

For your TANDY Computer

\$3.25

AUSTRALIAN

RAINBOW

INCORPORATING

GOCO

September, 1984

No. 39

NOTICE BOARD!



PRINT #-2,

3325
 AUSTRALIAN RAINBOW GOCO
 INCORPORATING

September, 1988 Volume 1 No. 1

NOTICE BOARD!

Dedicated with Love to the Memory of Greg Wilson

AUSTRALIAN
 EDITOR & PUBLISHER

Graham Morphett

CO-EDITOR

Kevin Mischevski

EDITOR'S ASSISTANT

Christine Lucas

AND GRATEFUL ASSISTANCE FROM

Brian Dougan Peggy Annabel

Richard & Judy Rod Hoskinson

Helga Wilson Patric Simonis

Annette Morphett Glen Mischevski

Jim & Sheryl Bentick

COVER DESIGN

Jim BENGOCK

All Programs in this issue of RAINBOW

are available on cassette tape

SEE CENTRE PAGE FOR DETAILS

OS-9.

Kevin HOLMES is the man to contact for information on OS-9. All Rainbow OS-9 content is sent to him and in turn to those interested, along with a monthly newsletter.

Kevin has joined the U.S. Users' Group, and wants to form a local branch of that group here to give you access to all their public domain software, and to keep you abreast of the latest news.

Kevin appreciates any assistance you can provide in the form of software or hints.

His address is:-

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

Printed by :
 Australian Rainbow Magazine
 P.O. Box 1742
 Southport. Qld. 4215.
 Reg Publication N805033.

You couldn't stay around this office this last month for very long, seeing the various publications coming through the door, reading your letters as they arrived, looking at new software, and playing with the latest CoCo hardware, without saying with John Christou in the latest CoCoPug magazine:

"Because of all this new equipment, my feeling is that CoCo is one of the best buys around."

We've all felt that we had pretty good cause to be proud of our CoCo's at one time or another, but the reason for this latest bout of joy, is probably the realization that with programs like "Graphcom", (a brand new 'window' program that goes a long way towards emulating in one very cheap swoop, the Apple Mackintosh,) OS-9, an operating system so powerful, it will take several years for the average user to understand the extent of the uses to which it can be put; hardware mods like 80 column cards, monitors, and modems; services like active support groups, an interested supplier (yes - meaning Tandy), bulletin boards, and not to mention the best User's magazines in the business (!), we have in the CoCo a machine which on balance, is very difficult to better.

Outside our cosy Tandy world, the IBM PC has come and flopped, the Mackintosh has arrived and everyone has said 'gee wizz' but I don't really think it suits me, and owners of Apple IIe's are wondering why 12 months ago they paid up to 3 times the current price for their computer (meaning, in many cases more than \$10,000.00).

Outside, no one can accept the simple truth that programs of equal quality in any area, be it games, business, education, or research, are available at half or even no price, with our CoCo. Out there they are still paying \$300.00 for cheap word processors!

CoCo has it's faults, but it is possible to change each for the better. Don't be sucked in - when the others are long forgotten, CoCo will remain in one form or another.

This issue marks the gradual return to normality that has to come if you and I are to retain our sanity. In future issues of Australian Rainbow Magazine, look for reviews of the latest American Software and Hardware, news of the latest happenings in the Computer world, and more of those great games and articles that you have come to expect from the team at American Rainbow.

You also get to catch up on CoCo's 'big brother', the Model 2000, and CoCo's portable pal, the model 100.

Welcome to the new Australian Rainbow Magazine!

CoCoConf

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

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

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

The BASSEMBLER is a UNIQUE macro-assembler that allows you to inter-mix assembly language and BASIC in the same program - you can literally have one line assembly and the next line BASIC.

The BASSEMBLER understands the complete 6809 assembly language, macros and other pseudo-ops, labels and LINE NUMBERS too.

A disassembler/monitor Program is included with the BASSEMBLER.

Easy to follow instructions and demonstration programs come with the BASSEMBLER and EXPANDED BASIC.

64K is required. Specify disk or tape (giving model of recorder) and specify COCO model (1 or 2).

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

TO ORDER/REPLY

Buy now, direct from the author:

TINO DELBOURGO
15 WILLOWDENE AVENUE, Sandy Bay, Hobart, Tasmania
7005. Phone (0122) 25 3896

CoCoConf '85 is a reality.

In June '85, (probably the weekend after the Queens Birthday), we are gathering in Surfers Paradise to talk computers and have a bit of fun.

Already, we have commitments from a number of key people who will be there specifically to further your knowledge of your favorite computer, whether it be CoCo, MiCo, Model 100 or the 2000.

CoCoConf will be held in conjunction with the Gold Coast Computer Expo, and your registration fees will include free entry to this very fine Expo. The Expo will feature computers and hardware from a range of Distributors, so you'll be able to see just how well our machines do compare.

Back at CoCoConf, there will be tutorials and/or lectures on appropriate subjects, Software Agents' stalls, we'll see the latest hardware, and have the chance to meet some of those names and voices we've never met face to face. Saturday night is special. Many of the details of Saturday night we are keeping secret for now!

However, the job that we really want to do is to institute the inaugural H. G. Wilson award for services to the Computer Community. The wording of that still sounds a little clumsy to me, so if you feel you can say it better, please do, and please tell me! But you get the point - if you have someone in your computer community who you feel we should honour, then write about that person, preferably as a club.

We will print your entries in Australian CoCo so that everyone has an opportunity to consider all contenders. On the Saturday night, we will all vote on these entries, and the award will be made there and then!

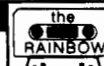
So, if you want to learn more about your computer, see how it compares with other computers, see the latest hardware, see the latest software, meet some friends, honour a friend, or just have some fun, then the place to be in June '85, is CoCoConf!

INDEX

RAINBOW

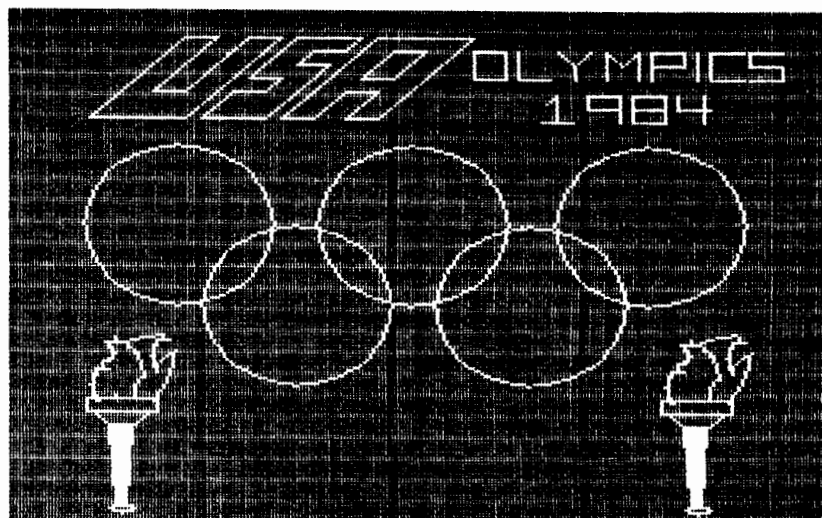
Print #2.....	2	Key Bombers.....	43
Index.....	3	Dragon's Gold...45	
CoCoConf.....	3		
Olympics.....	4		
Interfacing		GoCo	
circuits.....	5		
Colonel Potter's		MS-DOS.part II..49	
Horse.....	11	Flag of the	
Everything about		Month.....	51
CoCo.....	13	dBASE II -	
The Trip.....	16	part 1.....	52
Graphics		MS-DOS.part III.54	
Education....	26	Square Deal.....	55
Order Form.....	29	Zoid Patrol.....	56
		Creating	
Cooking with		Directories...58	
CoCo.....	31	Professor CoCo..58	

GRAPHICS

16K
ECB

RING IN THE OLYMPICS !

By Bill Duke



This is a 16K ECB graphics program that displays the five connecting rings universally known as the symbol of the Olympics. What better time to salute our athletes than in our games issue. Good luck, USA!

The listing:

```

10 PMODE4, 1: PCLS: SCREEN1, 1
20 GOSUB90
30 CIRCLE (126, 80), 30
40 CIRCLE (53, 80), 30
50 CIRCLE (200, 80), 30
60 CIRCLE (90, 110), 30
70 CIRCLE (163, 110), 30
80 GOTO80
90 DRAW"BM55, 10; G30R30E30L10G20L
10E20L10
100 DRAW"BM90, 10; G15R10G5L10G10R
20E20L10E5R10E5L20
110 DRAW"BM115, 10; G30R10E10R10G1
0R10E30L30
120 DRAW"BM120, 15; G10R10E10L10
130 DRAW"BM145, 15; D10R10U10L10
140 DRAW"BM159, 15; D10R10
150 DRAW"BM172, 15; F5E5G5D5
160 DRAW"BM187, 15; D10U10F5E5D10
170 DRAW"BM201, 15; D10U10R10D5L10
180 DRAW"BM215, 15; D10
190 DRAW"BM219, 15; R10L10D10R10
200 DRAW"BM231, 15; R10L10D5R10D5L
10
210 DRAW"BM171, 31; D10R5L10R5U10G
3
220 DRAW"BM181, 31; D5R10U5L10R10D
10
230 DRAW"BM195, 31; D5R10D5L10U5R1

```

```

0U5L10
240 DRAW"BM209, 31; D5R10L2U5D10
250 LINE (33, 185) - (31, 155), PSET
260 LINE (36, 185) - (38, 155), PSET
270 DRAW"BM32, 155; R5
280 DRAW"BM33, 185; R3
290 PAINT (33, 175), 5
300 DRAW"BM31, 155; H5R18G5E5R2U5L
22D5R2
310 CIRCLE (29, 140), 4, , 2, .30, .85
320 CIRCLE (29, 128), 4, , 1, .85, .25
330 CIRCLE (33, 142), 10, , 2, .70, .95
340 CIRCLE (43, 145), 5, , 2, .47, .70
350 CIRCLE (22, 127), 30, 1, 1, .0, .12
360 DRAW"BM50, 128; G4
370 CIRCLE (48, 129), 4, , 2, .35, .85
380 CIRCLE (47, 151), 30, , 1, .69, .75
390 CIRCLE (35, 186), 4, , .5
400 LINE (213, 185) - (211, 155), PSET
410 LINE (216, 185) - (218, 155), PSET
420 DRAW"BM212, 155; R5
430 DRAW"BM213, 185; R3
440 PAINT (213, 175), 1
450 DRAW"BM211, 155; H5R18G5E5R2U5
L22D5R2
460 CIRCLE (209, 140), 4, , 2, .30, .85
470 CIRCLE (209, 128), 4, , 1, .85, .25
480 CIRCLE (213, 142), 10, , 2, .70, .9
5
490 CIRCLE (223, 145), 5, , 2, .47, .70
500 CIRCLE (202, 127), 30, 1, 1, .0, .1
2
510 DRAW"BM230, 128; G4
520 CIRCLE (228, 129), 4, , 2, .35, .85
530 CIRCLE (227, 150), 30, , 1, .69, .7
5
540 CIRCLE (215, 186), 4, , .5
550 RETURN

```

(Bill Duke, a freshman in high school, has a 64K CoCo with one drive, printer and modem. He mainly works with graphics and his printer.)



TUTORIAL

16K
ECB

Part II —

The Programmable Chip

Interfacing Your Own Circuits

By T. Whit Athey and Susan C. Athey

In the first article of this series (July, Page 138), I described a general-purpose I/O interface for the Color Computer. The interface consists primarily of an 8255A Programmable Peripheral Interface (PPI) chip, installed on a modified Radio Shack printed circuit (PC) board which plugs into the cartridge slot. You can add your own circuits to the board where they can communicate with your CoCo.

In this article I will describe some examples of circuits which can be added to the interface board. By building the interface and connecting it to other devices, you can learn about the way the CoCo works and about digital circuits in general.

The 8255A has three eight-bit I/O ports, designated A, B, and C. In the main circuit I will describe, port A will be used as a data bus, and two lines of port C will be used as control lines, to communicate with and control a General Instruments AY-3-8910 Sound Generator Chip.

The AY-3-8910

The AY-3-8910 is a 40-pin LSI chip. It has 16 internal registers which control the frequency and amplitude of three independent tone generators, a noise generator, an envelope generator, and two 8-bit I/O ports. Almost any kind of sound can be produced under program control. The production of sound is dependent only on the contents of the internal registers, and these will change only when the register contents are overwritten. Continuous microprocessor attention is not required.

The 16 PSG registers are not directly addressable in the way that the four registers of the 8255A are. There is only the one 8-bit data bus over which data and addresses must be transmitted to the AY-3-8910, so the selection of the desired register must be done separately. You can think of the 16 registers of the PSG as being connected to its data bus through a multiplex switch. Only one register at a time can be connected to the data bus (for reads or writes). This process of "connecting" a register to the bus is called "latching."

To latch a sound chip register to the data bus, two operations must be carried out. First, both control lines (from the 8255A port C) must go high (to +5 V). This is a signal to the chip that the byte about to arrive over the data bus is the register number of the register to be latched. Second, the register number must be transmitted through port A, e.g., with a *POKE &HFF40,[reg. no.]*, over the data bus.

Once a register is latched it will remain latched until another register is latched. While it is latched, any write (or read) operation to the 8255A port A will also be a write (or read) operation to the 8255A port A will also be a write to the sound chip register currently latched. The two control lines from port C must be set for a write (C0 = 1, C1 = 0) during the write operation.

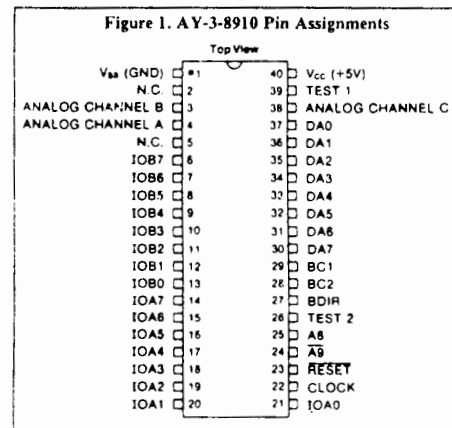
For example, the following sequence of operations would be carried out to write the byte 3B (Hex) to register two (assuming that both A and C ports of the 8255A are already programmed for output).

First, latch register 2:

- POKE &HFF42,3 (set C0 and C1 high)
- POKE &HFF40,2 (send register no. to data bus)
- POKE &HFF42,0 (reset control lines)

Then, write the byte 3B:

- POKE &HFF40,&H3B (put byte on data bus)
- POKE &HFF42,1 (set control lines for write)
- POKE &HFF42,0 (reset control lines)



Pin No.	Label	Function
1	GND	Ground
2	N.C.	(Not connected)
3	AUDIO-A	Audio channel A
4	AUDIO-B	Audio channel B
5	N.C.	(Not connected)
6	IOB7	I/O port B, bit 7
7	IOB6	I/O port B, bit 6
8	IOB5	I/O port B, bit 5
9	IOB4	I/O port B, bit 4
10	IOB3	I/O port B, bit 3
11	IOB2	I/O port B, bit 2
12	IOB1	I/O port B, bit 1
13	IOB0	I/O port B, bit 0
14	IOA7	I/O port A, bit 7
15	IOA6	I/O port A, bit 6
16	IOA5	I/O port A, bit 5
17	IOA4	I/O port A, bit 4
18	IOA3	I/O port A, bit 3
19	IOA2	I/O port A, bit 2
20	IOA1	I/O port A, bit 1
21	IOA0	I/O port A, bit 0
22	CLOCK	Clock reference signal
23	RESET	Logic zero resets registers to "0". Extra "address" or "chip select lines, logic 0 selects A9, logic 1 selects A8. These should be tied to GND and VCC if only one PSG chip is being used.
24	A9	
25	A8	
26	TEST2	A test point — not used here.
27	BDIR	Bus direction control line
28	BC2	Bus control line 2 — not used here
29	BC1	Bus control line 1
30	DA7	Data/address bit 7
31	DA6	Data/address bit 6
32	DA5	Data/address bit 5
33	DA4	Data/address bit 4
34	DA3	Data/address bit 3
35	DA2	Data/address bit 2
36	DA1	Data/address bit 1
37	DA0	Data/address bit 0
38	AUDIO-C	Audio channel C
39	TEST1	A test point — not used here
40	VCC	To +5 volts

These instructions illustrate the simple steps required to latch and write to a register, but unfortunately, they don't work in practice because of a timing problem resulting from the slowness of BASIC. The problem is that the sound chip wants to see the write indication on its control lines for no longer than 10 microseconds, while the BASIC POKE instruction requires about 4500 microseconds (that's still a lot less than a second!). The latch routine above will work okay, but the write routine will have to be done in machine language (using only three simple instructions). This can still be handled from BASIC using a USR subroutine (or EXEC statements for those of you without Extended BASIC). This will be discussed further in the software section.

The AY-3-8910 Programmable Sound Generator Chip

A pin diagram for the programmable sound generator chip (PSG) is shown in Figure 1, and the function of each line is given in Table 1. A functional block diagram is shown in Figure 2. The lines labeled BDIR and BCI are the two control lines which were discussed above. The functions of the chip are determined by these two lines as shown in Table 2.

All functions of the PSG chip are controlled by the computer through a series of writes to the 16 registers (designated R0-R15). Registers R0-R5 are used to select the frequencies (actually the periods) of the three-tone generators. Register R6 selects the frequency of the noise generator (white noise). Register R7 is for mixer control and I/O enable (for the two I/O ports). The amplitudes of the signals generated by the three tone channels are controlled by registers R8-R10 when in the level amplitude mode. Registers R11-R13 are for control of the envelope generator, and the last two registers, R14 and R15, are the two I/O ports. The operation of these 16 registers will now be discussed in more detail.

Tone Generator Control (Registers R0-R5)

The frequency of each square wave generated by each of the three tone generators is controlled by Registers R0-R5. Register R0 and the lower four bits of Register R1 form a 12-bit tone period value (the reciprocal of the frequency). The exact frequency of the tone which is produced depends on the reference clock signal. The period values needed to produce musical notes for two clock frequencies are shown in Table 3. The upper (most significant) four bits of Register R1 are not used. The other two tone generators are controlled in exactly the same way with Registers R2-R5.

Noise Generator Control (Register R6)

The frequency of the noise source is controlled with Register R6. The lower five bits of this register form a five-bit period value. You can try stepping through the range of periods available (&H01 to &H1F) to select the period with the desired effect.

Mixer Control-I/O Enable (Register R7)

Register R7 is a multi-function enable/disable register which controls which of the tone or noise sources is connected to the mixers, and also defines the direction for the bidirectional I/O ports. The control bits and their functions are shown in Table 4.

Amplitude Control (Registers R8-R10)

The amplitudes of the three tone channels are determined by the contents of the lower five bits of registers R10, R11, and R12. The upper three bits are not used. The fifth bit is used to select either fixed level or variable (envelope) amplitudes (0 = fixed, 1 = variable). The lower four bits contain the amplitude value when bit five is zero, and are ignored when bit five is one. There are 16 amplitude levels, 0000-1111 (binary), which can be selected.

When bit five is one, the envelope control is enabled. A

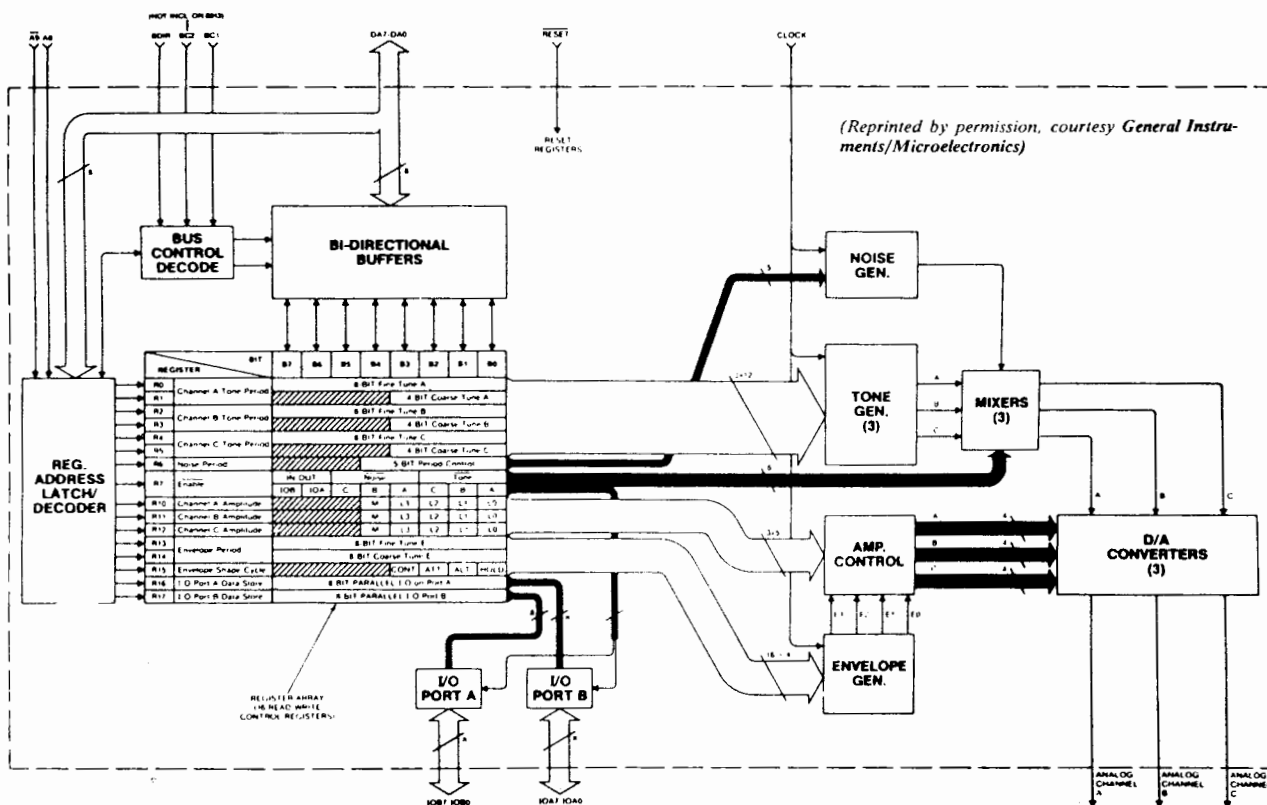


Figure 2. Programmable Sound Generator Block Diagram

description of the envelope generator and its control follows.

Envelope Generator Control (Registers R11-13)

The shape of the envelope is determined by the contents of the lower four bits of register R13. The envelope shape is the time history of the amplitude (amplitude as a function of time). Figure 3 shows the different shapes which are available and the control values which select each shape.

Registers R11-R12 contain the time for one cycle of the envelope. These two registers are used as one 16-bit register, R12 containing the most significant bits and R11 containing the least significant bits. R12 can be considered the course tune value and R11 the fine tune value. The envelope periods available range from a value of 0001 (Hex) to FFFF (Hex), corresponding to .285 milliseconds to 18.7 seconds (assuming a .897 MHz clock).

I/O Ports (Registers R14-R15)

The PSG chip has two 8-bit I/O ports which can be used in a manner similar to the three I/O ports of the 8255A chip. When an 8-bit word is written to register R14, the data also appears on the eight PSG pins for port A. When the PSG ports are in the input mode, any data present on the pins will be present in the corresponding register (R14 or R15). A read of the register will retrieve the data to the computer for processing. The mode of the ports (input or output) is determined by the contents of the two highest bits of R7 as discussed earlier.

Adding The PSG To Your I/O Board

If you have finished your I/O board according to last month's article, the addition of the PSG will be fairly easy. You will need only the PSG chip and a 40-pin socket (see the previous article for socket sources). Unfortunately, General Instruments, the manufacturer of the PSG, and their distributors are not really very interested in the hobbyist business, and this makes the purchase of a PSG chip a little tricky. Arrow Electronics is a General Instruments distributor with many offices around the country, but they usually have a \$50 minimum order. If you have a lot of other things to order anyway, then try Arrow. You might also find it at some Pioneer outlets. If you can't find a source, HIB Associates (3505 Hutch Place, Chevy Chase, MD 20815) has again agreed to fill mail orders (include \$2 shipping and handling on all orders, five percent sales tax on Maryland orders). The PSG is priced at \$9.50, and the 40-pin socket is \$1.

Table 2. PSG Control Line Functions

BC1	BDir	Function
0	0	Inactive
0	1	Write
1	0	Read
1	1	Latch

A suitable location for the PSG is directly across the board from the 8255 (use the same row numbers) on the last black rectangle. In this position, as was the case for the 8255, all of the pins will be accessible without removing the computer cover when the board is plugged in. Bend and solder the corner pins of the socket to the copper pads, and bend and solder two more pins near the center of the socket. Now follow the wiring list in Table 3. Make sure that all pins to be connected via the copper pads are bent and soldered to the pads. When you are finished, recheck your work against the

Table 4. Mixer-I/O Control

Noise Enable Truth Table

R7 Bits			Noise Enabled on Channel		
B5	B4	B3	C	B	A
0	0	0	C	B	A
0	0	1	C	B	-
0	1	0	C	-	A
0	1	1	C	-	-
1	0	0	-	B	A
1	0	1	-	B	-
1	1	0	-	-	A
1	1	1	-	-	-

Tone Enable Truth Table

R7 Bits			Tone Enabled on Channel		
B2	B1	B0	C	B	A
0	0	0	C	B	A
0	0	1	C	B	-
0	1	0	C	-	A
0	1	1	C	-	-
1	0	0	-	B	A
1	0	1	-	B	-
1	1	0	-	-	A
1	1	1	-	-	-

I/O Port Truth Table

R7 Bits		I/O Port Status	
B7	B6	IOB	IOA
0	0	Input	Input
0	1	Input	Output
1	0	Output	Input
1	1	Output	Output

Table 3. Notes, frequencies, and codes

Note	Frequency (1.97 MHz clock) Hz	Frequency (.897 MHz clock) Hz	Hex Code	Note	Frequency (1.97 MHz clock) Hz	Frequency (.897 MHz clock) Hz	Hex Code
C	32.7	16.3	D5D	C	522.7	261.4	0E6
C#	34.6	17.3	C9C	C#	553.8	276.9	0CA
D	36.7	18.4	BE7	D	588.7	294.4	0B1
D#	38.9	19.4	B3C	D#	621.5	310.7	0B4
E	41.2	20.6	A9B	E	658.0	329.0	0BA
F	43.7	21.8	A02	F	699.1	349.6	0A0
F#	46.2	23.6	973	F#	740.8	370.4	097
G	49.0	24.5	8EB	G	782.2	391.1	08E
G#	51.9	25.9	86B	G#	828.6	414.3	087
A	55.0	27.5	7F2	A	880.8	440.4	07F
A#	58.3	29.1	780	A#	932.2	466.1	078
B	61.7	30.9	714	B	989.9	495.0	071
C	65.4	32.7	6AE	C	1045.4	522.7	06B
C#	69.3	34.6	64E	C#	1107.5	553.8	065
D	73.4	36.7	5F4	D	1177.5	588.7	051
D#	77.8	38.9	59E	D#	1242.9	621.5	05A
E	82.4	41.2	54D	E	1316.0	658.0	055
F	87.3	43.7	501	F	1398.3	699.1	050
F#	92.5	46.3	489	F#	1471.9	735.9	04C
G	98.0	49.0	475	G	1575.5	787.8	047
G#	103.9	51.9	435	G#	1669.6	834.8	043
A	110.0	55.0	3F9	A	1747.8	873.9	040
A#	116.5	58.2	3C0	A#	1864.3	932.2	03C
B	123.5	61.7	38A	B	1962.5	981.2	039
C	130.8	65.4	357	C	2110.6	1055.3	035
C#	138.6	69.3	327	C#	2237.2	1118.6	032
D	146.8	73.4	2FA	D	2330.4	1165.2	030
D#	155.6	77.8	2CF	D#	2485.8	1242.9	02D
E	164.7	82.4	2A7	E	2663.4	1331.7	02A
F	174.5	87.3	281	F	2796.5	1398.3	028
F#	184.9	92.4	25D	F#	2943.7	1471.9	026
G	195.9	97.9	23B	G	3107.2	1553.6	024
G#	207.5	103.8	21B	G#	3290.0	1645.0	022
A	220.2	110.1	1FC	A	3495.6	1747.8	020
A#	233.0	116.5	1E0	A#	3728.7	1864.3	01E
B	246.9	123.5	1C5	B	3995.0	1997.5	01C
C	261.4	130.7	1AC	C	4143.0	2071.5	01B
C#	276.9	138.4	194	C#	4474.4	2237.2	019
D	293.6	146.8	17D	D	4660.9	2330.5	018
D#	310.7	155.4	168	D#	5084.6	2542.3	016
E	330.0	165.0	153	E#	5326.7	2663.3	015
F	349.6	174.8	140	F	5593.0	2796.5	014
F#	370.4	185.2	12E	F#	5887.4	2943.7	013
G	392.5	196.2	11D	G	6214.5	3107.2	012
G#	415.8	207.9	10D	G#	6580.0	3290.0	011
A	440.4	220.2	0FE	A	6991.3	3495.6	010
A#	466.1	233.0	0F0	A#	7457.4	3728.7	00F
B	495.0	247.5	0E2	B	7990.1	3995.0	00E

wiring list until you are sure you have it right (improper connections may damage the chip or the computer). Now you can test the circuit using the test program shown in the next section.

