A0003
               *    FDB INPUT          [REF 3]
               * [REF 3: Uncomment when AUTO code is installed]
               *A0003 DID HAVE $C6B7 WHICH JUST RETURNED
               ***********************
               ** DO A PAUSE AFTER EACH 15 LINES IN DIR
CBD5                  ORG   A0017
               * INITIALIZE COUNTER
CBD5 BDD886           JSR   NOTBRK
               *
CC26                  ORG   A0018
               * DO PAUSE IN DIR
CC26 BDD849           JSR   LINHLD
               *
               *
               ***********************************
               * PATCH TO ADD DATE TO FILE WHEN OPENING
C570                  ORG   A0006
C570 7ED83C           JMP   FILDAT     PUT DATE INTO FILE
               *******
               * PATCH FOR DSKINI EXTRA TRACKS
               *********************************
D571                  ORG   A0030
D571 9180             CMPA  <NTRACK
               *
D594                  ORG   A0031
D594 9180             CMPA  <NTRACK
               *
D446                  ORG   A0027
               * FIX DSKI$/DSKO$ TO ALLOW UP TO 40 TRACKS
D446 27               FCB   39         TOP TRACK NUMBER
               *
D4AB                  ORG   A0028      FIRST LINE OF DSKINI
D4AB 1603DE           LBRA  DSKINI     GOTO NEW CODE
               * DID HAVE LBEQ $A61F
               *
               *PATCH BACKUP
D1B2                  ORG   A0023
D1B2 7ED8AC           JMP   BCKPAT     BACKUP PATCH
               * RETURN TO A0024
               *
               * THIS PATCHES BACKUP SYNTAX CHANGES
               *    MAKE TRACK COUNT A VARIABLE
D1AF                  ORG   A0025
D1AF 9680             LDA   <NTRACK    WAS LDA #23
               *
               * THIS PATCHES KILL TO CHECK FOR ERASING FILE
C6CB                  ORG   A0011
C6CB 7ED8D4           JMP   KILLCK     DO KILL CHECK CODE
               *
               *****Following patches set the drive step rate
               *Affects all drives, select rate of slowest drive
               *
D6CD                  ORG   A0033      RESTORE step rate
D6CD 02               FCB   2          =20mS;3=30mS;1=12mS;0=6mS

D723                  ORG   A0034      SEEK step rate
D723 16               FCB   $16        =20mS;$17=30mS;$15=12mS;$14=6mS
               ******************************************
               * Patch code to existing commands
               *
               * ALL NEW CODE RESIDES IN THE UPPER
               * AREA OF DISK ROM NOT USED
               * BY DISK BASIC, STARTING AT
               * A0035.
D7DD                  ORG   A0035
               *
               *
               *************************
               * PATCH FIXES THE BUG IN PCLEAR
               *
               *
               *
               * DO ROUTINE, FIX IS TO REVISE PARSER POINTER
               * AT $A6 FOR CHANGE IN LOCATION
               * OF BASIC
               *
D7DD 81C0      PCLEAR CMPA  #$C0       IS IT PCLEAR?
D7DF 1026F64B         LBNE  A0020      NO, EXIT TO PREVIOUS HOOK
D7E3 9D9F             JSR   $9F        PARSE OVER PCLEAR TOKEN
D7E5 BDB70B           JSR   $B70B      GET & EVAL. 1ST ARG.
D7E8 5D               TSTB             IS IT ZERO?
D7E9 274E             BEQ   FCERR      YES, SO ERROR
D7EB C111             CMPB  #17        IT IS >16?
D7ED 244A             BHS   FCERR      YES, ERROR
D7EF B606             LDA   #6         MULTIPLY BY 1536(1 SCREEN)
D7F1 3D               MUL               6*256=1536
D7F2 DBBC             ADDB  $BC        ADD TO START OF
D7F4 1F98      -      TFR   B,A        1SRT GRAPHIC SCREEN
D7F6 C601             LDB   #1
D7F8 1F02             TFR   D,Y        COPY THIS+1 TO Y
D7FA 1093B7           CMPD  $B7        IS THIS PAGE RESERVED?
D7FD 253A             BLO   FCERR      YES, SO ERROR
D7FF 9319             SUBD  $19        SUB. START OF BASIC
D801 1F03             TFR   D,U        SAVE VALUE TEMPORARILY
D803 D31B             ADDD  $1B        ADD END OF BASIC
D805 1F01             TFR   D,X        SAVE NEW END ADDRESS
D807 C300C8           ADDD  #200       STACK SIZE
D80A 9321             SUBD  $21        STACK TOP ADDRESS
D80C 242B             BHS   FCERR      NO ROOM, ERROR
D80E 8D0B             BSR   DIRECT     CHECK IF DIRECT
D810 2706             BEQ   L1         YES SO DIRECT NO FIX
D812 1F30             TFR   U,D        RECOVER OFFSET
D814 D3A6             ADDD  $A6        REVISE PARSER POINTER
D816 DDA6             STD   $A6        AND SAVE IT
D818 7E96B4    L1     JMP   $96B4      CONTINUE PCLEAR ROUTINE
               *
               *THIS CHECKS IF IN DIRECT MODE, Z=1 IF SO
D81B 3410      DIRECT PSHS  X          PRESERVE A AND B
```

```
D81D 9E68                 LDX   $68         GET LINE NUMBER
D81F 3001                 LEAX  1,X         IS IT $FFFF?
D821 3510                 PULS  X           RECOVER D
D823 39                   RTS
                    *************************
0000                      IFEQ  REV         DOS 1.0
                    *  PATCH FIXES A SIMILAR BUG
                    *  IN THE FILES COMMAND
D824 931B           FILES  SUBD  <$1B       END OF BASIC ADDRESS
                    * D HAS OFFSET DUE TO MOVE OF BASIC
D826 3406                 PSHS  A,B         SAVE RESULTS
D828 8DF1                 BSR   DIRECT      CHECK IF DIRECT MODE
D82A 2706                 BEQ   SKIP        YES SO DIRECT COMMAND
D82C ECE4                 LDD   ,S          GET D OFF STACK FIX OFFSET
D82E D3A6                 ADDD  $A6         ADD TO PARSER POINTER
D830 DDA6                 STD   $A6         SAVE IT
D832 3506           SKIP  PULS  A,B         RECOVER OFFSET
D834 D319                 ADDD  <$19        ADD BASIC START ADDRESS
D836 7ED0E8               JMP   $D0E8       CONTINUE FILES CODE
                          ENDC
                    *
D839 7EB44A         FCERR JMP   $B44A       ?FC ERROR
                    *
                    ************************
                    *FILE DATE TO DIRECTORY
D83C B70976         FILDAT STA  $976        FINISH WHAT WAS DOING
D83F A742                 STA   2,U         DITTO
D841 FC014E               LDD   DATUM       GET DATE
D844 ED45                 STD   5,U         PUT INTO BUFFER
D846 7EC575               JMP   A0007       CONTINUE OPENING FILE
                    ************************
                    * DIR command revisions
                    *
                    *
                    ********************************************
                    * directory output of file creation date
D849 3404           LINHLD PSHS B           SAVE GRANULE COUNT
D84B BDBDCC               JSR   $BDCC       OUTPUT IT TO SCREEN
D84E BDCC41               JSR   SPACE       OUTPUT 1 SPACE
D851 3504                 PULS  B           RECOVER GRANULE COUNT
D853 C109                 CMPB  #9          HOW MANY DIGITS?
D855 2203                 BHI   ATCLM       DONT NEED EXTRA SPACE
D857 BDCC41               JSR   SPACE       OUTPUT A SPACE
D85A AE62           ATCLM LDX   2,S         GET DIRECTORY PNTR
D85C EC8810               LDD   16,X        GET DATE FROM DIRECTORY
D85F 3406                 PSHS  D           SAVE VALUE
D861 C608                 LDB   #8          SEE IF ROOM FOR STRING
D863 BDB50F               JSR   $B50F       WONT RETURN IF NOT
                    *X POINTS TO STRING SPACE
D866 3506                 PULS  D           GET DATE AGAIN
                    *       JSR DATOUT PUT DATE IN IT  [REF 5]
D868 C6F8                 LDB   #-8         CHARACTERS TO FIX
D86A A685           OUTCHR LDA  B,X         GET CHARACTER
D86C BDA282               JSR   CHROUT      OUTPUT IT
D86F 5C                   INCB              REDUCE COUNTER
D870 26F8                 BNE   OUTCHR      DO SOME MORE
                    *********************************
                    * DIRECTORY PAUSE TO SCREEN ONLY
                    *
D872 0D6F                 TST   DEVNUM      CHECK IF TO SCREEN
D874 2615                 BNE   CR          DON'T PAUSE IF DIR NOT TO SCREEN
D876 7A01D5               DEC   LCOUNT      DECREASE CURRENT LINE COUNT
D879 2610                 BNE   CR          OUTPUT NEXT LINE
D87B BDA1B1         WAIT  JSR   GETKEY      GET KEYBOARD ENTRY
D87E 27FB                 BEQ   WAIT        IF NONE YET
                    *
D880 8103                 CMPA  #3          IS IT BREAK?
D882 2602                 BNE   NOTBRK      NO
D884 3264                 LEAS  4,S         REMOVE OLD RETURN
                    * AND X LEFT ON STACK
                    *
D886 C610           NOTBRK LDB  #16         REST LCOUNT
D888 F701D5               STB   LCOUNT
D88B 39             CR    RTS
                    ********************
                    ** PATCH DSKINI0 TO FORMAT UP TO 40 TRACKS
                    ** SYNTAX IS DSKINI drive,number of tracks,skip factor
                    ** NUMBER OF TRACKS IS 35 - 40, DEFAULTS TO 35
D88C 1027CD8F       DSKINI LBEQ $A61F       DN ERROR
D890 BDD169               JSR   A0022       CHECK FOR 0-3 DEVICE #
D893 C623                 LDB   #35         DEFAULT # OF TRACKS
D895 9DA5                 JSR   $A5         ANY MORE ON INPUT LINE?
D897 2703                 BEQ   NOVALS      NO MORE VALUES
D899 BDB738               JSR   $B738       GET TRACK VALUE
D89C 8D03           NOVALS BSR  TRKCHK      CHECK FOR VALID DRIVE #
                    *
D89E 7ED4B2               JMP   A0029       RETURN TO REGULAR CODE
                    *
D8A1 C123           TRKCHK CMPB #35         LOWEST LEGAL VALUE
D8A3 2594                 BLO   FCERR       ?FC ERROR
D8A5 C128                 CMPB  #40         HIGHEST LEGAL VALUE
```

```
D8A7 2290                 BHI   FCERR       ?FC ERROR
D8A9 D780                 STB   NTRACK      SAVE IN TEMP BUFFER
D8AB 39                   RTS
                    *
                    ***********************
                    **** PATCH TO BACKUP ****
                    *
D8AC 3404           BCKPAT PSHS B           SAVE SOURCE DRIVE NO.
D8AE C623                 LDB   #35         DEFAULT TRACKS
D8B0 D780                 STB   NTRACK      SAVE DEFAULT VALUE
D8B2 3504                 PULS  B           RECOVER SOURCE DRIVE NO.
D8B4 9DA5                 JSR   $A5         ANY MORE ON LINE?
D8B6 2719                 BEQ   BUPOUT      NO SO EXIT
D8B8 812C                 CMPA  #',         LOOK FOR A COMMA
D8BA 2708                 BEQ   GTTRK       YES SO GET NO OF TRACKS
                    * LOOK FOR "TO" TOKEN
D8BC C6A5                 LDB   #$A5        "TO" TOKEN
D8BE BDB26F               JSR   $B26F       CHECK FOR IT SN ERROR IF NOT
D8C1 BDD169               JSR   A0022.      get second drive and check it
                    *  now we have second drive in b
                    ***********************
                    *  NOW GET NO OF TRACKS
D8C4 3404           GTTRK PSHS  B           PRESERVE SECOND DRIVE #
D8C6 9DA5                 JSR   $A5         ANY MORE ON LINE?
D8C8 2705                 BEQ   BUPEXT      NO SO CONTINUE OLD CODE
D8CA BDB738               JSR   $B738       PARSE , GET VALUE
D8CD 8DD2                 BSR   TRKCHK      FOR VALID DRIVE #
D8CF 3504           BUPEXT PULS B           RECOVER SECOND DRIVE VALUE
D8D1 7ED18E         BUPOUT JMP  A0024       CONTINUE OLD CODE
                    **********************
                    * REVISE KILL ROUTINE TO CHECK FOR ERASURE
                    *
D8D4 BDC65F         KILLCK JSR  A0009       CHECK FOR FILE
D8D7 BDC6B8               JSR   A0010       DID WE GET A MATCH?
                    * WON'T RETURN HERE IF WE DIDN'T
D8DA 3416                 PSHS  X,A,B       SAVE REGISTERS
D8DC BDD81B               JSR   DIRECT      only confirm in direct mode
D8DF 2638                 BNE   NOCNF       Dont confirm delete
D8E1 C60A                 LDB   #10         CHARACTER COUNT
D8E3 8ED8FD               LDX   #CHKMSG     POINT TO MESSAGE
D8E6 BDB9A2               JSR   STROUT      OUTPUT THIS
D8E9 BDA1B1               JSR   GETKEY      GET ANSWER
D8EC BDA282               JSR   CHROUT      OUTPUT IT
D8EF 3402                 PSHS  A           SAVE IT
D8F1 BDB958               JSR   RETURN      OUTPUT A CR
D8F4 3502                 PULS  A           GET RESPONSE
D8F6 8159                 CMPA  #'Y         IS IT YES
D8F8 2714                 BEQ   CONFRM      CONFIRM DELETION
D8FA 3516                 PULS  X,A,B
D8FC 39                   RTS               EXIT AND DON'T DELETE
D8FD 53             CHKMSG FCC  %SURE(Y/N)?%
D907 44             CNFMSG FCC  /DELETED/
                    *
D90E 8ED907         CONFRM LDX  #CNFMSG     POINT TO CONFIRM MESSAGE
D911 C607                 LDB   #7          CHARS  IN IT
D913 BDB9A2               JSR   STROUT      OUTPUT THIS
D916 BDB958               JSR   RETURN      PLUS A CR
D919 3516           NOCNF PULS  X,A,B       RECOVER REGS
D91B 7EC6CF               JMP   A0012       CONTINUE KILL COMMAND
                    ***************************************
                    * COMMAND TABLE AND JUMP CODE
                    *
                    *
                    ********************************************
                    *  ADDED BASIC COMMANDS AND FUNCTIONS      *
                    ********************************************
                    *
                    *
                    *     COMMAND TABLE
                    *
D91E 43             COMTBL FCC  /COL/
D921 C4                   FCB   'D+128
D922 57                   FCC   /WPOK/
D926 C5                   FCB   'E+128
D927 46                   FCC   /FAS/
D92A D4                   FCB   'T+128
D92B 53                   FCC   /SLO/
D92E D7                   FCB   'W+128
D92F 58                   FCC   /XE/
D931 D1                   FCB   'Q+128
D932 41                   FCC   /AUT/
D935 CF                   FCB   'O+128
D936 53                   FCC   /SWA/
D939 D0                   FCB   'P+128
D93A 45                   FCC   /ERROR/
D93F D3                   FCB   'S+128
D940 42                   FCC   /BAU/
D943 C4                   FCB   'D+128
D944 4C                   FCC   /LDI/
D947 D2                   FCB   'R+128
```