In this design, the CoCo's clock signal (.897 MHz) on pin 6 of the cartridge connector is used as the clock signal for the PSG. Using a 1.79 MHz pulse would be better, allowing another octave of higher frequency tones. An external clock circuit may be used for this purpose if necessary. I will be glad to send you a simple clock circuit diagram if you include a self-addressed, stamped envelope with your request (6913 Breezewood Terrace, Rockville, MD 20852).

Table 5. Wiring List.

Connect the copper strip running down underneath the new 40-pin socket to the ground bus. The two nearest strips parallel to the ground bus should be connected to Vcc (+5 V), forming a Vcc bus (one of these may already be connected to Vcc. Connect a 1000 ohm resistor between pin 38 of the AY-3-8910 and the ground bus. Connect a 0.1µF capacitor between Vcc and ground near pin 40 of the AY-3-8910 (or between its pin 40 and ground).

From	To	Signal
Vcc bus	AY-3-8910-25	Vcc
Vcc bus	-28	Vcc
Vcc bus	-40	Vcc
GND bus	-1	Ground
GND bus	-24	Ground
8255-14	-27	Port C, line 0 (control)
-15	-29	Port C, line 1 (control)
-4	-37	DO
-3	-36	DI
-2	-35	D2 (Data lines
-1	-34	D3 from
-40	-33	D4 Port A)
-39	-32	D5
-38	-31	D6
-37	-30	D7
AY-3-8910-3	-4	Audio
-3	-38	Audio
-3	CC Cartr.-35	Audio
CC Cartr.-6	AY-3-8910-22	Clock (.89 MHz)
-5	-23	Reset

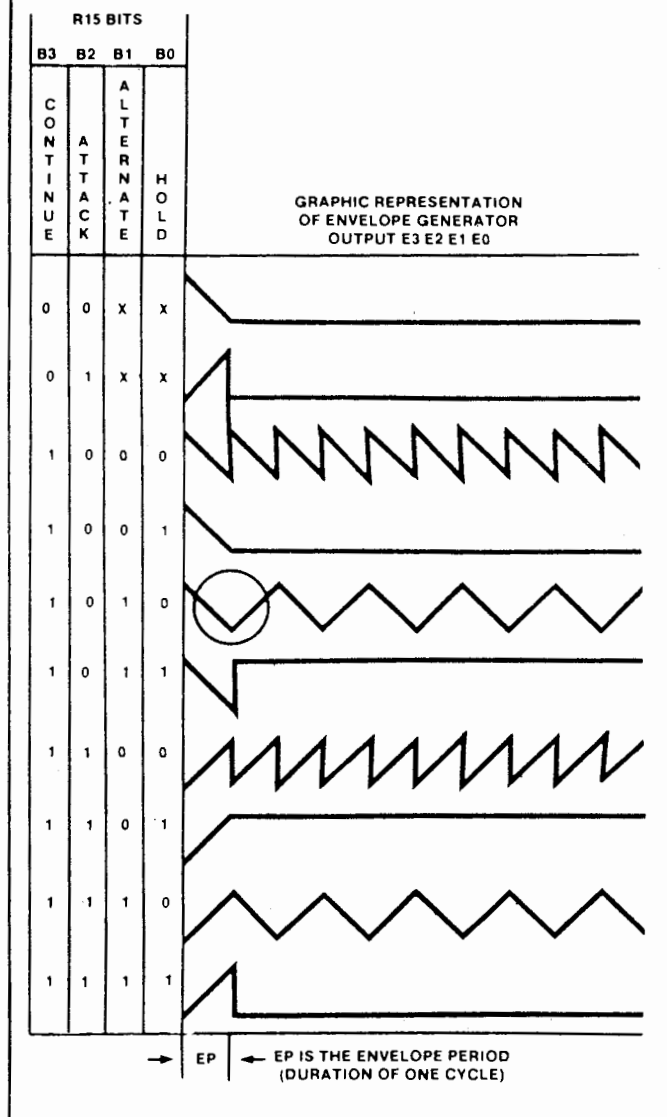
You may also want to connect AY-3-8910-3 (audio out) to an audio jack for playback through a separate sound system.

Software

The most important routines for controlling the PSG are the three which latch, write, and read. These must be written in machine language for proper timing and for maximum speed. Listing 1 shows these routines along with a short test program. It is assumed that the control lines are connected to PPI C-0 and C-1, and that the eight data lines are connected to port B.

The machine language instructions are shown in annotated assembly language in Listing 2. The statement numbers of the corresponding data statements are given in parentheses.

Figure 3. Envelope Shape/Cycle Control



This example of assembly language programming is so simple that no previous experience should be necessary to understand the way it works. Only four instructions are used, although one of them (STA) is used in two different addressing modes. In Listing 2 the operation code is in the left column and the number or address it operates on is in the right column. The dollar sign means that the number following is in Hex, and the "#" sign means that the number following should be treated as a number rather than as an address. *LDA #80* means that the CPU A register should be loaded with the Hex number 80. *STA \$FF43* means that the contents of the CPU A register should be transferred to address FF43 (which, in our case is the control register of the PPI). *LDA \$FF40* means that the contents of address FF41 should be transferred to the A register. *CLRA* simply clears the CPU A register, and *RTS* returns control back to the program which is called the routine.

Listing 1 has a few statements which require Extended BASIC, but it may be modified to get around this problem. If you don't have Extended BASIC, simply make the following changes:

220 EXEC PA

260 EXEC PA+&H17

290 EXEC PA+&H2A

Delete Lines 8910, 8920, 8930

Radio Shack doesn't advertise the fact the EXEC works in this manner, but it does.

To test the PSG, RUN the test program. In response to the prompt, OP,REG,DATA:, enter the operation (1 for write, 2 for read), the PSG register number, and the data to be written (enter zero here when reading). To get a tone from audio channel A, we must write and enable code to register 7 (see Table 4), a frequency value to register 0 (and sometimes to register 1 as well), and an amplitude code to register 8.

For example, try the following:

OP,REG,DATA: 1,7,&H3E

OP,REG,DATA: 1,0,&HD6

OP,REG,DATA: 1,8,&H0F

The note C should sound. Now test the read feature by reading the contents of register 7 (which we just set to 3E, or 62 in decimal, in the above procedure):

OP,REG,DATA: 2,7,0

The computer should respond by printing 62 on the screen. Test out the other features of the PSG in a similar manner.

Many sophisticated sound effects are possible with the PSG, and making the sounds does not tie up the CPU full time. Listing 3 shows a few of the examples which my 12-year-old daughter wrote for a science fair project. These statements should be added to the test program (Listing 1), replacing Lines 100-140.

As a different example of the possibilities with the PSG, I have added a second PSG to my board (a third control line to PSG pin 25 serves as a chip selector). I then connected the lines of the two additional I/O ports on each chip to switches under the keys of an old three-octave chord organ keyboard. With a few more subroutines, I now have a six-voice (real-time) electronic organ. My next step will be (just through additional software) to add synthesizer effects. As you can see, with the PSG you are only limited by your imagination.

The sound generator chip is only one example of the kinds of circuits which you can add to your interface board. A great security system could be built with each on/off or indicator switch from critical locations around your house wired in (through a buffer chip like the 74LS241) to the PPI I/O port lines.

The important thing is to build something. It's the best way to start learning about the wonderful world of digital circuits.

Listing 1:

```

10 CLEAR1000,&H2FFF
20 PA=&H3000
30 'SELECT CARTRIDGE SOUND
40 AUDIOON
50 POKE&HFF01,&H34
60 POKE&HFF03,&H3C
70 GOSUB8900
90 '
100 'TEST PROGRAM
105 '
110 INPUT"OP,REG,,DATA: ";OP,R,S
120 GOSUB210
130 IFOP=1 THENGOSUB240ELSEGOSUB2
90
140 GOTO110
    
```

310..... 43
END 136

```

200 'LATCH
210 POKEPA+&H0B,R
220 X=USR1(0)
230 RETURN
240 'WRITE
250 POKEPA+&H18,S
260 X=USR2(0)
270 RETURN
280 'READ
290 X=USR3(0)
300 PRINT"CONTENTS=";PEEK(PA+&H2
9)
310 RETURN
8900 'READ AND STORE MACHINE LAN
GUAGE ROUTINES
8910 DEFUSR1=PA
8920 DEFUSR2=PA+&H17
8930 DEFUSR3=PA+&H2A
8940 'LOAD USR1, USR2, & USR3
8950 FORI=PA TOPA+&H41
8960 READY:POKEI,Y
8970 NEXTI
8980 RETURN
9000 'REGISTER LATCH ROUTINE
9010 DATA&H86,&H80
9020 DATA&H87,&HFF,&H43
9030 DATA&H86,&H03
9040 DATA&H87,&HFF,&H42
9050 DATA&H86,0
9060 DATA&H87,&HFF,&H40
9070 DATA&H4F
9080 DATA&H87,&HFF,&H42
9090 DATA&H87,&HFF,&H40
9100 DATA&H39
9110 'WRITE DATA ROUTINE
9120 DATA&H86,0
9130 DATA&H87,&HFF,&H40
9140 DATA&H86,&H01
9150 DATA&H87,&HFF,&H42
9160 DATA&H4F
9170 DATA&H87,&HFF,&H42
9180 DATA&H87,&HFF,&H40
9190 DATA&H39
9195 DATA0
9200 'READ ROUTINE
9210 DATA&H86,&H82
9220 DATA&H87,&HFF,&H43
9230 DATA&H86,&H02
9240 DATA&H87,&HFF,&H42
9250 DATA&H86,&HFF,&H40
9260 DATA&H87,&H30,&H29
9270 DATA&H4F
9280 DATA&H87,&HFF,&H42
9290 DATA&H87,&HFF,&H40
9300 DATA&H39
    
```

Listing 2:

```

09000 *REGISTER LATCH ROUTINE
09010 LDA #000 SET PPI REG. FOR
09020 STA $FF43 B/C PORT OUTPUT.
    
```

```

09030 LDA  ##03 SET BOTH CONTROL LINES
09040 STA  ##FF42 (C PORT) HIGH.
09050 LDA  0 OPERAND GETS POKED REG. NO.
09060 STA  ##FF41 WRITE REG. NO. TO PORT B.
09070 CLRA ZERO A REGISTER.
09080 STA  ##FF42 CLEAR CONTROL LINES.
09090 STA  ##FF41 CLEAR DATA LINES.
09100 RTS RETURN.
09110 *WRITE DATA ROUTINE
09120 LDA  0 OPERAND GETS POKED DATA.
09130 STA  ##FF41 WRITE TO DATA LINES.
09140 LDA  ##01 SET CONTROL LINES FOR
09150 STA  ##FF42 WRITE DATA.
09160 CLRA ZERO A REGISTER.
09170 STA  ##FF42 CLEAR CONTROL LINES.
09180 STA  ##FF41 CLEAR DATA LINES.
09190 RTS RETURN.
09200 *READ DATA ROUTINE
09210 LDA  ##02 SET PPI CONTROL REG. FOR
09220 STA  ##FF43 B PORT INPUT.
09230 LDA  ##02 SET PSG CONTROL LINES
09240 STA  ##FF42 FOR READ.
09250 LDA  ##FF41 READ DATA ON DATA LINES.
09260 STA  (PA+##3F) SAVE IN SPARE LOCATION.
09270 CLRA CLEAR A REGISTER.
09280 STA  ##FF42 CLEAR CONTROL LINES.
09290 STA  ##FF41 CLEAR DATA LINES.

```

1590 202
1840 159
END 140

Listing 3:

```

100 PRINT "SOUND EFFECTS BY SUSAN
ATHEY"
105 PRINT
110 PRINT "TYPE THE LETTER OF YOU
R CHOICE."
115 PRINT
120 PRINT "(A) SWEEP AMPLITUDE"
125 PRINT "(B) SWEEP TONE FREQUEN
CY"
130 PRINT "(C) GUN SHOTS + EXPLOS
ION"
135 PRINT "(D) EUROPEAN SIREN"
140 A$=INKEY$: IFA$="" THEN 140
145 IFA$="A" THEN GOSUB 1510
150 IFA$="B" THEN GOSUB 1610
155 IFA$="C" THEN GOSUB 1710
160 IFA$="D" THEN GOSUB 1910
165 CLS: GOTO 100
170 '
1500 ' SWEEP AMPLITUDE
1501 '
1510 R=0: S=&HD6: GOSUB 210: GOSUB 25
0
1520 R=7: S=&076: GOSUB 210: GOSUB 25
0
1530 FORA=0 TO &H0F
1540 R=8: S=A: GOSUB 210: GOSUB 250
1550 NEXTA
1560 FORA=&H0F TO 0 STEP -1
1570 R=8: S=A: GOSUB 210: GOSUB 250
1580 NEXTA
1590 FORDLY=1 TO 200: NEXTDLY
1595 RETURN
1600 '
1601 ' SWEEP FREQUENCY
1602 '
1610 R=7: S=&076: GOSUB 210: GOSUB 25
0
1620 R=8: S=&H0F: GOSUB 210: GOSUB 25
0
1630 FORI=1 TO 3
1640 FORN=&HD6 TO &H6B STEP -1
1650 R=0: S=N: GOSUB 210: GOSUB 250
1660 NEXTN
1670 NEXTI
1680 RETURN
1700 '
1701 ' GUNSHOTS AND EXPLOSION
1702 '
1710 V=16: G=15: F=0
1720 FOR Y=1 TO 5
1730 R=6: S=G: GOSUB 210: GOSUB 250
1740 R=7: S=7: GOSUB 210: GOSUB 250
1750 R=8: S=16: GOSUB 210: GOSUB 250
1760 R=9: S=16: GOSUB 210: GOSUB 250
1770 R=10: S=16: GOSUB 210: GOSUB 250
1780 R=12: S=V: GOSUB 210: GOSUB 250
1790 R=13: S=0: GOSUB 210: GOSUB 250
1800 FORDLY=1 TO 10: NEXTDLY
1810 NEXT Y
1820 IFF=5 THEN RETURN
1830 FORDLY=1 TO 200: NEXTDLY
1840 V=56: F=5: G=0: GOTO 1720
1900 '
1901 ' SIREN
1902 '
1910 FORT=1 TO 10
1920 R=7: S=&076: GOSUB 210: GOSUB 25
0
1930 R=8: S=&H0C: GOSUB 210: GOSUB 25
0
1940 R=9: S=&H0B: GOSUB 210: GOSUB 25
0
1950 R=0: S=&H6B: GOSUB 210: GOSUB 25
0
1960 R=2: S=&H69: GOSUB 210: GOSUB 25
0
1970 FORDLY=1 TO 100: NEXTDLY
1980 R=0: S=&H47: GOSUB 210: GOSUB 25
0
1990 R=2: S=&H46: GOSUB 210: GOSUB 25
0
2000 FORDLY=1 TO 100: NEXTDLY
2010 NEXT T
2020 RETURN

```

GAME

16K
ECB



BUT WHO WAS COLONEL POTTER'S HORSE?

Milt Tanzer

For several years (I won't say how many because that is one of the questions) the TV series *M*A*S*H* stayed at the top of the charts as one of the most watched programs on TV. It even topped the Super Bowl game on viewing audience.

One evening, after the series had ended, our family sat reminiscing about the program. "Remember how Hawkeye and B.J. used to pick on Frank Burns?" someone said. "Sure, it started with the pilot episode when they put him in a full body cast," someone else answered. "That wasn't B.J. He wasn't on the show yet. That was Hawkeye and Trapper." Before we knew it, we were totally involved in testing each other's memory about the many characters and happenings at the *M*A*S*H* unit. Over the next few months we made a list of 200 trivia questions and answers about the series, spanning the entire 11 years the show ran. (Oh, I just gave you one answer.)

I decided to write a program for the CoCo that would give all *M*A*S*H* fans the opportunity to share the fond memories of everyone's favorite TV series.

Since I could not find a way to have the computer accept a correct answer that varies slightly from the data line answer, I suggested to the player in the instructions not to take a missed question too seriously . . . after all, it's only a game.

The listing which follows is the 16K version of the game and can be found on this month's RAINBOW ON TAPE. Additionally, a longer (32K) version with many more *M*A*S*H* trivia questions is also on RAINBOW ON TAPE.

135.....	152
245.....	2
370.....	47
465.....	84
540.....	70
END	43

The listing:

```

1 'BY MILT TANZER
2 ' 2921 NE 46TH ST.
3 ' LIGHTHOUSE POINT, FLA. 33064
5 CLEAR 1000
10 CLS:PMODE 3,1
15 PCLS:SCREEN1,1
20 DRAW"C3;BM44,156;D18;R12;U18"
25 CIRCLE(38,168),10,4,1,.25,.85
30 CIRCLE(32,164),16,4,1,.12,.90
35 LINE(44,156)-(120,156),PSET
40 LINE(56,171)-(120,160),PSET
45 DRAW"BM48,137;D18;R3;U18;L3"
50 DRAW"BM1,140;R100;U1;L100"
55 DRAW"BM12,182;F4;R60
60 CIRCLE(122,158),9,4
65 DRAW"BM37,184;E9;R8;F9
70 DRAW"BM56,171;E14;F10;E10;F7;
E7;F5;E5"
    
```

```

75 DRAW"BM20,181;F2;R52"
80 PAINT(47,165),3,3
85 FORT=1T01000:NEXT
90 DRAW "BM16,10;D40;R8;U28;F12;
E12;D28;R8;U40;L8;G12;H12;L8"
95 PLAY"D;L7;C;L4;D;C;D;C;P4"
100 LINE(92,52)-(76,92),PSET
105 LINE(76,92)-(84,92),PSET
110 LINE(84,92)-(88,84),PSET
115 LINE(88,84)-(104,84),PSET
120 LINE(104,84)-(108,92),PSET
125 LINE(108,92)-(116,92),PSET
130 LINE(116,92)-(100,52),PSET
135 LINE(100,52)-(92,52),PSET
140 LINE(96,64)-(88,76),PSET
145 LINE(88,76)-(104,76),PSET
150 LINE(104,76)-(96,64),PSET
155 PLAY"L6;C;L4;D;L6;C;L4;D;C;D;
C;P4"
160 DRAW"BM144,96;G8;D8;F8;R26;D
8;L32;F8;R24;E8;U8;H8;L24;U8;R32;
H8;L24"
165 PLAY"L4;C;O2;A;L4;O3;C;D;F"
170 DRAW"BM196,140;D40;R8;U16;R2
4;D16;R8;U40;L8;D16;L24;U16;L8"
175 PLAY"L4;GFDC;L2;D"
180 PAINT(20,12),4,3:PAINT(96,54
),4,3
185 PAINT(148,98),4,3:PAINT(200,
144),4,3:FOR X=1T0500:NEXT
190 COLOR4,3
195 LINE(64,52)-(76,64),PSET
200 LINE(76,52)-(64,64),PSET
205 LINE(64,58)-(76,58),PSET:FOR
X=1T0500:NEXT
210 LINE(120,88)-(132,100),PSET
215 LINE(132,88)-(120,100),PSET
220 LINE(120,94)-(132,94),PSET:FOR
OR X=1T0500:NEXT
225 LINE(180,132)-(192,144),PSET
230 LINE(192,132)-(180,144),PSET
235 LINE(180,138)-(192,138),PSET
:FORX=1T0500:NEXT
240 PLAY"L4;C;O2;A;O3;CDFGFDC;L1
;D"
245 DRAW"BM180,16;D16;L16;D20;R1
6;D16;R20;U16;R16;U20;L16;U16;L2
0"
250 PAINT(188,24),4,4
255 FOR T=1T01000:NEXT
260 PRINT@71,"WELCOME TO..."
265 PRINT@138,"M*A*S*H"
270 PRINT@202,"TRIVIA"
275 PRINT@385,"DO YOU WANT INSTR
UCTIONS?(Y/N)":INPUT I$
280 IF I$="Y" THEN GOTO450
285 CLS:PRINT@165,"PLEASE BE PAT
IENT"
290 PRINT@322,"I'M THINKING UP Q
UESTIONS"
    
```

```

295 PLAY"D;L6;C;L4;D;C;D;C;P4"
300 DIM Q$(20),A$(20)
305 FORX=1TO20
310 READ Q$(X),A$(X)
315 NEXTX
320 CLS:C=0
325 FOR Y=1TO20
330 CLS:PRINT@64," * * * * *
* * * * * * * * *
335 PRINTQ$(Y):INPUT Z$
340 IFZ$=A$(Y) THEN GOTO375
345 SOUND100,10
350 IF Z$(>)A$(Y) THEN PRINT@266,
"MY ANSWER IS: "
355 PRINT@320,A$(Y)
360 FOR T=1TO2500:NEXT
365 NEXTY
370 GOTO415
375 R=RND(5):PRINT""
380 FORX=1TO5:SOUND50,1:SOUND150
,1:NEXT
385 IFR=1 THEN PRINT"THAT'S RIGH
T..YOU'RE PRETTY GOOD"
390 IFR=2 THEN PRINT"HEY, YOU'RE
OK"
395 IF R=3 THEN PRINT"TERRIFIC..
.RIGHT AGAIN"
400 IFR=4 THENPRINT"YOU DID IT A
GAIN
405 IFR=5 THENPRINT"RIGHT...YOU
SURE KNOW MASH"
410 C=C+1:FOR T=1TO1500:NEXT:GOT
O365
415 CLS:PRINT:PRINT"THAT'S 20 QU
ESTIONS"
420 PLAY"D;L6;C;L4;DCDC;P4"
425 PRINT:PRINT"YOU GOT ";C;"RIG
HT OUT OF 25"
430 RESTORE:PRINT"CARE TO TRY MO
RE?(Y/N)":INPUT C$
435 IFC$="Y"THEN GOTO320
440 PRINT:PRINT"THANKS FOR PLAYI
NG."
445 PLAY"L4;C;O2;A;L4;O3;CDF;L3;
GFDC;L2;D":END
450 ' INSTRUCTIONS
455 CLS:PRINT:PRINT:PRINT"YOU WI
LL BE ASKED 20 TRIVIA QUES
TIONS ABOUT THE TV SERIES
M*A*S*H
460 FOR T=1TO2500:NEXT:CLS
465 PRINT:PRINT"TYPE IN YOUR ANS
WER AND <ENTER>"
470 PRINT:PRINT"HINT: EXCEPT WHE
RE ASKED TO GIVE FULL NAMES
USE ONLY THE CHARACTER'S NICK
NAME (RADAR, HAWKEYE,ETC.)
475 PRINT"IF THE CHARACTER HAS N
O NICKNAME USE THEIR LAST NAME O
NLY (POTTER, MULCAHY,ETC.)

```

```

480 PRINT@418,"PRESS <ENTER> TO
CONTINUE":INPUT C
485 CLS
490 PRINT"AT TIMES, THE COMPUTER
MAY TELL YOU YOUR ANSWER IS NOT
CORRECT JUST BECAUSE YOUR WORD
ED IT DIFFERENTLY.
495 PRINT"PLEASE DON'T TAKE A MI
SS TOO SERIOUSLY":PRINT
500 PRINT"AFTER ALL...IT'S ONLY
A GAME!!"
505 PRINT:PRINT"...SO IF YOU'RE
READY TO RELIVE YOUR MEMOR
IES OF"
510 PRINT@362,"M*A*S*H
515 PRINT@418,"PRESS<ENTER> TO S
TART":INPUT C
520 CLS:GOTO285
525 DATA H.Q.PHONE OPERATOR,SPAR
KY
530 DATA POTTER'S HOBBY,PAINTING
535 DATAWHO PLAYED TRAPPER,WAYNE
ROGERS
540 DATAMULCAHY'S SISTER'S OCCUP
ATION,NUN
545 DATANEIGHBORHOOD BAR, ROSIE'
S
550 DATAWHAT DOES MASH STAND FOR
,MOBILE ARMY SURGICAL HOSPITAL
555 DATANAME AND RANK OF INTELLI
GENCE OFFICER,COLONEL FLAGG
560 DATAWHO QUIT DRINKING WHEN H
E SAW HIS BAR BILL,HAWKEYE
565 DATARADAR'S GUINEA PIG,DAISY
570 DATAONLY ACTOR TO STAR BOTH
IN THE TV SERIES AND THE MOVIE,
GARY BURGHOFF
575 DATANAME AND RANK OF THE SHR
INK,MAJOR SIDNEY FREEDMAN
580 DATAPOTTER'S HOMETOWN AND ST
ATE,"HANNIBAL,MISSOURI"
585 DATAWHAT DID B.J. DO THAT BU
GGED HAWKEYE,COMBED HIS MUSTA
CHE
590 DATABURNS' FAVORITE DRINK,SH
IRLEY TEMPLE
595 DATANAME AND RANK OF NURSE W
HO WAS PRIZE IN A RAFFLE,LT. DI
SH
600 DATAWHO KEPT BOMBING THE MAS
H UNIT,FIVE O'CLOCK CHARLIE
605 DATAHOW DID BURNS PASS HIS M
EDICAL EXAMS,BOUGHT THE ANSWERS
610 DATAKLINGER'S FAVORITE TEAM,
TOLEDO MUD HENS
615 DATAWHERE DID BLAKE'S PLANE
CRASH,SEA OF JAPAN
620 DATANAME OF BARBECUE RIB PLA
CE IN CHICAGO, ADAM'S RIB

```

TUTORIAL

Everything

You Always
Wanted To
Know About
The CoCo But
Radio Shack
Didn't Tell
You

By Andy Kluck

In response to the introductory page of the first section of Radio Shack's *Getting Started with Color BASIC*, which invites the reader to "prove us wrong (if you can)," I have made an attempt to compile a list of some of the major errors and omissions in the Color Computer's documentation. In this article and in the following installments, I will also outline some of the techniques that can be helpful in using the CoCo that were not mentioned in the manuals. One of the examples assumes that the *PLUCK* function has been previously defined with the Extended BASIC *DEF FN* statement:

```
100 DEF FN PL(X)=PEEK(X)*%H100+ PEEK(X+1)
```

This function returns the value of the two-byte integer at the address specified by the argument, and is useful for examining Color BASIC's pointers. Hexidecimal numbers in this text are identified by the dollar sign; this should be replaced by *&H* in Extended BASIC expressions. I will refer to Radio Shack's *Getting Started with Color BASIC*, *Going Ahead with Extended Color BASIC*, and *Color Computer Disk System Owners Manual and Programming Guide* as the Cbasic, Xbasic, and Dbasic manuals respectively. Some of this information has been previously published, and is included here in the interest of completeness. If you find any mistakes, please be advised that they are intentional; I tried to include something for everyone, and some people are always looking for mistakes. (I'll bet you had fun with Radio Shack's manuals.)

Release Numbers — When Extended or Disk BASIC is activated, the sign-on message gives the revision number of the highest level ROM in the system, ignoring the revision numbers of the other ROM or ROMs. These statements may be used to determine the hidden revision numbers of the Color BASIC and Extended ROMs:

```
PRINT PEEK(41301) 48 ' Revision of Color BASIC
PRINT PEEK(33023) 48 ' Revision of Extended BASIC,
```

if applicable.

General Information — Color BASIC only accepts line numbers from 0 to 63999. Whenever program lines are added, edited, renumbered, or deleted, a *CLEAR* is executed. A question mark may be used as an abbreviation for *PRINT*, and a single quote (shifted 7) may be substituted for *REM*.

Variables and Spacing — According to the Cbasic manual, variable names may be any combination of one or two letters. Actually, the second character may be a letter or a digit, and they may be followed by as many letters and digits as you want; however, only the first two are significant, so *+ing* uses six bytes of string space, but *50 B\$="string"* uses no string space because the BASIC only needs to remember where the string is located in the program text. There also must be enough string space for temporary strings that are formed while expressions are evaluated. Usually it is best to overestimate the amount of string space by a few hundred bytes unless the program and variables use up almost all the RAM, since *INPUTed* strings may be up to 249 bytes long. To allocate half of the available memory to strings:

```
CLEAR 0: CLEAR MEM/2
```

It is often helpful to know how much string space is left unused. In Level II BASIC, the *FRE* function, when used with a string argument, causes the free and "in use" string space to be separated and returns the number of free string bytes. Color BASIC doesn't have this function, but it can be duplicated by using the Color Computer's "garbage collection" routine and then taking the difference between the bottom of used strings and bottom of string space pointers:

```
EXEC 46481: FRE= FNPL(35)- FNPL(33)
```

gives the amount of free string space. According to the Color BASIC manual, "Without *CLEAR*, the Computer reserves 200 characters." Actually, a *CLEAR 200* is done only when BASIC is first entered. The number of reserved string bytes is not affected by *LOAD* or *RUN* and is inherited from the last program; therefore, no program should assume that 200 bytes are reserved, since the last program run may have reserved 0 or 10000. For this reason, all substantial programs should use *CLEAR* to reserve string memory. Statements of the form: *CLEAR 200,23999* allocate the first number of bytes for string space and set the end of the string pool to the address of the second number minus one. Since the end of the string pool plus one is the highest RAM area used by BASIC, the area starting with the given address plus one, 24000 in this case, is made available for the user's machine language routines. Note that BASIC positions the stack to build down from the bottom of the string pool, so in this example the stack will be in the area just below $24000-200 = 23800$.

PCLEAR — According to the Extended BASIC manual, a *PCLEAR 4* is done automatically, and *PCLEAR* is necessary only "when you want to reserve a different number of pages." In fact, the number of *PCLEARed* pages is also inherited from the last program run, so this number may be anything from one to eight when a new program is loaded, and no program should make assumptions about this number. The manual also says that *PCLEAR* should be the first or second statement in the program, right after *CLEAR*. This advice could cause problems in many cases because the *PMODE* parameters are also unaffected by *LOAD* operations. For example, if the last program used

PMODE 4,5 and the new program tries to *PCLEAR* any fewer than eight pages without first setting *PMODE* to a reasonable value, an FC Error will occur. Furthermore, whenever *PCLEAR* is used with a different number of pages than the last *PCLEAR*, the BASIC program is moved up or down in memory according to the new number of pages. This, in itself, wouldn't cause any problems, except for the now infamous *PCLEAR* bug in the Xbasic 1.0 ROM —after *PCLEAR* copies the program to its new position, it doesn't set the interpret pointer at \$A6 to the new copy. This can have several results. Sometimes an unexplained Syntax Error occurs on the line with *PCLEAR*. Often the program runs normally until the *PCLS* statement is present, which erases the old copy of the program, since it is now in the area of the graphics screen, and causes either an error or a forced *END*. In rare cases, *PCLEAR* may result in a jump to another part of the program. Usually, when *PCLEAR* causes an error, the program will work if it is *RUN* a second time, since it has already been moved to the correct address. To prevent this problem in the first place, two steps have been suggested. First, if *PCLEAR* reduces the number of graphics pages, it should be at the end of the program, and if it increases the number of pages, it should be at the beginning. This prevents the immediate error that occurs when the *PCLEAR* statement is overwritten by another part of the program. Second, to set the interpret pointer to its correct position after *PCLEAR*, use a *GOTO* statement that references a line number less than the current line. To prevent as many errors as possible, I recommend using something like:

```
For PCLEAR 1:
10 GOTO 63990
20 CLEAR 500
30 REM PROGRAM STARTS HERE
63980 END
63990 PMODE 0,1:PCLEAR 1:GOTO 20
```

```
For PCLEAR 2 through 8:
10 GOTO 63990
20 GOTO 40
30 CLEAR 500: PCLEAR 5: GOTO 20
40 REM PROGRAM STARTS HERE
63980 END
63990 PMODE 0,1:PCLEAR 1:GOTO 30
```

Here the *PMODE* statement in Line 63990 prevents a possible FC Error in the *PCLEAR 1*, which, in turn, prevents an error in the *CLEAR 500*, which might occur, for example, if the last program used *PCLEAR 8* and there is not enough free memory for *PCLEAR 8*, and 500 bytes for strings. The extra *GOTOs* force BASIC to recover from the bug. The *CLEAR* statements and the *PCLEAR* in Line 30 of the second example should be adjusted according to the needs of your program. A shorter version, for example: *10 PMODE 0,1: CLEAR 0: PCLEAR 5: CLEAR 500* would be sufficient for use with Xbasic 1.1, but since many users are stuck with 1.0, programs to be distributed to others should allow for the bug. Note that *PCLEAR* also does an implied *CLEAR*, erasing variables and defined functions.

PCLEAR 0 — They said it couldn't be done, and they were right — Various methods have been suggested for effecting a *PCLEAR 0*. An often published example is:

```
POKE 25,6:NEW
```

There are two problems with this. First, it moves the

BASIC program to \$601, which is only correct for plain Extended BASIC. If this is used in Disk BASIC, it jumbles system pointers and variables with the likely result that when you try to load a BASIC program, part of it will be written out on the disk in place of the File Allocation Table. (I hope you made backups.) Second, BASIC will give an error if you try to execute a *NEW* or *RUN* without a zero in the byte before the program. If Xbasic has just been started, there is a zero in this location, but use of graphics page 1 may change this. To fix these problems, try this revised *PCLEAR 0*:

```
POKE 25,PEEK(&HBC):POKE PEEK(&HBC)*256,0:
NEW
```

Address \$BC contains the high byte of the start address of graphics page one, which is 6 for non-Disk Extended BASIC and varies with *FILES* in Disk BASIC. But this is still not a real *PCLEAR 0*; I would call it a *PNEW 0*. To *PCLEAR 0* from inside a program, part of the *PCLEAR* routine can be used to do the necessary moving of the program:

```
10 GOTO 63950
20 CLEAR 200 ' or whatever
30 REM PROGRAM STARTS HERE
63940 END
63950 POKE &H3C0,&H5F:POKE &H3C1,&H5C
' CLR B, INCB- $01 in B, Clear Carry
63960 POKE &H3C2,&H96:POKE &H3C3,&HBC
' LDA $BC
63970 POKE &H3C4,&H1F:POKE &H3C5,&H02
' TFR D,Y
63980 POKE &H3C6,&H7E:POKE &H3C7,&H96:
POKE &H3C8,&HA3 ' JMP $96A3
63990 EXEC &H3C0:GOTO 20 ' PCLEAR 0
```

This works with Xbasic 1.0 and 1.1. Of course, since any *PCLEAR 0* places the BASIC program where the graphics screens are supposed to be, any use of graphics statements afterwards should be avoided unless special arrangements have been made. For example, the addition of these lines:

```
30 POKE &HBA,&HE6:POKE &HB7,&HFE
40 POKE &HB9,&H20:POKE &HB6,3
```

will direct the action of any graphics statements to the area \$E600 through \$FDFE. This prevents graphics statements from causing problems; and, in the 64K BASIC in RAM mode, it allows normal use of one *PMODE 3* or *4* screen in otherwise unused RAM. For example,

```
50 PCLS 1:SCREEN 1,0:CIRCLE (128,96),90,4
60 GOTO 60
```

draws a big *PMODE 3* circle using "free" memory with *PCLEAR 0* still in effect if the SAM has already been set to the 64K RAM mode and BASIC has been copied into the upper page of RAM with a *MOVEROM* program. The *PCLEAR 0* effectively disables *PMODE*, but *PMODE 3* or *4* may still be selected by *POKEing* the 3 or 4 into location &HB6; *PMODEs 0* through *2* require some additional *POKEs* to set up correctly. Because *PCLEAR* is disabled by Line 30, the only way to bring the system back to normal is something like:

```
POKE &HBA, PEEK(&HBC):POKE &HB7, PEEK
(&HBC)+6:PCLEAR 1:PMODE 0,1
```

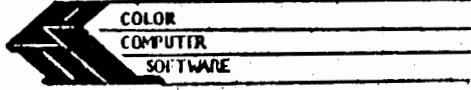
PCLEAR 0 graphics should not be used if Extended BASIC has been relocated for extra free memory or any

RAM in the range \$F600-\$FDFF is in use. Finally, *FILES* should be avoided while this is in effect.

FILES — According to the Disk BASIC manual, the statement *FILES 1,400* reserves space for 1 file and 400 bytes of buffer space for it. Actually, the first number specifies only the number of files that can be accessed by *OPEN*; one extra file control block is reserved for use by *LOAD(M)*, *SAVE(M)*, *MERGE*, and *COPY*. The second specifies the number of bytes to be reserved for random file buffers. To prevent an OB Error, this number must be \geq the sum of the record lengths of random (direct) files to be open at the same time. Since this buffer space is not used by files opened for sequential access, the program on Page 55 demonstrating the necessity of *FILES 1,400* would work just as well without it. A more appropriate example would have been to open a random (direct access) file with a record length of 400. There are also problems in the *FILES* routine itself. Use of *FILES* involves displacement of the graphics area, and *FILES* will sometimes set the start of page one to an odd page boundary in Dbasic 1.0. Since the SAM chip can only handle graphics on even pages, this results in garbage appearing at the top of the high resolution picture. To prevent this, test your *FILES* statement on a computer with Dbasic 1.0 (In RAM, if necessary, I hope to soon publish a routine to install different BASICs from disk files into the upper-half of the 64K RAM.) before putting it in the program and check the contents of location \$BC. If *PRINT PEEK(&HBC)* gives an odd number after *FILES*, add 256 to the second number in *FILES* and try again. *FILES* also may require moving the BASIC program, and in Disk BASIC 1.0 it has the same bug, with similar solutions, that *PGLEAR* does. Note that many BASIC and machine language programs assume that graphics Page 1 begins at \$E00 and makes use of this area. If *FILES* causes Disk BASIC's file handler variables to move into this area, these programs could cause a crash; therefore, it is a good idea to print a warning to the user when a program's use of *FILES* causes *PEEK(&HBC)* to exceed 14. Finally, executing *FILES* closes all disk files and does an automatic *CLEAR*.



Computer Enthusiast: "Why did the computer see the dentist?"
 Son: "I don't know, Dad. Why?"
 Computer Enthusiast: "To straighten out its byte."



NEW!!!
AUSTRALIAN GEOGRAPHY

The first 3 programmes in a set aimed at all geographic aspects of each individual state. Programmes cover general agriculture, weather, towns, industry, rivers etc. in tutorial and questionnaire form. A splendid aid to teaching Australian geography in school or home. Two programmes on each tape. All 16Kecb \$14.95 each.

SOUTH AUSTRALIA
NEW SOUTH WALES
QUEENSLAND

Plus all the latest arcades and utilities at the best prices.

R.D.L. COLOR COMPUTER SOFTWARE
 31 NEDLAND CRES.,
 PT. NOARLUNGA STH.,
 S.A. 5167
 (08) 386 1647

Editor:
 Have you heard these two?
 Wife of Computer Enthusiast: "Why did the computer see a chiropractor?"
 Computer Enthusiast: "I don't know. Why?"
 Wife: "Because it had a slipped disk."

BACK MORE WINNERS

BACK WINNERS BY USING THE POWER OF YOUR COMPUTER TO PICK WINNERS.

PROVEN ANALYTICAL PROGRAMS BEING USED BY SUCCESSFUL FULL TIME PUNTERS.

WE ARE COMPUTER RACING SPECIALISTS.

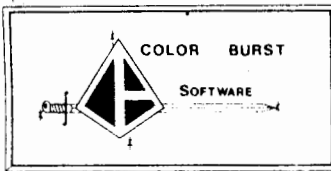
Price includes Manual and postage.

<u>PASTMASTER</u> \$75	<u>CHIP PICKER</u> \$30	(Other specialist programs available on request.)
<u>HARNESS MASTER</u> \$50	<u>DATA RATER</u> \$50	

Write for further information or phone (055) 23 3793 - (055) 23 3205.

SILICON SYSTEMS SOFTWARE - P.O. Box 392 - PORTLAND - Victoria 3305.





Color Burst Software

QUALITY PROGRAMMES FOR THE TRS80 COLOR COMPUTER

P.O. BOX 256, ROSEVILLE, NSW, 2069 PHONE: (02) 467-1619

FOR A FREE CATALOG SEND NAME AND ADDRESS TO COLOR BURST SOFTWARE, P.O. BOX 256, ROSEVILLE, NSW, 2069. OR PHONE: (02) 467-1619

ARC • ARCADE	SIM • SIMULATION	ADV • ADVENTURE	GRA/ADV • GRAPHIC ADVENTURE	GME • GAME	EDC • EDUCATIONAL
ASTRO BLAST (16K ARC) 28.00		JUNIORS REVENGE (32K ARC) 33.00		COSMIC CLONES (16K ARC) 29.00	APPLICATION PROGRAMS TAPE DISK
BLACK SANCTUM (16K ADV) 22.00		KOMET KAZE (16K ARC) 23.00		CESSNA LANDER (16K SIM) 22.00	AUTOTERM 46.00
CALIXTO ISLAND (16K ADV) 22.00		LABYRINTH (16K ARC) 23.00		CHOPPER STRIKE (16K ARC) 32.00	CC DATABASE/LETTERWR 52.00 57.00
DEFENSE (16K ARC) 25.00		MAD BOMBER (16K ARC) 22.00		COLOR FURY (32K ARC) 32.00	COLOR DFT 30.00 35.00
DOONKEY KING (32K ARC) 28.00		MARS LANDER (16K ARC) 22.00		DEMON ASSAULT (16K ARC) 29.00	ELITE CLAC 63.00 68.00
FROG TREK (16K ARC) 17.00		MEGAPEDE (16K ARC) 29.00		DESERT PATROL (32K ARC) 29.00	ELITE WORD 68.00 -
GALACTIC HANGMAN (32K GME) 17.00		THE NIBBLER (16K ARC) 23.00		DOODLE BUG (16K ARC) 31.00	FILMASTER 35.00 -
GEOGRAPHY PAC (16K EDC) 33.00		MS. NIBBLER (16K ARC) 23.00		EL BANDITO (16K ARC) 29.00	SCHEMATIC DRAFTING PR - 62.00
GHOST GOBBLERS (16K ARC) 22.00		MUDDIES (32K ARC) 32.00		FOOD WAR (32K ARC) 29.00	SMALL BUSINESS ACCT. - 150.00
HUD flight/sim 16K SIM) 20.00		OUTHOUSE (32K ARC) 32.00		FRYA DRACA (32K ARC) 29.00	TELEWRITER-64 57.00 68.00
KATERPILLAR ATTK (16K ARC) 28.00		PATTI-PAK (32K ARC) 32.00		GLAZZONS (16K ARC) 29.00	TERMTALK 45.00 50.00
KEYS OF WIZARD (16K ADV) 22.00		PLANET RAIDERS (32K ARC) 29.00		HAYWIRE (16K ARC) 29.00	YIP DATABASE - 68.00
LANGER/JOUST (32K ARC) 28.00		RAIL RUNNER (16K ARC) 29.00		ICE MASTER (32K ARC) 29.00	YIP SPELLER - 68.00
MATH DRILL (16K EDC) 22.00		SEAMOLFE (32K ARC) 29.00		INTERCEPTOR (32K ARC) 23.00	YIP TERMINAL 52.00 57.00
MS. GOBLER (32K ARC) 28.00		SHARK TREASURE (16K ARC) 29.00		PARAMDIOS AMON. (16K ADV) 23.00	YIP WRITER 63.00 68.00
PLANET INVASION (16K ARC) 25.00		SPACE RAIDERS (16K ARC) 29.00		PYRAMIC (16K ADV) 23.00	TIMS (TAPE INFO. MAN. 515) 128.00 -
SPACE COMMAND (16K ARC) 10.00		TIME BANDIT (32K GRA/ADV) 29.00		SEA QUEST (32K ADV) 29.00	UTILITIES TAPE DISK
STORY PROBLEMS (16K EDC) 22.00		TUT (32K GRA/ADV) 29.00		SHERMANGANS (32K GRA/ADV) 29.00	64COL. MOD I/III EMULATOR - 23.00
TEXT GUESS (16K GME) 10.00		VENTURER (16K ARC) 29.00		WIZARD 64 (64K ADV) 32.00	64K BOOT/PAGER 17.00 -
TIMS DATA BASE (16K UTL) 28.00		WACKY FOOD (32K ARC) 26.00		FLIPPER (16K ARC) 20.00	64 DISK UTIL. PACK - 26.00
TRAPFALL (16K GRA/ADV) 31.00		XYGIDD (16K ARC) 23.00		INTERGALACTIC FORCE (16K ARC) 3D 25.00	DISK MANAGER - 29.00
VIKING (16K SIM) 22.00		ZEUS (16K ARC) 29.00		EIGHT BALL (32K GME) 29.00	DISK UTILITY 23.00 -
WHIRL BIRD RUN (16K ARC) 28.00		ADVENTURE TRIL (16K GRA/ADV) 29.00		BLOC HEAD (16K ARC) 29.00	HIDDEN BASIC 23.00 -
ZAKSUND 3D (32K ARC) 30.00		CINECON MOON 3D (16K GRA/ADV) 29.00		REDWOOD GOLF (32K GME) 29.00	MDISK 23.00 -
BUMPERS (16K ARC) 29.00		CALIXTO ISLAND (32K GRA/ADV) 30.00		TALKING FINAL CNTMNM (32K ARC) 29.00	PRITTY PRINTER 23.00 -
ZONE 6 3D (16K ARC) 25.00		CIRCLE WORLD (16K ADV) 23.00		STAGECOACH (32K GRA/ADV) 25.00	QUICKSORT 12.00 -
DEMON SEED (32K ARC) 32.00		DEATHSHIP (16K ADV) 18.00		INSPECTOR CLUESEAU (32K GRA/ADV) 25.00	ROMBACK 20.00 -
CASHMAN (32K ARC) 32.00		DERELICT (16K ADV) 23.00			SUPER SCREEN 35.00 -
QUEST (16K GRA/ADV) 23.00		EARTHQUAKE (16K ADV) 23.00			SUPER ZAP - 40.00
WIZARDS TOWER (32K GRA/ADV) 23.00		FEMOTOS REV. 3D (32K GRA/ADV) 29.00			TAPE UTILITY 35.00 40.00
DUNGEONS OF DEATH (32K GRA/ADV) 23.00		GREYMOON (16K SIM) 23.00			YIP DISK ZAP 68.00
BAG-IT-MAN (32K ARC) 29.00		HAUNTED HOUSE (16K ADV) 18.00			
CATERPILLAR (16K ARC) 29.00		INCA TREASURE (16K ADV) 23.00			
CAVE HUNTER (16K ARC) 29.00		MARS (16K ADV) 23.00			

MAIL ORDERS POSTED WITHIN 24 HOURS.

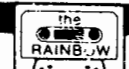
Make Cheque/Money Order out to COLOR BURST SOFTWARE.

Address Orders to: COLOR BURST SOFTWARE, P.O. Box 256, Roseville, NSW, 2069

OR Phone: (02) 467-1619

GAME

32K ECB



THE TRIP

Jacques Bourgeois



The Trip is a 32K Hi-Res Adventure game that you play mostly with a joystick. The main element of the game is a car which you ride on a main road. This main road crosses 10 different side roads on which you go to find the objects you need.

It is a mix of Adventure and arcade-type games. Luck is involved as well as skill and head work. Most of the elements of the game are random, so you can play it over and over again and find it challenging every time.

It is not easy to get through, and even a skilled *Tripper* may take two hours to finish it or make a wrong move and be killed.

Free your friend Joe, prisoner of the evil forces. At the start of the game, you do not know where the evil forces hide, and you have nothing in your possession.

You are represented by the flashing dot on the screen and can move around using your right joystick. Be careful, however, because moving in some positions may get you into trouble.

To get into action, you have to use your fire button. For example, if you see a sign moving in front of you, push the fire button and you will be able to read it. The computer will react by flashing a message for a few seconds or by changing the graphics. If nothing happens, it means that you are not in the right position on the screen, or that there is nothing to do there.

You can press 'I' any time the cursor is present on the screen to get an inventory of what you have. Pressing the Space Bar in the same conditions will automatically bring you back to the car (if you have one) and it can sometimes help in difficult situations. 'I' and the Space Bar will not work when there is text on the screen.

The car, you will find, is very sophisticated. It will give you instructions the first time you will get into it by *speaking to you*. The program makes use of the audio possibilities of your CoCo, but for that, you will have to prepare a short recording.

Make a save of the program on an empty cassette. Right after the program, record the following text with a microphone, pinching your nose to give the impression that a computer is talking:

I am the Car.

Please listen carefully because I will not repeat these instructions.

I am an all push button car. You have to push a button to turn me ON or OFF, to move around or to open the door to get out of me.

Once on the main road, you will encounter a number of side roads. Each one bears a number which will appear at the upper-left or right side of your screen. You can turn on a side road as long as you can see the road and the number. If you turn when there is no road, you will end up in the field and lose control of me.

Once on a side road, your screen will become blank. You then have to open the door and get out to see what happens there.

To get back to me and the main road, try hitting the Space Bar.

Good luck and have a good trip.

This recording will self-destruct within 10 seconds.

If you leave the recorder on play after loading the game, the message will be heard through your TV the first time you enter the car. The recorder will shut off after one minute of play, so make your recording less than one minute long.

The computer will be in a long loop during that time and will not accept any command. If you want to get rid of that feature, delete the end of Line 99 from :IFPEEK. This program will not run with the disk drive plugged in.

11	118	100	195	179	96
24	109	112	219	190	207
35	216	122	67	201	183
42	97	133	201	208	1
52	130	138	146	218	181
63	131	147	200	228	93
71	5	156	10	238	155
78	205	162	248	244	205
85	218	168	84	END	218

The listing:

```

1 CLEAR2,32766: CLEAR300: POKE6549
5,0: IFPEEK(32767)<7THENPOKE32767
,8: GOTO16ELSE22
2 FORX=1TO1000: NEXT: RETURN
3 FORX=1TO3000: NEXT: RETURN
4 IF (JOYSTK(0)<4ORJOYSTK(0)>59OR
JOYSTK(1)<4ORJOYSTK(1)>59) AND (PE

```

```

EK(65280)=126ORPEEK(65280)=254) T
HEN4
5 IFC8=1THENT=T+1: IFT>30THENT=0:
TX=TX-1: IFAP=1THENLINE(TX,144)-(
TX,152),PSET
6 A$=INKEY$: IFA$=" "ANDAB=1ANDAT
=0ANDC6=0THEN91ELSE IFA$="I"THENC
LS:GOSUB241
7 IFINKEY$="I"THENC LS:GOSUB241
8 IFAT=1THENCX=JOYSTK(0)*4: CY=JO
YSTK(1)+108: GET(CX,CY)-(CX+3,CY+
3),M: PUT(CX,CY)-(CX+3,CY+3),A2:
FORX=1TO9: NEXT: PUT(CX,CY)-(CX+3,
CY+3),M: RETURN
9 PUT(CX,CY)-(CX+3,CY+3),M: IFJOY
STK(0)<4THENCX=CX-8: IFCX<1THENCX
=1: GET(CX,CY)-(CX+3,CY+3),M ELSE
GET(CX,CY)-(CX+3,CY+3),M: SOUND12
5,1
10 IFJOYSTK(0)>59THENCX=CX+8: IFC
X>252THENCX=252: GET(CX,CY)-(CX+3
,CY+3),M ELSEGET(CX,CY)-(CX+3,CY
+3),M: SOUND125,1
11 IFJOYSTK(1)<4THENCY=CY-3: IFCY
<1THENCY=1: GET(CX,CY)-(CX+3,CY+3
),M ELSEGET(CX,CY)-(CX+3,CY+3),M
: IFJOYSTK(0)>3ANDJOYSTK(0)<60THE
NSOUND125,1
12 IFJOYSTK(1)>59THENCY=CY+3: IFC
Y>188THENCY=188: GET(CX,CY)-(CX+3
,CY+3),M ELSEGET(CX,CY)-(CX+3,CY
+3),M: IFJOYSTK(0)>3ANDJOYSTK(0)<
60THENSOUND125,1
13 IFPPPOINT(CX-1,CY-1)=10RPPPOINT
(CX-1,CY-1)=5THENPUT(CX,CY)-(CX+
3,CY+3),A2: RETURNELSEPUT(CX,CY)-
(CX+3,CY+3),A1: RETURN
14 FORX=1TO2000: NEXT: RETURN
15 DATAh,e,,t,r,i,p
16 A$=STRING$(32," "): CLS0: PRINT
@226,"WELCOME": PRINT@234,"TO":
PRINT@160,A$: PRINT@288,A$:
17 PLAY"L301EEEL1C": Y=236
18 FORZ=1TO8: Y=Y+2: READA$: PRINT@
Y,A$: IFA$<>" "THENSOUND250,3: GOS
UB2: NEXTELSEGOSUB2: NEXT
19 PLAY"L301AAAL1F": GOSUB2
20 PRINT@352," C. JACQUES BOURG
E O I S - 1983
21 FORX=1TO255STEP3: SOUNDX,1: NEX
T
22 CLS0: PRINT@256," WE HOPE YOU
WILL STAY ALIVE HA!
HA! HA!"
23 DIMA1(3,3),A2(3,3),M(3,3),R(1
0),RS(10),C(16,16),D(3,3),E(3,3)
,F(3,3)
24 Y=RND(10): FORX=1TO10: IFR(X)=Y
THEN24
25 IFR(X)=0THENR(X)=Y: IFX=10THEN

```

```

26 ELSE GOTO 24 ELSE NEXT
26 Y=RND(10):FORX=1TO10:IFRS(X)=
Y THEN 26
27 IFRS(X)=0 THEN RRS(X)=Y:IFX=10 TH
EN 28 ELSE GOTO 26 ELSE NEXT
28 PMODE3:PCLS4:GET(10,10)-(26,2
6),C:PCLS:GET(0,0)-(3,3),A1
29 B0$="U48R48D48NL48H48
30 LINE(0,0)-(9,9),PSET,BF::PAIN
T(2,2),2,1:GET(1,1)-(4,4),A2
31 PMODE3:PCLS:SCREEN1,0
32 DRAW"BM0,96C3R255":DRAW"BM0,8
0E2F2":FORX=1TO63:DRAW"BM-4,+1;E
2F2BU1E2F2":NEXT:PAINT(2,84),2,3
:PAINT(1,1),3,3:DRAW"BM0,96C2R25
5
33 FORZ=1TO50:X=RND(255):Y=RND(9
5):PSET(X,Y,2):NEXT:CIRCLE(50,20
),10,2:PAINT(50,20),2,2
34 AT=0:COLOR4,1:LINE(8,96)-(56,
68),PSET,BF:LINE(84,104)-(220,52
),PSET,BF:CIRCLE(182,52),24,4:PA
INT(182,51),4,4:DRAW"BM182,28C1U
8
35 DRAW"BM16,76;C3R32D20L32U20":
PAINT(32,84),2,3:DRAW"BM16,80;R3
2":FORY=1TO3:DRAW"BM-32,+4;R32":
NEXT:DRAW"BM170,88;R24D16L24U16"
:PAINT(182,96);3,3
36 LINE(84,52)-(220,56),PRESET,B
F:FORX=164TO196STEP8:IFX<>180THE
NLINEX(X,56)-(X+4,104),PRESET,B:P
AINT(X+2,57),1,1:NEXT ELSE NEXT:LI
NE(182,20)-(182,28),PSET:LINE(84
,104)-(220,104),PSET
37 COLOR4,1:FORX=88TO148STEP20:L
INE(X,64)-(X+12,76),PRESET,BF:PA
INT(X+2,65),2,4:LINE(X+6,60)-(X+
6,80),PSET:NEXT:LINE(86,70)-(162
,70),PSET:FORY=64TO88STEP24:LINE
(204,Y)-(216,Y+12),PRESET,BF:PAI
NT(212,Y+1),2,4:LINE(202,Y+6)-(2
18,Y+6),PSET:NEXT
38 LINE(210,60)-(210,102),PSET:F
ORX=92TO128STEP36:LINE(X,84)-(X+
28,100),PRESET,BF:PAINT(X+2,90),
2,4:NEXT:FORY=80TO96STEP4:LINE(8
8,Y)-(160,Y),PSET:NEXT:FORX=98TO
114STEP8:LINE(X,83)-(X,101),PSET
:LINE(X+36,83)-(X+36,101),PSET:N
EXT
39 COLOR3,1:LINE(4,172)-(16,96),
PSET:LINE(48,96)-(60,172),PSET:L
INE(158,172)-(170,104),PSET:LINE
-(194,104),PSET:LINE-(206,172),P
SET:DRAW"BM0,172R4BR56R98BR48R50
":PAINT(30,191),3,3:DRAW"BM160,1
04C4R40
40 COLOR2,1:FORX=1TO255:Y=RND(10
)-2:LINE(X,191)-(X,191-Y),PRESET
:NEXT:DRAW"BM136,168U12L8U12R16D
12L8BR92L8U12R16D12L8D12":PAINT(
132,148),2,2:PAINT(226,148),2,2
41 FORX=132TO224STEP92:FORY=148T
O152STEP4:FORW=0TO9STEP2:PSET(X+
W,Y,3):NEXTW,Y,X
42 IFPEEK(32766)=7THENCX=36:CY=1
10ELSECX=180:CY=170
43 A1=0:A2=0:GET(CX,CY)-(CX+3,CY
+3),M
44 IFA1=1THENDRAW"BM16,76C4R32D2
0L32U20":PAINT(32,84),3,4:PAINT(
32,78),3,4:PAINT(32,92),3,4
45 IFA2=1THENDRAW"BM179,88C2R7D1
5L7U15":PAINT(183,90),2,2
46 GOSUB4
47 IFCY<172ANDCX>0AND(CX<173ORCX
>189)THENIFPPOINT(CX-1,CY-1)=1OR
PPOINT(CX+4,CY+4)=1THENCLS:PRINT
@101,"YOU HEAR A DOG BARKING":A3
=A3+1:GOSUB14:ELSEA3=0
48 IFA3>1THENA3=0:PRINT@192,"YOU
STAYED TOO LONG ON THE GRASSTHE
INVISIBLE DOG COMES AND EATSYOU
.":IFAC=1THEN243ELSE248
49 SCREEN1,0:IF(CX<84 ANDA1=0 AN
DCY<96)OR(CX>84 ANDA2=0 ANDCY<104)
THENPUT(CX,CY)-(CX+3,CY+3),M:CY=
CY+4:GET(CX,CY)-(CX+3,CY+3),M
50 IFA2=1ANDCX>172ANDCX<190ANDCY
<104THEN58
51 IFA1=1ANDCX<48ANDCX>12ANDCY<9
6THEN61
52 IF(PEEK(65280)=126ORPEEK(6528
0)=254)THEN53ELSEGOTO46
53 IFCX<48ANDCX>12ANDCY<9
4ANDCY<100THENA1=1:GOTO44
54 IFCY<172THEN56ELSEIFCY<182AND
CX>126ANDCX<142THENCLS:PRINT@232
,"THE SIGN READS:":PRINT@295,"KE
EP OFF THE GRASS":GOSUB3:GOSUB2:
SCREEN1,0
55 IFCY<182ANDCX>218ANDCX<238THE
NCLS:PRINT@232,"THE SIGN READS:":
PRINT@290,"BEWARE OF THE INVISI
BLE DOG":GOSUB3:SCREEN1,0
56 IFA2=0ANDCX>172ANDCX<190ANDCY
<107THENA2=1:GOTO45
57 GOTO46
58 CLS0:PRINTSTRING$(32,CHR$(159
)):FORX=30TO34:FORY=31TO1STEP-1
:SET(X,Y,2):NEXTY,X:PRINT@96,"A
TRAP OPENS ";PRINT@128,"UNDER Y
OU AND";PRINT@160,"YOU FALL IN
A";PRINT@192,"DEEP DARK PIT";
59 FORY=1TO31:SET(32,Y,RND(8)):S
OUNDY*8,2:SET(32,Y,2):NEXT:FORX=
31TO34:SET(X,31,4):NEXT:PRINT@14
6,"YOU ARE THE ";PRINT@210,"R
ED BLOB ";PRINT:GOSUB3:IFAC