```
               Ø466         IFDF  PARPRT      ASSEMBLE FOR PARALLEL PORT
               Ø467 *KEEP THIS LAST IN LIST FOR TOKEN COMPATABILITY
D948 5Ø        Ø468         FCC   /PARALLE/        [REF 6]
D94F CC        Ø469         FCB   'L+128                [REF 7]
               Ø47Ø         ENDC
               Ø471 * [REF 6 & 7: If no conditional assembler and
               Ø472 *  parallel port is used, delete IFDF and ENDC
               Ø473 *  lines. If not used, delete all 4 lines.]
               Ø474 *
               Ø475 *****
               Ø476 *    COMMAND JUMP TABLE
               Ø477 * MUST BE IN SAME ORDER AS COMMANDS
               Ø478 *
D95Ø           Ø479 CTABLE EQU   *           TABLE START
D95Ø DA4E      Ø48Ø COMDSP FDB   COLD        COLD RESTART
D952 DA4F      Ø481        FDB   WPOKE
D954 DA5Ø      Ø482        FDB   FAST
D956 DA51      Ø483        FDB   SLOW
D958 DA52      Ø484        FDB   XEQ
D95A DA53      Ø485        FDB   AUTO
D95C DA55      Ø486        FDB   SWAP
D95E DA54      Ø487        FDB   ERRCMD
D96Ø DA56      Ø488        FDB   BAUD
D962 DA57      Ø489        FDB   LDIR        PRINT DIRECTORY
               Ø49Ø *
               Ø491 *KEEP THIS LAST IN LIST FOR TOKEN COMPATABILITY
               Ø492         IFDF  PARPRT      ASSEMBLE FOR PARALLEL PORT
D964 DA58      Ø493         FDB   PARA              [REF 8]
               Ø494         ENDC
               Ø495 * [REF 8: If no conditional assembler and
               Ø496 *  parallel port is used, delete IFDF and ENDC
               Ø497 *  lines. If not used, delete all 3 lines.]*
D966           Ø498 CTBLEX EQU   *           TABLE END
               Ø499 *****
ØØØB           Ø5ØØ NUMCMD EQU   (CTBLEX-CTABLE)/2 NO. OF CMDS
               Ø5Ø1 ********************************************
               Ø5Ø2 *   FUNCTION TABLE
               Ø5Ø3 *
D966 53        Ø5Ø4 FUNTBL FCC   /SCAN/
D96A A4        Ø5Ø5        FCB   '$+128
D96B 44        Ø5Ø6        FCC   /DATE/
D96F A4        Ø5Ø7        FCB   '$+128
D97Ø 45        Ø5Ø8        FCC   /ELIN/
D974 C5        Ø5Ø9        FCB   'E+128
D975 45        Ø51Ø        FCC   /ECOD/
D979 C5        Ø511        FCB   'E+128
ØØØØ           Ø512        IFEQ  REV
D97A 45        Ø513        FCC   /ENAME/
D97F A4        Ø514        FCB   '$+128
               Ø515        ENDC
D98Ø 57        Ø516        FCC   /WPEE/
D984 CB        Ø517        FCB   'K+128
               Ø518 *****
               Ø519 *   FUNCTION JUMP TABLE
               Ø52Ø *
D985           Ø521 NTABLE EQU   *           FUNCTION TABLE START
D985 DA59      Ø522 FUNDSP FDB   SCAN
D987 DA5A      Ø523        FDB   DATE
D989 DA5B      Ø524        FDB   ERRLIN
D98B DA5C      Ø525        FDB   ERRCOD
ØØØØ           Ø526        IFEQ  REV
D98D DA5D      Ø527        FDB   ERNAME
               Ø528        ENDC
D98F           Ø529 ARGMRK EQU   *
               Ø53Ø * put all functions without an argument above
               Ø531 * this equate
D98F DA5E      Ø532        FDB   WPEEK
D991           Ø533 NTBLEX EQU   *           TABLE END
               Ø534 *******
ØØØ6           Ø535 NUMFUN EQU   (NTBLEX-NTABLE)/2 NO. OF FUNCTS
               Ø536 ********************************************
               Ø537 * THIS IS EXECUTED DURING STARTUP
               Ø538 *
               Ø539 * Output revision banner
D991 8EDAØF    Ø54Ø ADDCOM LDX   #BANNER-1   POINT TO BEFORE BANNER
D994 BDB99C    Ø541        JSR   $B99C       USE BASIC'S OUTPUT ROUTINE
               Ø542 ***********************
D997 7FØ149    Ø543        CLR   AUTOFG       SET UP FOR NO AUTO
D99A 7FØ14A    Ø544        CLR   INTFLG       OLD LINE REPEAT FLAG
D99D 8ED91E    Ø545        LDX   #COMTBL     POINT X TO COMMAND TABLE
D9AØ CEØ13E    Ø546        LDU   #$13E       START OF COMMAND VECTOR TABLE
D9A3 AF41      Ø547        STX   1,U         SAVE COMMAND TABLE ADDRESS
D9A5 86ØB      Ø548        LDA   #NUMCMD     GET NUMBER OF COMMANDS
D9A7 A7C4      Ø549        STA   ,U          SET IT IN TABLE
D9A9 8EDA28    Ø55Ø        LDX   #COMCOD     COMMAND CODE
D9AC AF43      Ø551        STX   3,U
               Ø552 ************
D9AE 86Ø6      Ø553        LDA   #NUMFUN     GET NUMBER OF FUNCTIONS
D9BØ A745      Ø554        STA   5,U         SAVE IT IN TABLE
D9B2 8ED966    Ø555        LDX   #FUNTBL     GET FUNCTION TABLE ADDRESS
D9B5 AF46      Ø556        STX   6,U         SAVE IT IN TABLE
D9B7 8EDA37    Ø557        LDX   #FUNCOD     GET FUNCTION CODE ADDRESS
D9BA AF48      Ø558        STX   8,U
D9BC 6F4A      Ø559        CLR   1Ø,U        SET END OF TABLES FLAG
D9BE 8EB277    Ø56Ø        LDX   #$B277      ?SN ERROR
D9C1 AFC812    Ø561        STX   18,U        STORE IN NEXT HOOK SLOTS
D9C4 AF4D      Ø562        STX   13,U        FOR COMS & FUNCT.
D9C6 6F4F      Ø563        CLR   15,U        SET TOKEN GROUP TO ZERO
D9C8 9E8A      Ø564        LDX   ZERO
D9CA AFC81Ø    Ø565        STX   16,U        CLEAR DATUM
               Ø566 * JSR RESET ERROR TRAP VALUES   [REF 9-1]
               Ø567 * REDIRECT ERRORS TO ERRTRP BY CHANGING JUMP ADDRESS
               Ø568 *AT $18F
               Ø569 *   LDD #ERRTRP     [REF 9-2]
               Ø57Ø *  STD $18F         [REF 9-3]
               Ø571 * [REF 9: Uncomment when ERRORS code is installed]
               Ø572
               Ø573 *
               Ø574         IFDF  PARPRT      DO FOR PARALLEL [REF 1Ø]
               Ø575 * [REF 1Ø & 11: If no conditional assembler and
               Ø576 *  parallel port is used, delete IFDF and ENDC
               Ø577 *  lines. If not used, delete these and
               Ø578 *  all lines in between.]
               Ø579 * REDIRECT CALLS FOR OUTPUT VIA [AØØ2] A282
               Ø58Ø * TO ALLOW PARALLEL PORT OPERATION.
D9CD CCDA5F    Ø581        LDD   #PAROUT     PARALLEL PORT ROUTINE
D9DØ FDØ168    Ø582        STD   $168
               Ø583 * NOW INITIALIZE PARALLEL PORT
               Ø584 ********************************************
               Ø585 *  BASIC PATCH FOR PARALLEL OUTPUT
               Ø586 ********************************************
               Ø587 *
               Ø588 *
               Ø589 * THE UART BAUD RATE MSB ($95) IS SET TO 1 TO
               Ø59Ø * ACTIVATE THE PARALLEL INTERFACE.  SET TO ZERO
               Ø591 * FOR THE SERIAL OUTPUT. THIS MEANS 3ØØ BAUD AND
               Ø592 * HIGHER WILL ACTIVATE THE SERIAL PORT, 11Ø OR LOWER
               Ø593 * WILL ACTIVATE THE PARALLEL PORT.
               Ø594 * THIS IS THE DEFAULT CONDITION.
               Ø595 ****************************************
               Ø596 * PIA LAYOUT
               Ø597 *                BIT Ø UNUSED INPUT
               Ø598 *                BIT 1 UNUSED INPUT
               Ø599 *                BIT 2 UNUSED INPUT
               Ø6ØØ *      FF24      BIT 3 UNUSED INPUT
               Ø6Ø1 *                BIT 4 UNUSED INPUT
               Ø6Ø2 *                BIT 5 UNUSED INPUT
               Ø6Ø3 *                BIT 6 UNUSED INPUT
               Ø6Ø4 *                BIT 7 PRINTER BUSY=1
               Ø6Ø5
               Ø6Ø6 *     FF25       SET TO $4 FOR ALL INPUTS
               Ø6Ø7
               Ø6Ø8 *                BIT Ø PARALLEL OUTPUT
               Ø6Ø9 *                BIT 1 PARALLEL OUTPUT
               Ø61Ø *                BIT 2 PARALLEL OUTPUT
               Ø611 *      FF26      BIT 3 PARALLEL OUTPUT
               Ø612 *                BIT 4 PARALLEL OUTPUT
               Ø613 *                BIT 5 PARALLEL OUTPUT
               Ø614 *                BIT 6 PARALLEL OUTPUT
               Ø615 *                BIT 7 PARALLEL OUTPUT
               Ø616
               Ø617 *     FF27       SET TO $2C FOR OUTPUTS & CB2
               Ø618
               Ø619 *BUSY IS ALSO CONNECTED TO CB1 BUT NOT USED
               Ø62Ø *PIA DETECTS BUSY TO NOT BUSY TRANSITION
               Ø621 *
               Ø622 * SET UP PIA FOR PARALLEL PORT
D9D3 8EFF26    Ø623        LDX   #DATA       POINT X TO PIA
D9D6 86FF      Ø624        LDA   #$FF
D9D8 A784      Ø625        STA   ,X          SET DATA DIRECTION REG TO $FF
D9DA 862C      Ø626        LDA   #$2C        SET FOR AUTO STROBE
D9DC A7Ø1      Ø627        STA   1,X         CONTROL REGISTER
D9DE 86Ø4      Ø628        LDA   #$4         SET UP BUSY PIA
D9EØ A71F      Ø629        STA   -1,X        POINT FF24 TO DATA REG
               Ø63Ø * SET UP OF PIA COMPLETE
               Ø631 * SET UP DEFAULT BAUD RATE
D9E2 CCØ1CA    Ø632        LDD   #$1CA       BASICS 12Ø BAUD
D9E5 DD95      Ø633        STD   BDFLAG      SET VALUE
               Ø634        ENDC              END CONDITIONAL    [REF 11]
               Ø635
               Ø636 **********************
               Ø637 *RUN AUTOEXEC FILE
               Ø638 *
D9E7 8EDAØ2    Ø639        LDX   #AUTFIL     POINT X TO COMMAND LINE
D9EA CEØ2DD    Ø64Ø        LDU   #$2DD       BASIC INPUT BUFFER
D9ED C6ØE      Ø641        LDB   #FILEND-AUTFIL NUMBER OF CHARACTERS
D9EF 3444      Ø642        PSHS  B,U         SAVE COUNT AND BUFFER PNTR
D9F1 BDA59A    Ø643        JSR   $A59A       MOVE X TO U B BYTES
D9F4 8655      Ø644        LDA   #$55        WARM FLAG
D9F6 9771      Ø645        STA   $71         SET IT
D9F8 BDB95C    Ø646        JSR   $B95C       SET O/P PARAMETERS
D9FB 3514      Ø647        PULS  B,X         CHAR COUNT $ BUFR PTR IN X
D9FD 3Ø1F      Ø648        LEAX  -1,X        BACK OFF POINTER
D9FF 7EAC7F    Ø649        JMP   $AC7F       STARTUP BASIC
```

```
                Ø65Ø *RETURN TO BASIC ROM
DAØ2 52         Ø651 AUTFIL FCS   /RUN"AUTOEXEC"/ Ø BYTE ENDED
DA1Ø            Ø652 FILEND EQU   *
DA1Ø 52         Ø653 BANNER FCS   /REV(C)1984 C.STEARMAN(ØD)(ØD)/
                Ø654 ****************************************
                Ø655 *   COMMAND CODE
                Ø656 *This is executed during token interpretation
                Ø657 * to jump to correct code
                Ø658 *
DA28 81E3       Ø659 CONCOD CMPA  #HITOKN+NUMCMD HIGHEST LEGAL CODE
DA2A 23Ø3       Ø66Ø        BLS   GOODVL       GOT A GOOD VALUE
                Ø661 *
DA2C 7EB277     Ø662 SNERR  JMP   >$B277       ?SN ERROR JUMP
                Ø663 *
DA2F 8ED95Ø     Ø664 GOODVL LDX   #COMDSP      POINT TO DISPATCH TABLE
DA32 8ØE1       Ø665        SUBA  #HITOKN+1    LOWEST TOKEN IN RANGE
                Ø666 * MAKES A HAVE OFFSET INTO DISPATCH TABLE
DA34 7EADD4     Ø667        JMP   >$ADD4       CALCULATE AND EXECUTE IT
                Ø668 ****************************************
                Ø669 *  FUNCTION CODE
                Ø67Ø *This is executed during token interpretation
                Ø671 * to jump to correct code
                Ø672 *
DA37 C15A       Ø673 FUNCOD CMPB  #$4E+(2*NUMFUN)
DA39 22F1       Ø674        BHI   SNERR        BAD CODE
DA3B CØ5Ø       Ø675        SUBB  #$5Ø         LOWEST FUNCTION NUMBER
DA3D C1Ø8       Ø676        CMPB  #ARGMRK-NTABLE-2 Number of functions not
                Ø677 * requiring an arguement, X 2 +2
                Ø678 *
                Ø679 *ACTUAL TOKEN IS 5Ø/2 + BØ = A8
DA3F 2FØ7       Ø68Ø        BLE   NOARG        FIRST FUNCTIONS HAVE
                Ø681 *                          NO ARGUMENT
                Ø682 *ALL OTHERS DO AND ITS OBTAINED
                Ø683 * FIRST HERE
DA41 34Ø4       Ø684        PSHS  B            SAVE TOKEN OFFSET
DA43 BDB262     Ø685        JSR   $B262        EVAL BRACKETTED ARGUMENT
DA46 35Ø4       Ø686        PULS  B            RESTORE OFFSET
DA48 8ED985     Ø687 NOARG  LDX   #FUNDSP      POINT TO FUNCT. DISPATCH TABLE
DA4B 7EB2CE     Ø688        JMP   $B2CE        GO LOOKUP AND JUMP
                Ø689 ****************************************
DA4E 39         Ø69Ø COLD   RTS                      [REF 12]
DA4F 39         Ø691 WPOKE  RTS                     [REF 13]
DA5Ø 39         Ø692 FAST   RTS                      [REF 14]
DA51 39         Ø693 SLOW   RTS                      [REF 15]
DA52 39         Ø694 XEQ    RTS                       [REF 16]
DA53 39         Ø695 AUTO   RTS                      [REF 17]
DA54 39         Ø696 ERRCMD RTS                     [REF 18]
DA55 39         Ø697 SWAP   RTS                      [REF 19]
DA56 39         Ø698 BAUD   RTS                      [REF 2Ø]
DA57 39         Ø699 LDIR   RTS                      [REF 21]
DA58 39         Ø7ØØ PARA   RTS                      [REF 22]
DA59 39         Ø7Ø1 SCAN   RTS                      [REF 23]
DA5A 39         Ø7Ø2 DATE   RTS                      [REF 24]
DA5B 39         Ø7Ø3 ERRLIN RTS                     [REF 25]
DA5C 39         Ø7Ø4 ERRCOD RTS                     [REF 26]
DA5D 39         Ø7Ø5 ERNAME RTS                     [REF 27]
DA5E 39         Ø7Ø6 WPEEK  RTS                      [REF 28]
DA5F 39         Ø7Ø7 PAROUT RTS                     [REF 29]
                Ø7Ø8
                Ø7Ø9
                Ø71Ø
DA5F            Ø711 ZZLAST EQU   *-1          last used address value
                Ø712 *
                Ø713 * ZZLAST must not be greater than $DFFF for
                Ø714 * DOS 1.Ø and $DEFF for DOS 1.1.   The latter
                Ø715 * has the OS-9 Boot program and SWI set routines
                Ø716 * from $DFØØ to $DF4C
                Ø717 *
                Ø718 *
                Ø727        OPT   LIS
D991            Ø728        END   ADDCOM
        NO ERROR(S) DETECTED
```

Listing 2:

```
10 'DATE LOADER
11 DIM DAYS(12)
12 DATA 31,28,31,30,31,30,31,31,30,31,30,31
13 FOR I=1 TO 12
14 READ DAYS(I)
15 NEXT
30 INPUT"DATE(MM,DD,YY)";M,D,Y
50 IF M<0 OR M>12 THEN 1000
70 IF Y<0 THEN 1000
80 IF D<1 THEN 1000
90 IF M=2 THEN 120
100 IF D>DAYS(M) THEN 1000 ELSE 150
110 ' DO FEBRUARY
120 IF(INT(Y/4)<>Y/4)AND(D>DAYS(M))THEN 1000
130 ' LEAP YEAR
140 IF D>29 THEN 1000
150 DATE =(Y*INT(2^9))+(M*INT(2^5))+D
160 WPOKE &H14E,DATE
170 END
1000 PRINT"ERROR":GOTO30
```

GOCO

No matter which way I approach GoCo this month I find my words sounding negative and the real life situation is far from negative. None the less there are problems with this magazine, which is something that I'm sure you've guessed from the issues you've seen.

Problem number one is the continuing non arrival of PCM Magazine from the States. Last month was a case in point. We made up a "stand in" Magazine just in case PCM didn't turn up, and it was just as well, because it didn't! We didn't feel as bad as we might because Micheal Patkin and Bob Delbourgo had contributed good quality Australian articles. Australian articles in fact would be the saviour of the magazine, if enough were sent.

Problem number two, the problem Greg was wrestling with, is that this is very much a magazine which will take time to develop a viable number of readers. Until then it depends on Rainbow to keep it alive.

Problem number three is my own. There just hasn't been time to organize an economic method of getting you your Bar Code listings. The intent is to send photocopies to those who indicate that they want them, but getting time to do that has been a problem. That's one I will have to solve.

That's the negative, now the positive. I want very much to see this magazine go well. I happen to like both the computers that it supports. I started my computing life with a Tandy PC1, a 1.5K honey that was very useful. I was sorry to see it's demise. So to an extent, I have had experience with portable computers and am aware of just how useful they can be to people on the move.

And the Model 2000 is a favourite because I've seen its potential, particularly in the business community. One of the joys of this last month was finding out about the Maze program mentioned in the Education pages.

There are at least 90 model 100 owners reading this magazine, and we get regular content from 3 or 4 authors only. How about making this magazine work like Australian CoCo/MiCo. MiCo is a simular magazine to GoCo in terms of subscriber numbers, yet we hear I'll wager, from just about every MiCo owner at least twice a year. And they are a tenacious lot. They will not allow anything to interfere with their magazine. They do something about it though - their programs jam my post box every Monday and Tuesday morning. In fact if I have to get a bigger post box it'll be because of them!

I love a challenge, and if we find that we are getting strong Aussie and NZ input, then we'll just have to look sooner at ways of improving the presentation of this magazine.

Telecommunicating With PoCo

From Mind Games To A Researcher's Dream

Randy Graham

I am still mentally on vacation. Let's play a little mind game before we settle back down to serious work. I have three wondrous objects with magical properties with which to dazzle your imagination. This is a word game — no pictures. As I describe these objects, see if you can guess what they are. Ready?

The first is a little piece of solid metal about three-inches long, an inch wide and a quarter-inch thick. The first inch has been cut into a circular shape and has a hole in the end. The rest of the length has been cut down to about a half-inch wide. Transverse grooves this is part of the magic — have been cut along its length, and a random series of notches has been cut into one edge by a mumbling wizard.

How can I use this magic talisman? Well, all over the world, in every town and village, on every college and university campus, there are special buildings where the knowledge and wisdom of the world is collected into a series of hard-copy random access files, neatly shelved and stored for easy access. With my magical object, I can enter any of these buildings and browse among their files to my heart's content. What do you think of that?

Is this all there is to your word game? Boy, the summer heat sure has gotten to your brain. So you have a master key to all the world's libraries — that is not enough to make you wise or knowledgeable. I will give you half a point for describing a book as the original random access file. It was a great advance over the scroll which for years limited people to serial access of knowledge.

But there are two reasons I am not impressed with your magic. One is that all of the buildings are open to the public anyway. The only benefit of having your own key is that you could go in nights and weekends when it was less crowded. The real problem is time and money. You still must go to the libraries wherever they are, city by city, campus by campus. You will not live long enough to explore them all and digest all their contents. It is one of the limitations of modern scholarship. One of your other magic objects had better be a flying carpet.

Not impressed, huh? OK, let me bring out my next one. This is a piece of molded plastic about eight-inches long, two-inches wide and an inch thick. It has an earphone molded into one end and a microphone in the other end. Between these parts is a set of numbered keys. A small wire runs from it to the wall, around the house, and out to the world where it is joined to many other wires.

Now, I happen to know that in each of these shrines of knowledge which you call libraries, there is a group of professional devotees who care for the collection and know it intimately. By pressing a series of numbers on my object, I can talk to them directly and get them to look up and share the information for me; it is no longer necessary to travel to the place.

OK, you have a telephone and can call the librarians. Now you are limited to their available time and patience. No librarian is going to go through the whole catalog, find everything you want and read it to you over the phone. Granted, as time permits, many of them will research your questions and may even mail you materials if it is not too valuable to circulate. Time and money; time and money. You have not yet solved your problem.

Well, I guess I had better get out my real whizbanger and destroy your cynicism once and for all. Look at this: a plastic box 8½" x 11" x 2", weighing in at three pounds. Around three of its edges are a series of sockets, into one of which is plugged that same wire which runs to the outside world. On top, a nice keyboard and LCD display . . .

Are you trying to tell me that your PoCo can take you right into a library's innards where all the catalogs and books are electronically accessible?

Close, but no cigar -- yet. That day is still a glow on the horizon. It is coming; it is almost here. Libraries are computerizing their card catalogs as fast as they can. And as fast as they can install the equipment, they are putting terminals out for patrons to use in the library. And they are beginning to set up interlibrary networks, so far used only by the librarians.

I was recently given the opportunity to access such a network as an experiment to see how effectively it could be used by a nonlibrarian. It was like a visit to Tomorrowland. I called the local library, gave the access code and searched the catalog. The only advantage now is that I knew what I wanted and what was availabie before setting out for the building. If I could have accessed several catalogs from home, I could have set out with a shopping list and an itinerary.

As I said, they are setting up interlibrary networks to allow librarians to locate requested items and to arrange interlibrary loans. There may be a library in your town which is already on such a network. But you had still better make sure that the librarian has a sense of humor before you go up to the desk and ask for all the books in the world about penguins. However, if you are trying to locate an out-of-print book for a legitimate research project, help may be as close as your nearest university library.

Books online will probably have to wait for videodisks and automated retrieval and a few other enhancements. Those of us in the Grecian Formula generation may never see that day, but it is hazardous to make rash forecasts in this business. After all, some of the people who worked on the first real computers in the plugboard-and-relay days are still alive and very active. (If you ever get a chance to hear Capt. Grace Hopper, USN, talk about those days, cancel all appointments and go!)

But back to the PoCo. So far in this article, I have been trying to mentally condition you to meet such a gigantic new world of information that it will boggle your mind — if you are who I think you are. I think that you are a professional who bought a Model 100 as a working tool, or you were lucky enough to have one bought for you by your employer because it would increase your professional effectiveness. And I think you have found it to be such; I think you are going around telling everyone that you don't know how you ever managed without it. I think you are less interested in playing computer games or going to computer club meetings than you are in writing and research. If I have come even close to describing you, then you are the typical PoCo user

If you are engaged in scholarly and/or scientific work at a professional level, then you regard yourself as knowledgeable in your field. And this professional competence can never be a static status. You are continually reading, writing, and going to seminars and workshops, "keeping abreast." Books aren't so bad; you read them all and quickly gain insight into the author's new wrinkle, you read the reviews and buy selectively or check them out of the library. You run over the footnotes and bibliographies to see if he has seen anything you missed.

It's the journals that kill us. They keep coming in the mail and piling up on the desk. We religiously keep the pile close by, scan the contents and sample the articles. You make notes, insert markers, scribble flags on the cover and hope you remember them all. Your competence is tested whenever you need an article published somewhere by somebody a few years ago in one of the journals. Remember the maddening Master-of-Everything at school who could reach over and pull out an article published 10 years ago in the Zimbabwe Journal of Knowledge, and use it to rebut your latest argument? If we could only remember *everything*!

It gets worse when you start publishing — you know all this, don't you? The first thing -- well no, the first thing is to get a grant. But then, the first task in scientific and scholarly research is to "search the literature." That phrase strikes fear into every noble heart. The footnotes and bibliographies in articles and books are not really intended to inform or educate; they are designed to prove to critics and reviewers that the author has *searched the literature*. References in the text, such as "Contrary to what Sillybum said in the Zeitgeist *Festschrift* . . ." are supposed to be further proof. Every researcher has nightmares about getting an article back with a note from the editor that "We feel this subject was adequately covered by Bumbleman in his recent article in the JABC."

And so we spend all those hours and days in the library with the indexes and abstracts and footnotes, trying to insure that we have not missed anything important. Is there any way to cut down on this time so that we can get on with the project the whole world and our boss is waiting for? YES!

That is why I have made you go on this walking tour of familiar old battlefields known to every researcher, some of them stained with our own blood. I can not yet give you the magic formulae to bring the books of the world into your living room, but how about a series of huge catalogs which index and abstract all the journal articles, symposia, research reports and other occasional publications which have ever been published anywhere by anyone in your field — all available from your desk? Now are you dazzled and bemused? What if I throw in a bonus that will allow you, with a few keystrokes, to order that reprints of any of these exotic materials be sent to you? Is that worth some time and effort — and no small expense? OK, then let's go.

This is where I have been planning to take you since this series began. I have tried to persuade you that you have on your desk or in your briefcase a telecommunications device for which Galileo or Newton would have killed to possess. I urged you to get some familiarity with it in inexpensive ways by accessing bulletin boards, and to have some fun with it on The Source and CompuServe and Delphi. Now it is time to graduate from the Piper Cub and the 747 to the Space Shuttle. No more "bus tours" of information "shopping malls." The next stop is the launching pad.

You see, there are right now, as you read this, several gigantic information utilities that are prepared to deliver what I have been describing. Heretofore they have been used by librarians. All students of library science are now spending their time in the computer room and all librarians now practicing their profession are going back to school to master computer applications. I make that sweeping statement with very little fear of contradiction. They have all found that electronic information retrieval is where they will spend the rest of their days.

You may have already met this new science. You may have the support of a library which is using the online information services. If so, you have depended on the expert knowledge of the librarian/information retrieval specialist. But then if you are lucky, you have always been able to utilize the expertise of these professionals. In the old print days, if you asked a funny question, the librarian turned to the special shelf of books behind her/him and browsed reflectively, maybe called another librarian somewhere for help. Or, you may have been able to go to your company librarian and say, "I have been assigned to develop a brandy-flavored bubble gum for the Young Executive market. See what you can find on brandy, bubble gum and the Young Executive market." That request would take some time and produce notes, bibliographies and some books and journals with pages marked.

Now you will find that your information specialist (we all have new names) may want to interview you, perhaps using a printed form. What journals, what years, exactly what are you looking for. He/she is designing a "search." The librarian of today knows the databases; she/he does not know your field. This helper may come back with some printout for further discussion -- "refining the search." Finally, and more quickly than by a physical search, the material is before you. Whether or not it is what you need and want depends mostly on how well you and the librarian were able to communicate.

Can we shorten the process even more? What about all the time and effort two professionals spent trying to communicate? Why not teach the "end user" (this is the latest buzz word, even if it makes you sound like you are terminally ill; the end user of the hamburger is the guy who eats it) — I say, why not teach the end user to do his/her own searching and information retrieval? This is the current, latest, hottest trend in information services. And there you are, right up at the boundary with your PoCo. Yessir, we are going to remodel you into a sort of semi-pro librarian, so that when you want it, you can go and get it. Now, if that isn't magic, then my bag of tricks is empty.

And so, go back and complete all unfinished assignments. And next time be prepared to meet the big boys. PCM

Making Pies

Dave Pifer

Whatever your reason may have been for buying a Tandy 2000, you must admit that its graphics capabilities had some impact on you. Tandy put together two outstanding demonstration programs for the 2000. One of them you saw in action at your local computer center and the other was included on the DOS disk supplied with the computer

The program included with your computer, *GRAPHICS.BAS*, generates a nice display of pie charts, bar charts, and line graphs. But, if you examine a listing of *GRAPHICS.BAS*, you will see that the programmers simply plotted every point, line, and circle specifically for the demonstration

It doesn't take long to realize that designing a similar program, on your own, would involve a lot of work and a thorough understanding of the BASIC's complex graphic statements. Clearly, this is far too time-consuming an exercise to go through every time you want to create a new pie chart.

PIEPLOT.BAS is a program to generate custom pie charts on the Tandy 2000 with the high-resolution color graphics option. *PIEPLOT* accepts input from the user in simple terms and does all the complex plotting and display calculations automatically.

Capabilities

PIEPLOT is capable of drawing a pie chart in increments of 1 percent. In theory, you can design a chart with a hundred 1 percent slices, although such a chart would be too cluttered to be of any real use. At the other extreme, you could have one 100 percent slice. Percentages less than 1 percent are rejected during the input routine. Fractional percentages are accepted.

PIEPLOT keeps track of the percentage of the total chart used and goes to its plotting routines as soon as 100 percent is reached or exceeded. If your last percentage input pushes the total over 100 percent *PIEPLOT* adjusts it to bring the total down to 100 percent.

Eight colors are available and are assigned to the individual slices during the input cycle. Color selections outside the zero to seven range are rejected.

Individual slices of the pie can be titled during the user input routine. Titles can be as long as 12 characters. Longer titles are rejected. *PIEPLOT* draws a short line from the title of the slice to the border of the slice itself.

Slices of the chart can be highlighted by pulling them out and away from the main body of the pie. This effect is achieved by specifying a negative percentage during the input routine.

How to use *PIEPLOT*

PIEPLOT is a pure BASIC program so it can simply be run from BASIC's OK prompt by typing: RUN"PIEPLOT.BAS". Similarly, it can be loaded and run from the DOS A> prompt by typing: BASIC PIEPLOT.BAS.

Once *PIEPLOT* is running, the screen clears and eight color blocks are drawn down the left side of the screen. These are the colors available to design a chart. Colors are specified by the one digit number (0 through 7) shown beside block.

Next, *PIEPLOT* asks you for the title of the chart. You could specify up to an 80 character name, but shorter names are more appealing. This title will be displayed on the first line, centered at the top of your chart.

You are now ready to begin defining the chart itself. *PIEPLOT* displays the percentage of chart remaining: starting at 100 and decrementing as you add slices. Then you are prompted to enter the percentage, color code, and title for the individual slices. Repeat this process until you have reached 100 percent.

Once you have completed the input cycle, *PIEPLOT* automatically plots and displays a pie chart using your parameters. Once the chart has been drawn, *PIEPLOT* will beep and go into a loop awaiting any keystroke. Typing any key will return you to the data entry routine.

Hints and Ideas

Although *PIEPLOT* will take the tedium out of making pie charts on your Tandy 2000, some planning on your part will improve the results. For example, if you are creating a chart with several small slices, you will have a nicer looking pie if you distribute these evenly about the chart. The same is true involving longer titles for the slices.

PIEPLOT begins drawing the first slice at the 3 o'clock position on the circle and works counterclockwise. Keeping this rotation in mind will help you mix your colors for a more pleasing display. Remember, your last slice will be located next to your first one so plan accordingly.

Since creating a pie chart is now so much easier, experimentation will help you select your favorite arrangements. In the time it used to take to plot out one chart, you can easily create a dozen pies using *PIEPLOT*.

If you select black as the color for one of your slices, *PIEPLOT* will seem to hang-up for a few seconds while it repaints the black background. Don't panic! Nothing has gone wrong with the program as will be verified when the title suddenly appears.

I have been using *PIEPLOT* extensively, and know of no bugs in the program. Error trapping within the program easily handles any mistakes during the input cycle for quick recovery.

How It Works

The structure and flow of *PIEPLOT* is fairly straightforward. I have used no tricks, so you should be able to modify it easily if you so desire. Here is a brief explanation of the major routines within *PIEPLOT*.

Lines 100-190 are simple variable initialization routines. In Line 120 I set the variable RADIAN equal to one radian to simplify calculations later on.

Lines 170 and 180 select the colors for use in the active palette. To change the colors simply modify the DATA statement in Line 170.

Line 190 is a self-test routine which has been made into a REMARK for normal operation. If you delete the remark, *PIEPLOT* will skip the user input routine and go directly to plotting a chart with 20 equal 5 percent slices and 12-digit titles. This is really just a stress test to check out the display.

Lines 1000-1190 accept the user input and store it into arrays for use in the plotting and display routines. PR(n) holds the true signed percentage value of the slice and is used to determine if a slice is pulled out of the circle or not. PC(n) holds the percentage as degrees and is used in the actual plotting routines.

Lines 5000-5210 draw the pie slices. The degrees in PC(n) are converted into the starting and end points in radians used by BASIC's CIRCLE command. If the slice is not to be pulled away from the body of the chart it is drawn in Lines 5120 and 5130.

If the slice is to be segmented, then Lines 5150-5170 calculate an imaginary line to bisect the circle. Then a point on this line is used as the center point instead of the standard one. This imaginary point is also used to supply X and Y coordinates to the PAINT command in Line 5210.

This imaginary line is used again to display the title in Lines 5230-5250 and once more for the connecting line between the title and the slice in Lines 5260-5320. Adjustments are made to the line and title position depending on whether it is on the left side of the screen or the right. These adjustments are made in Lines 5290-5310 and use the current cursor row and column positions in their calculations.

Lines 10000-10030 are a delay and erase routine used by the error handling lines in the input section.

Lines 15000-15050 are a subroutine to display the palette during the input routine. Colors are displayed as boxes against slightly larger boxes of another color to separate them more clearly from the background.

Lines 20000-20020 freeze the display until you strike any key and then start the program all over. If you want to add a screen dump routine or save the chart to disk using the BSAVE command, this would be the place to do it.

Line 30000 is just a little trick I use to save programs while I'm developing them. Instead of trying to remember the file name, a simple GOTO 30000 saves the program and I'm back to work.

Uses And Modifications

While *PIEPLOT* will not rival com-

```
2 GOTO 100
4 '
8 '
10 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
12 '                        PIEPLOT.BAS  VERSION 5/20/84
14 '                 FOR THE TANDY 2000 WITH HI-RES COLOR BOARD
16 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
18 '
20 '  (C) Copyright 1984 by David R. Pifer, 951 Perry Hwy., Pgh, PA 15237
22 '+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
34 '
36 '
90 '              GENERAL INITIALIZATION
100 KEY OFF:CLS
110 DIM PC(101),CL(101),PR(101),TL$(101)
120 RADIAN=1.745329E-02
130 CL=1
140 CLS:SCREEN 3:COLOR 5,0
150 X=320:Y=200
160 RD=160
165 '              PICK SOME NICE COLURS FOR OUR PALETTE
170 DATA 0,2,7,15,14,12,11,13
180 FOR CT=0 TO 7:READ PC:PALETTE CT,PC:NEXT CT
185 '              NEXT LINE IS A SELF TEST ROUTINE NOT NORMALLY EXECUTED
190 REM TL$="AUTOTEST":FOR CT=0 TO 19:PR(CT)=-5:PC(CT)=5*-.01*360:CL(CT)=5:TL$(C
T)="123456789012":NEXT CT:GOTO 1500
195 '              GO AND DISPLAY THE PALETTE IN USE
200 GOSUB 15000
990 '              USER INPUT ROUTINE
1000 CT=0
1010 LOCATE 1,20:INPUT"Title of this chart";TL$
1020 IF TL$="" THEN TL$="Pie Chart"
1030 RW=4
1040 LOCATE RW,10
1050 PRINT 100-PT;"% remaining      ";
1060 INPUT "PERCENTAGE, COLOR, TITLE: ";PC(CT),CL(CT),TL$(CT)
1070 PR(CT)=PL(CT):PC(CT)=ABS(PC(CT))
1080 IF TL$(CT)="" THEN TL$(CT)=" "
1090 IF PC(CT)<1 THEN BEEP:LOCATE RW,10:PRINT"Percentage must be >=1      ";:GOTO
 10000
1100 IF CL(CT)<0 OR CL(CT)>7 THEN BEEP:LOCATE RW,10:PRINT"Color must be in range
```

Table 1:
Variables Used In PIEPLOT.BAS

PC()	Array to hold the size of each slice in degrees
CL()	Array to hold the color value of each pie slice
PR()	Array holds the percentage of a slice for use in segmenting
TL$()	Array to hold the title of each slice
RADIAN	Set equal to 1 radian to calculate and draw slices
X	Horizontal center point of the circle
Y	Vertical center point of the circle
RD	Radius of the circle
CT	Misc. counter and temporary variable
PC	Misc. temporary variable
TL$	Title of the pie chart
RW	Row value used to format displays
TT	Misc. counter variable
PT	Keeps track of the percentage of the circle used
TP	Set to the last array element to be used
LC	Line color used in all graphics statements
EP	Ending point of the slice's arc
SP	Starting point of the slice's arc
ST	Absolute value of SP
ED	Absolute value of EP
CL	Set equal to the CL() value of the current slice
PX	Horizontal point inside of pie slice
PY	Vertical point inside of pie slice
CR	Current cursor line used for title displays
KT	Misc. counter variable
X1	Horizontal coordinate used for palette display
Y1	Vertical coordinate used for palette display
PS	Holds cursor position for use in titling routines
A$	Misc. input string