```

```

=1THEN243ELSE248
60 GOTO46
61 CX=244:CY=180
62 POKE32766,7:AH=RND(5)-1:SCREE
N1,0:PCLS3:FORW=1TO8:X=RND(18)+1
62:Y=RND(42)+58:PSET(X,Y,2):NEXT
:LINE(220,108)-(228,116),PRESET,
BF
63 COLOR4,3:FORX=0TO1:FORY=0TO1:
LINE(162+12*X,58+22*Y)-(162+12*(
X+1),58+22*(Y+1)),PSET,B:NEXTY,X
:FORY=60TO84STEP24:NEXT:COLOR2,1
:LINE(223,110)-(224,111),PSET,BF
:GET(CX,CY)-(CX+3,CY+3),M
64 GOSUB4
65 IFPEEK(65280)=126ORPEEK(65280
)=254THEN66ELSEGOTO64
66 IFCX>160ANDCX<189ANDCY>54ANDC
Y<102THENCLS3:PRINT@166,"NICE WE
ATHER OUTSIDE";:PRINT@225,"THE F
ULL MOON GIVES A STRANGE";:PRINT
@259,"LOOK TO ALMOST EVERYTHING"
;:GOSUB3:SCREEN1,0:GOTO64
67 IFCX>218ANDCX<230ANDCY>106AND
CY<114THENIFA6=1THENA6=0:GOTO62E
LSE69
68 IFA6=1THEN78ELSE64
69 A6=1:LINE(0,56)-(136,100),PSE
T,BF:FORX=12TO108STEP24:LINE(X,6
4)-(X+20,92),PRESET,BF:CIRCLE(X+
5,84),3,2:NEXT
70 LINE(0,164)-(255,191),PRESET,
BF:LINE(0,163)-(100,128),PSET,BF
:FORX=4TO76STEP24:LINE(X,136)-(X
+20,162),PRESET,BF:CIRCLE(X+4,14
4),3,2:NEXT
71 COLOR4,1:FORY=1TO3:LINE(4,102
+Y)-(100,102+Y),PSET:LINE(4,114+
Y)-(100,114+Y),PSET:NEXT:FORX=14
TO86STEP12:FORY=1TO3:LINE(X+Y,10
4)-(X+Y,116),PSET:IFX=14ORX=50OR
X=86THENLINE(X+Y+7,102)-(X+Y+7,1
06),PRESET:NEXTY,X ELSENEXTY,X
72 IFAG=1THEN78ELSEFORX=86TO198S
TEP28:IFX=86ORX=170ORX=198THENC I
RCLE(X,166),14,3:PAINT(X,176),3,
3:CIRCLE(X,166),7,2:PAINT(X,166)
,2,2:A$="BM"+STR$(X):DRAWA$+",16
6C3NU7ND7NE7NF7NG7NH7NR7L7":NEXT
ELSENEXT
73 COLOR2,1:FORX=0TO1:LINE(163+X
,58+X)-(188-X,101-X),PSET,B:LINE
(175+X,58)-(175+X,101),PSET:LINE
(163,79+X)-(188,79+X),PSET,B:NEX
T:LINE(159,100)-(192,104),PSET,B
F
74 DRAW"BM24,168C4R44BR36R48BR64
R12BM24,152M92,128NM108,108R8BE8
U12R60BM182,114L34ND14BL8ND14L26
D14R26BR8R54BM222,152R6D10BL30BU

```

```

4L28
75 CIRCLE(32,175),24,4,1,.63,.7:
CIRCLE(24,144),16,4,1,.05,.3:CIR
CLE(36,144),28,4,1,.32,.4:CIRCLE
(100,120),8,4,1,0,.25:CIRCLE(86,
180),22,4,1,.6,.9:CIRCLE(170,180
),22,4,1,.6,.75:CIRCLE(198,180),
22,4,1,.75,.9 :CIRCLE(145,255),8
8,4,1,.81,.96
76 PAINT(24,164),4,4:PAINT(24,15
6),2,4:COLOR2,1:LINE(114,113)-(1
41,129),PSET,B:DRAW"BM179,113C2L
33D16R58":PAINT(226,154),4,4
77 CIRCLE(64,168),20,3,1,.5,.75:
CIRCLE(64,168),24,3,1,.5,.75:DR A
W"BM40,168C3R4BM64,148R158H4L152
":PAINT(210,145),3,3:CIRCLE(145,
255),88,4,1,.81,.96:PAINT(44,165
),3,3:DRAW"BM110,168U58R34D58":C
L=RND(5):DRAW"BM114,138C2R8D1L8D
1R8"
78 IFCX<125ANDCX>111ANDCY>134AND
CY<142THEN79ELSEGOTO80
79 IFA7=1ANDAG=0THEN89ELSECLS7:P
RINT@230,"THIS DOOR IS LOCKED";:
GOSUB2:SCREEN1,0
80 FORX=14TO110STEP24:IFCX>=X AN
DCX<X+8ANDCY<87ANDCY>79THENPAINT
(CX+9,CY),3,2:CIRCLE(X+3,84),3,3
:GET(CX,CY-4)-(CX+3,CY-1),M:GOTO
81ELSENEXT:GOTO64
81 X=12+24*AH:IFA7=0ORAG=1THEN82
ELSEGOTO64
82 IFCX>X ANDCX<X+18THEN83ELSEGO
TO64
83 IFAR=0ANDAG=1THENGOTO88ELSEIF
AR=0THENFORW=X+4TOX+16STEP4:FORY
=68TO88STEP4:PSET(W,Y,RND(2)*2):
NEXTY,W:GOSUB2:CLS3:PRINT@96,"TH
ERE IS A PANEL FILLED WITH REDAN
D YELLOW KEYS":PRINT"DO YOU WANT
A KEY (Y/N)";ELSEGOTO64
84 INPUTA$:IFA$="Y"THEN85ELSEESC R
EEN1,0:GOTO64
85 PRINT@327,"WHAT COLOR (Y/R)";
:INPUTZ$:A7=1:IFZ$="R"THENCLS:A6
=0:FORY=1TO10:PRINT@234,"***ALAR
M***":FORW=1TO200STEP25:SOUNDW,1
:NEXTW:PRINT@234,"***alarm***":N
EXTY:CLS:PRINT"THE INVISIBLE DOG
COMES AND EATSYOU.":IFAC=1THEN2
43ELSEGOTO248
86 IFZ$<>"Y"THEN85
87 SCREEN1,0:GOTO64
88 CLS:X=RND(2000):PRINT:PRINT"Y
OU FIND"X"GOLD COINS":G=G+X:GOSU
B241:AR=1:GOTO64
89 IFPEEK(32767)<>7THENA5=1:AUDI
OON:MOTORONELSEA5=1
90 AU=0:A7=0:A8=1:C5=0:C8=0:PX=1

```

```

04:PY=161:TX=156:T=0
91 IFC8=0THENZ1=3:Z2=2:GOTO94ELS
EZ1=2:Z2=3:IFTX>92THENCLS:PRINT:
PRINT"YOU FORGOT TO TURN THE MOT
OR OFFWHEN YOU LEFT THE CAR. YO
U ARE LUCKY THERE IS SOME GAS LE
FT. BE CAREFUL NEXT TIME":GOSUB3:
GOTO94ELSEGOTO92
92 CLS:PRINT:PRINT"YOU LEFT THE
CAR WITHOUT TURNINGTHE MOTOR OFF
. IT RAN OUT OF GAS
":PRINT:PRINT"the game is ove
r":GOTO249
93 IFC8=1THENZ1=2:Z2=3
94 A6=0:PCLS1:SCREEN1,0:AE=0:DRA
W"BM24,24C3M0,108R255M232,24BM20
8,0L160":CIRCLE(52,27),28,3,1,.5
2,.73:CIRCLE(204,27),28,3,1,.77,
.98:PAINT(0,26),4,3:PAINT(255,26
),4,3
95 A$="R28D20L28U20":DRAW"BM4,12
0"+A$+"BM72,120"+A$+"BM116,120"+
A$+"BM160,120"+A$+"BM224,120"+A$
:COLOR3,2:LINE(92,144)-(168,152)
,PSET,B:LINE(TX,144)-(TX,152),PS
ET:PAINT(166,150),3,3:LINE(100,1
60)-(160,188),PSET,B:LINE(10,164
)-(48,176),PSET,B
96 LINE(208,164)-(228,176),PSET,
B:LINE(230,164)-(250,176),PSET,B
:PAINT(120,1),3,3:PAINT(0,109),4
,3:COLORZ1,Z2:LINE(208,164)-(228
,176),PSET,B:LINE(230,164)-(250,
176),PSET,B
97 DRAW"BM28,132C3U4L12U4M-8,+6M
+8,+6U4R12BM228,132U4R12U4M+8,+6
M-8,+6U4L12":PAINT(27,129),3,3:P
AINT(230,130),3,3:DRAW"BM80,136U
12R10F2D2G2L10M92,136BR32R10E2U2
H2L8H2U2E2R10BR32R10F2D8G2L10U12
":C$="U4R2F1D2G1L2BR7":A$="BR1H1
U2E1R1F1D2G1L2BR7
98 B$="U4R4D2L4F1M+2,+1R1":DRAW"
BM17,172"+C$+"BR1"+A$+"BL1"+A$+B
$+"S4"
99 AT=1:DRAW"BM213,172S5"+A$+"S4
U4F4NU4BR8"+A$+"BL1U2NR2U2R3BR3N
R3D2NR2D2BM88,152C1L4U4NR4U4R4BM
172,152U4NR4U4R4":IFPEEK(32767)<
>7ANDA5=1ANDAN=0THENFORX=1TO4400
0:NEXT:FORX=1TO25:SOUNDRND(255),
1:NEXT:AN=1:POKE32767,7:AUDIOOFF
:MOTOROFF:GOTO103
100 IFA5=1THEN103
101 DRAW"BM22,32C1R210M255,108BL
255M22,32BM3,96M128,32M253,96":P
AINT(118,35),1,1:PAINT(138,35),1
,1:FORW=1TO20:X=RND(208)+24:Y=RN
D(31):IFPOINT(X,Y)=3THENPSET(X,
Y,2):NEXTELSENEXT
102 COLOR3,2:LINE(122,108)-(128,
32),PRESET:LINE-(134,108),PRESET
:LINE-(122,108),PRESET:PAINT(128
,100),2,2:COLOR4,3:LINE(0,0)-(25
5,191),PSET,B
103 COLOR3,2:SCREEN1,0:AP=1:GOSU
B5:IFTX<92THENCLS:PRINT:PRINT"YO
U RAN OUT OF GAS":PRINT:PRINT"th
e game is over":GOTO249
104 IF(PEEK(65280)=126ORPEEK(652
80)=254)THEN105ELSE122
105 IFCY<176ANDCY>162THEN106ELSE
110
106 IFAQ=1ANDCX>8ANDCX<47THENC6=
1:GOTO232
107 IFCX>206ANDCX<227ANDC8=0THEN
C8=1:LINE(208,164)-(228,176),PRE
SET,B:LINE(230,164)-(250,176),PS
ET,B
108 IFCX>230ANDCX<251ANDC8=1THEN
C8=0:LINE(208,164)-(228,176),PSE
T,B:LINE(230,164)-(250,176),PRES
ET,B:C5=0
109 GOTO103
110 IFC8=0ORAQ=1THEN103
111 IFCX>158ANDCX<187THENC5=1:IF
A5=1THENCLS:AS=1:AU=1:A7=0:AT=0:
A6=0:AG=0:PRINT:PRINT"YOU CRASHE
D INTO THE WALL OF THEGARAGE.":G
OSUB2:IFAC=1THEN243ELSECLS:GOTO2
48
112 IFCX>70ANDCX<99THENC5=0ELSEG
OTO115
113 IFA5=1THENCLS:PRINT:PRINT"YO
U ARE OUT OF THE GARAGE":GOSUB2:
A5=0:GOTO114ELSECLS:PRINT:PRINT"
YOU CAN'T GO BACKWARD ON THE
ROAD.":GOSUB3:SCREEN1,0:GOTO101
114 PRINT:PRINT"THE CAR IS PROGR
AMMED TO GET YOUOUT OF TOWN AUTO
MATICALLY. IT THEN STOPS AND W
AITS FOR YOUR INSTRUCTIONS.":G
OSUB3:GOSUB2:SCREEN1,0:GOTO101
115 IFCX<144ANDCX>114THENC5=0
116 IFC5=0THEN103
117 IFCX<31ANDCX>2ANDC5=1ANDC8=1
THENC1=1ELSEIFCX>222ANDCX<251AND
C5=1ANDC8=1THENC2=1
118 IFC1=1THENIFAA=1THENR=RG:GOT
O130ELSEIFC1=1ANDC5=1THEN121
119 IFC2=1THENIFAB=1THENR=RD:GOT
O130ELSEIFC2=1ANDC5=1THEN121
120 GOTO123
121 FORX=1TO10:SCREEN1,1:SOUND1,
2:SCREEN1,0:SOUND200,2:NEXT:CLS:
PRINT:PRINT"YOU TURNED WHEN THER
E WAS NO SIDE ROAD SO YOU ENT
ERED THE FIELDS AT HIGH SPEED
AND LOST CONTROL OF THE CAR":
AS=1:C1=0:C2=0:AU=1:A7=0:AT=0:A6

```

```

=0:AG=0:GOTO243
122 IFC5=0ORAQ=1THEN103
123 T=T+2:COLOR3,1:IFAB=1THENLINE
(SD,YD)-(XD,YD),PRESET:SD=SD+.5
:XD=XD+4:YD=YD+2:IFXD>250THENPUT
(235,4)-(252,20),C:AB=0ELSELINE(
SD,YD)-(XD,YD),PSET:GOTO125
124 X=RND(20):IFX=10THENX=RND(10
):IFX=RG THEN125ELSEGOSUB229:DRA
W"BM235,20C2"+L$:RD=X:AB=1:XD=12
8:YD=32:SD=233
125 COLOR3,1:IFAA=1THENLINE(SG,Y
G)-(XG,YG),PRESET:SG=SG-.5:XG=XG
-4:YG=YG+2:IFXG<4THENPUT(4,4)-(2
0,20),C:AA=0ELSELINE(SG,YG)-(XG,
YG),PSET:GOTO127
126 X=RND(20):IFX=10THENX=RND(10
):IFX=RD THEN127ELSEGOSUB229:DRA
W"BM4,20C2"+L$:RG=X:AA=1:XG=128:
YG=32:SG=22
127 PUT(PX,PY)-(PX+3,PY+3),A1:PX
=PX+4:IFPX>157THENPX=104:PY=PY+4
:IFPY>184THENPY=161
128 PUT(PX,PY)-(PX+3,PY+3),A2
129 GOTO103
130 GOTO232
131 SCREEN1,1:PCLS5:CIRCLE(128,9
6),125,6,.75:PAINT(0,0),6,6:COLO
R6,7:LINE(0,176)-(255,176),PSET,
BF:PAINT(128,177),6,6:LINE(32,16
8)-(224,175),PRESET,BF
132 BR$="BE4E12F4G16U7"
133 DRAW"BM40,168S4C8U132BE12R52
BF12D132L74BM60,34R32BD24L32":CI
RCLE(50,35),12,8,1,.5,.75:CIRCLE
(106,35),12,8,1,.75,1:CIRCLE(60,
46),12,8,1,.25,.75:CIRCLE(92,46)
,12,8,1,.75,.25:PAINT(41,167),8,
8:LINE(60,40)-(92,44),PSET,BF:PA
INT(44,167),8,8
134 LINE(60,48)-(92,52),PRESET,B
F:DRAW"BM112,60C7"+BR$:PAINT(120
,56),7,7:LINE(48,72)-(104,148),P
SET,BF:DRAW"BM104,148C5L56M76,72
M104,148":PAINT(76,76),5,5:CIRCL
E(76,130),14,7:PAINT(76,130),7,7
135 CX=126:CY=182:GET(CX,CY)-(CX
+3,CY+3),M:AD=RND(2):IFAD=1THEND
RAW"BM132,167C8S7"+B0$+"S4":PAIN
T(136,163),8,8:PAINT(187,122),7,
8
136 IFQW=0THEN138ELSECLS:PRINT:P
RINT"YOU OWE"OW"COINS":PRINT:IFO
W<G THENG=G-OW:OW=0:PRINTTAB(8)"
THANK YOU!":GOSUB3:SCREEN1,1:GOT
O138
137 PRINT"YOU DO NOT HAVE ENOUGH
MONEY TO PAY YOUR DEBT. WE TAKE
WHAT YOU HAVE AND YOU WILL HAVE
TO COME BACK TO GET GAS.":OW=0
    
```

```

W-G:G=0:GOSUB3:GOSUB2:GOTO131
138 GOSUB4:SCREEN1,1:IFPEEK(6528
0)=126ORPEEK(65280)=254THEN139EL
SEGOTO138
139 IFCX>120ANDCX<130ANDCY>41AND
CY<56THENPUT(CX,CY)-(CX+3,CY+3),
M:DRAW"BM112,60S4C5"+BR$:PAINT(1
20,56),5,5:DRAW"BM112,60A1C7"+BR
$:PAINT(120,69),7,7:FORX=1TO25:S
OUND240,1:SOUND255,2:NEXT:CLS8:P
RINT@229,"YOUR TANK IS NOW FULL"
;:ELSE142
140 W=166-TX:TX=166:PRINT@288,W"
LITERS * 15 COINS/LITER":PRINT"T
OTAL COST:"W*15"COINS":GOSUB3:IF
G-W*15<0THENPRINT@384,"YOU HAVE
ONLY"G"COINS, SO YOU OWE"ABS(G
-W*15)"COINS":OW=ABS(G-W*15):G=0
:GOSUB3:ELSEG=G-W*15:GOTO141
141 SCREEN1,1:DRAW"BM112,60C5"+B
R$:PAINT(120,69),5,5:DRAW"BM112,
60A0C7"+BR$:PAINT(120,56),7,7:CO
LOR8,6:LINE(116,52)-(116,78),PSE
T:PAINT(114,64),8,8
142 IFAD=1ANDCX>134ANDPPOINT(CX-
1,CY-1)=8ANDPPOINT(CX+4,CY+4)=8T
HENGOSUB237:SCREEN1,1:GOTO138ELS
E138
143 R=RND(4):PCLSR:SCREEN1,0:CX=
128:CY=96
144 V=RND(4):IFV=R THEN144ELSECO
LORV,R:LINE(0,0)-(3,3),PSET,BF:G
ET(0,0)-(3,3),D
145 Z=RND(4):IFZ=R ORZ=V THEN145
ELSEPAINT(1,1),Z,R:GET(0,0)-(3,3
),E
146 FORX=0TO255STEP8:FORY=0TO188
STEP8:IFINT(Y/8)-Y/8=0THENPUT(X,
Y)-(X+3,Y+3),D:NEXTY,X
147 W=RND(4):IFW=R ORW=V ORW=Z T
HEN147ELSEFORZ=1TO5:X=RND(215)+2
0:Y=RND(151)+20:CIRCLE(X,Y),20,W
:PAINT(X,Y),W,W:SOUND200,3:NEXT
148 CX=RND(251):CY=RND(187):IFPP
OINT(CX-1,CY-1)=W ORPPOINT(CX+4,
CY+4)=W THEN148ELSEGET(CX,CY)-(C
X+3,CY+3),M
149 X=RND(249)+1:Y=RND(185)+1:GE
T(X,Y)-(X+3,Y+3),F:PUT(X,Y)-(X+3
,Y+3),E:ZZ=RND(28):FORV=1TOZZ:GO
SUB4:IFPPOINT(CX-1,CY-1)=W ORPPO
INT(CX+4,CY+4)=W THENGOSUB2:CLS:
GOTO243ELSEIFCX<X+4ANDCX>X-4ANDC
Y<Y+4ANDCY>Y-4THEN151ELSENEXT:PU
T(X,Y)-(X+3,Y+3),F
150 IFAF=1THENAF=0:SCREEN1,0:GOT
O149ELSEAF=1:SCREEN1,1:GOTO149
151 PCLS6:SOUND132,16:DRAW"BM160
,64C7"+B0$:PAINT(162,25),7,7:PAI
    
```

```

NT(162,62),8,7:SCREEN1,0
152 GOSUB4:IFPPOINT(CX-1,CY-1)=3
ANDPPOINT(CX+4,CY+4)=3AND(PEEK(6
5280)=126ORPEEK(65280)=254)THENG
OSUB237ELSE152
153 GOSUB4:GOTO153
154 PCLS5:R=RND(3)+5:SCREEN1,1
155 V=RND(3)+5:IFV=R THEN155ELSE
FORZ=1TO29:W=RND(15)+10:X=RND(25
5):Y=RND(191):SOUND1,1:CIRCLE(X,
Y),W,V:PAINT(X,Y),V,V:NEXT:X=RND
(232)+10:Y=RND(168)+10:CIRCLE(X,
Y),15,R:PAINT(X,Y),R,R
156 X$=STR$(X-6):Y$=STR$(Y+6):DR
AW"BM"+X$+" "+Y$+"S1C5"+B0$+"S4"
:PAINT(X-4,Y+4),R,5
157 CX=RND(251)+1:CY=RND(187)+1:
IFPPOINT(CX-1,CY-1)=V OR PPOINT(
CX+3,CY+3)=V THEN 157 ELSEGET(CX
,CY)-(CX+3,CY+3),M:PUT(CX,CY)-(C
X+3,CY+3),A1
158 GOSUB4:PUT(1,1)-(4,4),M:IFPP
OINT(1,1)=V OR PPOINT(4,4)=V THE
NGOSUB2:CLS:GOTO243ELSEIFPPOINT(
CX-1,CY-1)=R OR PPOINT(CX+4,CY+4
)=R THENGOSUB2:GOSUB237:GOTO158E
LSE158
159 CLS3:IFAV<>1THENPRINT@192,"
IT IS NOT POSSIBLE TO TURN ON
THAT ROAD, A POLICEMAN ASK YOU
TO GO BACK ON THE MAIN ROAD.":AV
=AV+1:GOSUB3:C1=0:C2=0:SCREEN1,0
:GOTO103ELSE234
160 AV=AV+1:CLS:PRINT:PRINT"THE
POLICEMAN HAS FALLEN ASLEEP.DO Y
OU WANT TO GO ON (Y/N)":INPUTA$
:IFA$="Y"THEN161ELSEIFA$="N"THEN
94ELSE160
161 X=RND(3):IFX=2THEN163ELSECLS
:PRINT"YOU GO ON THE ROAD FOR A
WHILE. SUDDENLY, YOU HEAR A STRA
NGE NOISE.":FORX=255TO150STEP
-1:SOUNDX,1:NEXT:CLS5:SOUND1,19:
CLS:PRINT"SOMETHING HITTED YOUR
CAR WHICH EXPLODED. YOU ARE BADL
Y INJURED.
162 AU=1:PRINT"THIS IS A TESTING
GROUND FOR NEWWEAPONS.":GOSUB3:
GOTO243
163 X=RND(10000):CLS:PRINT"YOU C
OME TO THE SCENE OF AN ACCID
ENT WHICH SEEMS TO HAVE TAKEN
PLACE ABOUT AN HOUR AGO. ACAR I
S STILL BURNING. NEAR IT ANARMY
TRUCK IS LYING ON ONE SIDE,DOORS
OPEN. IT IS FILLED WITH BAGS.
":G=G+X:Z1=2:Z2=3:AA=0:AB
164 PRINT"A FEW SOLDIERS ARE AWA
Y IN THE FIELDS, LOOKING ON THE
GROUND AT SOMETHING YOU DO NOT S
EE. YOU GRAB ONE OF THE BAGS A
ND GO AWAY WITH IT. WHILE DRIVING
BACK TO THE MAIN ROAD, YOU OPE
N IT AND FINDS THAT IT CONTAINS
"X:PRINT@448,"GOLD COINS:
165 PRINT@480,"PRESS <ENTER> TO
GO ON.":IFINKEY$=""THEN165ELSECL
LS:GOSUB241:GOTO94
166 CLS5:PRINT@448,"IT IS SNOWIN
G SO MUCH THAT YOU CAN'T SEE.":
GOSUB3
167 IFS=0THENPRINT"DO YOU WANT T
O (1)-COME BACK ON THE MAIN ROAD
(2)-STAY ON THIS ROAD":INPUTX
:IFX=1THENAA=0:AB=0:C1=0:C2=0:C5
=1:GOTO93ELSEIFX<>2THEN167
168 CLS5:PRINT@416,"THE CAR IS S
TUCK IN THE SNOW ANDCANNOT MOVE
ANYMORE. DO YOU WANTTO USE A BAG
OF SALT TO MELT THESNOW AND FRE
E YOU (Y/N)":INPUTA$:IFA$="Y"TH
EN169ELSEIFA$="N"THEN171
169 IFS>0THENS=S-1:PRINT"IT WORK
ED, YOU ARE NOW MOVING":GOSUB3:G
OTO172
170 PRINT:PRINT"YOU DO NOT HAVE
A BAG OF SALT":GOSUB3:GOSUB241
171 AA=0:AB=0:C5=0:C8=0:CLS5:PRI
NT@0,"AFTER LONG HOURS OF FREEZI
NG, YOU FALL ASLEEP.":GOSUB3:G
OSUB3:GOTO243
172 X=RND(10):IFRS(X)=3ORRS(X)=6
THEN172ELSEC6=1:C8=0:GOTO234
173 AG=1:GOTO31
174 R=RND(3)+5:V=RND(3)+5:IFV=R
ORV=7THEN174ELSEPCLS5:SCREEN1,1:
COLORV,R:U=0:FORZ=1TO35
175 X=RND(252):Y=RND(188):W=RND(
100)+Y:IFW>188OR(X>234AND Y<16)T
HEN175ELSELINE(X,Y)-(X+4,W),PSET
,BF:NEXT:FORZ=1TO25
176 Y=RND(187):X=RND(245):IFX>23
4OR Y<16THEN176ELSELINE(X,Y)-(X+
10,Y+4),PSET,BF:NEXT:W=0
177 W=W+1:IFW=V ORW=R THEN177ELS
ECOLORW,R:DRAW"BM240,16C7S1"+B0$
+"S4":PAINT(250,6),8,7:PAINT(242
,14),7,7:CX=4:CY=185:GET(CX,CY)-
(CX+3,CY+3),M:TIMER=0
178 PUT(CX,CY)-(CX+3,CY+3),M:SX=
CX:TY=CY:GET(SX,TY)-(SX+3,TY+3),
D:GOSUB4:IFPPOINT(CX-1,CY-1)=V O
R PPOINT(CX+4,CY+4)=V THENPUT(CX
,CY)-(CX+3,CY+3),M:PUT(SX,TY)-(S
X+3,TY+3),D:CX=SX:CY=TY:GET(CX,C
Y)-(CX+3,CY+3),M:PUT(CX,CY)-(CX+
3,CY+3),A2
179 IFTIMER>1200THENLINE(U,0)-(U
,191),PRESET:U=U+4:IFCX+4<U ORU>
252THENGOSUB2:CLS:GOTO243ELSEIFU

```

```

>128THENU=U+2
180 IFCX>236ANDCY<16AND(PEEK(652
80)=126ORPEEK(65280)=254)THENGOS
UB237
181 GOTO178
182 PCLS4:SCREEN1,0:DRAW"BM240,1
6C3S1"+BO$+"S4":PAINT(250,6),3,3
:PAINT(242,14),1,3:CX=1:CY=188:G
ET(CX,CY)-(CX+3,CY+3),M:COLOR2,1
183 GOSUB4
184 X=RND(255):Y=RND(191):W=RND(
255):Z=RND(191):IF(X>236ANDY<16)
OR(W>236ANDZ<16)THEN184ELSELINE(
X,Y)-(W,Z),PSET:SOUNDRND(255),2
185 IFCX>236ANDCY<16THEN189
186 FORU=CX TOCX+3:FORV=CY TOCY+
3:IFPPOINT(U,V)=2THENGOSUB2:SCRE
EN0,0:FORUE=0TO9:CLS(UE):SOUNDRN
D(255),2:NEXT:GOTO188
187 NEXTV,U:GOTO189
188 CLS:PRINT:PRINT"YOU HAVE BEE
N HIT. YOU ARE VERYWEAK AND UNA
BLE TO GO ON.":GOSUB2:GOTO243
189 IFPEEK(65280)=126ORPEEK(6528
0)=254THENGOSUB237:GOTO183ELSE18
3
190 CLS:PLAY"EEEC":FORX=0TO351:P
RINT@X,CHR$(RND(117)+128):NEXT:I
FAM=0THENAL=RND(5000)+5000:AM=1
191 IFC8=1THENTX=TX-3
192 PRINT@352,"YOU HAVE FOUND US
. DO YOU HAVE "AL"GOLD COINS, T
HE ELIXIR ANDTHE BAG OF SALT?
193 A$=INKEY$:X=RND(351):PRINT@X
,CHR$(RND(117)+128);:IFA$="N"THE
N206ELSEIFA$="Y"THEN194ELSE193
194 PRINT@352,"":IFG>=AL ANDS>=1
THENIFB>=1THEN200ELSEIFBP>=1THEN
199
195 PRINT@384,"YOU ARE A LIAR. Y
OU DO NOT HAVE ENOUGH";:IFG<AL T
HENPRINTTAB(11)"GOLD
196 IFB=0THENPRINTTAB(11)"ELIXIR
197 IFS<1THENPRINTTAB(11)"SALT
198 GOSUB3:CLS:GOSUB241:GOTO91
199 PRINT@352,"THE BOTTLE YOU HA
VE CONTAINS A GREEN LIQUID WHIC
H IS POISON. YOU WANTED TO KIL
L US. YOU WILL DIE":GOSUB3:FORX=
352TO510:PRINT@X,CHR$(RND(117)+1
28);:NEXT:PLAY"L301EEEL1CP2L301A
AAL1F":GOTO248
200 PRINT@384,"THERE IS ONE MORE
THING YOU MUSTDO BEFORE WE GIVE
YOU BACK YOUR FRIEND: you must
drink poison! ARE YOU READY TO
DO SO?
201 A$=INKEY$:X=RND(351):PRINT@X
,CHR$(RND(117)+128);:IFA$="N"THE
N202ELSEIFA$="Y"THEN203ELSE201
202 PRINT@352,"":PRINT"TOO BAD F
OR YOU! WE KEEP YOUR FRIEND. T
HANK YOU FOR THE GOODIES."
:GOSUB3:GOSUB3:CLS:GOTO249
203 IFAC=1ANDBP>=1THENCLS:PRINT:
PRINT"THE ELIXIR YOU DRANK SOONE
R PROTECTS YOU AND YOU DO NO
T DIE WHEN YOU DRINK THE POISON.
YOUR FRIEND JOE IS FREE AN
D YOU GO AWAY WITH HIM.":PRINT:P
RINT"congratulations! YOU'VE MAD
E IT":GOTO249
204 IFBP>=1THENCLS:PRINT:PRINT"Y
OU DRINK THE POISON AND YOU DIE"
:GOTO249
205 PRINT@352,"":PRINT"YOU DO NO
T HAVE ANY POISON. COME BACK
WHEN YOU'LL HAVE SOME.":GOSUB3:
GOTO91
206 PRINT@352,"":PRINT"THEN GO A
ND TRY TO FIND WHAT YOU NEED.
":GOSUB3:GOTO91
207 AT=1:CLS:PRINT"*****main roa
d general store*****":PRINT"WE SE
LL AND BUY. WE HAVE THE LOWES
T PRICES THIS SIDE OF THE ROAD
AND WE GIVE THE BEST MONEY FOR Y
OUR GOODIES.":PRINT"THE DEALS WE
OFFER ARE SO GOOD THAT WE HAD
TO FIX A LIMIT OF
208 IFC8=1THENTX=TX-3
209 PRINT"ONE TRANSACTION BY CUS
TOMER.":PRINT:PRINT"DO YOU WANT
TO BUY, SELL OR LEAVE (B/S/L
)?
210 A$=INKEY$:IFA$="S"THEN211ELS
EIFA$="B"THEN222ELSEIFA$="L"THEN
91ELSE210
211 X=RND(4):IFX=1THENZ=S:A$="BA
GS OF SALT"ELSEIFX=2THENZ=BP:A$=
"BOTTLES OF POISON":ELSEIFX=3THE
NZ=B:A$="BOTTLES OF ELIXIR OF LO
NG LIFE"ELSEA$="EMPTY BOTTLES":Z
=B
212 Y=RND(500):IFX=1THENY=RND(25
0)ELSEIFX=4THENY=RND(50)
213 CLS:PRINT:PRINT"WE WOULD BE
INTERESTED IN BUYING":PRINTA$:PR
INT"AND WILL PAY"Y"GOLD COINS":G
OSUB241:IFZ=0THENPRINT:PRINT"WE
SEE THAT YOU HAVE NONE AND NOT
HING ELSE INTERESTS US."ELSE216
214 PRINT"MAYBE YOU WOULD LIKE T
O BUY SOMETHING (Y/N)?"
215 A$=INKEY$:IFA$="Y"THEN222ELS
EIFA$="N"THEN219ELSE215
216 PRINT:PRINT"HOW MANY DO YOU
WANT TO SELL";:INPUTW:IFW>Z THEN
GOSUB241:CLS:PRINT"YOU DO NOT HA
VE"W:GOTO216

```

```

217 IFW=0THEN214ELSEG=G+W*Y:IFX=
1THENS=S-W ELSEIFX=2THENBP=BP-W
ELSEIFX=3THENB=B-W ELSEBV=BV-W
218 CLS:PRINT:PRINT"IT'S A DEAL"
:GOSUB241:PRINT
219 AT=0:CLS:PRINT"IT'S CLOSING
TIME NOW, YOU HAVE TO LEAVE.":GO
SUB3:GOTO91
220 PRINTTAB(5)"ARE YOU INTEREST
ED (Y/N)?"
221 A$=INKEY$:IFA$=""THEN221ELSE
IFA$="Y"AND(G-Y)<0THENCLS:PRINT:
PRINT:PRINT"YOU DO NOT HAVE THE"
Y"COINS":GOSUB3:A$="":RETURNELSE
RETURN
222 CLS:PRINT"**** ON SALE TODAY
****":PRINT:Y=RND(1500)+500:IFR
ND(2)-1THENPRINT"1 BOTTLE OF ELI
XIR:"Y"GOLD":PRINT"COINS":GOSUB2
20:IFA$="Y"THENG=G-Y:B=B+1:GOTO2
18
223 IFRND(2)=1THENY=RND(1000)+50
0:PRINT"1 BOTTLE OF POISON:"Y"GO
LD":PRINT"COINS":GOSUB220:IFA$="
Y"THENG=G-Y:BP=BP+1:GOTO218
224 IFRND(2)=2THENY=RND(500)+100
:PRINT"1 BAG OF SALT:"Y"GOLD":PR
INT"COINS":GOSUB220:IFA$="Y"THEN
G=G-Y:S=S+1:GOTO218
225 IFRND(2)=1THENY=RND(2000)+50
0:PRINT"1 SURPRISE BOX:"Y"GOLD":
PRINT"COINS":GOSUB220:IFA$="Y"TH
ENG=G-Y:GOTO227
226 GOTO219
227 X=RND(4):Y=RND(4):Z=RND(4):I
FX=Y ORZ=X ORZ=Y THEN227ELSEPCLS
X:SCREEN1,0:COLORY,Z:DRAW"BM100,
95S8"+B0$+"S4":PAINT(178,27),Z,Y
:PAINT(102,90),Y,Y:GOSUB2
228 W=RND(2):FORZ=1TOW:AE=0:GOSU
B237:NEXT:GOTO219
229 IFX=1THENL$="BR6U16"ELSEIFX=
2THENL$="NR16U4E4R12U4H4L12D2"EL
SEIFX=3THENL$="R16U8NL8U8L16"ELS
EIFX=4THENL$="BR12U16G12R16"ELSE
IFX=5THENL$="BU4F4R10E2U4H2L14U8
R16
230 IFX=6THENL$="R16U8L16D8U16R1
6D2"ELSEIFX=7THENL$="E16L16"ELSE
IFX=8THENL$="R16U16L16D16U8R16"E
LSEIFX=9THENL$="NU2R16U16L16D8R1
6"ELSEIFX=10THENL$="U16BR4D16R8U
16L8
231 RETURN
232 FORX=1TO10:IFR(X)=R THEN233E
LSENEXT
233 IFRS(X)<>6ANDRS(X)<>3THENPAI
NT(126,2),4,4ELSEONRS(X)/3GOTO16
6,159
234 AB=0:AA=0:C1=0:C2=0:C5=0:AR=

```

```

0:AP=0
235 IFRS(X)=6THEN160ELSEIFC6=1TH
ENAC=0:C6=0:GOTO236ELSEAQ=1:GOTO
103
236 AT=0:ONRS(X)GOTO143,131,,190
,154,,173,207,182,174
237 IFAE=1THENRETURNELSEAE=1:X=R
ND(6):CLS:PRINT:PRINT"THE BOX OP
ENS.":PLAY"L1A#P8V10T3L2B-9":PRI
NT"IT CONTAINS:
238 IFX<3THENY=RND(2000):G=G+Y:P
RINTTAB(10)Y"GOLD COINS
239 IFX=4THENS=S+1:PRINTTAB(10)"
A BAG OF SALT"ELSEIFX=5THENBP=BP
+1:PRINTTAB(5)"A BOTTLE OF GREEN
LIQUID":INPUT"DO YOU WANT TO DR
INK IT (Y/N)":A$:IFA$="Y"THENPRI
NT"YOU DRINK AND IT IS POISON":B
P=BP-1:BV=BV+1:GOSUB3:GOTO243
240 IFX=6THENPRINTTAB(10)"NOTHIN
G"ELSEIFX=3THENPRINTTAB(5)"A BOT
TLE OF RED LIQUID":B=B+1:INPUT"D
O YOU WANT TO DRINK IT (Y/N)":A$
:IFA$="Y"THENBV=BV+1:B=B-1:CLS:P
RINT"YOU JUST DRANK ELIXIR OF LO
NG LIFE. YOU ARE NOW SURE THAT
YOU WILL NOT DIE.":AC=1
241 PRINT:PRINT"YOU NOW HAVE.":P
RINT0"GOLD COINS":PRINTS"BAGS OF
SALT":PRINTB"BOTTLES OF ELIXIR
OF LONG LIFE":PRINTBP"BOTT
LES OF POISON":PRINTBV"EMPTY BOT
TLES":PRINT:PRINT"---PRESS ANY K
EY TO CONTINUE.--"
242 IFINKEY$=""THEN242ELSESCREEN
1,0:RETURN
243 C2=0:C1=0:IFAC=0THEN245ELSEP
RINT:PRINT"YOU SHOULD BE DEAD NO
W. YOU ARE LUCKY YOU DRANK ELIXI
R OF LONG LIFE. ";
244 IFAU=1THENPRINT"HOWEVER, YOU
DON'T HAVE A CAR SO YOU WILL HA
VE TO FIND ONE":GOSUB3:GOSUB3:C8
=0:A7=0:AG=0:GOTO31ELSEC8=0:PRIN
T"THIS WAY I CAN SEND YOU BACK
ON THE ROAD.":GOSUB3:GOTO91
245 X=RND(3):IFX=2THENPRINT:PRIN
T"YOU ARE LUCKY, YOU COULD HAVE
DIED. YOU SPEND SOME TIME AT T
HEHOSPITAL AND NOW YOU ARE OK.
YOU STILL HAVE EVERYTHING";IF
AU=1THENPRINT" EXCEPTTHE CAR WHI
CH WAS DESTROYED IN THE ACCIDEN
T.":C8=0:AG=0:AR=0
246 IFX=2THENPRINT:PRINT"PRESS <
ENTER> WHEN YOU WILL FEELREADY T
O GO ON."ELSE248
247 IFINKEY$=""THEN247ELSEIFAU=0
THEN91ELSEAU=0:PRINT:PRINT"YOU L
OST THE CAR, SO YOU WILL HAVE

```



```

TO FIND ANOTHER ONE":GOSUB3:A7=0
:CB=0:A5=0:AS=0:GOTO31
248 PRINT@331,"YOU'RE DEAD";
249 PRINT@448,"***** ANOTHER GA
ME (Y/N)? *****";
250 A$=INKEY$: IFA$="Y" THEN RUNELS
EIFA$="N" THEN POKE65494,0: ENDELSE
IFA$="" THEN 250 ELSE GOTO249
251 A$=INKEY$: IFA$="" THEN 251 ELSE
PRINT@231,"ONE MOMENT PLEASE!":R
UN
    
```

```

000000
####00 ##### COMPUTER SOFTWARE SHOP
# 000000# =====
# 00##### ENQUIRIES:- PATRICK SIMONIS
# 000000 # P.O.BOX 372, Sprinswood 4127-
##### PH. 07/209.31.77
BANKCARD & MASTERCARD WELCOME
    
```

* QLI'S LARGEST & BEST *

I am the fastest and LARGEST growing software distributor in Australia, for the following reasons, I have the latest and largest range of top quality software in Arcade Games 16 and 32K, Adventure Games, Utilities, and Business over 160 programs and more on the way.

I also have parallel printer interfaces, HIL-57 Keyboard, Disk Controller (J.DOS and/or 1.0 or 1.1 DOS), C10 & C15 blank cassettes etc...

I offer service with a smile, so please call me on 07/209-31-77 between the hours of 8am and 9.30pm or write to us at the above address.

AUSSIE

make GOOD NEWS

in Rainbow GoCo and MiCo

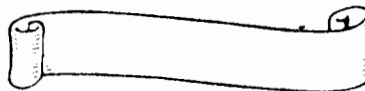
for ads

Peggy

5283391

Annabel

Sydney



Blaxland

P.O. Box 125 BLAXLAND 2774



Computer

Package Deals

Services

MAIL ORDER

64 K Ram Upgrade Kit

posted \$ 89

FULL INSTRUCTIONS INCLUDED +2 TAPE PROGRAMS TO USE YOUR 64K COCO
(PLEASE STATE OLD, NEW OR CC2 BOARD WHEN ORDERING)

10 x C-10's posted \$ 11 10 x C-30's posted \$ 16

DATA LIFE VERBATIM SS DD DISKETTES Box of 10 \$ 35

Mailing Labels per 1,000 \$ 16

Cassette Labels per 100 \$ 4

PRINTER PAPER 533 A BOX 2,000 sheets. freight extra

GOOD QUALITY PAPER 570 A BOX 3,200 SHEETS

Large Range of Software

Acting as BROKER for your TANDY Purchases

Phone: 047 39-3903

Education Through Graphics

Problems and Education

One of the greatest weaknesses of public school education is the overpowering obsession to teach facts rather than how facts can be acquired and used to solve problems. Facts are blocked off into subject matter areas with little cross-fertilization. As the body of knowledge (facts) acquired by the human race accelerates in size, the traditional teaching methods become overwhelmed by the task of cramming all this information into tidy blocks of school time.

We, as parents and friends, can provide meaningful learning experiences for children after the classroom doors are locked at the end of the school day. The Color Computer is an ideal tool for this purpose outside (as well as inside) the classroom. It is inexpensive and quite powerful when compared to other personal computers. Its ease of use and friendliness make it ideal for first-time computer users. A beginner can immediately use the CoCo with the introduction of a minimum of facts. Attention can then be turned to solving real-life problems.

Problem solving is the key to successful learning. The traditional method of education teaches a block of facts, supposedly logically arranged. Then an attempt is sometimes made to apply the "learned" facts to a set of similar problems which have right or wrong answers. When an arbitrary percentage of problems have been "solved" with matching right answers, students are rewarded with the next logical block of facts.

Problems faced in the real world do not have cut and dry "2+2=4" solutions. Real results are not black or white, right or wrong. There are many solutions to real problems, some better than others in a given situation.

Color LOGO provides one of the most free-form ways of learning that I have found. Anyone of any age can immediately encounter creative experiences with a minimum of factual knowledge with Color LOGO. The first LOGO commands encountered have a direct relationship to body-movements that are already familiar to a child.

Examples:

FORWARD, BACK, LEFT, RIGHT which can be abbreviated: FD , BK , LT , RT.

Shapes and turtle movements can be explored before the child is even aware of the concept of programs or procedures. In fact, the child will naturally develop a desire to write a complete LOGO procedure after experimenting with a few basic LOGO commands.

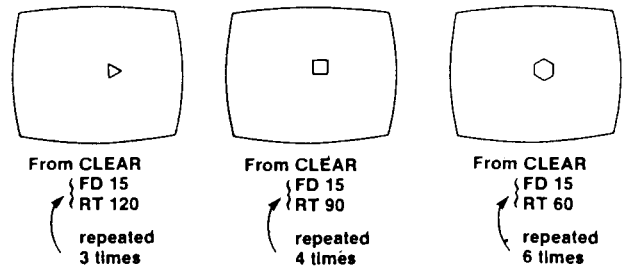
As you work with a child, listen closely for questions such

as: "How can I . . . ?", "What if . . . ?" These are clues that the child is ready to move forth to new learning experiences. Be careful that you do not provide a solution to the problem raised. The child is merely seeking clues, or new tools, for solving the problem.

The Problem Develops

Let's suppose that a young child, named Sue, is experimenting with Color LOGO.

Figure 1:



Sue, "That looks like the shape the bees use when making a honeycomb. How can I make two of them alongside each other? Is there a short way to make a shape? Can the computer remember how I made the last shape?"

This sounds like the time to introduce procedures. A procedure is simply a way to have the computer remember the steps that Sue has previously used in drawing her hexagon.

```

TO HEX ← procedure's name
  FD 15
  RT 60
  FD 15
  RT 60
  FD 15
  RT 60
  FD 15
  RT 60
  FD 15
  RT 60
  END
  
```

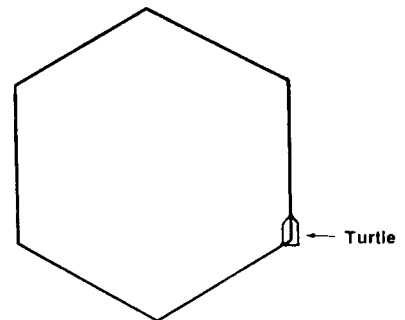
The procedure can be shortened by introducing the REPEAT command:

```

TO HEX
  REPEAT 6(FD 15 RT 60)
END
  
```

To make a second hexagon alongside the first, Sue could pick up the turtle's pen and move it to the right side of the first hexagon. Then the pen could be lowered in preparation for drawing another hexagon.

Figure 2:

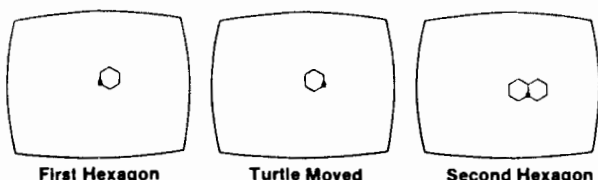


A second procedure is added to control the placement and drawing of the hexagons.

```
TO PAIR
  CLEAR
  HEX
  HEX
  PU RT 120 FD 15 LT 60
  FD 15 LT 60 PD
  HEX
  HEX
  END
  -- clear the screen
  -- use the HEX procedure
  -- use HEX procedure again
```

```
TO HEX
  REPEAT 6(FD 15 RT 60)
  END
```

Figure 3:



Notice that the turtle ended at the point where the second hexagon was started. Now Sue asks, "Can I make a complete circle of shapes around the first one?"

Your answer, "Well it might be possible. Where would you have to move the turtle to start the next one?"

Sue, "I think I'd try moving it FORWARD 15 from the second shape and then turn LEFT 60. In fact, if I did that six times, I might get them all."

You, "Why don't you try it?"

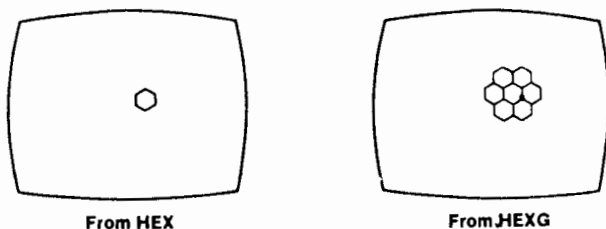
Sue, "Okay. I'll change the PAIR procedure and call it BEE. I'll call my new procedure HEX6."

```
TO BEE
  CLEAR
  HEX
  HEX6
  HEX6
  END
  -- call to new procedure added
```

```
TO HEX
  REPEAT 6(FD 15 RT 60)
  END
```

```
TO HEX6
  PU RT 120 FD 15 LT 60
  FD 15 LT 60 PD
  REPEAT 6(HEX FD 15 LT 60)
  END
```

Figure 4:



Sue, "It works! WOW! Now, I want to put another circle of shapes around that."

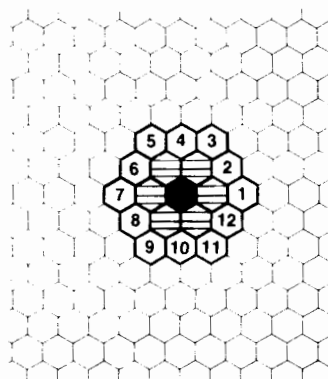
You, "How many little shapes do you think it will take?"

Sue, "I can try to fit them in my mind, or I can draw them on paper and count them."

You, "I have some paper here that has a grid made of the same shapes that you are working with. You can shade in the shapes you have so far, and then see how many you will

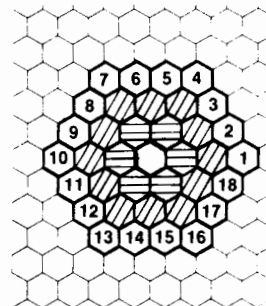
need." Sue drew them out like this:

Figure 5:



Sue discovered that she needed 12 new shapes to encircle the others. After a few false starts, Sue came up with the additional procedure HEX12. Of course she also needed to change the BEE procedure to call HEX12.

Figure 6:



```
TO BEE
  CLEAR
  HEX
  HEX6
  HEX12
  END
  -- new line
```

```
TO HEX
  REPEAT 6(FD 15 RT 60)
  END
```

```
TO HEX6
  PU RT 120 FD 15 LT 60
  FD 15 LT 60 PD
  REPEAT 6(HEX FD 15 LT 60)
  END
```

```
TO HEX12
  PU RT 120 FD 15 LT 60
  FD 15 LT 60 PD
  REPEAT 6(HEX FD 15 LT 60
  HEX FD 15 RT 60 FD 15 LT 60)
  END
```

} new procedure

Depending upon the child you are working with (age, ability, interest, or whatever), this development of HEX

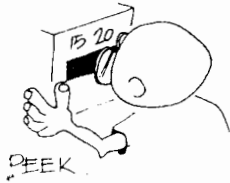
procedures may continue. By following the previous procedures with similar logic, can you write another HEX procedure to circle the outside 12 hexagons? How many small hexagons will be needed? If you draw it on a Hex-grid as we have, you will see that 18 new hexagons are needed.

Study the figure, and see if you can come up with the necessary additions. If you want to go still farther, you will have to shorten the sides of the hexagons. How many hexagons would be needed in the next ring? Let's see . . . first one, then six, then 12, then 18, then . . . ?

Now that you have seen the beginning of drawing HEX RINGS, send me a general Color LOGO procedure for drawing any desired number of HEX RINGS. The number of rings is to be included as a variable in the name of the program so that:

HEXRING 3 would draw three rings of Hex shapes around the center hexagon.

HEXRING 5 would draw five rings of Hex shapes around the center hexagon.



*For all
commercial
advertising in
Australian
RAINBOW
contact
TOT^oadvertising
(075) 39 2003*

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

ToT^oADVRSPT

ARE YOUR WALKING FINGERS GETTING FOOTSORE ?

Tired of typing in those long, but wonderful, programs from each issue of the RAINBOW? Now, you can get RAINBOW ON TAPE and give those tired fingers a rest. With RAINBOW ON TAPE, you'll be able to spend your time enjoying programs instead of just typing...typing...typing them! All you need to do ever again is pop a RAINBOW ON TAPE cassette into your recorder, CLOAD and RUN any one you want.

RAINBOW ON TAPE is available as a single issue. It is the perfect complement for the RAINBOW itself.

LATEST & BACK ISSUES \$12 each

The Best Color Computer Magazine
Offers The Best Tape Service

**Average is 17 Programs
AVAILABLE NOW**

Commenced April '82

ORDER RAINBOW ON TAPE TODAY!

Please note that subscriptions to this Tape Monthly are filled spasmodically due to constant delay in receipt of American Master

DID YOU READ?

RAINBOW Jan '83

- CREATING YOUR OWN ADVENTURE
- INVITATIONS MADE EASY
- MATH PAL
- GET INTO THE HOBBIT OF PLAYING
- LOAD BASIC PROGRAMS 'UP TOP'
- CREATING 3 DIMENSIONAL GRAPHICS

Get 'em while they last — there's still a few copies left

Every issue is **NEW** until you 've **READ IT!**

RAINBOW Feb '83

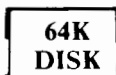
- ORGANISE YOUR TAPES
- STAYIN ALIVE AT OUTPOST FIVE
- HEX PAD CONNECTION
- MANAGEMENT DECISION MODEL
- HAUNTED HOUSE ADVENTURE
- RANDOMIZE RND NUMBERS
- HANG'N AROUND 'HANGMAN'
- CHEMICAL BONDING SIMULATION
- NON-STANDARD INTER FACING

RAINBOW Mar '83

- HALF LIFE OF NUCLEAR DECAY
- TECHNIQUES FOR PLOT-TING SCREEN GRAPHS
- D.O.S. DETACH
- DUMB TERMINAL ROUTINE
- VARIABLES TUTORIAL
- THE WORD - LEARN YOUR GRAMMAR
- DEACTIVATE THE BOMB
- TWO ILLUSIONS

EXPANDING BASIC

*COOKING
With
CoCo*



Colin J.
Stearman

PART II

In which we construct a simple plug-in cartridge programmer for the 2764 8K EPROM.

This month we continue to build the tools needed to enhance the CoCo Disk Operating System (DOS). Last month we developed a means to store the complete BASIC operating system on a special system floppy disk. Now I will describe a simple construction project to build a plug-in programmer for one of the most popular (and hence cheapest!) 8K EPROM's currently available — the 2764. The primary purpose of this project is to allow us to put the modifications into an EPROM which will replace the ROM containing the original DOS. But once built, the programmer can be used to put any code you wish into a 2764.

Design Philosophy

I'm a firm believer in the "KISS" principle I learned many years ago ("Keep It Simple, Stupid!"). So this programmer uses three integrated circuits, a transistor and a few resistors and capacitors. The bulk of the work is done by the driving software. This means there are no timing circuits or other complex logic to worry about. The result is a simple project to build and get working.

Circuit Description

I do not propose to provide a long description of how a 2764 is programmed. In general, it is programmed by presenting the address and data to the chip, then pulsing the program input pin while supplying 21 volts to another.

If you look at the schematic in Figure 1, you will see that the key to the programmer's simplicity lies in the two 6821 peripheral interface adapters (PIA). These are the same chips used inside the CoCo for interfacing with the outside world. These two chips provide the 2764 with all address and data information along with other control lines. The only

other chip is an inverting buffer to decode the address information to the PIAs.

One of the PIA outputs drives a transistor which activates a relay to control the 21-volt source. It's not the most elegant way of doing it, but certainly the simplest. A light emitting diode (LED) tells you when the programmer is programming. The diode around the relay suppresses transients during switching and the other two stop currents flowing to the wrong places. The capacitors are all for power supply filtering. These are not shown in Figure 1 for clarity. Locate a 0.1uF disk capacitor from +5V to 0V at each integrated circuit (polarity is not important) and one 10uF 12V electrolytic capacitor anywhere on the board across these same lines (polarity is important here, the wire labelled "+" goes to the +5V line).

The PIA is a programmable device and its external connections may be programmed as inputs or outputs. This makes it possible for the software to both program the 2764 and then read back the resulting data.

The 21 volt source is easily obtained from three nine-volt batteries and a few other components as shown in Figure 1. This circuit is not built on the board and may not be needed if you already have an adjustable power supply.

Finally, the two sockets shown at U4 and U5 have nothing to do with the programmer itself, but provide a convenient method of putting two programmed 2764s into the CoCo memory map. One socket is wired to fill the address space from \$C000 to \$DFFF and the other from \$E000 to \$FFFF. (The last 256 bytes are not accessible in the latter because the addresses \$FF00 to \$FFFF are used internally as system input/output and vector addresses.)

Construction Hints

Radio Shack sells a printed circuit breadboard with the correct 40-pin edge connect for the CoCo expansion port. Check the parts list in Figure 1 for the number. This board is ideal for the project. The photograph shows the construction method I used. The components were conveniently laid out and then hooked up using a combination of the copper tracks on the board and solid hook-up wire. Maybe it's not the most elegant, but it's serviceable and functional. You could lay out and etch a custom printed circuit board and make a more professional job if you wished.

Take your time during the construction! The finished project will be plugged into your precious CoCo and could cause some nasty problems if you make an error. Use a meter or continuity tester to make sure you have wired correctly and that there are no shorts. The most likely cause of damage is having the power supply voltages coming out of CoCo going to the wrong places. The only internal supply used is the five-volt source from pin nine of the connector so check this line carefully. Pins one and two, which supply -12 volts and +12 volts are not used, so make sure they do not go anywhere on your board. Check Figure 2 for the edge connector pin numbering.

Source For Components

Those parts available from Radio Shack have been listed in Figure 1. The PIAs and 2764s are not available from them, nor is the Zero Insertion Force (ZIF) socket. The ZIF socket is not essential but is a good idea as it saves wear and tear on the 2764s. Most mail order houses can supply these components and I can recommend ACTIVE Electronics (800-343-0874) in Westboro, Mass. as a reputable firm. When ordering the 2764 ask for the 2764-3 which has an access time of 300nS. This is fast enough to work in the familiar "speed-up mode" that some CoCo programs use.

An enclosure for the board can be obtained from The Microworks or Colorware who both advertise in RAINBOW.