```
 0-7":GOTO 10000
1110 IF LEN(TL$(CT))>12 THEN BEEP:LOCATE RW,10:PRINT"Title must be less than 12
characters          ";:GOTO 10000
1120 PT=PT+PC(CT):PC(CT)=(PC(CT)*-.01)*360
1130 IF PT>=100 THEN CT=CT+1:GOTO 1160
1140 CT=CT+1
1150 RW=RW+1:GOTO 1040
1160 TP=CT-1
1170 PC(CT)=-1
1180 CL(CT)=CL(CT-1)
1190 TP=CT-1
4990 '               PLOT AND DRAW THE CHART
5000 CLS
5010 LC=3
5020 LOCATE 1,(40-LEN(TL$)/2):PRINT TL$;
5030 IF PR(0)<0 THEN 5050
5040 LINE (X,Y)-(X+RD,Y),LC
5050 FOR CT=0 TO TP
5060 EP=SP+(PC(CT)*RADIAN)
5070 IF EP>0 THEN EP=EP*-1
5080 IF SP>0 THEN SP=SP*-1
5090 CB=CL(CT)
5100 IF EP<-6.283186 THEN EP=-6.283186
5110 IF PR(CT)<0 THEN 5150
5120 CIRCLE (X,Y),RD,LC,SP,EP
5130 CIRCLE (X,Y),RD-1,LC,SP,EP
5140 '               LOCATE A POINT INSIDE SLICE FOR PAINT AND TITLES
5150 ST=ABS(SP):ED=ABS(EP)
5160 PX=COS(ST+((ED-ST)/2))
5170 PY=-SIN(ST+((ED-ST)/2))*(25/28)
5180 IF CT=0 AND PR(CT)<0 THEN LINE (X+PX*15,Y+PY*15)-((X+PX*15)+RD,Y+PY*15),LC
5190 IF PR(CT)<0 THEN CIRCLE(X+PX*15,Y+PY*15),RD,LC,SP,EP
5200 IF PR(CT)<0 THEN CIRCLE(X+PX*15,Y+PY*15),RD-1,LC,SP,EP
5210 PAINT (320+PX*90,200+PY*90),CL,LC
5220 '               FIND A HANDY PLACE FOR THE TITLES
5230 LOCATE (210+PY*200)/16,(320+PX*190)/8
5240 IF PX<0 THEN LOCATE CSRLIN,POS(0)-LEN(TL$(CT))
5250 PRINT TL$(CT);
5260 '               DRAW A LINE FROM THE TITLE TO THE SLICE
5270 CR=CSRLIN
5280 PS=POS(9)
5290 IF PX<0 THEN PS=PS-LEN(TL$(CT))/2 ELSE PS=PS-LEN(TL$(CT))
5300 IF PY>0 THEN CR=CR-1
5310 IF PR(CT)<0 THEN LINE (PS*8,CR*16)-(320+PX*175,200+PY*175),LC:GOTO 5330
5320 LINE (PS*8,CR*16)-(320+PX*160,200+PY*160),LC
5330 SP=EP
5340 NEXT CT:GOTO 20000
9990 '               ROUTINE TO HANDLE ERRORS IN INPUT ROUTINES
10000 FOR TT=0 TO 2000:NEXT TT
10010 LOCATE RW,10
10020 PRINT SPACE$(60);
10030 GOTO 1040
14990 '               HERE'S THAT PALETTE ROUTINE WE USED EARLIER
15000 KT=0
15010 LOCATE 1,1:PRINT"  your";:LOCATE 2,1:PRINT"palette:"
15020 FOR Y1=35 TO 245 STEP 30:LINE (0,Y1)-(15,Y1+30),7-KT,BF
15030 LINE(2,Y1+2)-(13,Y1+28),KT,BF:KT=KT+1:NEXT Y1
15040 FOR KT=0 TO 7:LOCATE 3+KT*2,3:PRINT KT;:NEXT KT
15050 RETURN
19990 '               ADMIRE THE CHART ROUTINE AND RESTART ON KEYSTROKE
20000 BEEP
20010 A$=INKEY$:IF A$=""GOTO 20010
20020 RUN
20030 END
29990 '               SAVE THE PROGRAM BEFORE THE LIGHTS GO OUT AND ALL IS LOST
30000 SAVE"pieplot.bas"
```

mercial graphics terminals, it will do more than an adequate job of preparing visuals for meetings, presentations, and your own amusement. Due to the variety of hardcopy devices available, I've made no attempt to integrate any such drivers into the program.

You should be able to add these easily into the program in Line 20000, which is the exit point for the display routines. Depending on your output device, you might want to change the display colors selected in Line 170.

With the outstanding resolution of the Tandy 2000, 35mm slides can be made of your pie charts without much trouble. Several articles have been published on how to photograph a computer display. And, any good photographic store should be able to supply you with enough information to get started in audio visual presentations using your 2000.

If you would like a copy of *PIE-PLOT*, but don't feel like typing in the program listing, I'll send you a copy for a $5 handling charge just send me a formatted Tandy 2000 diskette and a self-addressed, postage-paid return mailer. Mail to: Dave Pifer, 951 Perry Highway, Pittsburgh, PA 15237.

New Products

Re-ink Printer Ribbon

Computer Friends has introduced MAC INKER which automatically re-inks ribbons for any printer. Operation is very simple. The user loads the cartridge and presses the start button, and the correct amount of ink is metered and evenly distributed across the ribbon.

MAC INKER can reduce the cost of replacing ribbons. Re-inking when print-out quality begins to deteriorate, besides restoring print-out quality, extends almost indefinitely the life of the ribbon. The ink contains emulsified lubricant for safe dot matrix printhead operation. The average quality ribbon can be re-inked "at least 50 times."

MAC INKER is available with multicolored inks for any current printer. The cost is $54.95 to $69.95 depending on type. Most units are below $60.

For more information, write Computer Friends, 6415 S.W. Canyon Ct. Suite 10, Portland, OR 97225, or call (800) 547-3303.

A Summing Up — Symbolically

This exercise in symbolic arithmetic presents a randomly selected series of challenging problems for anyone interested in testing their algebraic skills.

Bob Delbourgo

Family magazines often carry a puzzle corner page. There you might find a selection of word, numerical and picture puzzles. Sometimes you will come across arithmetic puzzles in which sums are presented in letter form; the aim of the 'game' is to find the number represented by ea: letter. These puzzles are among the most challenging numerical ones you are likely to encounter.

I have devised a program which will offer you such symbolic sums at random. It is definitely educational and ought to appeal to math teachers who are striving to instill algebraic skills into their students and are looking for interesting problems. Each offering is different, by courtesy of PoCo. Also you have the choice of multiplication or addition exercises. Students who can arrive at the answers with less than 10 clues are truly brilliant! More likely than not, 15 or more clues will be needed to guess the number. represented by each letter. I should mention that the letters go from Q to Z (0 to 9) at random and that every number is associated with a separate letter. When guessing, please enter *every* number, as prompted. When asking for a clue, kindly give PoCo time to effect the substitution.

Now, a few words about the listing. The numbers are stored in the form of strings and the subroutine from Line 1000 to 1030 assigns the letters to the individual integers. The subroutine starting at Line 1100 concocts the various number strings (having an assigned length N). The entire series of number strings are compacted in Line 150, 580, 1200 or 1310 and each time a clue is called, the subroutine 1315-1350 comes into action, substituting a letter for a digit. N$ and L$ stand for the initial number and letter strings as a rule; S$ is for the sum and P$ represents the product on occasion. You should be able to follow the rest during play.

The listing:

```
10 CLEAR1000:CLS:PRINT@44,"TXY";:PRINT@8
0,"+YVU";:PRINT@123,"+UVY";:PRINT@163,"-
---";:PRINT@202,"*YZRS";
.0 PRINT@34,"YSR";:PRINT@73,"+ TY";:PRIN
T@112,"------";:PRINT@154,"YSR";:PRINT@19
2,"YZTP";:PRINT@232,"------";:PRINT@271,"
$="":P2$="":P3$="":S$="":LT$="":N$="":C
LS
60 PRINT@11,"SYMBOLIC ARITHMETIC":PRINTS
TRING$(40,"*");:PRINT" Each letter stand
s for a particular     number in what fo
llows. You will be     presented with a
sum and have to guess  the numeric value
 of each letter."
70 PRINTSTRING$(40,"*");:PRINT" <A>dditi
on or <M>ultiplication Sums?";
80 I$=INKEY$:IFI$="A"ORI$="a"THEN100 ELS
EIFI$="m"ORI$="M"THEN500

90 GOTO80
100 CLS:PRINT@1,"+-+-+-+-+ SYMBOLIC ADDI
TIONS +-+-+-+-+"
105 PRINT" Here you have a sum of three
6-digit    integers. You may ask for clu
es at any  stage by pressing <C>. With e
ach clue   you demand, your score decrea
ses by 1,  your initial score being 25."
110 PRINT" Press <G> when you're ready t
o guess."
112 PRINT@280," How many clues to start
(20 Max)";:INPUTC:IFC>20THEN112
*YXRUR";
30 PRINTCHR$(27);"p":PRINT@49,"+-+-+-+-+
-+-+-+-+-+-+";:PRINT@209,"+-+-+-+-+-+-+-
+-+-+-+";:FORI=89TO169STEP80:PRINT@I,"-"
;:PRINT@I+20,"-";:NEXTI:PRINT@129,"+";:P
RINT@149,"+";:PRINTCHR$(27)"q"
40 PRINT@90,"Symbolic Arithmetic";:PRINT
@171,"by  Bob Delbourgo";
45 DIML$(10)
50 TR=TR+1:FORI=1TO7+VAL(RIGHT$(TIME$,2)
):R=RND(1):NEXTI:N1$="":N2$="":N3$="":P1
115 SC=SC+25-C:LN=0
120 BEEP:PRINT" Setting it up, please wa
it ...";:GOSUB1000
130 N=6:GOSUB1100:N1$=N0$:L1$=L0$:N=6:GO
SUB1100:N2$=N0$:L2$=L0$:GOSUB1100:N3$=N0
$:L3$=L0$
140 N1=VAL(N1$):N2=VAL(N2$):N3=VAL(N3$):
N=N1+N2+N3:N$=STR$(N):N$=RIGHT$(N$,LEN(N
$)-1):GOSUB1150
150 LT$=L1$+L2$+L3$+S$:GOSUB1200:CLS
160 L1$=LEFT$(LT$,6):L2$=MID$(LT$,7,6):L
3$=MID$(LT$,13,6):S$=RIGHT$(LT$,LEN(S$))
170 PRINT:PRINTSPACE$(5);L1$:PRINTSPACE$
(5);L2$:PRINTSPACE$(5);L3$:PRINT"      --
----":PRINTSPACE$(11-LEN(N$));S$
180 PRINTCHR$(27)"p":PRINT@260,"SCORE ="
SC;:PRINTCHR$(27)"q":PRINT@20,"<C>-CLUE
<G>-GUESS";
190 I$=INKEY$:IFI$="C"ORI$="c"THEN200ELS
EIFI$="G"ORI$="g"THEN300ELSE190
200 C=1:BEEP:GOSUB1200:SC=SC-1:GOTO160
300 PRINT@55,"<--";:INPUTC1$:PRINT@95,"<
--";:INPUTC2$:PRINT@135,"<--";:INPUTC3$:
PRINT@214,"<--";:INPUTC$:CT$=C1$+C2$+C3$
+C$
310 IFCT$=NT$ THENCLS:PRINT" Well done i
ndeed!":PRINT" Your score is"SC"after"TR
"rounds.":FORI=10000TO200STEP-200:SOUNDI
,3:NEXTI:GOTO50
320 CLS:PRINT" Sorry, incorrect! Try aga
in!":FORI=10TO100:SOUNDI*I,1:NEXTI:SC=SC
-1:CLS:GOTO160
500 CLS:PRINT@1,"****** SYMBOLIC MULTIPL
ICATIONS ******":PRINT" Here you are giv
en the detailed product of a 5-digit num
ber and a 3-digit one."
510 PRINT" You may ask for 1 clue at a t
ime by      pressing <C>,  or <G> if you
are ready  to guess. Each clue given dim
inishes    your score by 1 (initially 35
)."
520 PRINT@280," How many clues to start
(30 max)";:INPUTC:IFC>30THEN520
525 SC=SC+35-C:LN=0
530 BEEP:PRINT" Setting it up, please wa
it ...";:GOSUB1000:
540 N=5:GOSUB1100:N1$=N0$:L1$=L0$:N=3:GO
SUB1100:N2$=N0$:L2$=L0$:GOSUB1100
550 N1=VAL(N1$):N2=VAL(N2$):P1=N1*VAL(RI
GHT$(N2$,1)):P2=N1*VAL(MID$(N2$,2,1)):P3
=N1*VAL(LEFT$(N2$,1))
560 P1$=RIGHT$(STR$(P1),LEN(STR$(P1))-1)
:N$=P1$:GOSUB1150:S1$=S$:P2$=RIGHT$(STR$
(P2),LEN(STR$(P2))-1):N$=P2$:GOSUB1150:S
2$=S$
570 P3$=RIGHT$(STR$(P3),LEN(STR$(P3))-1)
:N$=P3$:GOSUB1150:S3$=S$:N=N1*N2:N$=RIGH
T$(STR$(N),LEN(STR$(N))-1):GOSUB1150
580 LT$=L1$+L2$+S1$+S2$+S3$+S$:GOSUB1300
:CLS

590 L1$=LEFT$(LT$,5):L2$=MID$(LT$,6,3):S
1$=MID$(LT$,9,LEN(P1$)):S2$=MID$(LT$,9+L
EN(P1$),LEN(P2$)):S3$=MID$(LT$,9+LEN(P1$
)+LEN(P2$),LEN(P3$)):S$=RIGHT$(LT$,LEN(S
$))
600 PRINT@5,L1$;:PRINT@45,"* "L2$;:PRINT
@83,"--------";
610 PRINT@130-LEN(S1$),S1$;:PRINT@169-LE
N(S2$),S2$;:PRINT@208-LEN(S3$),S3$;:PRIN
T@242,"---------";:PRINT@290-LEN(S$),S$;
620 PRINTCHR$(27)"p";:PRINT@265,"SCORE =
"SC;:PRINTCHR$(27)"q";:PRINT@105,"<C> or
 <G>";
622 I$=INKEY$:IFI$="C"ORI$="c"THEN625ELS
EIFI$="G"ORI$="g"THEN630ELSE622
625 C=1:BEEP:GOSUB1300:SC=SC-1:GOTO590
630 PRINT@13,"<--";:INPUTC1$:PRINT@53,"<
----";:INPUTC2$:PRINT@132,"<--";:INPUTC3
$:PRINT@171,"<--";:INPUTC4$:PRINT@210,"<
--";:INPUTC5$:PRINT@290,"<--";:INPUTC$
640 CT$=C1$+C2$+C3$+C4$+C5$+C$
650 IFCT$=NT$THENCLS:PRINT" Well done in
deed!":PRINT" Your score is"SC"after"TR"
rounds.":FORI=200TO10000STEP200:SOUNDI,3
:NEXTI:GOTO50
660 CLS:PRINT:PRINT" Wrong, try again!":
SC=SC-1:FORI=10TO100:SOUNDI*I,1:NEXTI:CL
S:GOTO590
1000 L$="QRSTUVWXYZ":FORI=1TO50
1010 J=INT(10*RND(1)+1):K=INT(10*RND(1)+
1):IFK<=JTHEN1010
1020 J$=MID$(L$,J,1):K$=MID$(L$,K,1):L$=
LEFT$(L$,J-1)+K$+MID$(L$,J+1,K-J-1)+J$+R
IGHT$(L$,10-K):NEXTI
1030 FORI=0TO9:L$(I)=MID$(L$,I+1,1):NEXT
I:RETURN
1100 L0$="":N0$="":FORI=1TON-1:J=INT(10*
RND(1)):L0$=L0$+L$(J):N0$=N0$+RIGHT$(STR
$(J),1):NEXTI
1110 J=INT(10*RND(1)):IFJ=0THEN1110ELSEL
0$=L$(J)+L0$:N0$=RIGHT$(STR$(J),1)+N0$
1120 RETURN
1150 S$="":FORI=1TOLEN(N$):V=VAL(MID$(N$
,I,1)):S$=S$+L$(V):NEXTI:RETURN
1200 NT=LEN(N1$+N2$+N3$+N$):LN=LN+1:IFLN
=NT-1ORC=0THENRETURN
1205 NT$=N1$+N2$+N3$+N$:
1210 GOSUB1315:RETURN
1300 NT=LEN(N1$+N2$+P1$+P2$+P3$+N$):LN=L
N+1:IFLN=NT-1ORC=0THENRETURN
1310 NT$=N1$+N2$+P1$+P2$+P3$+N$:GOSUB131
5:RETURN
1315 FORI=1TOC
1320 R=INT(LEN(LT$)*RND(1)+1):M$=MID$(LT
$,R,1)
1330 IFM$="0"ORM$="1"ORM$="2"ORM$="3"ORM
$="4"ORM$="5"ORM$="6"ORM$="7"ORM$="8"ORM
$="9"THEN1320
1340 MID$(LT$,R,1)=MID$(NT$,R,1)
1350 NEXTI:RETURN
```

The Portable Computer As A Social Science Research Tool

David Kintsfather

Radio Shack sells a statistical analysis program for the PoCo (Cat. No. 26-3825) which appears from the description and documentation to be aimed at business users. A little experimentation has proven that this very economical program also has many applications in social science research.

Of course it is not going to rival the statistical packages in a university mainframe, but it handles the descriptive statistics and analysis of variance procedures which are most frequently used in day-to-day research projects. For the student or teacher in the field who does not have access to a mainframe this is a quick and easy way to analyze a variety of data.

The descriptive statistics and the one-way and two-way analysis of variance procedures are well documented and anyone with a rudimentary knowledge of research techniques should have no trouble using them.

One statistic which is barely mentioned in the documentation is the t-test for paired data (correlated t). This statistic is appropriate to analyze data from the pre-test post-t"st experimental design which is used extensively in the social sciences, and especially education.

In this design a group of subjects are tested for a variable such as math ability or attitude toward local government. Then the group is given the experimental "treatment," for example, a programmed math lesson or a television program advocating the importance of local government. The group is then retested. The correlated t statistic compares the mean (average) scores on the two tests and yields the probability of the change in mean scores (if any) being the result of chance rather than a real effect of the experimental treatment.'

To perform this test with PoCo, load the *STAT* and *SACR* programs from the statistical analysis package. Using *TEXT*, create a data file containing three columns of numbers separated by commas. The first column should contain the subject number. This is not needed to run the program but may be helpful in keeping track of the data. The second column will contain the scores on the pre-test and the third column will contain the post-test scores for each subject (see example):