The only other major item you might want to consider is an EPROM eraser. EPROMs are erased by exposure to ultraviolet light and can usually be programmed and erased many times. It is probable that you will wish to erase an EPROM you have programmed at some point and will need an eraser. If you live in the Sun Belt you might try leaving them outside in the sun for a week or two. But if you live in the north like me and forget what the sun looks like you'll have to buy an eraser. Hobby models are available for around \$60 (also from ACTIVE). They do the job in about 15 minutes and can erase 15 chips at once. UV is dangerous to the eyes and skin and these inexpensive models have no safety interlocks, so if you get one treat it with respect and NEVER look into the lighted lamp.

Software

Listing 1 shows the source code for the EPROM programmer. It is fully position independent and is an ideal candidate for loading into an EPROM. I put such a programmed EPROM into one of the sockets on the board so that the cartridge had both the hardware and software ready to go.

The program is menu driven and provides a variety of functions. Menu selection one will verify that all locations in the EPROM are erased. A colored bar shrinks as the EPROM is checked and if fully erased, this is reported. If not, the first unerased memory location is reported and the checking process stops. An EPROM is fully erased when all memory bits are a one. The programming process can only convert 1s to 0s, not the reverse. You can program a partially erased EPROM however, as long as the memory locations you do wish to program are erased.

Menu item two allows the data stored in any section of CoCo's memory to be programmed into the EPROM. This does not have to be the whole 8K and can be as little as one byte. All memory addresses are entered as hexadecimal and the EPROM memory locations are numbered from \$0000 to \$1FFF. As the programming proceeds, the cell being programmed is indicated and also automatically verified. If a cell does not return the same data as was programmed in it, the address is shown and a "BAD EPROM" message issued. If it is just not erased, this will be reported as such. In either case the programming stops.

The third menu item allows the contents of the EPROM to be dumped as a hexadecimal and ASCII character table. This is useful for inspecting the contents of the EPROM. The EPROM start and stop addresses are supplied and the output can be directed to the screen or printer. If the screen is chosen, the output will pause and wait for any key-press after each screen is filled. In either case the BREAK key will stop output and return to the request for dump range. Pressing the ENTER key for this returns to the main menu.

Menu item four permits individual inspection and programming of EPROM memory locations. The up and down arrows scan through consecutive memory locations displaying their contents. If a new value is entered an attempt is made to program that cell. This is done by pressing the 'P' key at the appropriate address and then entering the data. Sometimes it is possible to correct minor errors in a programmed EPROM this way. A new address may be selected by pressing 'N' and entering the desired address. 'X' will return to the main menu.

The fifth menu item will return the load start and end addresses of a cassette binary file, along with the execution

address. This is used to find out where a binary file from tape went in memory so that it can be transferred to the EPROM. This display does not take into account any load offset you might have used in the CLOAD command.

Menu item six simply returns you back to BASIC.

Assembling the Program

As I mentioned in the previous installment, I use MAC by Computerware as my assembler. However, many of you may have EDTASM+ or some other brand. Generally they are compatible, but there are some differences. For example, MAC allows binary numbers in the operand field. These are preceded by a percent sign. For other assemblers simply figure out what the number is in hexadecimal and enter it with a dollar sign in front instead.

MAC also has an FCS (Form Constant String) mnemonic. This is similar to FCC (Form Constant Characters), but allows hexadecimal codes to be imbedded in the string by enclosing them in angle brackets. Also it automatically adds a zero byte at the end of the string. Every FCS instruction can be replaced by a series of FCC and FCB (Form Constant Byte) mnemonics. For example, this line:

```
·FCS /<OD> Sample program <OD>Enter?/
```

would become:

```
FCB $0D return
FCC /Sample program/
FCB $0D return
FCC /Enter?/
FCB 0 terminating zero byte
```

You may also see mnemonics OPT, NAM and TTL in the listings. These are just directives to MAC and can be omitted.

Once you have entered the source code and it assembles without error, save a copy of the machine code binary file to a cassette. This will be needed to first "fire up" the programmer as the disk system will be disconnected.

Testing the Project

After you have thoroughly checked the circuit board for errors there is nothing else but to plug it in and try it. If you have a meter you might monitor a five-volt point somewhere on the board before powering up. Owners of the Multi-Pak Interface should plug the programmer into slot one and select this on the front switch. If you do not own one, remove the disk controller and plug the programmer directly into the computer.

Now cross your fingers and power up. (If you have the Multi-pak, just power that up and verify the five-volt line with your meter first.) Now power up CoCo. If the screen does not clear and the copyright notice does not appear in the normal time, power down immediately and further check your construction.

If everything is alright so far, CLOADM and EXEC the programmer driver software from your cassette. The title and menu should appear. If not, recheck your typing of the source code.

Without a 2764 in the ZIF socket, select menu item one. If the programmer is working you will see a purple horizontal bar which shrinks from the right as each of the 1024 bytes are verified. (If there is no 2764 chip in the socket, it looks like a fully erased chip to the programmer.) When all 8K have been checked the 2764 will be declared fully erased. Pressing

ENTER will return you to the menu. If you get this far, things are looking pretty good.

Now try menu item five and verify that the start, end and execute addresses of the programmer software just loaded from cassette are returned. Make a note of these numbers.

Next is a dry run at programming. Connect the 21.5 volt external source using clip leads. Still without a 2764 in the ZIF socket, select menu item two. For the start and end address in RAM use the start and end address from the previous steps. For the EPROM target address use 0. As soon as you enter the zero, the program will announce the attempt to program the EPROM at address zero and then indicates you have a bad EPROM at the location. As you have not plugged in an EPROM, this is to be expected. You should have heard the relay actuate briefly and the LED may have flashed on momentarily. Press ENTER twice to return to the main menu. Things are still looking good.

Now plug in an erased 2764 into the ZIF socket. Use menu item one to verify it is erased. If so, return to menu item two and reenter the RAM start and end values as before. Target the code to begin at EPROM address \$0000. When you press ENTER the relay should "click" in and the LED come on. As each address is programmed its EPROM address is shown on the screen. Remember that data for each address is being verified as it goes along, so there is little likelihood of wrong data being programmed in, unless it was wrong in the first place. It takes 50mS to program each location, so an entire 8K takes a little over six minutes. This is not a limitation of the software but rather a requirement of the EPROM. The programmer software is not 8K long so will not take that long.

When the last byte of the block has been programmed, the addresses of the range of bytes programmed is displayed. Pressing ENTER once would allow you to program another part of this EPROM or another one. (You could put some other program in the unused portion of the EPROM just programmed, if you wish.) Pressing ENTER again returns you to the main menu.

It would be a good idea to dump the data just programmed to double check it. This is done with menu item three. Dump the range programmed and spot check the data for errors. It should be alright.

Now power down the system and remove the 2764 from the ZIF socket and put it into the spare socket on the programmer labelled \$E000 - \$FEFF. Power up again and type in EXEC&HE000. The EPROM programmer software should immediately start up.

If you got this far without problems I think you can breathe a sigh of relief. . . the unit seems to be working fine. If not, check and double check everything and after all else fails, drop me a line and a SASE and I'll try to figure out what went wrong.

Using the Programmer with the Disk

It is a good idea to get a copy of the unmodified Disk BASIC on a cassette and if you have the Multi-Pak to also put it into an erased EPROM. The latter is the case because the Multi-Pak Interface allows you to use the programmer with the disk system. Put the disk controller in slot four and the programmer in slot one. Initially select slot four.

To save disk BASIC to cassette, with the disk system running and a blank cassette in the tape drive, type: CSAVEM"DBASIC",&HC000,&HDFFF,&HA027.

If you have the Multi-Pak interface, the next few steps will put Disk BASIC into an EPROM so that it can be put into the other socket on the programmer. If you don't have this

interface there is little point in doing this as the CoCo cannot have the programmer and disk controller available to it at the same time. However, Disk BASIC on a cassette will come in useful later.

For those with the interface, continue by powering down and selecting slot one. Then power up to Extended Color BASIC. Type in the following commands:

```
CLEAR 200,&H3FFF
CLOADM"DBASIC",&H4000-&HC000+65536
EXEC &HE000
```

Assuming you have the programmer software in an EPROM in the socket as \$E000, it should start up and you can program a fully erased 2764 with the data stored in RAM at \$4000 though \$5FFF. This, of course, is Disk BASIC.

When the EPROM is programmed, power down and put the EPROM in the other socket on the programmer (\$C000 to \$EFFF, the normal addresses for Disk BASIC). With the selector still in position one, power up the system. You should get the normal Disk Extended Color BASIC banner. You are now running Disk BASIC from the EPROM. However, it will not work properly because the secondary chip select signal is going to slot one (because of the position of the switch) and it needs to go to the controller in slot four. This is accomplished by entering POKE 65407,3. Now the system will act normally until you press Reset. Then you'll have to do this POKE again.

You can now load machine code files from disk and then activate the programmer code. This is done by redirecting the secondary chip select to slot one with a POKE 65407,0, then EXEC&HE000 to start up the programmer code. Menu item six returns to Disk BASIC where the secondary chip select can once again be directed to slot four.

Wrapping It Up

If this was your first construction project and you got here with no problem, congratulations — you are now a qualified "hardware hacker." For those "old hands" this should have made a simple but rewarding project.

We now have all the necessary tools to enhance the DOS, so next month we will start that in earnest by revising some commands and maybe adding one or two new ones. Until then!

Listing 1:

EPROM.MAC COMPUTERWARE MACRO ASSEMBLER PAGE 1
2764 EPROM PROGRAMMER By C.J.STEARMAN (C)1984

```
0001 *****
0002 *   EPROM  PROGRAMMER   *
0003 *           BY          *
0004 *   COLIN  STEARMAN    *
0005 *
0006 *   (C)1984 C.J.Stearman *
0007 *****
0008 *
0009 *   THIS IS POSITION INDEPENDENT
0010 *
0011 *
0012 *
0013 *****
0014   ORG   $E000
0015 *
0016 *
0017 *****
0018 *   SOME EQUATES
0019 *
0020 CLEAR EQU   $A928   BASIC CLEAR SCREEN ROUTINE
0021 BUFFER EQU  $1DA   USES THE CASSETTE BUFFER
```

```

#008      #022 NUMK EQU 8          NUMBER OF K IN EPROM
1FFF      #023 TOPADD EQU (NUMK*1024)-1 TOP EPROM ADDRESS
          #024 *
          #025 *
          #026 *****
          #027 *
          #028 * MAINLINE OF PROGRAM
          #029 *
          #030 *
#000 170130 #031 EPROM LBSR INIT      SET UP THE PIAS
          #032 * NORMAL EPROM MODE IS TO READ THE EPROM
#E03 80A920 #033 MENU JSR CLEAR    CLEAR SCREEN
#E06 30B00A3 #034 LEAX MENU,PCR    POINT TO MENU TEXT
#E0A 17094E #035 LBSR OUTST0      OUTPUT THE MENU
          #036 *
          #037 *GET RESPONSE
#E0D 1709FA #038 LBSR INSTR0      GET RESPONSE INTO BUFFER
          #039 * FIRST SEE IF ONLY 1 CHARACTER ENTERED
#E10 B601DC #040 LDA BUFFER+2      SHOULD BE ZERO
#E13 26EE   #041 BNE MENU        IT WASNT
#E15 B601DA #042 LDA BUFFER      GET FIRST CHARACTER IN BUFFER
#E18 B131   #043 CMPA #'1        VERIFY ERASE
#E1A 2605   #044 BNE .COPY
#E1C 17021F #045 LBSR ERASE
#E1F 20E2   #046 BRA MENU
#E21 B132   #047 .COPY CMPA #'2    COPY RAM
#E23 2605   #048 BNE .DUMP
#E25 170310 #049 LBSR COPY
#E28 20D9   #050 BRA MENU
#E2A B133   #051 .DUMP CMPA #'3    DUMP EPROM
#E2C 2605   #052 BNE .CELL
#E2E 170543 #053 LBSR DUMP
#E31 20D0   #054 BRA MENU
#E33 B134   #055 .CELL CMPA #'4    INDIVIDUAL CELL PROGRAM
#E35 2605   #056 BNE .FILE
#E37 1706C8 #057 LBSR CELL
#E3A 20C7   #058 BRA MENU
#E3C B135   #059 .FILE CMPA #'5     CASSETTE FILE DATA RETURN
#E3E 2605   #060 BNE .BASIC
#E40 170810 #061 LBSR CFILE
#E43 20BE   #062 BRA MENU
#E45 B136   #063 .BASIC CMPA #'6    IS IT EXIT TO BASIC?
#E47 26BA   #064 BNE MENU        NO SO DO MENU ABAIN
#E49 80A920 #065 JSR CLEAR    BEFORE GOING TO BASIC
#E4C 39     #066 RTS        EXIT FOR CHECK SO FAR
          #067 *
#E4D 45     #060 MENU FCC /E P R O M   P R O G R A M M E R /
#E4E 2050205220
#E53 4F20402020
#E58 2020502052
#E5D 204F204720
#E62 522041204D
#E67 204D204520
#E6C 52
#E6D 3D     #069 FCC /*****
#E6E 3D3D3D3D3D
#E73 3D3D3D3D3D
#E78 3D3D3D3D3D
#E7D 3D3D3D3D3D
#E82 3D3D3D3D3D
#E87 3D3D3D3D3D
#E8C 3D
#E8D 00D0   #070 FDB 00D0      TWO (CR)
#E8F 20     #071 FCC / 1 - VERIFY ERASURE/
#E90 2031202D20
#E95 5645524946
#E9A 5920455241
#E9F 53555245
#EA3 00     #072 FCB 00D
#EA4 20     #073 FCC / 2 - PROGRAM EPROM FROM MEMORY/
#EA5 2032202D20
#EA8 50524F4752
#EAF 414D204550
#EB4 524F4D2046
#EB9 524F4D204D
#EBE 454D4F5259
#EC3 00     #074 FCB 00D
#EC4 20     #075 FCC / 3 - DUMP EPROM CONTENTS/
#EC5 2033202D20
#ECA 44554D5020
#ECF 4550524F4D
#ED4 20434F4E54
#ED9 454E5453
#EDD 00     #076 FCB 00D
#EDE 20     #077 FCC / 4 - PROGRAM INDIVIDUAL CELLS/
#EDF 2034202D20
#EE4 50524F4752
#EE9 414D20494E
#EEE 4449564944
#EF3 55414C2043
#EF8 454C4C53
#EFC 00     #078 FCB 00D
#EFD 20     #079 FCC / 5 - CASSETTE FILE DATA/
#EFE 2035202D20
#EF3 4341335345
#EF8 5454452046
#EFD 494C452044
#F12 415441
#F15 00     #080 FCB 00D
#F16 20     #081 FCC / 6 - RETURN TO BASIC/
#F17 2036202D20
#F1C 5245545552
#F21 4E20544F20
#F26 4241534943
#F28 00D0   #082 FDB 00D0D
#F2D 20     #083 FCC / SELECTION? /
#F2E 2053454C45
#F33 4354494F4E
#F38 3F20
#F3A 00     #084 FCB 0          MESSAGE TERMINATOR
          #085 *
          #086 *
          #087 *****
          #088 *****
          #089 * EPROM ACCESS ROUTINES
          #090 *****
          #091 *
          #092 *
#F41        #093 CONREG EQU 0FF41    LOWEST CONTROL REGISTER
#F44        #094 LOWADD EQU 0FF44    LOW ADDRESS OUTPUT
#F46        #095 HIADD EQU 0FF46    HIGH ADDRESS OUTPUT
#F40        #096 DATARG EQU 0FF40    DATA REGISTER
#F42        #097 CLINES EQU 0FF42    CONTROL LINES REGISTER
#F43        #098 VOLTS EQU 0FF43     RELAY CONTROL REGISTER
          #099 *
          #100 *
#101 *****
#102 * INITIALIZING ROUTINE
#103 *
#F3D 4F     #104 INIT CLRA          EXPOSE ALL THE DDRS
#F3C 8D1F   #105 BSR DDRSET
          #106 *
          #107 * NOW ALL DATA DIRECTION REGISTERS ARE EXPOSED
          #108 *
#F3E C6FF   #109 LDB 00FF          SET ALL ADDRESS LINES TO OUTPUTS
#F40 F7FF44 #110 STB LOWADD
#F43 F7FF46 #111 STB HIADD
          #112 *
#F46 C607   #113 LDB 07          SE CONTROL LINES TO OUTPUTS
#F48 F7FF42 #114 STB CLINES
          #115 *
#F4B 7FFF40 #116 CLR DATARG        TO MAKE IN INPUTS
          #117 *
#F4E 0604   #118 LDA 04          RESET THE CONTROL REGISTERS
#F50 0D0B   #119 BSR DDRSET        TO OUTPUTS
          #120 *
#F52 0634   #121 LDA 0034        SET CONTROL REG FOR RELAY OUTPUT
#F54 07FF43 #122 STA VOLTS        ENABLES CB2 AS OUTPUT AT ZERO
          #123 *
#F57 0601   #124 LDA 01          SET UP CONTROL LINES FOR READ
#F59 07FF42 #125 STA CLINES        DE,CS=0 PGM=1
          #126 *
          #127 RTS
          #128 *
          #129 **
#130 ***** SUBROUTINES *****
          #131 * SET CONTROL REGISTERS TO CONTENTS OF A
          #132 DDRSET LDB 04          # OF CONTROL REGISTERS
#F5D C604   #133 LDX 0CONREG        POINT I TO CONTROL REGISTERS
#F5F BEFF41 #134 CLRREG STA ,I++    CLEAR AND DOUBLE INCREMENT
#F62 A781   #135 DEC0          DECREASE COUNTER
#F64 5A     #136 BNE CLRREG        DO NEXT REGISTER
#F65 26FB   #137 RTS
#F67 39
          #138 *****
          #139 *
          #140 *
          #141 *****
          #142 * PROGRAM EPROM ROUTINE
          #143 *****
          #144 *
          #145 * THIS PROGRAMS THE PROM FROM DATA STARTING
          #146 * AT ADDRESS IN "START", FOR THE NUMBER OF
          #147 * BYTES IN "COUNT", AT EPROM ADDRESS "TARGET"
          #148 * THESE LOCATIONS ARE RESERVED IN THIS ROUTINE
          #149 * START ENDS UP WITH LAST ADDRESS DATA WAS
          #150 * TAKEN FROM RAM. TARGET HAS LAST ADDRESS
          #151 * WRITTEN TO IN EPROM.
          #152 * B HAS ERROR CODE

```

#153 *	1= NCT ERASED			#FFA C602	#242	LDB #2	VERIFY ERROR CODE
#154 *	2= BAD EPROM LOCATION			#FFB 2010	#243	BRA PEIIT	
#155 *	0= NO PROBLEM				#244 *		
#156 *	*****				#245 *	END PROGRAMMING LOOP	
#157 *					#246 *	*****	
#1D1	#158 START EQU #1D1	USED CASSETTE NAME AREA		#FFA AD9FA000	#247	VERTOK JSR [POLCAT]	BREAK PRESSED?
#1D3-	#159 COUNT EQU START+2			#FFE 2705	#248	BEQ DOWNCT	NO SO DECREASE COUNT
#1D5	#160 TARGET=EQU START+4			1000 5F	#249	CLRB	READY FOR RETURN CODE
	#161 *			1001 8103	#250	CMPA #3	BREAK VALUE
#F58 B6FF43	#162 PROGRM LDA VOLTS	SET REG VALUE		1003 2705	#251	BEQ PEIIT	YES SO EXIT
#F65 B808	#163 ORA #200001000	SET BIT 8 TO APPLY 21V		1005 301F	#252	DOWNCT LEAX -1,X	REDUCE COUNT
#F6D B7FF43	#164 STA VOLTS			1007 2608	#253	BNE PLOOP	NOT DONE YET
	#165 *	21V IS NOW APPLIED		1009 5F	#254	CLRB	NO ERROR CODE
	#166 *	WAIT A WHILE FOR RELAY TO CLOSE		100A 335F	#255	PEIIT LEAU -1,U	DECREASE TO LAST LOADED ADDRESS
#F70 BEFFFF	#167 LDX #0FFFF			100C FF01D5	#256	STU TARGET	
#F73 301F	#168 RLYDLY LEAX -1,X	DECREMENT X		100F 313F	#257	LEAY -1,Y	DO SAME FOR RAM ADDRESS
#F75 26FC	#169 BNE RLYDLY			1011 10BF01D1	#258	STY START	SAVE LAST RAM ADDRESS
	#170 *			1013 B6FF43	#259	PREIIT LDA VOLTS	GET VOLTS REGISTER
	#171 *	PRESERVE Y AND U REGISTERS		1018 84F7	#260	ANDA #111110111	TURN OFF 21V
#F77 3460	#172 PSHS U,Y			101A B7FF43	#261	STA VOLTS	
	#173 *			101D 35E0	#262	PULS U,Y,PC	RECOVER REGISTERS & RETURN
	#174 *			#263 *			
#F79 10BE01D1	#175 LDY START	POINT Y TO RAM START		#264	*****		
#F7D FE01D5	#176 LDU TARGET	POINT U TO TARGET		#265 *			
#F80 5F	#177 CLRB			#266 *	THIS PULSES THE PGM LINE LOW FOR 50MS		
#F81 BE01D3	#178 LDX COUNT	GET BYTE COUNT		#267 *			
#F8A 1027009D	#179 LBEQ PEIIT	ALL DONE PROGRAMMING		101F B6FF42	#268	PULSE LDA CLINES	GET LINES
	#180 *			1022 84FE	#269	ANDA #111111110	MAGE PGM LOW
	#181 *	MOVE CURSOR AHEAD 4		1024 B7FF42	#270	STA CLINES	
#F88 FC0000	#182 LDD CURLOC			#271 *			
#F8B C30004	#183 ADDD #4			1027 3410	#272	PSHS X	FOR DELAY COUNT
#F8E FD0000	#184 STD CURLOC			1029 1A50	#273	ORCC #201010000	PREVENT INTERRUPTS
	#185 *	*****		102B BE1600	#274	LDX #1600	FOR 50 MS
	#186 *	PROGRAMMING LOOP		102E 301F	#275	DLOOP LEAX -1,X	REDUCE COUNT
#F91 1F30	#187 PLOOP TFR U,D	SET UP EPROM ADDRESS		1030 26FC	#276	BNE DLOOP	KEEP LOOPING
#F93 3341	#188 LEAU 1,U	INCREMENT EPROM ADDRESS		1032 1CAF	#277	ANDCC #110101111	ALLOW INTERRUPTS
#F95 B7FF46	#189 STA HIADD			#278 *			
#F98 F7FF44	#190 STR LOWADD			1034 B6FF42	#279	LDA CLINES	GET LINES
	#191 *			1037 B801	#280	ORA #200000001	SET PGM HI
	#192 *	DISPLAY WORKING ADDRESS		1039 B7FF42	#281	STA CLINES	
#F98 3436	#193 PSHS Y,X,I,"			103C 3590	#282	PULS X,PC	RECOVER X AND RETURN
#F9D FC01D3	#194 LDD COUNT	DONT IF IT IS 1		#283	*****		
#FA0 10B30001	#195 CMPD #1			#284	*****		
#FA4 270F	#196 BEQ NODISP	NO DISPLAY		#285 *	VERIFY ROUTINE *		
#FA6 ECE4	#197 LDD ,S	RECOVER VALUE IN D		#286	*****		
#FAB BE0000	#198 LDX CURLOC	MOVE CURSOR BACK 4		#287 *			
#FAB 301C	#199 LEAX -4,X			#288 *	THIS VERIFIES ERASURE OF THE EPROM		
#FAD BF0000	#200 STX CURLOC			#289 *	PROVIDES A GO/NOGO RESPONSE		
#FB0 1F01	#201 TFR D,X	MOVE VALUE TO X		#290 *	OF ERASURE OF ENTIRE EPROM		
#FB2 170016	#202 LBSR HEXOUT	DISPLAY IT		#291 *			
#FB5 3536	#203 NODISP PULS Y,X,D	RECOVER VALUES		103E BDA92B	#292	ERASE JSR CLEAR	CLEAR SCREEN
	#204 *	GET DATA TO BE LOADED INTO REG B		1041 3420	#293	PSHS Y	PRESERVE REGISTER Y
#FB7 E6A0	#205 LDB ,Y+	AND INCREMENT ADDRESS		1043 309D007B	#294	LEAX ERAMSG,PCR	PUT UP TITLE
	#206 *			1047 170711	#295	LBSR OUTST0	
#FB9 B6FF40	#207 LDA DATARG	GET DATA AT THIS ADDRESS		#296 *			
#FBC 81FF	#208 CMPA #0FF	SHOULD BE THIS		#297 *	PUT UP PROGRESS MONITOR		
#FBE 2704	#209 BEQ EMPTY			104A BE0000	#298	LDX CURLOC	GET CURSOR LOCATION
#FC0 C601	#210 LDB #1	NOT ERASED CODE		104D 3000	#299	LEAX NUMK,X	MOVE OVER NUMBER OF X IN EPROM
#FC2 2046	#211 BRA PEIIT			104F BF0000	#300	STX CURLOC	AND SAVE IT
	#212 *			1052 CC0BF8	#301	LDD #0BF8	2 RED SQUARES
#FC4 B6FF42	#213 EMPTY LDA CLINES	GET CONTROL LINES		1055 10BE0000	#302	LDY #NUMK	COUNTER
#FC7 B802	#214 ORA #200000010	RAISE OE		1059 E0B1	#303	PUTMON STD ,I++	STORE ON SCREEN
#FC9 B7FF42	#215 STA CLINES			105B 313F	#304	LEAY -1,Y	DECREASE COUNT
	#216 *			105D 26FA	#305	BNE PUTMON	
#FCC 7FFF41	#217 CLR CONREG	MAKE DATA LINES OUTPUTS		#306 *			
#FCF B6FF	#218 LDA #0FF			105F 10BE0000	#307	LDY #0	START ADDRESS
#FD1 B7FF40	#219 STA DATARG			1063 AD9FA000	#308	VLOOP JSR [POLCAT]	TEST FOR BREAK
#FD4 B604	#220 LDA #4	RESET TO OUTPUT REG		1067 2706	#309	BEQ NOBRK	HE KEY PRESSED
#FD6 B7FF41	#221 STA CONREG			1069 8103	#310	CMPA #3	BREAK?
	#222 *			106B 2602	#311	BNE NOBRK	
#FD9 F7FF40	#223 *	SAVE DATA IN B IN EPROM		106D 35A0	#312	PULS Y,PC	RETURN
	#224	STB DATARG	PUT ON DATA LINES	#313 *			
#FDC 8D41	#226	BSP PULSE	THE PGM LINE LOW	106F 1F20	#314	NOBRK TFR Y,D	
	#227 *			1071 B7FF46	#315	STA HIADD	SET UP ADDRESS ON PIA
	#228 *	NOW VERIFY		1074 F7FF44	#316	STB LOWADD	
#FDE 7FFF41	#229	CLR CONREG	MAKE DATA LINE INPUTS	1077 B6FF40	#317	LDA DATARG	GET DATA
#FE1 7FFF40	#230	CLR DATARG		107A 81FF	#318	CMPA #0FF	IS IT ERASED
#FE4 8604	#231	LDA #4		107C 261C	#319	BNE NOTMTY	NOT ERASED
#FE6 B7FF41	#232	STA CONREG		107E 3121	#320	LEAY 1,Y	INCREASE
	#233 *			#321 *	ADJUST PROGRESS COUNTER IF NEEDED		
	#234 *	ENABLE CHIP		1080 1F20	#322	TFR Y,D	
#FE9 B6FF42	#235	LDA CLINES		1082 50	#323	TSTB	IF NOT ZERO CONTINUE
#FEC 84FD	#236	ANDA #211111101	OE LOW	1083 2609	#324	BNE DONEYT	DOME YET
#FEE B7FF42	#237	STA CLINES		1085 B403	#325	ANDA #200000011	SEEE IF THESE ARE ZERO
	#238 *			1087 2605	#326	BNE DONEYT	NO SO S:1P
#FF1 F1FF40	#239 *	NOW COMPARE DATA ON DATARG WITH CONTENTS AT Y		1089 C0BF8F	#327	LDD #0BF8F	GREEN SQUARES
#FF4 2704	#240	CMPB DATARG	DATA WAS LEFT IN B FROM LOAD	108C E0B3	#328	STD ,--X	DECREASE MONITOR FROM RIGHT
	#241	BEQ VERTOK	TT WAS THE SAME	#329 *			
				109E 10BC2000	#330	DONEYT CMY #TOPADD+1	ADDRESS LIMIT

```

1092 26CF 0331 BNE VLOOP
1094 308D0079 0332 LEAX GOOD,PC1 IS FULLY ERASED
1098 2019 0333 BRA VEIIT
0334 *
109A FC0088 0335 NOTHTY LDD CURLC
109D C30020 0336 ADDD #32 MOVE TO NEXT LINE
10A0 F00088 0337 STD CURLC
10A3 308D0059 0338 LEAX ADDNMT,PCR GET ADDRESS MESSAGE
10A7 1706B1 0339 LBSR OUTST#
10AA 1F21 0340 TFR Y,I
10AC 17071C 0341 LBSR HEIOUT PUT LAST ADDRESS UP
10AF 308D0058 0342 LEAX BAD,PCR
10B3 1706A5 0343 VEIIT LBSR OUTST#
10B6 308D0063 0344 LEAX VERIFY,PCR
10BA 17069E 0345 LBSR OUTST#
0346 *
10BD 1706AA 0347 LBSR INSTR# GET KEYBOARD RESPONSE
10C0 35A0 0348 PULS Y,PC RECOVER Y AND RETURN
0349 *
10C2 20 0350 ERASB FCC / EPROM ERASURE VERIFICATION/
10C3 2020455452
10C8 4F4D204552
10CD 415355245
10D2 2056453249
10D7 4649434154
10DC 494F4E
10DF 00 0351 FCB #00 (CR)
10E0 20 0352 FCS / *****(<0D><0D>)/
10E1 20203D3D3D
10E6 3D3D3D3D3D
10E8 3D3D3D3D3D
10F0 3D3D3D3D3D
10F5 3D3D3D3D3D
10FA 3D3D3D00D
10FF 00
0353 *
1100 00 0354 ADDNMT FCS /(<0D><0D>)ADDRESS /
1101 0041444452
1106 4553532000
110B 20 0355 BAD FCS / NOT /
110C 4E4F542000
1111 00 0356 GOOD FCS /(<0D><0D>) FULLY /
1112 0D20202046
1117 554C4C5920
111C 00
111D 45 0357 VERIFY FCS /ERASED<0D><0D>PRESS *ENTER* TO CONTINUE /
111E 5241534544
1123 0D0D505245
1128 5353202245
112D 4E54455222
1132 20544F2043
1137 4F4E54494E
113C 55452000
0358 *****
0359 *****
0360 * EPROM PROGRAMMING *
0361 *****
0362 *
0363 *
0364 * THIS GETS START ADDRESS AND END ADDRESS IN
0365 * RAM AND START TARGET ADDRESS IN EPROM
0366 * CHECKS FOR ERRORS THEN TRANSFERS DATA
0367 *****
0368 *
1140 0DA920 0369 COPY JSR CLEAR SCREEN
1143 308D00CC 0370 LEAX CPYTTL,PCR GET HEADER
1147 170611 0371 LBSR OUTST# PUT IT UP
0372 *
0373 * GET START ADDRESS IF NULL THEN RETURN
114A 308D0171 0374 LEAX STRTIT,PCR
114E 17066E 0375 LBSR INPUT#
0376 * DID WE GET A NULL
1151 B601DA 0377 LDA BUFFER GET FIRST BYTE
1154 8100 0378 CMPA #00 IS IT CR?
1156 2601 0379 BNE BETST
1158 39 0380 CEIIT RTS
0381 *
1159 170604 0382 BETST LBSR HEIINT CONVERT INTO REG X
115C BF01D1 0383 STZ START GET START ADDRESS
115F 5D 0384 TSTB CHECK FOR ERRORS
1160 26DE 0385 BNE COPY
0386 *
1162 308D0170 0387 LEAX ENDS6,PCR GET ENDING RAM ADDRESS
1166 170656 0388 LBSR INPUT# GET ENDING ADDRESS
1169 B601DA 0389 LDA BUFFER TEST FOR NULL
116C 8100 0390 CMPA #00 IS IT CR?
116E 27E0 0391 BEQ CEIIT
0392 *
1170 1705ED 0393 LBSR HEIINT GET VALUE ENTERED
1173 5D 0394 TSTB DID WE GET ERROR?
1174 26CA 0395 BNE COPY RESTART
0396 * X NOW HAS ENDING
1176 1F10 0397 TFR X,D PUT INTO ACC D
1178 B301D1 0398 SUBD START FIND DIFFERENCE
117D 2542 0399 BLD DERROR DATA ERROR MESSAGE
117D C30001 0400 ADDD #1 TO MAKE IT ACTUAL COUNT
1180 F001D3 0401 STD COUNT SAVE IT
0402 *
0403 *NOW GET TARGET ADDRESS
1183 308D0166 0404 LEAX TBTHS6,PCR
1187 170635 0405 LBSR INPUT#
118A B601DA 0406 LDA BUFFER NULL ENTRY?
118D 8100 0407 CMPA #00 (CR)
118F 27C7 0408 BEQ CEIIT SO EIIT ROUTINE
0409 *
1191 1705CC 0410 LBSR HEIINT GET VALUE IN X
1194 5D 0411 TSTB ERROR?
1195 26A9 0412 BNE COPY RESTART
0413 *
0414 * X NOW HAS TARGET ADDRESS
1197 BF01D5 0415 STX TARGET
119A 8C1FFF 0416 CMPX #TOPADD HIGHEST ALLOWED VALUE
119D 2229 0417 BHI TOOH1 GO TO ERROR MESSAGE
119F 3410 0418 PSHS X PUT TARGET ONTO STACK
11A1 CC2000 0419 LDD #NUMK#1024 EPROM SIZE
11A4 A3E1 0420 SUBD ,B++ SUBTRACT TARGET & CLEAN STACK
0421 * D NOW HAS AVAILABLE BYTES ABOVE TARGET
11A6 10B31D3 0422 CMPD COUNT
11AA 2525 0423 BLD NOROOM NOT ENOUGH ROOM
0424 * ALL SEEMS OK DO PROGRAM
0425 * FIRST DISPLAY WORKING ADDRESS TEST
11AC 308D01AC 0426 LEAX WRKADD,PCR
11B0 1705A8 0427 LBSR OUTST#
0428 *
11B3 17FD02 0429 LBSR PROORM
0430 *
11B6 5D 0431 TSTB FOR ERROR CODE
11B7 2729 0432 BEQ GOODPR GOOD PROGRAM
11B9 C101 0433 CMPD #1 NOT ERASED
11BB 2741 0434 BEQ UNERAS
11BD 204E 0435 BRA BADLOC BAD PROM LOCATION
0436 *****
0437 **
0438 *
11BF 308D0080 0439 DERROR LEAX DIFF,PCR START ABOVE END MSG
11C3 170595 0440 LBSR OUTST#
11C6 2010 0441 BRA .KEY
0442 *
11CB 308D00A0 0443 TOOH1 LEAX HIGH,PCR TARGET TOO HIGH
11CC 17058C 0444 LBSR OUTST#
11CF 2007 0445 BRA .KEY
0446 *
11D1 308D00B2 0447 NOROOM LEAX NROOM,PCR NOT ENOUGH ROOM IN EPROM
11D5 170503 0448 LBSR OUTST#
0449 *
11D8 308D00C7 0450 .KEY LEAX EKEY,PCR WAIT FOR ENTER
11DC 1705E0 0451 LBSR INPUT#
11DF 16FF5E 0452 LBRA COPY
0453 *
11E2 308D011E 0454 GOODPR LEAX GOODP1,PCR GOOD PROGRAM
11E6 170572 0455 LBSR OUTST#
11E9 BE01D1 0456 LDX START GET LAST RAM ADDRESS
11EC 17050C 0457 LBSR HEIOUT OUTPUT IT
11EF 308D012D 0458 LEAX GOODP2,PCR
11F3 170565 0459 LBSR OUTST#
11F6 BE01D5 0460 LDX TARGET GET LAST PROM ADDRESS
11F9 1705CF 0461 LBSR HEIOUT OUTPUT IT
11FC 20DA 0462 BRA .KEY
0463 *
11FE 308D0139 0464 UNERAS LEAX UNERSD,PCR NOT ERASED
1202 170556 0465 .LEAVE LBSR OUTST#
1205 BE01D5 0466 LDX TARGET GET LAST EPROM ADDRESS
1208 1705C0 0467 LBSR HEIOUT OUTPUT IT
1209 20CB 0468 BRA .KEY
0469 *
120D 308D013D 0470 BADLOC LEAX BADPRN,PCR BAD PROM LOCATION
1211 20EF 0471 BRA .LEAVE
0472 *
0473 *****
0474 *
1213 20 0475 CPYTTL FCC / RAM TO EPROM TRANSFER/
1214 2020202052
1219 41020544F
121E 204550524F
1223 4020545241
1228 4E53464552
122D 00 0476 FCS /(<0D>) *****(<0D><0D>)/
122E 2020202020
1233 303D3D3D3D

```

```

123B 303030303D
1230 303030303D
1242 303030303D
1247 30000000
0477 *
124B 00 047B DIFF FCS /(<0D><0D>START HIGHER THAN END ADDRESS<0D>/
1251 0053944152
1251 5420484947
1256 4845522054
125B 48414E2045
1260 4E4204144
1265 4452455353
126A 0000
126C 00 0479 HIGH FCS /(<0D><0D>TARGET ADDRESS TOO HIGH<0D>/
126D 0054415247
1272 4554204144
1277 4452455353
127C 20544F4F20
1281 484947480D
1286 00
1287 00 0480 WROOM FCS /(<0D><0D>NOT ENOUGH ROOM IN EPROM<0D>/
1288 0D4E4F5420
128D 454E4F547
1292 4820524F4F
1297 4D20494E20
129C 4550524F4D
12A1 0000
12A3 00 0481 EKEY FCS /(<0D>PRESS 'ENTER' TO CONTINUE /
12A4 5052455353
12A9 2022454E54
12AE 4552222054
12B3 4F20434F4E
12B8 54494E5545
12B0 2000
12B2 00 0482 STRTXT FCS / RAM START ADDRESS: /
12C0 202052414D
12C5 2033544152
12CA 5420414444
12CF 524553533A
12D4 2000
12D6 00 0483 ENDMSB FCS / RAM END ADDRESS: /
12D7 2020202052
12DC 414D20454E
12E1 4420414444
12E6 524553533A
12E9 2000
12ED 45 0484 TBTNSB FCS /EPROM TARGET ADDRESS: /
12EE 50524F4D20
12F3 5441524745
12FB 5420414444
12FD 524553533A
1302 2000
1304 00 0485 G0DDP1 FCS /(<0D><0D> LAST RAM ADDRESS USED: /
1305 0D20204C41
130A 5354205241
130F 4D20414444
1314 5245535320
1319 555345443A
131E 2000
1320 00 0486 G0DDP2 FCS /(<0D>LAST EPROM ADDRESS USED: /
1321 4C41335420
1326 4550524F4D
132B 2041444452
1330 4553532053
1335 5345443A20
133A 00
133B 00 0487 UNERSD FCS /(<0D><0D>NOT ERASED AT /
133C 0D4E4F5420
1341 4552415345
1346 4420415420
134B 00
134C 00 0488 BADPRM FCS /(<0D><0D>BAD EPROM AT /
134D 0D42414420
1352 4550524F4D
1357 2041542000
135C 00 0489 WRKADD FCS /(<0D><0D>PROGRAMMING EPROM AT /
135D 0D50524F47
1362 52414D4D49
1367 4E47204550
136C 524F4D2041
1371 542000

```

```

0493 *****
0494 *
1374 0DA920 0495 DUMP JSR CLEAR SCREEN
1377 30000121 0496 LEAX DMPTTL,PCR DUMP TITLE
137B 1703DD 0497 LBSR OUTST0
0498 *
137E 300D0140 0499 LEAX DSTRT,PCR GET START ADDRESS
1382 17043A 0500 LBSR INPUT0
0501 * DID WE GET A NULL?
1385 0601DA 0502 LDA BUFFER
138B 810D 0503 CMPA #00D <CR>
138A 2601 0504 BNE DCONT CONTINUE ROUTINE
0505 *
138C 39 0506 RTS RETURN TO MENU
0507 *
138D 1703DD 0508 DCONT LBSR HEXINT INTO X REG
1390 5D 0509 TST0 AN ERROR
1391 26E1 0510 BNE DUMP RESTART IF SO
1393 8C1FFF 0511 CMPX #TOPADD CHECK RANGE
1396 22DC 0512 BHI DUMP RESTART IF OVER
139B 3410 0513 PSHS X PRESERVE START
0514 *
139A 300D013C 0515 LEAX ESTRT,PCR GET END ADDRESS
139E 17041E 0516 LBSR INPUT0
13A1 17038C 0517 LBSR HEXINT INTO X REG
13A4 5D 0518 TST0 FOR ERROR
13A5 2605 0519 BNE RSTART RESTART IF SO
0520 * CHECK FOR OVER RANGE
13A7 8C1FFF 0521 CMPX #TOPADD
13AA 2304 0522 BLS GDUMP RANGE OK
13AC 3262 0523 RSTART LEAS 2,S CLEAN STACK
13AE 20C4 0524 BRA DUMP RESTART
13B0 1F10 0525 GDUMP TFR X,D TO SEE IF START IS AFTER END
13B2 A3E4 0526 SUBD ,S START ON STACK
13B4 20F6 0527 BNI RSTART NOT SO RESTART
0528 *
13B6 3410 0529 PSHS X PRESERVE END ADDRESS
13B8 300D012E 0530 LEAX DEV,PCR WHICH DEVICE?
13BC 170400 0531 LBSR INPUT0 S OR P
13BF 0601DA 0532 LDA BUFFER GET FIRST LETTER
13C2 C604 0533 LDB #4 FOR SCREEN DUMP WIDTH
13C4 8150 0534 CMPA #'P IS IT PRINTER?
13C6 2607 0535 BNE SCR NO SO LEAVE DEVNUM
13C8 06FE 0536 LDA #0-2 PRINTER DEVICE CODE
13CA 07006F 0537 STA DEVNUM
13CD C610 0538 LDB #16 FOR ITEM COUNT
13CF 3530 0539 SCR PULS X,Y X HAS END, Y START
13D1 50 0540 NEGB FOR MASK
13D2 3404 0541 PSHS B SAVE ON STACK
13D4 1F20 0542 TFR Y,D ROUND DOWN START
13D5 E4E4 0543 ANDB ,S ROUNDED DOWN NOW
13D8 1F02 0544 TFR D,Y PUT IT BACK IN Y
13DA 3410 0545 PSHS X SAVE END ON STACK
0546 * SET LINE COUNT FOR SCREEN
13DC 0610 0547 LDA #16 # OF LINES
13DE 070103 0548 STA COUNT
0549 *
13E1 1F21 0550 DMLOOP TFR Y,X OUTPUT ADDRESS
13E3 3020 0551 PSHS Y SAVE Y
13E5 060D 0552 LDA #00D
13E7 AD9FA002 0553 JSR [CHROUT] START NEW LINE
13EB 1703DD 0554 LBSR HEXOUT OUTPUT ADDRESS
13EE 3520 0555 PULS Y RECOVER Y
13F0 C606 0556 LDB #6 SPACES COUNT
13F2 17009B 0557 LBSR SPACES OUTPUT THEM
13F3 1F20 0558 INLOOP TFR Y,D GET START ADDRESS
13F7 07FF46 0559 STA HIADD
13FA F7FF44 0560 STB LOWADD SET UP EPROM ADDRESS
13FD 3121 0561 LEAY 1,Y INCREMENT ADDRESS
0562 **** OUTPUT THE HEX CHARACTERS
13FF F6FF40 0563 LDB DATARG GET FROM EPROM
1402 3420 0564 PSHS Y PRESERVE VALUE
1404 17039B 0565 LBSR HXPAIR PUT IN BUFFER
1407 0E01DA 0566 LDX #BUFFER POINT TO IT
140A 17034E 0567 LBSR OUTST0
140D 3520 0568 PULS Y RECOVER Y
140F C601 0569 LDB #1
1411 0D7D 0570 BSR SPACES
0571
1413 1F20 0572 TFR Y,D RECOVER COUNT IN D
1415 6362 0573 COM 2,S FOR LOOK AT LOWER BITS
1417 E462 0574 ANDB 2,S COUNT MASK
1419 3401 0575 PSHS CC PRESERVE TEST RESULT
141B 6363 0576 COM 3,S PUT IT BASK AS IT WAS

```

```

0490 *****
0491 *****
0492 * DUMPS EPROM CONTENTS TO SCREEN OR PRINTER *

```



```

0732 * CHECK B FOR ERROR CODE B(<>) FOR ERROR
159A 5D 0733 TSTB
159B 1026FF7F 0734 LBNE DISDAT BAD SO DO NOTHING
159F 8C1FFF 0735 CMPI #0PADD MUST NOT BE HIGHER THAN THIS
15A2 22EC 0736 BHI TOOH1B
15A4 8F01D5 0737 STX TARGET
15A7 16FF74 0738 LBRA DISDAT BO DISPLAY IT
0739 *****
15AA 8150 0740 NEWDAT CMPA #P PROGRAM THE LOCATION
15AC 2630 0741 BNE DEXIT EXIT ROUTINE
15AE 303D009D 0742 TOB1G LEAX NDATA,PCR NEW DATA MESSAGE
15B2 17020A 0743 LBSR INPUT# GET DATA
15B5 1701A8 0744 LBSR HEXINT GET VALUE IN X
0745 * TEST B FOR ERROR CODE B(<>) FOR ERROR
15B8 5D 0746 TSTB
15B9 1026FF61 0747 LBNE DISDAT DO NOTHING
15BD 8C00FF 0748 CMPX #9FF HIGHEST ALLOWED DATA
15C0 22EC 0749 BHI TOB1G
15C2 1F10 0750 TFR X,D
15C4 F701D7 0751 STB TEMP FOR PROGRAMMING
15C7 17F99E 0752 LBSR PROGRM TRY TO PROGRAM IT
15CA 5D 0753 TSTB
15CB 1027FF4F 0754 LBEX DISDAT ALL OK
15CF C101 0755 CMPB #1 NOT ERASED
15D1 2710 0756 BEQ NOERSD NOT ERASED
0757 ***
15D3 308DFD75 0759 LEAX BADPRM,PCR
15D7 1701B1 0764 WRITE LBSR OUTST#
15DA BE01D5 0761 LDX TARGET
15DD 1701EB 0762 LBSR HEXOUT
15E0 16FF38 0763 LBRA DISDAT
0764 *
15E3 308DFD54 0765 NOERSD LEAX UNERSD,PCR UNERASED MESSAGE
15E7 20EE 0766 BRA .WRITE
0767 *****
15E9 8150 0768 DEXIT CMPA #X IS IT EXIT?
15EB 1026FF2F 0769 LBNE DISDAT NO SO REDISPLAY
15EF 39 0770 RTS RETURN TO MENU
0771 *****
0772 *****
0773 * THIS MOVES CURSOR 1 RIGHT IF NOT AT END OF SCREEN
15F0 FC0080 0774 MOVCRS LDD CURLOC
15F3 100305FF 0775 CMPD #05FF AT END?
15F7 2706 0776 BEQ ATEND
15F9 C30001 0777 ADDD #1 MAKE A SPACE
15FC FD0080 0778 STD CURLOC
15FF 39 0779 ATEND RTS
0780 *****
1600 20 0781 CELMSG FCC / INDIVIDUAL CELL PROGRAMMING/
1601 2020494E44
1606 4956494455
1608 414C204345
1610 4C4C205052
1615 4F4752414D
161A 4D494E47
161E 00 0782 FCB #0D
161F 20 0783 FCS / *****(<0D><0D>/
1620 20203D3D3D3D
1625 3D3D3D3D3D3D
162A 3D3D3D3D3D3D
162F 3D3D3D3D3D3D
1634 3D3D3D3D3D3D
1639 3D3D3D3D3D0D
163E 0D00
1640 00 0784 NADDRS FCS /(<0D>NEW ADDRESS? /
1641 4E45572041
1644 4444524553
164B 533F2000
164F 00 0785 NDATA FCS /(<0D>NEW DATA? /
1650 4E45572044
1655 4154413F20
165A 00
0786 *****
0787 * RETURNS CASSETTE FILE DATA
0788 *****
0789 *
0790 * THIS RETURNS THE ADDRESSES OF THE LAST CLOADM
0791 *
0792 *
01E7 0793 STADD EQU 407 START ADDRESS
007E 0794 ENDADD EQU 126 END ADDRESS
01E5 0795 EXECAD EQU 405 EXEC ADDRESS
0796 *
165B 8DA920 0797 CFILE JSR CLEAR SCREEN
165E 308D0035 0798 LEAX FILMSG,PCR HEADING
1662 1700F6 0799 LBSR OUTST#
1665 BE01E7 0800 LDX STADD GET START ADDRESS
1668 170160 0801 LBSR HEXOUT OUTPUT IT
0802 *
166B 308D0071 0803 LEAX ENDTAT,PCR GET END MESSAGE
166F 1700E9 0804 LBSR OUTST#
1672 9E7E 0805 LDX ENDADD
1674 301F 0806 LEAX -1,X MOVE TO ACTUAL END
1676 170152 0807 LBSR HEXOUT
0808 *
1679 308D0078 0809 LEAX EXEMSG,PCR GET EXE MESSAGE
167D 1700DB 0810 LBSR OUTST#
1680 BE01E5 0811 LDX EXECAD
1683 170145 0812 LBSR HEXOUT
0813 *
0814 * MOVE CURSOR DOWN 2 LINES
1686 FC0080 0815 LDD CURLOC
1689 C30020 0816 ADD #32
168C FD0080 0817 STD CURLOC
0818 *
168F 308DFC10 0819 LEAX EKEY,PCR GET ENTER MESSAGE
1693 170129 0820 LBSR INPUT#
0821 *
1696 39 0822 RTS
0823 *****
1697 20 0824 FILMSG FCC / CASSETTE FILE DATA/
1698 2020202020
169D 2043415353
16A2 4545444520
16A7 46494C4520
16AC 44415441
16B0 00 0825 FCB #0D
16B1 20 0826 FCS / *****/
16B2 2020202020
16B7 203D3D3D3D3D
16BC 3D3D3D3D3D3D
16C1 3D3D3D3D3D3D
16C6 3D3D3D3D
16CA 0D00 0827 FDB #0D0D
16CC 20 0828 FCS / START ADDRESS: /
16CD 2020205354
16D2 4152542041
16D7 4444524553
16DC 533A2000
16E0 00 0829 ENDTIT FCS /(<0D> END ADDRESS: /
16E1 2020202020
16E6 20454E4420
16EB 4144445245
16F0 53533A2000
16F5 00 0830 EXEMSG FCS /(<0D> EXECUTE ADDRESS: /
16F6 2020455845
16FB 4355544520
1700 4144445245
1705 53533A2000
0831 *****
0832 *****
0833 * UTILITY LIBRARY *
0834 *****
0835 *****
0836 * INSTR# GETS A STRING FROM KEYBOARD AND PUTS#
0837 * IT INTO "BUFFER" TERMINATED BY A ZERO BYTE. *
0838 *****
0839 * BASIC POINTERS
0800 0840 CURLOC SET #00 CURSOR LOCATION
0800 0841 POLCAT SET #A000 KEYBOARD POLL
0802 0842 CHRDT SET #A002 CHARACTER OUTPUT
080F 0843 DEVNUM SET #6F # FOR SCREEN, -2 FOR PRINTER
0844 *****
0845 *****
170A 10BE01DA 0846 INSTR# LDY #BUFFER POINT Y TO BUFFER START
170E 8D40 0847 CRSR BSR CURSOR PUT BLACK SQUARE UP
1710 AD9FA000 0848 GETKEY JSR [POLCAT] LOOK FOR KEY
1714 27FA 0849 BEQ GETKEY NOTHING ENTERED YET
1716 8100 0850 CMPA #00 BACKSPACE
1718 2617 0851 BNE CHKRET
171A 108C01DA 0852 CMPY #BUFFER AT START OF BUFFER
171E 27EE 0853 BEQ CRSR NO BACKSPACE POSSIBLE
1720 8660 0854 LDA #00 BLANK
1722 A79F0080 0855 STA [CURLOC] STORE AT CURRENT LOCATION
1726 313F 0856 LEAY -1,Y DECREASE CURSOR LOCATION
1728 DC80 0857 LDD CURLOC GET CURSOR LOCATION
172A 830001 0858 SUBD #1 REDUCE D BY ONE
172D DD80 0859 STD CURLOC RESET CURSOR LOCATION
172F 20DD 0860 BRA CRSR
0861 *
0862 * IF CR THEN PUT INTO BUFFER, WITH A ZERO BYTE
0863 * THEN EXIT
1731 8101 0864 CHKRET CMPA #00D CARRIAGE RETURN
1733 2600 0865 BNE INKEY NO SO PUT INTO BUFFER
1735 A7A0 0866 STA .Y+ PUT CR INTO BUFFER
1737 AD9FA002 0867 JSR [CHROUT] PUT RETURN ON SCREEN
1738 6FA4 0868 .EXIT CLR Y SET LAST BYTE TO ZERO

```

```

1730 39      0069      RTS
              0070 *
              0071 * PUT CHARACTER INTO BUFFER, CHECK FOR
              0072 * SPACE FIRST, IF BUFFER HAS 254 PUT IT
              0073 * THEN SET 256 BYTE TO ZERO AND EXIT
              0074 *
173E 0120    0075 INKEY CMPA #32      FIRST PRINTABLE CHARACTER
1740 25CC    0076 BLO CRSR      NOT PRINTABLE SO LOOP
1742 A7A0    0077 STA ,Y+        PQ INTO BUFFER
1744 AD9FA002 0078 JSR (CHROUT)  OUTPUT ENTERED CHARACTER
1748 108C020B 0079 CMPY #BUFFER+254  BUFFER FULL?
174C 25C0    0080 BLO CRSR      NOT FULL
174E 20E8    0081 BRA .EXIT
              0082 *
              0083 * CURSOR ROUTINE
1750 8600    0084 CURSOR LDA #128      BLACK SQUARE
1754 A79F0088 0085 STA (CURLOC)
1756 39      0086 RTS
              0087
              0088 *****
              0089 *****
              0090 *OUTST* TAKES A STRING POINTED TO BY REG X *
              0091 *AND PUTS IT TO OUTPUT DEVICE. TERMINATED *
              0092 *BY A ZERO BYTE IN BUFFER *
              0093 *****
              0094 * BASIC POINTER
A002        0095 CHROUT SET #A002    OUTPUT ROUTINE
              0096 *
1757 AD9FA002 0097 .DSPLY JSR (CHROUT)  OUTPUT CHARACTER
175B A600    0098 OUTST* LDA ,X+      GET CHARACTER
175D 26FB    0099 BNE .DSPLY    DISPLAY IF NOT ZERO
175F 39      0080 RTS
              0081 *****
              0082 *****
              0083 *HEXINT GETS A HEX NUMBER FROM BUFFER AND *
              0084 *PUTS IT IN REG X. REG B IS ZERO IF NO *
              0085 *ERROR. WILL GET FIRST 4 CHARACTERS IN *
              0086 *BUFFER OR TO <CR> OR ZERO BYTE *
              0087 *****
              0088
1760 108E01DA 0089 HEXINT LDY #BUFFER    POINT Y TO BUFFER
1764 BE0000   0090 .LDX #0          CLEAR X FOR NUMBER
1767 B604     0091 LDA #4           CHARACTER COUNTER
1769 E6A0     0092 GTHEX LDB ,Y+    GET CHARACTER FROM BUFFER
176B 271E    0093 BEQ HEXIT       AT END OF BUFFER
176D C10D    0094 CMPB #0D        IS IT A <CR>?
176F 271A    0095 BEQ HEXIT       YES SO AT END
1771 C130    0096 CMPB #0        IS IT LESS THAN 0?
1773 2524    0097 BLO HEXERR     NO SO ERROR
1775 C139    0098 CMPB #09       GREATER THAN 9
1777 2214    0099 BHI ALPHA      MAY BE A - F
1779 C030    009A SUBB #0        MAKE A NUMBER
              009B *
              009C *B NDM HAS VALUE ENTERED
177B 1E01    009D HEX EXG D,X    SWAP REGISTERS FOR SHIFT
              009E * SHIFT D LEFT 4 PLACES
0004        009F RPT
              00A0 ASLB
              00A1 ROLA
              00A2 ENDR
1770 58      + ASLB
177E 49      + ROLA
177F 58      + ASLB
1780 49      + ROLA
1781 58      + ASLB
1782 49      + ROLA
1783 58      + ASLB
1784 49      + ROLA
1785 1E01    0099 EXG D,X    PUT IT BACK INTO X
1787 3A      009A ABI      ADD VALUE INTO REGISTER X
1788 4A      009B DECA
1789 26DE    009C BNE GTHEX
178B 5F      009D HEXIT CLR
178C 39      009E RTS
              009F **
178D C141    009A ALPHA CMPB #A    LESS THAN "A"
178F 250B    009B BLO HEXERR  YES SO ERROR
1791 C146    009C CMPB #F    HIGHER THAN "F"
1793 226A    009D BHI HEXERR  YES SO ERROR
1795 C037    009E SUBB #A-10    SET TO VALUE
1797 20E2    009F BRA HEX
              00A0 **
1799 C601    009A HEXERR LDB #1
179B BE0000  009B LDA #0
179E 39      009C RTS
              009D *****
              009E *****
              009F * HPAIR CONVERTS CONTENTS OF REG B INTO A *
              00A0 * STRINGS IN BUFFER TERMINATED BY A ZERO *
0950 * BYTE. NO <CR> IS ADDED TO THE STRING
0951 *****
0952
0953 *
0954 HPAIR LDY #BUFFER    POINT TO BUFFER
0955 * GET HIGH NIBBLE FROM B
1793 1F98    0956 TFR B,A        INTO A
0957 RPT 4      MOVE DOWN 4 PLACES
0958 LSR
0959 ENDR
1795 44      + LSR
1796 44      + LSR
1797 44      + LSR
1798 44      + LSR
0950 *
0951 *
0952 *
0953 *
0954 *
0955 *
0956 *
0957 *
0958 *
0959 *
0960 *
0961 *
0962 *
0963 *
0964 *
0965 *
0966 *
0967 *
0968 *
0969 *
0970 *
0971 *
0972 *
0973 *
0974 *
0975 *
0976 *
0977 *****
0978 *****
0979 * INPUT* OUTPUTS A STRING POINTED TO BY REG *
0980 * X, THEN RECEIVES A STRING FROM KEYBOARD *
0981 * AND PUTS IT INTO "BUFFER" TERMINATED WITH *
0982 * A ZERO. IF X IS ZERO NO STRING IS OUTPUT. *
0983 * MAX. CHARACTERS IN BUFFER IS 255. *
0984 *****
0985 * BASIC POINTERS
0986 CURLOC SET #B        CURSOR LOCATION
0987 POLCAT SET #A000     KEYBOARD POLL
0988 CHROUT SET #A002    CHARACTER OUTPUT
0989 DEVNUM SET #6       # FOR SCREEN, -2 FOR PRINTER
0990 *****
0991 *
0992 *
178F BC0000  0993 INPUT* CMPX #0      ANY TEXT TO OUTPUT
17C2 2703    0994 BEQ NOTEXT
17C4 17FF94  0995 LBSR OUTST*        OUTPUT TEXT STRING
17C7 17FF40  0996 NOTEXT LBSR INSTR*  GET INPUT STRING
17CA 39      0997 RTS
              0998 *
              0999 *
              1000 *
              1001 *****
              1002 *****
              1003 * HEXOUT TAKES CONTENTS OF X AND PUTS IT ON *
              1004 * SCREEN. USES HPAIR TO DO IT IN 2 PARTS *
              1005 * OUTST* IS ALSO USED *
              1006 *****
              1007
17CB 1F10    1008 HEXOUT TFR X,D      PUT DATA INTO REG D
17CD 1E89    1009 EXG A,B          PUT HIGH BYTE IN B
17CF 17FFCD  1010 LBSR HPAIR        PUT INTO SCREEN
17D2 3410    1011 PSHS X           PRESERVE VALUE
17D4 BE01DA  1012 LDY #BUFFER      POINT TO START OF STRING
17D7 17FFB1  1013 LBSR OUTST*     PUT OUT THE STRING
17DA 3506    1014 PULS D          RECOVER VALUE IN D
17DC 17FFC0  1015 LBSR HPAIR      PUT LOW BYTE ON SCREEN
17DF BE01DA  1016 LDY #BUFFER      POINT TO START OF STRING
17E2 17FF76  1017 LBSR OUTST*     PUT OUT THE STRING
              ---
EPRM.MAC      COMPUTERWARE MACRO ASSEMBLER PAGE 22
2764 EPRM PROGRAMMER BY C.J.STEARMAN (C)1984
17E5 39      1018 RTS
              1019 *
              1020 *****
              1021 *
              1022 TTL 2764 EPRM PROGRAMMER BY C.J.STEARMAN
              1023 NAM EPRM.MAC
              1024 *
0E00        1025 END EPRM
NO ERROR(S) DETECTED

```




Disk Drive Speed Check

By Roger Schrag

Have you ever suddenly been barraged by I/O Errors when trying to load a program from disk? Sometimes this is a sign that your disk drive needs some routine adjusting.