```
File Name:  DATA

1,103,117
2,87,92
```

The program will prompt you to create a data format, telling PoCo what all those numbers in the *DATA.DO* file mean. To use the formatting system provided we will call the pre-test scores "PRE" and designate them variable '1' in position two (remember we put subject number in position one). Post-test scores will be called "POST" and will be designated variable '2' in position three. When interpreting the results, keep in mind that we are really dealing with two sets of scores relating to the SAME variable.

When running the correlation and regression program *(SACR)*, designate number one "POST" as the dependent variable (dv) and number two "PRE" as the independent variable (iv). The program will then run and yield a number of statistics.

The one of greatest interest is the probability figure. If it is .05 or lower, then for most social science research we will accept the change in mean scores as "statistically significant." This means that there is at least a 95 percent probability that the observed change in scores would take place when the experimental treatment is applied to other groups of subjects. Keep in mind that this is a so-called two-tailed test which makes no assumption about whether the scores will get bigger or smaller following our experimental treatment.

Two other restrictions apply to the use of the correlated t statistic. The two sets of scores must have a positive correlation (the program will give you this statistic) and the variable must yield a normal distribution (bell shaped) in the general population. Most attitude and performance measurements are normally distributed (meaning that the majority of people are of average performance or hold moderate views with progressively fewer people at the extremes) and the use of a standardized test which has been validated will insure the accuracy of this statistical treatment.

Many people would like to be able to evaluate the effectiveness of their new curriculum materials, TV commericals, brochures, etc., but avoid research because of the difficulty of analyzing the data. With PoCo and the statistical analysis package anyone should be able to get reliable results quickly and easily (and it's fun, too). PCM

The 96K Model 100 – or How To Fit Two Model 100s In The Same Box

Jim Hawk

Memory expansion devices for the Model 100 now offer a variety of choices, and prices, for beefing up your portable computing power. Three different approaches will be examined: a tiny 32K RAM board that hides in the 100's expansion compartment, a heftier 64K outboard unit, and a 128K "bubble" memory that attaches underneath the 100. All three machines represent the efforts of entrepreneurial companies cashing-in on Tandy's openness about the 100's inner workings. The basic expandability inherent in the design also helps out: the RS-232 port in back, the 40-pin "expansion bus" underneath, and the ROM socket create some exciting opportunities for inventors. Tandy Radio Shack officials would be the first to admit they don't have the time or resources to fully exploit the potential of all their computer lines, and their long-standing policy of encouraging outside designers to get into the act has really paid off this time.

Double your Memory, Double your Fun

A small company on the outskirts of Detroit has figured out how to make a two-inch square circuit board (fitting inside the 100's not-so-secret compartment underneath) that doubles the memory of a fully-chipped 32K Model 100. Peter VanHeusben is president, chief designer and founder of PG Design Electronics, Inc. in Richmond, Mich. As one of the first owners of a 100, he started the company by offering 8K RAM chips for just over half the official price. VanHeusben still felt the need for more memory, so the "32K CMOS Expansion RAM" came about. As a designer of automotive control systems, he drew on his experience in low-power high reliability circuits to pack four 8K memory chips, a battery backup and rigid connecting pins into a palm-size unit. A "review" on the 32K Expansion RAM is virtually unecessary: it works *exactly* like the Model 100's regular memory. The unit comes packed in a classy-looking, cloth-covered box like you'd see expensive software come in. pull out the notebook and it contains 15 pages of documentation and a holding case for the 32K RAM circuit board. The installation procedure requires only a moment to plug in the circuit board and replace the plastic door underneath.

After typing in a one-line BASIC program and saving it as *Bank 0.BA*, you switch to the other 32K simply by placing the cursor over this program and hitting ENTER. A blink of an eye later, the first menu is replaced by a second one, complete with the five built-in software programs. All of a sudden, even though your 100 still looks the same, you have two separate main menus and up to 64K of usable memory. Add the 32K of built-in software that frees up the RAM entirely for your own use, and you have the "96K Model 100." One could theoretically keep a separate memory bank for work related text, address list and programs, and a separate one for personal use. Or, if your spouse has become convinced she/he can use your 100, you each have your own 32K of memory to use — "His & Hers" memory, if you will. Another possibility (if you can afford the $325 price per unit) is to own several units that can be plugged-in as necessary.

One design limit: the RAM board relies on its own internal batteries for continuous memory (two silver-oxide watch batteries costing about $1.50 each) and both must be replaced every four to six months. So count on $12 to $15 a year for batteries. PG Design does include a *Batt.DO* file in the unit to serve as a reminder of when the original batteries were installed. A second program embedded in the memory is *TEST.BA* which sorts through each and every byte before declaring "Good RAM" - reassuring for novice electronics installers like myself. A third program that's included by PG Design is the reverse of the one-line BASIC statement that gets you back to the main menu (logically labelled *Bank 1.BA* to differentiate it from the original menu's *Bank 0.BA* on the same theory as labelling disk drives). The only other drawback is due in part to the limitations of the 100's eight-bit central processing unit: only 32K of RAM can be utilized at any given time. Getting information from one 32K "bank" to the other can currently only be done by saving to cassette from one, then loading back into the other.

Unforunately, the "cut and paste" functions don't work between the two memory banks, but PG Design president VanHeusben says a software enhancement for this is in the works.

The bottom line is this: you get a duplicate master menu, complete with the five built-in programs, 19 files and 32K of additional memory for $325, about one-third the cost of a comparably equipped Model 100. It all fits neatly inside the expansion compartment, although it does preclude the use of other peripherals like the Disk Video Interface. After obtaining a production unit for testing and getting accustomed to having 64K of memory, I've concluded it'll be very difficult to send it back to PG Design — the highest compliment any reviewer can give.

Need more memory?

For nearly the same price, almost twice the usable memory is available from a company called Cryptronics, inc. of Fountain Valley, California now marketing the Porta-Pac 100. The catch is that it's an outboard unit about the same size and weight as the Model 100. The basic setup is priced at $395 for 64K of memory. Due to the internal "RAM disk" operating system, 60K is available to the user. However, internal expansion slots have been provided so that three more 64K RAM chips could be added for a maximum usable storage capacity of 240K.

The Cryptronics approach is to use cheaper, but more power hungry N-MOS chips instead of the power-thrifty but more expensive C-MOS RAM chips. The benefit is a price/performance attraction for those looking for the most "bang for the buck" when expanding the 100's memory. Instead of watch batteries like the PG Design unit, the battery needed to power the Porta-Pac is similar to the type used on a portable video cassette recorder: a 1.6 Amp-hour lead-acid gel cell made by Panasonic. Company spokesman Gary Kakert says it's better to think of the Porta-Pac as a unit requiring AC power, but also capable of mobile operation for a limited time. The battery-life figures bear him out: the 64K memory operates just two to eight hours before needing a recharge and even in "hibernation" the battery gives out in 48 hours. To meet this need for recharging, Cryptronics provides an AC adapter and a 12-volt cigarette lighter cord so you charge up the memory's battery while driving.

One unique feature of the built-in ROM software: by coordinating the memory through a Z-80 microprocessor, the Porta-Pac can theoretically store one file running 60K long.

Normally, each 64K memory chip has 32 separate files.

Like the PG Design unit, the Porta-Pac also cannot match the 100's internal memory capacity to "cut and paste" from one "drive" or section of memory to another. Kakert says the necessary additional software "may be possible in the future." The proprietary operating system was designed to serve not only the Model 100, but also desktop computers utilizing MS-DOS or CP/M based systems. Since the connection is made through the RS-232 port, Cryptronics made it possible to transfer data from your portable computer to a bigger office-type system. So far, buyers have been using the Porta-Pac mainly for Model 100 applications only. Since the product has just been finalized, review units have yet to be shipped. We'll wait and see whether the built-in DOS of the Porta-Pac can match the near-total operating transparency of PG Design's 32K RAM board.

Half a Megabyte!

A third firm getting into the Model 100 memory expansion field is Sound-Sight Communications in Hollywood, Calif. The Model 100 128K Bubble Memory Module can be expanded internally to accept three more of the Intel-manufactured 128K bubble memories, adding up to an astounding half a megabyte. The inch-thick unit is the same width and depth as the Model 100 and attaches to the bottom. The joined unit no longer fits in the 100's carrying case, but with half a megabyte in additional storage, who's arguing? Designers for SoundSight rejected both CMOS and N-MOS memory chips in favor of newer space-age developments in miniaturized bubble memories. Because each memory byte is kept in its own microscopic "magnetic bubble," the memory is non-volatile. In other words, data will not be lost even if the batteries go dead

Power consumption is low because the bubble memory is only powered up when, in disk-operating parlance, it "reads or "writes." A four-watt power burst is needed for the typical one-second of operating time, but even a single AA-cell can power the 128K bubble memory module. Up to three batteries can be loaded into the unit to extend the time between battery changes to about a month.

Perhaps the best way to describe this bubble-memory technology is to first take a look at how earlier-generation computer memories worked. The dinosaurs of 20 years ago used a similar magnetic on-off method of storing data — except they used expensive and bulky copper wires and each "bubble" was about the size of a pea. Using this design, 128K would have taken up a large warehouse and cost millions, so you can see the immensity of scale-reduction that has occurred. Now, the entire 128K magnetic bubble memory can sit in the palm of your hand thanks to breakthroughs from Intel Corporation. Their latest version also answers the main criticism of some previous bubble memories' speed. The chief engineer and designer at SoundSight, Henry Lubey, says the new 128K chips match or better the speed of normal RAM chips.

In a hint at things to come, he says the upcoming four mega-bit system (or four million characters) from Intel is 25 times faster The one "Achilles Heel" for magnetic bubole devices is the fact that a strong magnetic field could disrupt the entire memory. Fortunately, only something like a tape eraser using an AC powered electro-magnet would do damage — TVs, radios, and other consumer devices producing minor magnetic fields would have no effect.

As it's been said, "there's no free lunch," so let's get down to the cost involved. The basic module, with one 128K bubble component installed will run you $1,050. If you still need more memory, up to three additional 128K devices can be installed at a cost of $450 each. So, a half-megabyte system will run much more than the Model 100, a whopping $2,400! Because of the imposing initial price, individuals have not yet been rushing to buy this remarkable peripheral. But sales manager Fred Lowe told me he's already had over $6 million worth of inquiries from Fortune 500 companies, NASA and even two branches of U.S. Navy research. The 14-digit math accuracy of the 100 has made it a favorite of researchers — and now the vastly expanded memory means programmers can write up to 30K-sized subroutines and create programs that exceed the Model 100's RAM by sampling between the memories of the 100 and the bubble. Originally, the firm was going to offer an outboard version of the bubble memory -- but due to "zero interest" SoundSight decided to manufacture only the 128K which attaches to the bottom of a Model 100.

The 100 Comes of Age

Back in March of 1983, I don't think the most optimistic among us could have predicted such remarkable peripherals would come out within the year and a half that the Model 100 has been on sale. Three entirely different approaches to Model 100 memory expansion are now hitting the market, and the portable disk machines that will extend this capability even further are finally about to come out.

Products mentioned in this article:

32K RAM expansion by PG Design Electronics, Inc., 66040 Gratiot, Richmond, MI 48062, (313) 727-2744

64K "Porta-Pac" by Cryptronics, Inc., 11711 Coley River Circle, Suite 7, Fountain Valley, CA 92708, (714) 540-1174

You Can 'Playit' On Your PoCo

Larry M. Roberson

You didn't pay good money for the Model 100 just to play "lap sized" video games. Most of you bought it because of its powerful built-in software, and its flexibility of use while traveling. I've been a salesman longer than I care to remember, and have found many uses for the PoCo, business oriented and otherwise.

One of the non-business uses for PoCo which I have discovered to be a lot of fun is in the area of music. As a result, I have written the accompanying program, which I call *PLAYIT*.

You don't have to be a musician to use *PLAYIT*. This program is mostly for those of you with no musical talents at all. I took piano lessons for years, and played "I Dropped My Dolly In The Dirt" at each recital. With this in mind, *PLAYIT* is better compared to one of those toy wooden flutes. Well, actually the program is better than that, but I'll let you decide.

PLAYIT uses the sound capability of the Model 100 to produce notes over five octaves. Sharps or flats can be produced with it, and you can also change the length of the tones produced. If you really get inspired, you can "record" the tones for the computer to play later You also have the ability to record pauses, or empty notes. Use of these pauses can improve the listening quality of your creations. Here's how it works.

The 'menu' of the program tells you most of what you need to know. The first prompt asks if you want to play *MUSIC.DO*. If you have recorded a tune into this text file, you may press ENTER and the program will play it for you. During the first run of the program you won't have anything in this file, so tap any key and then press ENTER.

You can now play with the tones. Pressing a number key between one and eight will produce a tone. The octave that the program is using and the length of the tone will be displayed on the screen. To change the octave, press any of the first five function keys. The length of the tone is controlled by function keys six through eight. I have found it easiest to lock the NUM key. This allows you to play the tones with your right hand, and change the ranges with your left hand. You will notice that the screen is updated each time you press a function key.

Oh yes, if you want to "sharpen" a note, press the '+' key immediately before you press the number key. When you want a flattened note press the '-' key. The next number key pressed will be flattened a bit. The '+' and '-' only work for the next key pressed. Unlike a 'standard' musical instrument, all of the normal tones produced may be offset with this process. There are only two exceptions. In the lowest octave, the first two notes cannot be flattened. Simply, the machine can't go that low It won't hurt anything if you try it, but the tone produced will not be exactly right.

When you want to record the last note played, press the 'P' key. In other words, you can peck away until you produce the note that sounds right. Pressing the zero key records a pause or an empty note. These pauses will be the same length as a regular note. Pressing the ENTER key while in the "peck" mode restarts the program. When you are playing *MUSIC.DO*, you may notice that you have made mistakes while recording the notes. Hitting the 'E' key will cause the last note in the *MUSIC.DO* file to be erased. Each recorded note eats up 10 bytes of free memory. For those who want to type it in and take off, that's all there is to it. The balance of this article deals with more technical aspects of the program.

Details, Details

If you look at the *MUSIC.DO* file you will see lines of data consisting of the values used in the SOUND command. Each line represents a single note. Empty notes are represented by zero entries for the tone number. You can insert, delete or change notes using the functions of the *TEXT* program, if you wish. However, be sure to use the same format. This format was chosen so the data would be easier to read and manipulate. Notes recorded with the program will always be added to the end of the *MUSIC.DO* file. You may wish to transfer finished parts of a tune to another file, so you won't have to listen to them over and over.

Line by line

Lines 90-93 — opens the data file, sets up the menu and displays it on the screen.

Lines 94 — displays the prompt. If you press ENTER the program jumps to Line 600. Type any character before you press ENTER and the program will set up the initial ranges (subroutines) and the main part of the program will begin.

Line 95 — turns on the function key interrupts and defines them with subroutine line numbers

Line 101 — contains the data necessary to define the first octave range of the SOUND command.

Line 110 — reads the above data into an array for future reference

Line 190 — polls for a key press. When the key is a number key 1-7 the program jumps to line 206.

Line 195 — a bunch of "if-then" statements are eliminated here by this single instruction. T$ is defined in Line 92. T$ consists of all possible combinations of the keys used to record tones, pauses, sharps and flats. Notice that some of them have upper- and lowercase variations. When INSTR is used it will return a zero when the key just pressed (A$) cannot be found within T$. When INSTR can find A$ within T$, it returns its position. Of course, one unit is added to whatever INSTR comes up with. In other words this technique produces code that is smaller and much faster than separate IF — THEN statements.

Line 201 — when the key press is a 'P' the values used for the last SOUND command are printed to the *MUSIC.DO* file.

Line 202 — when zero is pressed an empty note is printed to the MUSIC file.

Line 203 — calculates offset factor for sharps.

Line 204 — calculates offset factor for flats.

Line 205 — restarts the program.

Line 206 — makes the sound. The program "looks up" the appropriate sound value from the array where they are stored (T(1) through T(8)). This value is then divided by a factor to produce other octaves. Page 180 of the Model 100 manual shows the values used for this trick. Notice that the values in the data statement correspond to the first octave values in the table. What makes the "factor" trick possible is that each higher octave value is cut in half. The sharps or flats are produced by offsetting the sound value by .0562 of its normal value. The error handling routine (Line 700) comes in when you try for a flattened 1 or 2 (G or A) in the lowest octave. The values produced would be out of the range of the SOUND command.

Lines 500-599 — define the function key interrupt routines. Variable 'F' is the factor values for changing octaves. Variable 'L' is the note length. Variables F1 and L1 are used to update the screen.

Lines 600-699 — the section plays the tune stored in *MUSIC.DO*. Look closely at Line 620 and you will see that the program counts the number of lines it has played so far. There must be a pause loop in this section. Without such a loop, the tones run together. This loop is located in Line 640. After playing 25 notes, BASIC will stop to do some housekeeping. So, every 26th note, the pause contained in Line 640 is shortened to almost nothing. The result is a relatively smooth flow of the tones. Line 650 actually plays the "pause" note, but it's beyond the range of human hearing. The result is a silent note.

PCM BAR CODED LISTING

The listing:

```
10 'PLAY IT PROGRAM FROM CHIPMMONK SYSTE
MS CO. -- BY LARRY M. ROBERSON COPYRIGHT
 1984 ALL RIGHTS RESERVED
20 'phone(214-692-7414) CIS#(71515,2004)
90 DEFINTA-Z:CLS:ONERRORGOTO700:OPEN"MUS
IC.DO"FORAPPENDAS1:PRINTCHR$(27)+"p"+"
       CHIPMONK SYSTEMS CO.           ";P
RINT"*****PLAY IT***** C. LARRY ROBERSON
 1984"+CHR$(27)+"q";:PRINT"SELECT LENGT
H WITH FUNCTION KEYS (6-8)
92 T$="Pp0+=-_"+CHR$(13)
93 PRINT"SELECT OCTAVE WITH FUNCTIO'
S (1-5)":PRINT"PLAY WITH NUMBER KEY:
8)":PRINT"[P] NOTE  [0] PAUSE  (+-)
P/FLAT"
94 PRINT@240,"<ENTER> = LISTEN TO M
DO";:INPUTR$:PRINT@240,SPACE$(38);:
""THEN600ELSEIFR$="E"ORR$="e"THEN66
GOSUB510:GOSUB560
95 KEYON:ONKEYGOSUB510,520,530,540.
60,570,580
101 DATA12530,11172,9952,9394,8368.
7032,6269
110 FORI=1TO8:READT(I):NEXTI
190 A$=INKEY$:IFA$=""THEN190ELSEIFA
ANDA$<"9"THEN206
195 ONINSTR(T$,A$)+1GOTO190,201,20
203,203,204,204,205
200 STOP
201 PRINT#1,STR$(INT(S/F)+G)+CHR$(4
R$(L):GOTO190
202 PRINT#1,"    0,"+STR$(L):GOTO19
203 G=INT((S/F)*.0562)*-1:GOTO190
204 G=INT((S/F)*.562):GOTO190
205 RUN
206 A=VAL(A$):S=T(A):SOUND(S/F)+G,L
GOTO190
510 F=1:F1=1:GOTO555
520 F=2:F1=2:GOTO555
530 F=4:F1=3:GOTO555
540 F=8:F1=4:GOTO555
550 F=16:F1=5:GOTO555
555 PRINT@240,"OCT"F1;:RETURN
560 L=2:L1=6:GOTO585
570 L=8:L1=7:GOTO585
580 L=32:L1=8:GOTO585
585 PRINT@260,"LEN"L1;:RETURN
599 STOP
600 CLOSE:OPEN"MUSIC.DO"FORINPUTAS1
610 IFEOF(1)THEN CLOSE:RUN
620 INPUT#1,S,L:C=C+1:IFC>24THENC=0:Y=2E
LSEY=120
625 IFS=0THENGOSUB650:GOTO640
630 SOUND8,L
640 FORX=1TOY:NEXTX:GOTO610
650 SOUND0,L:RETURN:'FORZ=1TO120:NEXTI:R
ETURN
660 CLOSE:PRINT"ERASING LAST NOTE":OPEN"
MUSIC.DO"FORINPUTAS1:OPEN"TEMP.DO"FOROUT
PUTAS2:GOTO670
665 IFEOF(1)THENCLOSE:KILL"MUSIC.DO":NAM
E"TEMP.DO"AS"MUSIC.DO":RUNELSEPRINT#2,A$
670 LINEINPUT#1,A$:GOTO665
699 STOP
700 IFERR<>5THENPRINT"ERROR";ERR;"LINE"E
RL:STOPELSESOUND16382,L:G=0:RESUME190
```

PCM

MS·DOSsier

Mastering MS-DOS

Edlin Function Keys

Danny Humphress

In the August edition of MS-DOSsier, we began exploring MS-DOS's handy line editor, Edlin. Edlin, you'll remember, allows you to create and edit disk files. Although the Edlin commands we learned last month are more than enough for simple file editing, the many other features provided within Edlin make it easy to solve almost any file editing problem. We'll be working with some of these features today as we work our way closer to the level of "MS-DOS Masters."

For example, we'll be working with a file with the names of several U. S. cities:

Atlanta, GA
Chicago, IL
Denver, CO
Los Angeles, CA
Louisville, KY
Miami, FL
New York, NY
Orlando, FL
Palo Alto, CA
San Francisco, CA

You should have a backup of a MS-DOS version 2.00.00 (or later) disk booted and in drive A: or on your hard disk. The line editing features discussed in last month's MS-DOSsier will be used, so refer to the August issue if you need a refresher course.

Start Edlin and create a file called "CITIES" with the following command line:

```
EDLIN CITIES
```

The Edlin program will start and display:

```
NEW FILE
*_
```

Use the 'I' insert command to enter the names of the 10 cities listed above. Be sure to enter them exactly as they appear — spelling, capitalization, and order — or the exercises we're going to do will not work properly.

Special Editing Keys

Edlin makes use of the function keys to make it easy to edit individual lines. Last month we used [→] to copy characters from the original line (or template) to the new line. We used INSERT to insert characters in a line and DELETE to delete template characters. We also used [F3] to copy all remaining characters from the template to the new line. If you do not understand how these features work, practice with them referring to the August edition of MS-DOSsier.

Let's play a little geographical musical chairs and move Chicago to Texas just for fun. If you did a good job entering the list, Chicago should be in the second line. At the asterisks prompt, tell Edlin to edit line two by typing:

```
2 ENTER
```

The screen should look something like this:

```
* 2
  2:*_
```

We want to copy the word, "Chicago," the comma and the space from the old line (the template) to the new version of the line. Using what we learned last month, we could press [→] nine times to copy the first nine characters to the new line. This is fine for short lines, but wouldn't it be nice, with our example, to tell Edlin to copy all the characters up to the state name with a single keystroke. Of course, I wouldn't have mentioned it if I weren't about to give you a solution, now would I? At the "2:*" line editing prompt, type:

```
[F2] I
```

The screen will now look like this:

```
* 2
  2:*Chicago, _
```

[F2] tells Edlin to copy all the characters from the current position in the old line (template) to the character that you specify. In this case, we told Edlin to copy all the characters from the beginning of the old line up to the "I" in "IL." We can now resume editing the line by typing the new state where "IL" was. Type "TX" and press ENTER. The screen will look like this:

```
* 2
  2:*Chicago, TX

*_
```

List the file to see what you've done. You remember how to do that, don't you? Type:

```
1P ENTER
```

Just as we can use [F2] to *copy* up to a certain character, we can use another function key to *delete* up to a specified character. [F4] tells Edlin to delete all characters from the current position in the template to a certain character.

Let's have a little fun with San Francisco by removing the space and "Fran" to make it "Sancisco"! No particular reason, just for practice. Since San Francisco is on the 10th line, tell Edlin to edit that line by typing:

```
10 ENTER
```

We want to keep all the characters up to space, "San," so press [F2] and the spacebar. The screen will look like this:

```
* 10
  10:*San_
```

Now, we want to delete "Fran," or all the characters up to the "c." Press [F4] and a lowercase 'c' like this:

```
[F4] c
```

The screen will not change, but something has happened! Press [F3] to copy the remaining characters from the template (don't press ENTER yet), and you'll see this:

```
* 10
  10:*Sancisco, CA_
```

The space and "Fran" were deleted from the template.

Now that we've made a mess of San Francisco, let's put it back to the way it was. Not by more editing, but with Edlin's panic key, [F8]. This key tells Edlin to void the current edit line — make no changes. This can be very handy when you discover that you've made a mistake in editing. Press [F8] and your screen will look like this:

```
* 10
  10:*Sancisco, CA \
  _
```

A backslash "\" appears where your cursor was, and the cursor moves to the next line. In effect, you've voided all changes made to this line and have started a fresh edit on it. Press [F3] to copy all the characters from the template, and you'll see that the old version is still intact:

```
* 10
  10:*Sancisco, CA \
  San Francisco, CA_
```

Now, just press ENTER to save the edited (or un-edited) line.

It may seem very awkward for you, at this point, to remember to press F-this or F-that to copy this and delete that in a line when it would be simpler to just retype the silly thing. Once you're used to using the editing keys — you'll be surprised how quickly you'll get the hang of it — it will seem natural.

I'll let you in on a secret about the Edlin function keys. They work the same when you're entering MS-DOS commands! The old (template) line is always the last command line you entered. You can repeat a command by pressing [F3] and ENTER or you can edit the line if you make a mistake or want to change parameters. I can't express how helpful this can be once you've begun to use it regularly — especially with long command lines.

The line editing function keys are just a part of the features available to you with Edlin. We will conclude this three part study of Edlin in October with a close examination of Edlin commands. You already know about some of these, such as 'P'age and 'E'nd, but there are several more that make Edlin a very helpful tool.

Stocking Up On Reinvested Dividends

Robert Frowenfeld

Last month I provided you, the trusted PCM reader, with one of my programs I use on a regular basis. The more I thought about it this month, the more I realized I have a few more programs I've been keeping for you. Programs that I think may help in managing your money (you know, the root of all evil). If you're a regular "on the roader," you may have guessed by now that I dabble in the stock market (lose my shirt is more like it). They say it's not whether you win or lose but how you play the game, so I play it a little differently . . . I use my Model 100 to do a lot of the record keeping that is necessary when you try to keep track of your investments. So, again this month, "On The Road" will provide you with another program to help you manage your financial future.

If you own stock in a corporation that pays dividends, and you have those dividends reinvested, you know that keeping track of all those quarterly statements can get to be quite a bother. Furthermore, if you have stock in a utility, such as a power company, telephone company, etc., sometimes the price you pay for a share of stock isn't always the price you claim when you sell it. How so? Well, many utilities encourage you to reinvest your dividends by giving you a discount (typically five percent) on buying additional shares. This is great for the company in that it doesn't really cost them anything to give you the new shares on paper. And, after all, the money they don't send you in the form of a dividend check they can use to finance operations and even invest. But a few years ago the IRS decided that the price you claim for having purchased the stock in this fashion must follow certain rules. Since these rules aren't always exactly the same as the rules the utility company follows when they determine the price of the stock when they buy it for you, a discrepancy can occur. To make a long story short, this month's "On The Road" assists you in keeping track of all the information you need to monitor the reinvestment of any stock dividends. By the way, you should be aware of the fact that, currently, qualifying utilities are able to reinvest your dividends for you, and don't have to pay income taxes on the dividends until you sell the stock. In this manner you can actually defer the payment of taxes.

This month's program is appropriately named *DIV.BA* for *DIV*idend. Figure 1 shows the main menu of *DIV*. Your options are pretty straightforward. You can select a stock, enter a new dividend, edit a prior dividend or print a dividend summary.

If you look closely, you may see a lot of similarities between this program and last month's. Sure enough, in Line 2, if you change the variable MD to equal 16, this program will run very neatly on your Model II, 12, or 16 under TRSDOS.

Before you can work with a stock you have to select it with option one. But, before you do this, you must set up a *TEXT* file with two pieces of information. The first piece of data indicates the number of shares of that stock you own. The second goes on the second line; this is an asterisk (*) which tells the program that this is the end of the file. Simply name your text file with the stock's ticker symbol, i.e., TAN, IBM, etc. Enter the number of shares, then press ENTER. Next, type an asterisk and again press ENTER. Finally, press the F8 function key to return to the Model 100's main screen and you're done!

Once you've selected your stock, you're ready to enter data. By selecting option two, you can enter the current quarter's dividend. You will notice something very interesting here. After you enter the date and the current dividend, the program displays the previous amount of shares you owned before they were reinvested. This is the field "P. Shrs." Below this is the amount of money ready to be reinvested; this is simply the current dividend multiplied by the previous number of shares, and is displayed in the field labeled "Dollars." Next, you are asked to enter the price per share at which the company purchased shares for you. The program then computes the number of shares to be purchased to four decimal places. Now comes the tricky part. As noted above, sometimes the basis, or price you actually must claim that you paid for the shares, is different from the price at which they were purchased. If this is the case, enter the basis price per share as indicated on your quarterly statement. If the basis is the same as the purchase price, just re-enter the purchase price. *DIV* will now compute your taxable income based on the basis purchase price of your shares and will automatically update the field labeled "N. Shrs" to reflect the new number of shares you own. This number will appear next quarter in the "P. Shrs" (previous shares) field.

If you somehow made a mistake and need to edit an entry, option three from the main menu will assist you. When the entry screen comes up, you are asked to enter the date of the transaction you wish to edit (remember, 1/1/84 is different from 01/01/84, so be consistent!). *DIV* will search all your entries, display the one you've selected, and ask you which field to edit. You can change the date, price per share, dividend, or basis price.

Hold on there! If you change any of these numbers (except the date, of course), you're going to make all future number of shares, dollars, etc. wrong, right? Well, in theory, that's correct. If you change a dividend, that will change the number of shares purchased and, in turn, mess up the number of new shares on this record so that you won't have the correct number of previous shares for the next quarter. Well relax! I've thought this one out and you don't have to worry. Any time you make a change in an entry, *DIV* will automatically recalculate all dividends, shares, income, *everything*. Just like a spreadsheet!

Last, but not at all least, is the option to print out the summary (see Figure 4). This lists all the fields on the input screen in neat columns so you can see how the power of reinvesting dividends adds stock to your portfolio.

Well Dow Jonesers, there you have it. Keep track of your dividends and maybe, just maybe, some day you'll remember you got it from "On The Road," you'll drive up to Prospect, Ky., in your Rolls Royce and you'll say: "Who was that programmer? I wanted to thank him."

Til next time.

Figure 1

```
          Main Menu
    PCM Dividend Manager
Stock: None
        1  Select Stock
        2  Input Dividend
        3  Edit Dividend
        4  Print Summary
        5  End Program
           Select:
```

Figure 2 Inputting Dividend

```
       Input Dividend
  ATT     Records on file: 7
  Date:04/01/79    Dividend: 1.2500
P. Shrs:115.720       Shares: 2.4830
Dollars:144.650        Basis:61.1875
Pr/Shr:58.2350        Income:151.984
                     N. Shrs:118.204
      Entry Correct (Y/N):
```

Figure 3 Editing An Existing Entry

```
Record #4   Edit Dividend
     ATT     Records on file: 7
1     Date:01/01/79 3  Dividend: 1.1500
   P. Shrs: 106.066        Shares: 2.1144
   Dollars:121.980  4      Basis:60.6250
2    Pr/Shr:57.6888        Income:128.188
                          N. Shrs:108.180
Enter field to edit, or "F1" to exit
```

Figure 4

Dividend Summary Listing

Stock: ATT

Date	Shares Held	Dividend Rate	Dollar Amount	Price Share	Shares Bought	Tax Basis Per Share	Taxable Income
04 01 78	100.000	1.15	115.00	58.3775	1.970	61.2187	120.60
07 01 78	101.970	1.15	117.27	57.0475	2.056	59.9062	122.15
10 01 78	104.026	1.15	119.63	58.6506	2.040	62.4375	127.35
01 01 79	106.066	1.15	121.98	57.6888	2.114	60.6250	128.19
04 01 79	108.180	1.25	135.23	58.2350	2.322	61.1875	142.09
07 01 79	110.502	1.25	138.13	54.7200	2.524	57.6250	145.46
10 01 79	113.026	1.25	141.28	52.4519	2.694	55.1250	148.48

SOFTWARE

MASS - 11pc Word Processor Available For The 2000

Microsystems Engineering Corporation of Chicago has begun offering a "feature-rich" word processor package for the Tandy 2000 and other MS-DOS computers — *MASS - 11pc*.

According to Microsystems, the features, functionality and speed of the product are virtually identical to the parent product, *MASS - 11*, running on Digital's VAX family of super minicomputers under the VMS operating system.

Features incorporated include: generation of table of contents, split-screen editing, list processing, utilities, footnoting, column mathematics, four-function calculator, multiple columns, scientific equation editing, redlining, user-defined keys, automatic page numbering, stored text, automatic headers and footers, multiple wrap tabs, sub- and superscripts, line drawing, and multiple font and pitch changes. This sytem also provides for automatic pagination with widow and orphan control.

MASS - 11pc is also compatible with commerical electronic mail services.

Contact Microsystems Engineering Corp., 2400 W. Hassell Road, Suite 400, Hoffman Estates, IL 60195; (312) 882-0111.

Port-A-Soft Offers Downloading Service

Port-A-Soft has announced that through implementations of new equipment and software, it is able to download to/from approximately 250 different microcomputer diskette formats as well as industry standard nine track tape. The 250 different diskette formats come from more than 12 different operating systems, including CP/M, MS-DOS, PC-DOS, North Star DOS, Apple DOS, Apple Pascal, IBM P-System, Turbodos, Valdocs, NEC DOS, TRS-DOS, and LDOS. Additionally, data can be downloaded to/from diskettes produced by various DEC minicomputer and IBM mainframe operating systems. Various proprietary word processing formats are supported as well.

In addition to providing the downloading service to customers, Port-A-Soft also sells both hardware and software to enable customers to perform their own downloading where such is desired.

For more information, and a catalog of downloading services, hardware and software, contact Port-A-Soft, 423 East 800 North, Orem, UT 84057, phone (801) 226-6704.

The listing:

```
1 CLEAR 500:CLS:DEFSTR A,R,U:DEFINT I-N
2 MD=100:NN=40
3 GOTO 15
4 FOR I=0 TO 7:IR=I:IC=0:A=BL$+" ":GOSUB
 8:NEXT I:RETURN 'clear screen
5 MD=16:GOTO 15
6 X=0:Y=0:IF ABS(FL)=1 THEN IN$=INPUT$(1
) ELSE LINE INPUT IN$
7 X=VAL(IN$):IF IN$<>"" THEN Y=ASC(IN$):
RETURN ELSE RETURN
8 IF MD=16 THEN PP=(IR+8)*80+IC+20 ELSE
PP=IR*40+IC
9 PRINT@PP,A;:RETURN
10 IF MD=16 THEN PP=(IR+8)*80+IC+20 ELSE
 PP=IR*40+IC
11 IF X>999 THEN FF$=F$(1) ELSE IF X>99
THEN FF$=F$(2) ELSE FF$=F$(3)
12 PRINT@PP,USING FF$;X;:RETURN
13 IF X>999 THEN FF$=F$(1) ELSE IF X>99
THEN FF$=F$(2) ELSE FF$=F$(3)
14 LPRINT USING FF$+"  ";X;:RETURN
15 IF MD=16 THEN R=CHR$(26):U=CHR$(25) E
LSE IF MD=100 THEN R=CHR$(27)+"p":U=CHR$
(27)+"q"
20 IF MD=16 THEN IR=-1:IC=-1:A=CHR$(128)
+STRING$(40,150)+CHR$(129):GOSUB 8:FOR I
=1 TO 8:IR=I-1:IC=-1:A=CHR$(148)+STRING$
(40," ")+CHR$(149):GOSUB 8:NEXT I:IR=8:I
C=-1:A=CHR$(131)+STRING$(40,150)+CHR$(13
0):GOSUB 8
25 IF MD=16 THEN EE$="*" ELSE EE$="F1":K
EY 1,"*"+CHR$(13)
30 BL$=STRING$(39," ")
32 NN=24
35 DIM DT$(NN),DV(NN),PS(NN),DL(NN),PR(N
N),BA(NN),IC!(NN),SP(NN),NS!(NN)
50 DATA "Select Stock","Input Dividend",
"Edit Dividend","Print Summary","End Pro
gram"
55 FOR I=1 TO 5:READ M0$(I):NEXT I
60 DATA "P. Shrs","Dollars"," Pr/Shr","
Shares","   Basis","  Income"," N. Shrs
65 FOR I=1 TO 7:READ DE$(I):NEXT I
70 F$(1)="####.##":F$(2)="###.###":F$(3)
="##.####"
90 ST$="None"
100 GOSUB 4
105 GOSUB 1000
110 IR=0:IC=9:A=R+" PCM Dividend Manager
 "+U:GOSUB 8
120 FOR I=1 TO 5:IR=I+1:IC=12:A=R+STR$(I
)+" "+U+" "+M0$(I):GOSUB 8:NEXT I
125 IR=1:IC=1:A="Stock: "+ST$+"    ":GOSU
B 8:IF ST$<>"None" THEN IR=1:IC=20:A="Re
cords on File:"+STR$(IX):GOSUB 8
130 IR=7:IC=16:A="Select: ":GOSUB 8
140 A=INPUT$(1):X=VAL(A):IF X<1 OR X>5 T
HEN 130
150 FX=X:ON FX GOTO 400,200,300,500,600
200 'input
202 IF ST$="None" THEN 100
205 A1=R+" "+M0$(FX)+" "+U:GOSUB 800
210 GOSUB 700:GOSUB 1300:GOSUB 760
212 DT$="":PS=0:DL=0:PR=0:BA=0:IC!=0:SP=
0:NS!=0
215 GOSUB 230:IF IN$="*" THEN GOSUB 1100
:GOTO 100 ELSE IR=7:IC=0:A=BL$:GOSUB 8:G
OSUB 235:GOSUB 240:GOSUB 245:GOSUB 250:G
OSUB 255:GOSUB 260:GOSUB 265:GOSUB 270:G
OTO 280
230 IR=2:IC=12:A="":GOSUB 8:GOSUB 6
232 IF DT$="" AND IN$="" THEN 230 ELSE I
F IN$<>"" THEN DT$=IN$
233 IR=2:IC=12:A=STRING$(8," "):GOSUB 8:
A=DT$:GOSUB 8:RETURN
235 'enter dividend
236 IR=2:IC=33:A="":GOSUB 8:GOSUB 6
237 IF DV=0 AND IN$="" THEN 235 ELSE IF
IN$<>"" THEN DV=X
238 IR=2:IC=33:X=DV:GOSUB 10:RETURN
240 'display prev. shares
242 IF IX=0 THEN PS=SH ELSE PS=NS!(IX)
243 IR=3:IC=12:X=PS:GOSUB 10:RETURN
245 'display dollars available
246 DL=DV*PS:DL=INT((DL+.005)*100)/100
248 IR=4:IC=12:X=DL:GOSUB 10:RETURN
250 'input price per share purchased
251 IR=5:IC=12:A="":GOSUB 8:GOSUB 6
252 IF PR=0 AND IN$="" THEN 250 ELSE IF
IN$<>"" THEN PR=X
253 IR=5:IC=12:X=PR:GOSUB 10:RETURN
255  display shares purchased this trans
action
256 SP=DL/PR
258 IR=3:IC=33:X=SP:GOSUB 10:RETURN
260 'input tax basis
261 IR=4:IC=33:A="":GOSUB 8:GOSUB 6
262 IF BA=0 AND IN$="" THEN 260 ELSE IF
IN$<>"" THEN BA=X
263 IR=4:IC=33:X=BA:GOSUB 10:RETURN
265 'display income
266 IC!=BA*SP
268 IR=5:IC=33:X=IC!:GOSUB 10:RETURN
270 'display new shares
271 NS!=PS+SP
273 IR=6:IC=33:X=NS!:GOSUB 10:RETURN
280 IR=7:IC=10:A="Entry Correct (Y/N): "
:GOSUB 8:A=INPUT$(1):IF A="N" OR A="n" T
HEN 200 ELSE IF A$<>"Y" AND A<>"y" THEN
280 ELSE PRINT A;
285 IX=IX+1
290 DT$(IX)=DT$:DV(IX)=DV:PS(IX)=PS:DL(I
X)=DL:PR(IX)=PR:BA(IX)=BA:IC!(IX)=IC!:SP
(IX)=SP:NS!(IX)=NS!
295 GOTO 200
300  edit
302 IF ST$="None" THEN 100
305 A1=R+" "+M0$(FX)+" "+U:GOSUB 800
310 GOSUB 700:GOSUB 1300
315 IR=2:IC=0:A=R+STR$(1)+" "+U:GOSUB 8
316 IR=5:IC=0:A=R+STR$(2)+" "+U:GOSUB 8
317 IR=2:IC=20:A=R+STR$(3)+" "+U:GOSUB 8
318 IR=4:IC=20:A=R+STR$(4)+" "+U:GOSUB 8
330 IR=7:IC=2:A="Enter Date of Entry, or
```

```
"+EE$+"' to exit":GOSUB 8
335 IR=2:IC=12:A="":GOSUB 8
340 GOSUB 6:IF IN$="*" THEN GOSUB 1100:G
OTO 100 ELSE IR=7:IC=0:A=BL$:GOSUB 8
345 FOR I=1 TO IX:IF IN$=DT$(I) THEN 355
 ELSE NEXT I
350 IR=7:IC=3:A="Entry not found, press
any key ... ":GOSUB 8:A=INPUT$(1):GOTO 3
00
355 DT$=DT$(I):DV=DV(I):PS=PS(I):DL=DL(I
):PR=PR(I):BA=BA(I):IC!=IC!(I):SP=SP(I):
NS!=NS!(I)
360 GOSUB 233:GOSUB 238:GOSUB 243:
248:GOSUB 253:GOSUB 258:GOSUB 263:
268:GOSUB 273
362 IR=0:IC=0:A="Record #"+MID$(S
2):GOSUB 8
365 IR=7:IC=1:A="Enter field to e
 "+EE$+"' to exit: ":GOSUB 8
370 IN$=INPUT$(1):IF IN$="*" THEN
SE IF IN$<"0" OR IN$>"4" THEN 365
=VAL(IN$)
375 ON N GOSUB 230,250,235,260
377 IF N<>1 THEN GOSUB 245:GOSUB
UB 265:GOSUB 270
385 DT$(I)=DT$:DV(I)=DV:PS(I)=PS:
L:PR(I)=PR:BA(I)=BA:IC!(I)=IC!:SP
NS!(I)=NS!:GOSUB 1200
387 GOTO 362
390 IF MD=100 THEN GOSUB 6
395 GOTO 300
400 'select stock id
405 IR=7:IC=15:A=STRING$(15," "):
410 IR=1:IC=8:A=STRING$(8," "):GO
415 IR=1:IC=8:A="":GOSUB 8:GOSUB
420 A=LEFT$(IN$,1):IF A<"A" OR A
N 410
425 ST$=IN$
430 GOTO 100
500 'print totals
510 IF ST$="None" THEN 100
520 IR=7:IC=0:A=BL$:GOSUB 8:IR=7:
="Printing ... ":GOSUB 8
525 LPRINT TAB(35)"Stock: "ST$:LP
530 LPRINT TAB(10);"Shares   Divi
ollar   Price/   Shares  Tax Basi
ble"
535 LPRINT"  Date       Held        Ra
Amount   Share    Bought  Per Shar
ome
540 LPRINT"--------  -------  ----
------  -------  -------  -------
----
550 FOR I=1 TO IX:LPRINT USING"\
####.###     ##.##  ####.## ###.##
.###    ###.####  ####.##";DT$(I);F
V(I);DL(I);PR(I);SP(I);BA(I);IC!(
560 NEXT I
590 GOTO 100
600 'end program
610 IF MD=16 THEN CLS:END ELSE ME
700 'display screen
705 IR=2:IC=07:A="Date:          ":
8
710 IR=2:IC=24:A="Dividend:      ":
8
715 FOR I=1 TO 7:IR=I+2:IF (IR)5 A
) THEN IR=IR-3
720 IC=4:IF I>3 THEN IC=24
730 A=DE$(I)+":          ":GOSUB 8
740 NEXT I
755 RETURN
760 'F1
765 IR=7:IC=7:A="Enter "+R+" "+EE$+" "+U
+" for DATE to exit":GOSUB 8:RETURN
770 'continue
775 IR=7:IC=5:A="Press any key to contin
ue ... ":GOSUB 8:A=INPUT$(1):RETURN
800 'clear screen # print title
810 GOSUB 4
820 IR=0:IC=20-LEN(A1)/2
830 A=A1:GOSUB 8
840 RETURN
900 'open for input
910 IF MD=16 THEN OPEN"I",1,ST$+"/DAT" E
LSE OPEN ST$+".DO" FOR INPUT AS 1
915 INPUT #1,SH
920 RETURN
930 'open for output
940 IF MD=16 THEN OPEN"O",1,ST$+"/DAT" E
LSE OPEN ST$+".DO" FOR OUTPUT AS 1
950 RETURN
960 'input all data
1000 'read in all data
1005 IF ST$="None" THEN IX=-1:RETURN
1010 GOSUB 900:IX=0
1020 IF EOF(1) THEN CLOSE:RETURN ELSE IN
PUT #1,DT$
1030 IF DT$="*" OR EOF(1) THEN CLOSE:RET
URN
1040 INPUT #1,DV,PS,DL,PR,BA,IC!,SP,NS!
1050 IX=IX+1
1060 DT$(IX)=DT$:DV(IX)=DV:PS(IX)=PS:DL(
IX)=DL:PR(IX)=PR:BA(IX)=BA:IC!(IX)=IC!:N
S!(IX)=NS!:SP(IX)=SP
1070 GOTO 1020
1100 'save all data
1110 IR=7:IC=1:A=BL$:GOSUB 8:IR=7:IC=12:
A="Saving Data ... ":GOSUB 8:GOSUB 930
1115 PRINT #1,SH
1120 FOR I=1 TO IX
1125 GOSUB 1200
1130 PRINT #1,CHR$(34);DT$(I);CHR$(34);"
,";DV(I);",";PS(I);",";DL(I);",";PR(I);"
,";BA(I);",";IC!(I);",";SP(I);",";NS!(I)
1140 NEXT I:PRINT #1,CHR$(34);"*";CHR$(3
4):CLOSE
1150 RETURN
1200 'refiqure all transactions
1210 FOR J=1 TO IX
1220 IF J=1 THEN PS(J)=SH ELSE PS(J)=NS!
(J-1)
1230 DL(J)=DV(J)*PS(J):DL(J)=INT((DL(J)+
.005)*100)/100
1240 SP(J)=DL(J)/PR(J)
1260 IC!(J)=BA(J)+SP(J)
1270 NS!(J)=PS(J)+SP(J)
1280 NEXT J
1290 RETURN
1300 'display records
1320 IR=1:IC=6:A=R+" "+ST$+" "+U+" "+"Re
cords on file:"+STR$(IX):GOSUB 8:RETURN
```

PCM

DICE TOTAL PROBABILITIES

Bob Delbourgo

Here's a quick little program which enables you to explore the outcome of multiple throws of variously shaped die.

PoCo firstly calculates the mathematical probability of each possible outcome. This is then tested by evaluating the results of 2,000 random throws and plotting the results on a histogram along with the mathematical probability previously calculated.

The Listing:

```
1 CLS:PRINT@1,STRING$(38,157);:PRINT@281
,STRING$(38,157);:FORI=1TO6:PRINT@40*I,C
HR$(157);:PRINT@40*I+39,CHR$(157);:NEXT
2 PRINT@58,"DICE";:PRINT@97,"TOTAL";:PRI
NT@134,"PROBABILITIES";:PRINT@172,"BY BO
B DELBOURGO";:PRINT@210,"15, WILLOWDENE
AVENUE";:PRINT@244,"HOBART, TASMANIA, AU
STRALIA 7005";
3 SOUND5586,20:SOUND6269,20:SOUND7032,10
:SOUND7456,10:SOUND6269,20:SOUND8386,40:
SOUND7032,20:SOUND7032,20:SOUND7456,10:S
OUND8386,10:SOUND9394,10:SOUND8386,10:SO
UND7456,40
4 CLS:PRINT@13,"DICE PARAMETERS":PRINT:I
NPUT"# of faces on the die";M:IFM<0ORM<>
INT(M)THEN4
5 PRINT@120,"# of dice thrown simultaneo
usly";:INPUTN:IFN<0ORN<>INT(N)THEN5
6 CLS:PRINT"# Faces ="M:PRINT@19,"# Dice
 thrown ="N:PRINT"POCO will now CALCULAT
E the mathematicalchances for the totals
 which range from "N"to"N*M"in integer s
teps .."
7 FORI=1TOVAL(RIGHT$(TIME$,1)):R=RND(1):
NEXTI
8 D=(M-1)*N+1:DIMA(D),B(D),P(M*N),Q(D)
9 A(1)=1:IFI)1THENA(I)=0
10 FORT=1TON:B(1)=1
11 FORJ=2TOM-1:B(J)=B(J-1)+A(J):NEXTJ
12 FORJ=M TOD:C=A(J-M+1):FORK=2TOM:C=C+A
(J-M+K):NEXTK:B(J)=C:NEXTJ
13 FORI=1TOD:A(I)=B(I):B(I)=0:NEXTI,T:L=
INT(1+(M-1)*N/2)
14 FORI=1TOD-1:PRINTA(I);CHR$(32);:NEXTI
:PRINTA(D)
15 PRINT@290,"<SPACEBAR> to continue";
16 IFINKEY$<>CHR$(32)THEN16
17 CLS:PRINT:PRINT" POCO will now EVALUA
TE the results of   throwing"N"dice with
"M"faces 2000":PRINT" times. It plots a
histogram, comparing  it with the theore
tical prediction.    PLEASE WAIT a minu
te or so for POCO to  do its work ..."
18 F=INT(2000/N):E=A(L)*F/(M^N)
19 R=0:FORS=1TOF:PRINTCHR$(144);:FORK=1T
ON:R=R+INT(M*RND(1)+1):NEXTK:P(R)=P(R)+1
:R=0:NEXTS
20 FORI=N TOM*N:Q(I-N+1)=P(I):NEXTI:
21 CLS:LINE(20,3)-(220,53),1,B:PRINT@295
,"TOTALS";:PRINT@283,N;:PRINT@314,M*N;
22 C$="CHANCES":FORI=0TO6:PRINT@40*I+2,M
ID$(C$,I+1,1);:NEXTI
23 FORI=1TOD:LINE(20+200*(I-1)/D,53-35*Q
(I)/E)-(20+200*I/D,53),1,B:NEXTI
24 FORI=1TOD:X=20+200*(I-.5)/D:Y=53-35*A
(I)/A(L):PSET(X,Y):PSET(X-1,Y-1):PSET(X+
1,Y+1):PSET(X-1,Y+1):PSET(X+1,Y-1):NEXTI
25 C$="ANY KEY":FORI=0TO6:PRINT@40*I+38,
MID$(C$,I+1,1);:NEXTI:FORI=0TO6:PRINT@40
*I+38," ";:NEXTI:IFINKEY$=""THEN25ELSERU
N4
```

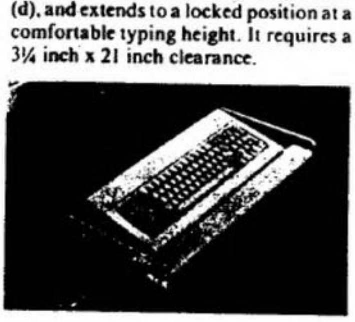
*

New Products

Slide Drawer For Detachable Keyboards

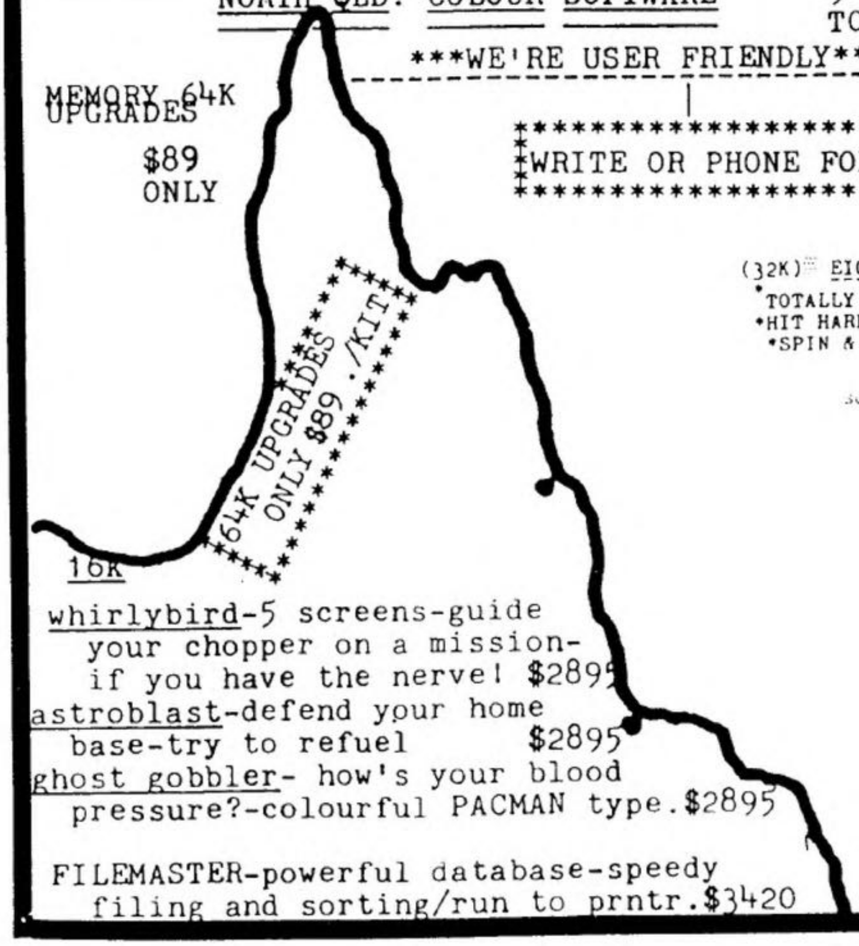
A space-saving addition for owners of microcomputers or terminals with detachable keyboards is the MicroComputer Under Carriage Keyboard Drawer, by MicroComputer Accessories, Inc.

This cantilever drawer, supported by ball bearing slides, can be installed and securely fastened under a desk top, shelf, or table top. The drawer holds a keyboard with dimensions up to 2¾ inches (h) by 20 inches (w) x 9½ inches (d), and extends to a locked position at a comfortable typing height. It requires a 3¼ inch x 21 inch clearance.



When not in use, the platform drawer remains a handy accessory by positioning the keyboard directly underneath the desk and out of the operator's way. Available in two colors: putty and black. Suggested retail price is $54.

For more information write to MicroComputer Accessories, Inc., 5721 Buckingham Parkway, Culver City, CA 90230, or phone (213) 641-1800.


64K UPGRADES ONLY $89 /KIT
16K

COLORVISION

SOFTWARE FOR THE T.R.S. 80 COLOR COMPUTER

PHONE - 03 890 0746 (AH) 'FIRST'

CASH MAN...	(32K)...	$31
LANCE?.....	(32K)...	$36
FLIGH:.....	(32K)...	$26
SEAQUEST...	(32K)...	$30
BLOCHEAD...	(16K)...	$30
DOODLE BUG.	(16K)...	$30
RAIL RUNER.	(16K)...	$26
ZONE 6.....	(16K)...	$30
FILEMASTER.	(16K)...	$35
ELITE*CALC.	(16K)...	$60

WE STOCK A LARGE RANGE OF COCO SOFTWARE. BUY A PROGRAM FROM US THIS MONTH AND YOU'LL RECIEVE A LIST OF POKES + EXECS PLUS A SHEET OF USEFUL HINTS ABOUT CASSETTE HANDLING. AND REMEMBER, WE OFFER OUR CUS-TOMERS THE BEST SERVICE AND THE BEST PRICES! PH:03 8900746

GEMINI PRINTER SPECIAL $645. SAVE HEAPS !!!!

COLORVISION COMPUTER SOFTWARE

1St Floor. 596 Station St. Box Hill. Vic. 3128.
PHONE: (03) 890 0746 Ah

SALLY LEE SALLY LEE SALLY LEE

SOFTWARE SPECTRUM
TRS-80 COLOR COMPUTER SOFTWARE
GAMES
ADVENTURES
UTILITIES
APPLICATION PROGRAMMES
EDUCATION PROGRAMMES
PRINTERS & DISC DRIVES

AVAILABLE FROM: SALLY LEE
39 NOLL ST
PORT PIRIE S.A
(086) 32 6196

SALLY LEE SALLY LEE SALLY LEE

SALLY LEE SALLY LEE SALLY LEE SALLY LEE

Want to Connect Your CO-CO to a PARALLEL PRINTER?

Want to RUN your DMP 100/200 up to 30% FASTER in serial mode?

THEN INSTALL A :

MK1 PARALLEL PRINTER INTERFACE

- Locally designed and manufactured
- Compatible with any Standard Centronics Parallel Printer
- Plugs into CO-CO Serial Port and includes all cables and connectors.
- Increases Printing Speed by up to 30% on Tandy DMP 100/200 Printers when using 4800 or 9600 Baud Rate on your CO-CO.
- Features Six switch selectable baud rates (300-9600)
- Power Pack is required for Printers not supplying power at pin 18 on the Parallel Connector

ONLY: $75 (plus P&P $4)

Add $10 for Power Pack if required.

Available from: Geoff Fiala
18 RUSSELL CRES,
WESTLEIGH. NSW.2120

or phone Sydney (02)-84 3172

Get useful Size Graphics Dumps

```
*************
*  G.S.P.R.  *
*************
```

1. <u>DOUBLE</u> & Single size Prints
2. Standard & Reverse Prints
3. Fully Relocatable Mach Lang
4. <u>Disk</u> Compatible
5. Works in ALL Pmodes
6. Shifts Image across page
7. Full Instructions supplied
8. Suit LPVII, VIII, DMP100-400 CP80, DT80&(STAR)GEMINI 10

Cost $15 Specify Printer &Ram
M. DRAKE P.O. BOX 140
WOOLLOONGABBA 4102 Qld

For all commercial advertising in Australian RAINBOW contact TOTO advertising Box 5730, Gold Coast Mail Centre Qld 4217 or phone (075) 39 2003

D & L WILSONS

serving melbourne's east

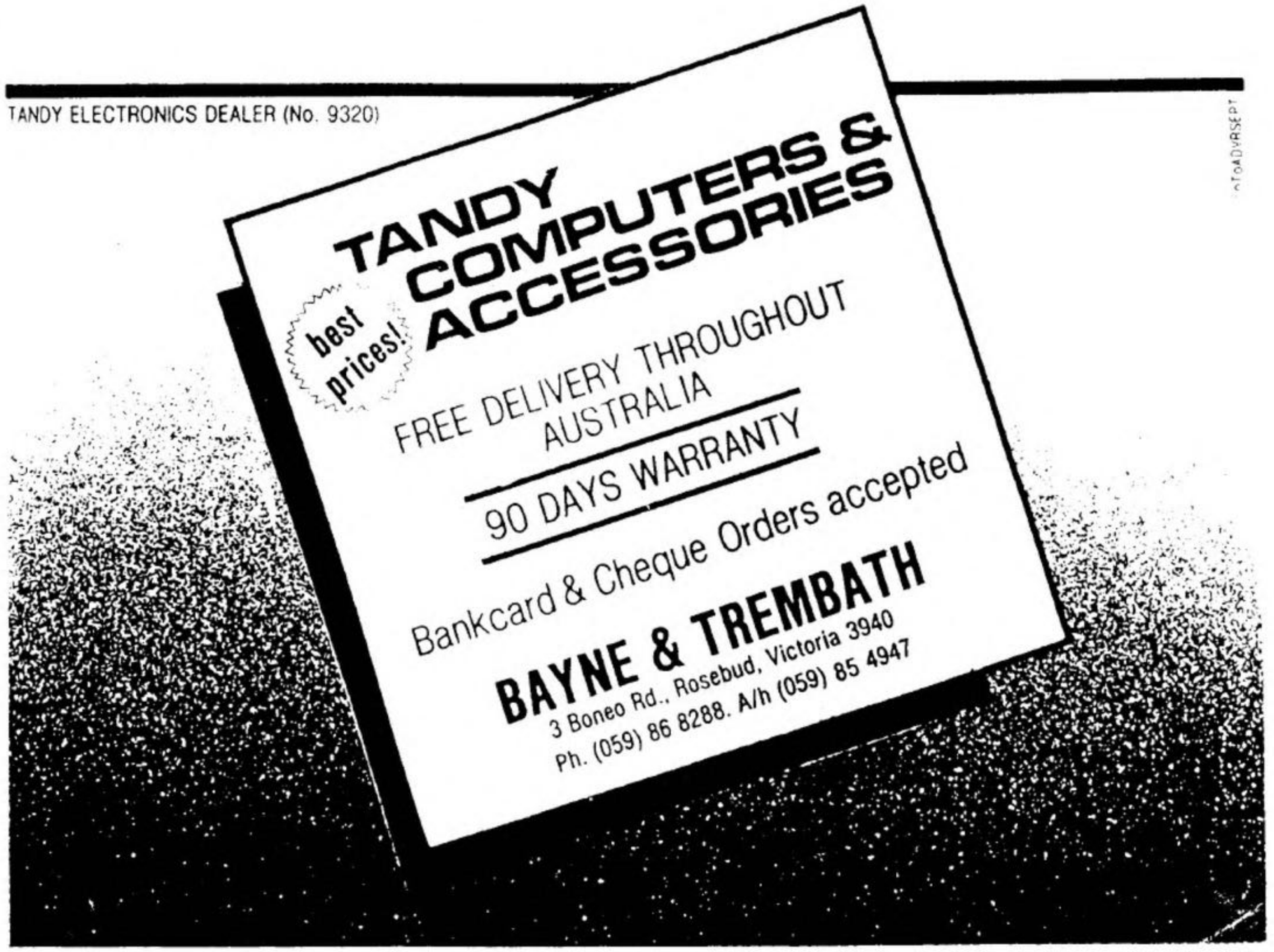
AUGUST [illegible] ? We have:- Senquest, [illegible], Stagecoach, Scepter of Kzirgla and more from $11.95 to $29.95.

We also have "VIDEO", the disk you have when you dont have a disk. Use your 64K at last.

AUGUST EXTRA SPECIAL. A great way to get organised. THE TWO TIERED COMPUTER DESK YOU CAN AFFORD !!! $79

doug & louise wilson -
6 stafford st -
blackburn - phone 898-4521

TANDY ELECTRONICS DEALER (No. 9320)



TANDY COMPUTERS & ACCESSORIES

best prices!

FREE DELIVERY THROUGHOUT AUSTRALIA

90 DAYS WARRANTY

Bankcard & Cheque Orders accepted

BAYNE & TREMBATH
3 Boneo Rd., Rosebud, Victoria 3940
Ph. (059) 86 8288. A/h (059) 85 4947

SOFTWARE SPECTRUM

correspondence: **P.O. Box 2101,**
Adelaide, South Australia, 5001.

deliveries: **25 Selby Street, Adelaide**
Phone: (08) 211 8763 or 51 4868

TRS 80® COLOR COMPUTER SOFTWARE SPECIALISTS

Serial to Parallel Printer Interface Switchable from 600–9600 BAUD $99.95 Non-switchable 4800 BAUD $69.95

HJL–57 Professional Keyboard $139.00 Disk Drives 40–Track with controller $627.00

Gemini 10x Printer with Interface & Word Processor $649.95

The VIP Library is further enhanced by the recent arrival of VIP CALC $64.95 on tape. $69.95 on disk (32 and 64k version supplied)

SOFTWARE

Learn to Read Music	$41.95	Musica	$41.95
Storm Arrows	$30.95	Beam Rider	$30.95
Cubix	$30.95	Shaft	$30.95
Tut's Tomb	$30.95	Black Sanctum (Graphics)	$30.95
Candy Co	$36.95	Erland	$30.95
Pengon	$30.95	Lunar Rover Patrol	$30.95

All products come with an unconditional Guarantee. If defective return within 15 days for replacement. **Dealer inquiries invited**

✂---

SEND $1.00 POSTAGE FOR CATALOGUE.

To: SOFTWARE SPECTRUM, Box 2101 GPO, Adelaide 5000 **New Zealand Bankcard welcome**

Mr/Mrs/Miss/Ms .... Address .... Postcode ....

Phone .... Please charge Bankcard No. □□□□ □□□□ □□□□ □□□□ Exp. Date .... Sig. ....

TRS80, COM 64, VIC 20, T194-4, TIMEX/SINCLAIR are the registered trademarks of Tandy Corp., Commodore Business Machines, Texas Instr. and Timex respectively.

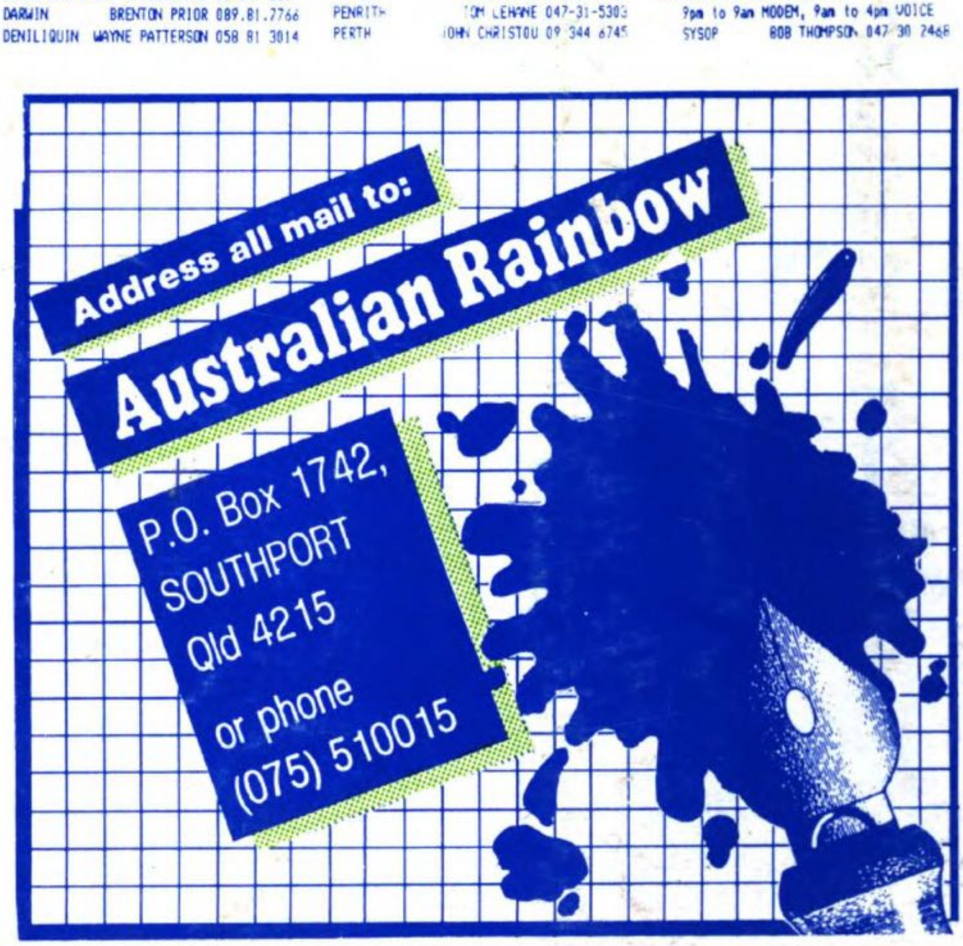
THE RAINBOW
Registered by Australia Post-
Registration No. QBG 4009
Graham Morphett, P.O.Box 1742, Southport. QLD. 4215.

user group CONTACTS

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

ADELAIDE	JOHN HAINES	08 278 3560
ADELAIDE NTH	STUN EISENBERG	08 250 6214
ALBURY	RON DUNCAN	060 43 1031
BAIRNSDALE	COLIN LEHMANN	051 57 1545
BALLARAT	MARK BEVELANDER	053 32 6733
BANKSTOWN	KEN HAYWARD	02 759 2227
BLACKTOWN	KEITH GALLAGHER	02-627-4627
BLAXLAND	BRUCE SULLIVAN	047 39 3903
BOWEN	TONY EVANS	077 86 2220
BRASSALL	BOB UNSWORTH	07 201 8659
BRIGHTON	GLENN DAVIES	08 296 7477
BRISBANE EAST	ROB THOMPSON	07 848 5512
BRISBANE NTH	JACK FRICKER	07 262 8869
BRISBANE STH	PATRIC SIMONIS	07 209 3177
BRISBANE SW	GRAHAM BUTCHER	07 376 3400
BRISBANE WEST	BRIAN DOUGAN	07 30 2072
BUNDABERG	JIM McPHERSON	071 72 8329
CAMBERWELL	TONY BALDWIN	03 728 3676
CAMPBELLTOWN	LEO GINLEY	02 605 4572
CANBERRA	SHAUN WILSON	062 51 2339
CARLISLE MiCo	STUART HALL	361 1922
CAULFIELD"?526.."	JEFF SHEEN	03 528 3724
CHATSWOOD	BILL O'DONNELL	02 411 3336
CHURCHILL	GEOFF SPOWART	051 22 1389
COLYTON TEENS	DWAYNE MANSON	02 623 5805
COOMA	SHEILA HAMMILL	064.82.3905
DANDENONG	BRETT CRUICKSHANK	03 547 6604
DARWIN	BRENTON PRIOR	089.81.7766
DENILIQUIN	WAYNE PATTERSON	058 81 3014
DUBBO	GRAEME CLARKE	068 89 2095
EMERALD	CAROL CATHCART	059 68 3026
FORSTER	GARY BAILEY	065 54 5029
FRANKSTON	BOB HAYTER	03.783.9748
GIPPSLAND STH	PAT KERMODE	056 74 4583
GOLD COAST	SHERYL BENTICK	075-39-2003
GOSFORD	PETER SEIFERT	043 32 7874
GRAFTON	DAVID HULME	066.42.0627
GREENACRES	BETTY LITTLE	08 261 4083
HOBART	BOB DELBOURGO	002 25 3896
IPSWICH	MILTON ROWE	07 281 4059
JUNEE	P. MALONEY	069 24 1860
KENMORE	GRAHAM BUTCHER	07 376 3400
LITHGOW	STUART RAYNER	063 51 4214
LIVERPOOL	LEONIE DUGGAN	02-607-3791
MACKAY	LEN MALONEY	079511333x782
MACLEOD	ROBIN ZIUKELIS	03 450211x465
MacQUARIEFIELDS	KIETH ROACH	02 618 2858
MAFFRA	MAX HUCKERBY	051 45 4315
MAITLAND	LYN DAWSON	049 49 8144
MELBOURNE	JEFF SHEEN	03 528 3724
MELTON	MARIO GERADA	03 743 1323
MILDURA	DOUG MATTHEWS	050 23 5701
MOE	STEPHEN SEMPLE	051 27 6841
MORWELL	GEORGE FRANCIS	051 34 5175
MT ISA	PAUL BOUCKLEY-SIMONS	077 43 6286
MUDGEE	BRIAN STONE	063-72-1958
NAMBUCCA HDS	WENDY PETERSON	065 68 6723
NEWCASTLE	LYN DAWSON	049 49 8144
NOWRA	ROY LOPEZ	044 48 7031
PARKES	DAVID SMALL	068 62 2682
PENRITH	TOM LEHANE	047-31-5303
PERTH	JOHN CHRISTOU	09 344 6745
PORT MacQUARIE	RON LALOR	065 83 8223
PORT NOARLUNGA	ROB DALZELL	08 386 1647
PORT PIRIE	KEVIN GOWAN	086 32 1368
RINGWOOD	ANDREW RAWLINGS	03 726 6521
ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
ROCKHAMPTON MiCo	TIM SHANK	079 28 1846
ROSEVILLE	KEN UZZELL	02 467 1619
SALE	BRYAN McHUGH	051 44 4792
SINGLETON	DAVID NICHOLS	065-73-1222
SPRINGWOOD	DAVID SEAMONS	047 51 2107
STURT	MARY DAVIS	08 296 7477
SUNBURY	JACK SMIT	03.744.1355
SUTHERLAND	IAN ANNABEL	02 528 3391
SWAN HILL	BARRIE GERRAND	050.32.3838
SYDNEY EAST	BOB JONES	02-331-4621
SYDNEY TEENS	ROD HOSKINSON	02 48 5948
TAMWORTH	ROBERT WEBB	067 65 7256
TONGALLA	TONY HILLIS	058 59 2251
TOOWOOMBA	MIKE CARTER	076 35 6911
BEGIN NTH	DAVID PROUT	076.32.7533
BEGIN STH	LEW GERSEKOWSKI	076 35 8264
ADVANCED	GRAHAM BURGESS	076 30 4259
TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064
TRARALGON	MORRIS GRADY	051 66 1331
UPPER HUNTER	TERRY GRAVOLIN	065 45 1698
WAGGA WAGGA	BRUCE KING	069 25 3091
WESTLEIGH	ATHALIE SMART	02 848 8830
WHYALLA NORRIE	CHRIS HUNTER	086 45 3395
WOLLONGONG	BRIAN McCAULEY	042 71 4265
WONTHAGGI	PAT KERMODE	056 74 4583

OS9 Bulletin Board System:
9pm to 9am MODEM, 9am to 4pm VOICE
SYSOP BOB THOMPSON 047 30 2468



Address all mail to:
Australian Rainbow
P.O. Box 1742,
SOUTHPORT
Qld 4215
or phone
(075) 510015

POSTAGE
PAID
AUSTRALIA