One of the things that can periodically slip out of line within your disk drive is its rotational speed. The disk drive is supposed to spin your diskettes at 300 revolutions per minute (rpm), give or take five percent.

Inside your disk drive there is a little knob which you may turn to adjust your disk drive's speed. This BASIC program will tell you how fast your drives are running. By repeatedly turning the knob slightly and then running the program, you may easily adjust your disk drives to perfect operating speed, thus saving a hefty repair bill.

If one of your drives is giving more than its fair share of I/O Errors, then run this program to see if indeed your drive's speed is off. The program will ask which drive you would like to check, and then will prompt you to insert an initialized diskette in the drive and press ENTER. Any diskette will do, as long as it has been initialized previously with the *DSKINI* command. The program then will draw up a chart of your drive's speed on 10 consecutive readings and the overall average.

If your drive is consistently more than about five rpm off from 300, you may wish to adjust the speed control. First, remove the outer cabinet by removing the four exterior screws. If your disk drive is a Radio Shack model, then the speed control is the bright yellow knob on the small circuit board on the same side of the drive as the large belt connecting the motor to the hub which grips the diskette.

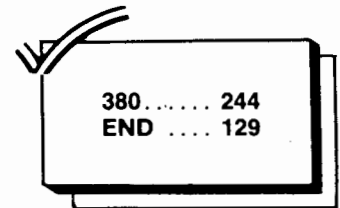
Use a flat blade screw driver to turn the knob *slightly* in one direction or the other. Then run the speed check program again. Do this repeatedly until your drive's speed is within about five rpm of 300. Your drive does not need to operate at exactly 300 rpm, and fluctuations of one or two rpm is perfectly normal.

This little program lets you determine if the source of your disk problems is a drive motor running off speed. This program lets you also fix it if you are somewhat mechanically inclined. Please note that opening your drive may void your warranty. However, this should not be a problem; if your drive were still under warranty, you would take it back

to the store if it exhibited any kind of problems whatsoever.

The program uses a short machine language subroutine to perform the actual timing. The data statements are set up so as to show the actual assembly language code that is being *POKEd* into memory.

All diskettes for the Color Computer have a pinhole in them called a "sector index hole." This pinhole passes in front of a sensor in the disk drive exactly once on each revolution. By reading this sensor, the machine language subroutine times how long it takes for the diskette to complete one revolution. From this information, the BASIC program is able to calculate how many revolutions the diskette would make in one minute if it continued spinning at that exact rate. This is the disk drive's rotational speed, measured in revolutions per minute.



The listing:

```

1 REM *****
2 REM * DISK DRIVE SPEED CHECK *
3 REM *****
4 REM
5 REM BY: ROGER SCHRAG
6 REM 2054 MANNING AVENUE
7 REM LOS ANGELES, CA 90025
8 REM
9 REM
100 CLS: CLEAR 500
110 READ B$: IF B$="END" THEN 130
120 A$=A$+B$: READ DUMMY$: GOTO 110
130 FOR X= 1 TO LEN(A$)/2
140 Y=VAL("&H"+MID$(A$,X*2-1,2))
150 POKE X+3585,Y: C=C+Y: NEXT X
160 IF C<>4001 THEN 490
170 S$=STRING$(32,61)
180 F1$="TRIAL ## SPEED ###.##"
190 F2$="AVERAGE ###.##"
200 PRINT "DRIVE SPEED CHECK"
210 PRINT S$
220 PRINT@128,"WHICH DRIVE";
230 INPUT DV: PRINT@142," "
240 IF DV<0 OR DV>3 THEN 220
250 PRINT: PRINT"PLEASE MOUNT ";
260 PRINT"AN INITIALIZED"
270 PRINT"DISK IN DRIVE"; DV;
280 INPUT"& PRESS ENTER"; X
290 CLS: PRINT"SPEED CHECK -- ";
300 PRINT"DRIVE"; DV: PRINT S$;
310 DSKI$ DV, 17, 1, A$, B$
320 POKE &HFF48, 3
330 TL=0: FOR TR=1 TO 10
340 POKE 2437, 120 : EXEC 3586
350 SP=PEEK(3584)*256+PEEK(3585)
360 IF SP=0 THEN 450
370 SP = SP * 0.026779174
380 PRINT USING F1$; TR, SP
390 TL=TL+SP: NEXT TR
    
```

```

400 AV=TL/10
410 PRINT TAB(9) "-----"
420 PRINT USING F2$; AV
430 PRINT@498,"PRESS ENTER";
440 INPUT X:CLS:GOTO 170
450 PRINT@384,"**** ERROR ****"
460 PRINT"PLEASE CHECK DISK ";
470 PRINT"IN DRIVE";DV
480 GOTO 430
490 PRINT"DATA ITEM INCORRECT"
500 STOP
510 REM
520 REM MACHINE LANGUAGE ROUTINE
530 REM
540 DATA"3413" ,"ST PSHS A,X,CC
550 DATA"1A50" ," ORCC #50
560 DATA"9E8A" ," LDX #8A
570 DATA"8602" ," LDA #02
580 DATA"3001" ,"L1 LEAX 1,X
590 DATA"2719" ," BEQ EX
600 DATA"B5FF48" ," BITA $FF48
610 DATA"27F7" ," BEQ L1
620 DATA"9E8A" ," LDX #8A
630 DATA"3001" ,"L2 LEAX 1,X
640 DATA"2710" ," BEQ EX
650 DATA"B5FF48" ," BITA $FF48
660 DATA"26F7" ," BNE L2
670 DATA"3001" ,"L3 LEAX 1,X
680 DATA"2705" ," BEQ EX
690 DATA"B5FF48" ," BITA $FF48
700 DATA"27F7" ," BEQ L3
710 DATA"BF0E00" ,"EX STX $E00
720 DATA"3593" ," PULS CC,X,A
,PC
730 DATA"END","MARK END OF DATA

```

Corrections

This note for non-disk users of the electronic spreadsheet program in "MoCalc — MiniCalc Gets a Big Brother" (April 1984, Page 186), author Barry Spencer says those who don't have Disk BASIC should replace the command *WRITE* with *PRINT* in Line 1020 and change *KILL* to *PRINT* in Line 1040.

H. Allen Curtis writes that we mistakenly indicated the minimum system for his *Bandy* program (June 1984 issue) to be Disk BASIC. This was done because of the use of *SAVE*, *LOAD*, and *WRITE* in the listing, but Curtis says, "The program logic is such that the lines containing those commands are bypassed when there is no disk controller connected." Thus, *Bandy* can be used in cassette-based 16K ECB — as indicated in the third paragraph of his article.

Curtis adds, however, that *LOAD*, *SAVE* and *WRITE* will not tokenize when used with a cassette-based system.

Reader Steven Ostrom, Minnetonka, Minn., tells us that there is an error in the "Simply Load and . . . Bingo!"

program (Page 92, April 1984 issue). The beginning of Line 520 should read *IF BB=>10* (not, *IF BB>10*). You need to add the "equal" symbol because, otherwise, when L=3 and BB=10, the middle row of the Bingo card will not have a free space.

Steven adds, "For my DMP-120, I had to add a printer delay (*POKE 151,25*) in addition to changing the elongation commands that the author noted. This *POKE* is necessary for many programs that print, due to a bug in the DMP-120, even after the Radio Shack upgrade."

Damon Swanson writes, "There is an occasional but potentially deadly bug in my modification to Steve Good's Spooler, ("Make the Good Spooler Better," May 1984, Page 23). The disaster will strike if an interrupt occurs while in the RAM (Type 1) memory map.

"The bug rarely bites because the program is only at risk during 11 of the almost 15,000 clock cycles between interrupts, and it causes disaster only when the print buffer is empty, i.e., when printing the first character to the buffer. Under these conditions, the interrupt handler, *START*, detects the empty buffer and jumps to high memory expecting ROM but finding random code in RAM.

"If the buffer has at least one character, *START* gets that character from RAM and sets the map back to ROM before calling any ROM routines. We can still have an error — dropping one character. The program is at risk for seven clock cycles yielding an error rate of about one in 2,000 characters."

"Bugs of this species are hard to recognize, impossible to test, and often ignored. But a good programmer will find them and destroy them. Fortunately, this bug is easily exterminated."

Continues Swanson, "Add an *ORCC #50* (Line 743) to kill the potential interrupt before switching to the RAM page and *ANDCC #SAF* (Line 757) to turn it on again in the modified spooler listing. Also correct the start address to allow the four extra bytes (*ORG \$7F61* or *\$7CF0* with *SCRPT*).

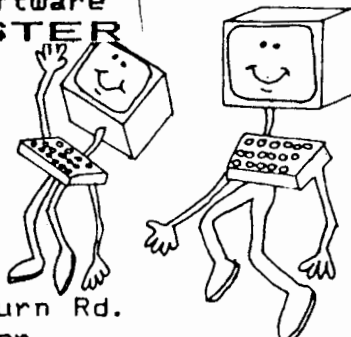
"Note there is no problem in the interrupt handler, *START*, which is synchronized with the interrupt and finishes long before another one comes along."

Thanks, Damon, your discussion calls attention to one of the more subtle problems of interrupt handling.

**Spectrum Software
DONCASTER**

- Adventures
- Arcade
- Business
- Utilities

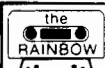
Mail orders



1/234 Blackburn Rd.
East Doncaster
Melb. Vic. 3109.
Call PETE
03-842-1205

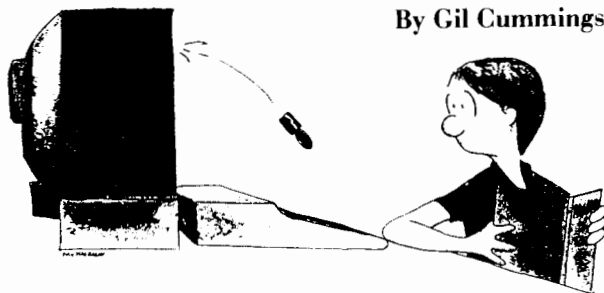
GAME

16K
ECB



ATTACK OF THE KEY BOMBERS

By Gil Cummings



My daughter took an instant shine to my computer. When it was new I couldn't power up without hearing a little voice pipe up, "Daddy, can I sit down and type *RUN*?" She had already seen the canned games that run on the Atari 2600 so she wasn't much impressed with my early efforts at programming. Every day she wanted to see something new and better. I found a winning formula pretty quick — use all the colors, lots of different sounds and plenty of visual and verbal rewards and go lightly on the "you lose" routines.

Even so, Julie's favorite game soon became, "Let's get Daddy away from the computer." I'd have something half baked in the RAM and when she tried to run it, it would throw an RG Error or some such put down. Then I'd tell her to wait until it runs a little better and instantly the face would cloud over with the "Oh, what a bummer!" look.

Or worse, the game would call for a lot of inputs, making the pace grind to a halt as she hunted for the keys. That kind of thing would end in something like "Oh, I can never win."

I've been at this for about two years now and in the process I've learned a little about computer programming and, more importantly, about fathering — a few things that I would never have known without access to a small computer. The Color Computer is the best kind of computer for this activity. Color and Extended Color BASIC make it easy to achieve the rewards of creativity, a real sense of satisfaction in seeing your dreams come true. Whatever you can dream you can bring into being through your CPU.

This little game helps kids make finding the right key a conditioned reflex. A small drama takes place above the keyboard as the bomber flits in and takes aim on a key. A bomb falls and it's up to the player to defend the helpless by pressing the targeted key, warding off the attack. The key colors and letters are stored in *DATA* statements, along with a symphony of inflight music. The graphics are all Lo-Res, the better to put all nine colors to work at once.

Line 2 provides room and board for an array in two dimensions, *LC\$* (level, character). There are four rows of keys, 10 keys in each row. I put '@' in the bottom row since they get a lot of action when programming. Lines 8 to 14 are subroutines for firing at the bomber, keeping score and making sure the bomber doesn't take aim on a key that's out of play.

The difficulty factor makes use of the tempo feature of the *PLAY* command. The lower the difficulty number, the slower the bomb whistles and the more time you have to respond with anti-aircraft fire. Lines 30-87 load in the colors and sounds, allowing the colors of the keys to change with each round of play. The bomber's inflight music corresponds to the key color.

Lines 100-165 draw the keys. Line 170 changes the row of keys presented with each round and awards bonus points for hitting five in a row. Line 190 paints the bomber (64 dazzling color combinations). Lines 200-290 control the play and build suspense as you wait for the bomb to drop. If the player presses the right key in time, it computes the score (more points for higher difficulty and quicker response). If the player misses, the key is out and a new round begins. A round is over when the player gets five in a row or loses the last key. A game is over after three rounds; then you get blue skies and a chance to do it again.

Instructions are in Lines 300-390, winning takes you to Lines 400-420 and losing all the keys gets you another chance at Line 500.

150.....	48
304.....	170
END	187

The listing:

```

2 DIM CB*(8),LC*(4,10):WB*=CHR*(
128)+CHR*(128)+CHR*(128):KC=0:SC
=0:GC=1:SS=1:GOSUB300:GOTO30
8 FOR I=23TO 3 STEP -1:SET(H,I,C
):SET(H-1,I,C):PLAY"05T64888E":R
ESET(H,I):RESET(H-1,I):NEXT I:PR
INT@KC*3-2,WB*:SC=SC+DF*(31-V):
GOSUB10:RETURN
10 PRINT@487,USING" SCORE ####
# # ";SC,SS;:RETURN
12 CR=0:SS=1:C=0:LE=RND(4):FOR I
=1 TO 10:XC(I)=1:NEXTI:RETURN
14 FORI=1 TO 10:IF XC(I)=1 THEN
RETURN ELSE NEXTI:GOTO500
30 CLS0:INPUT" EASY - HARD (1 -
16)";DF:DL*=STR*(DF*4):CLS0:FORC
=1TO8:READ CC
40 CB*(C)=CHR*(CC)+CHR*(CC)+CHR*
(CC)
50 NEXT C
60 FOR I=1 TO 4:FOR J=1 TO 10
70 READ LC*(I,J)
80 NEXT J,I
85 FOR I=1 TO 8:READ FL*(I):NEXT
87 GOSUB12
100 J1=RND(7):J2=J1+10
110 FOR I=0 TO 2
115 L=385+I*32
120 FOR J=J1 TO J2
130 IF J<9 THEN C=J ELSE C=J2-J
140 PRINT@L,CB*(C);
150 L=L+3
    
```

```

160 NEXT J
165 NEXT I
170 IF SS>5 THEN PRINT @487,"!!!
!BONUS!!500!!!!";:FORI=1 TO 5:PL
AY"O"+STR$(I)+"T16CDEFGAB":NEXTI
:SC=SC+500:GOSUB10:GOSUB12:GC=GC
+1:IFGC>3THEN400
172 FOR LO=410 TO 445 STEP 3
175 CR=CR+1
180 PRINT @LO,LC$(LE,CR);:NEXT L
O:GOSUB10
190 BC=RND(8):WC=RND(8):BR$=CHR$
(115+WC*16)+CHR$(124+BC*16)+CHR$
(115+WC*16):IF SS>5 THEN100
200 KC=RND(10):IF XC(KC)=0 THEN
200 ELSE DR=RND(4):PRINT@KC*3-2,
BR$;
205 H=KC*6-1:C=POINT(H,29):PLAYF
L$(C):IFDR=4THEN220ELSEPRINT@KC*
3-2,WB$;:GOTO200
220 H=KC*6-1
240 FOR V=3 TO 24:SET(H,V,C):SET
(H-1,V,C):PLAY"O4T"+DL$+"B-":RES
ET(H,V-1):RESET(H-1,V-1):IFINKEY
$=LC$(LE,KC)THENS$=SS+1:GOSUB8:G
OTO190ELSENEXTV
250 XC(KC)=0:PRINT@416+H/2,CHR$(
120);:PLAY"O1T2B-AGFE-DC":PRINT@
KC*3-2,WB$;:SS=1:GOSUB14:GOSUB10
290 GOTO 190
300 CLS:PRINT:PRINTTAB(42)"KEY B
OMBER"
302 PRINT:PRINT
304 PRINT"      LOOK OUT! UP IN T
HE SKY!      IT'S THE KEY BOMBER!
HE'S        TRYING TO BOMB YOUR
COMPUTER    KEYS. DON'T LET HIM
. WHEN      HE DROPS HIS BOMB YO
U HAVE      TO PRESS THE KEY BEF
ORE THE     BOMB LANDS ON IT. T
HEN YOUR    KEY IS SAFE!"
306 PRINTTAB(43)"GOOD LUCK!"
390 IFINKEY$=""THEN390 ELSE RETU
RN
400 CLS5:PRINT@(40),"THE SKIES A
BOVE";:PRINT@(103),"YOUR KEYBOAR
D ARE";:PRINT@(173),"SAFE!";
405 GOSUB 10
410 PRINT@266,"PLAY AGAIN?";
420 I$=INKEY$:IFI$=""THEN420ELSE
IFI$<>"N"THENRUNELSE PRINT@324,"
OK, TURN OFF THE COMPUTER.":END
500 PRINT@262,"TH-TH-TH-THAT'S A
LL!";
510 PRINT@331,"TRY AGAIN?";
520 I$=INKEY$:IFI$=""THEN520 ELS
EIFI$="N"THENEND
540 RUN
899 IF INKEY$=""THEN 899 ELSE CL
S:LIST-888

```

```

900 DATA 143,159,175,191,207,223
,239,255
910 DATA 1,2,3,4,5,6,7,8,9,0
912 DATA Q,W,E,R,T,Y,U,I,O,P
914 DATA A,S,D,F,G,H,J,K,L,;
916 DATA Z,X,C,V,B,N,M,"@",":",""
-"
920 DATA O2T8GAG,O2T16CDEFEDC,O2
T16EECCDGE,O3T8CDE,O3T8EDC,O3T16
EECCDDDB,O5T32CDEFGABAGFEDC,O5T8
BAG

```

Bying the Apple

- I'm looking for someone who has a program to read Apple II disks on a CoCo using either Disk BASIC or OS-9.

Larry Owen
Phoenix, AZ

I haven't heard of any, and there's a very good reason why it isn't done routinely. The CoCo and most other personal computers use a special disk controller chip, and these are usually designed to use a standard format developed by IBM some 10 years ago (although there are many variations and two different encoding schemes used now). The Apple II uses a rather ingenious circuit that Apple co-founder Steve Wozniak designed using readily available chips; it has stood the test of time so well that an improved single-chip version is used in the new Apple IIc and Macintosh, but it doesn't seem to be compatible with the "standard" disk controllers! (I'm sure that some of the more intrepid hackers among us will take this as a dare rather than a fact of life; I'd like to see somebody bridge this gap.)

Preserving Memories

- I was wondering if it is possible to make a small device which can be fitted to the CoCo which will keep the memory powered while the rest of the computer is switched off?

Richard Goodman
College Station, TX

Sadly, it isn't as easy as that. The CoCo (as with most other microcomputers) uses "dynamic" memory chips, rather than the more expensive "static" RAMs. A static memory system will hold its data as long as power is going into the chips, but with dynamic memory you have to constantly "refresh" the memory; in the CoCo it's done simply by accessing the memory every few milliseconds, and the job is handled nicely by the SAM chip. To keep dynamic RAMs running while the system is shut down, you would need to keep the refresh cycles going in some way, and you would have to keep your circuit from interfering with the SAM chip's operation. One false move in the transition from one to the other and you'll lose your data faster than you can say "6883!"

GAME

32K
ECB



DRAGONS GOLD

Charles Husak

Once upon a time in a far away land lived a dragon who guarded a fabulous fleece of gold. In this land there also lived a poor farmer named Arnold, who dreamed of the day when he could climb to the top of the dragon's lair, slay him, and take his golden fleece.

The trip to the top of the lair is very dangerous, but Arnold has learned from a sorcerer the secrets of the lair — that there is a staff of Ora at the very top which will destroy the dragon. But you must reach the top, get the staff, and push your joystick button to fire the magic light that will destroy the dragon. Getting the staff is difficult. You must jump over the Hornobers that come from under the ladders. Then you must climb the one-way ladders. Once you start you cannot climb down. Beware of the pits that open and close. You must wait till the pit closes before you can cross.

The Dragon's Gold, requiring 32K ECB, is a game of arcade action which I wrote for my eight-year-old daughter, Dianne. For those who can use the higher computer speed, enter 5 POKE 65495,0. Speeding up the computer will also increase the difficulty. To start the game after the title either push the Space Bar or the joystick button. And here's a hint before you begin the quest: Keep moving because if you stop, the Hornobers will speed up.

100.....	41	2756	51
195.....	94	3030	122
430.....	52	3155	1
2100	123	3240	57
2302	81	END	215

The listing:

```

6 DIM R(10),S(10)
7 TR=1
8 CLS:GOSUB3000:CLS
9 CLS:GOTO2000
10 T=&H1A:Z=1:POKE&H415A,0
11 C=183:D=162:P=0
12 POKE&H4123,&H59:POKE&H4125,&H
46:POKE&H4127,&H59
13 KL=0:POKE&H3F58,&H1A
15 PMODE4:SCREEN1,1:PMODE3
20 PCLS
30 DRAW"BM0,184;C7;R255"
40 PAINT(8,188),7,7
50 DRAW"BM0,144;C7;R255"
55 DRAW"BM0,152;C7;R208U8R18D8R3
0"
57 PAINT(250,148),7,7
    
```

```

60 PAINT(8,148),7,7
70 DRAW"BM0,112;C7;R33U8R18D8R21
0"
75 DRAW"BM0,104;C7;R255"
77 PAINT(250,108),7,7
80 PAINT(8,108),7,7
90 DRAW"BM0,72;C7;R208U8R18D8R30
"
95 DRAW"BM0,64;C7;R255
97 PAINT(250,68),7,7
100 PAINT(8,68),7,7
110 DRAW"BM0,44;C7;R24U4R20D8R4D
4R4D4R4D4R4D4"
120 PAINT(20,48),7,7
125 GOSUB130
126 GOTO160
130 DRAW"BM212,184;C7;U4R8D4U8L8
D4U8R8D4U8L8D4U8R8D4U8L8D4U8R8D4
U8L8D4U8R8D4"
140 DRAW"BM36,144;C7;U4R8D4U8L8D
4U8R8D4U8L8D4U8R8D4U8L8D4U8R8D4U
8L8D4U8R8D4"
150 DRAW"BM212,104;C7;U4R8D4U8L8
D4U8R8D4U8L8D4U8R8D4U8L8D4U8R8D4
U8L8D4U8R8D4"
155 RETURN
160 DRAW"BM8,44;C6;U4H4G2;BM10,4
4;U6E6U2;BM8,40;U3H6U2;BM10,40;U
10E3"
165 DRAW"BM18,30;C8;D6R2U6L2"
170 DRAW"BM70,63;C6;U2R6U2R6U2R6
U2R6U2R6D2R5D2R6U2R4U2R4D2R2E8R3
D2L1D2L2D2L2D2L2D2R10"
173 DRAW"BM240,63;C8;U15L3E5F5L4
D15"
180 DRAW"BM70,63;C6;R15U3F3R4U4F
4R20U2L4U2R22U2"
185 PAINT(72,62),6,6:PAINT(89,60
),6,6
187 PSET(118,55,7)
188 POKE&H3F58,&H1A:POKE&H3F59,&
H40
189 EXEC&H3F00:Y=65
190 POKE&H3F07,&H3F:POKE&H3F08,&
H5B
192 EXEC&H4099
195 A=JOYSTK(0)
200 IF A>50 THEN GOSUB2300
210 IF A<10 THEN GOSUB2330
215 B=JOYSTK(1)
220 IF B<10 AND PEEK(&H3F59) =&H5
A THEN GOSUB2650
230 IF B<10 AND PEEK(&H3F59) =&H4
4 AND C=143 THEN GOSUB 2660
240 Z=Z+1
250 IF Z=8 THEN GOSUB405
260 IF Z=20 THEN GOSUB 430
265 IF PEEK(&H3F59)>77 AND PEEK(
&H3F59)<82 THEN 266 ELSE 270
266 IF Z>8 AND Z<20 THEN GOSUB47
    
```

```

0
270 IF PEEK(339)=254 THEN GOSUB1
000
272 EXEC&H4099
275 EXEC&H3F00
278 EXEC&H4130
279 IF PEEK(&H415A)=100 THEN 480
280 IF PEEK(&H3F58)=&H0B THEN 27
80
310 SC=SC+10
400 GOTO190
405 DRAW"BM120,112;C1;UBR20DBL20
":PAINT(125,108),1,1
410 DRAW"BM120,152;C1;UBR20DBL20
":PAINT(125,149),1,1
420 RETURN
430 DRAW"BM120,112;C7;UBR20DBL20
":PAINT(125,108),7,7
440 DRAW"BM120,152;C7;UBR20DBL20
":PAINT(125,149),7,7
450 Z=0
460 RETURN
470 IF PEEK(&H3F58)<15 THEN RETU
RN
475 IF PEEK(&H3F58)>25 THEN RETU
RN
480 SR=SR+1
481 PLAY"L20;04;1;2;3;4;5;6;7;8;
9;10;11;12"
483 IF SR>2 THEN 490
484 GOTO10
490 TR=4
491 PLAY"L20;04;1;2;3;4;5;6;7;8;
9;10;9;8;7;6;5;4;3;2;1"
495 GOTO9
1000 A=PEEK(&H3F58)
1005 EXEC&H3F3C
1010 A=A-1
1020 POKE&H3F58,A
1030 EXEC&H3F1E
1033 FORX=1TO4:EXEC&H4099
1034 PLAY"03L30;4;5":NEXTX
1035 EXEC&H3F3C
1040 A=A+1
1050 POKE&H3F58,A
1055 EXEC&H3F00
1060 RETURN
2000 SR=0
2080 FORV=1TO8:GOSUB2200:NEXT:FO
RV=1TO8:GOSUB2200:NEXT
2090 FORV=1TO8:GOSUB2210:NEXT:FO
RV=1TO6:GOSUB2210:NEXT
2100 FORV=1TO8:GOSUB2220:NEXT:FO
RV=1TO7:GOSUB2220:NEXT
2105 POKE1534,143:POKE1535,143
2110 DEFUSR0=16515
2120 PRINT@136,"THE DRAGON'S GOL
D";:PRINT@207,"BY";
2130 PRINT@233,"CHARLES A HUSAK"
2140 IF TR=4 THEN PRINT @ 393,"S
ORRY YOU LOSE ";
2145 IF TR=5 THEN PRINT @ 389,"Y
OU HAVE WON THE GOLD";
2170 SCREEN0,1
2172 W=1
2175 R$="CFCFCFCFCGFEFGCECECECEFE
DEFCEFEDEFCAF"
2176 N$="T602L5"
2180 A=USR0(0):IFW=36 THEN W=1
2181 M$=MID$(R$,W,1):W=W+1
2182 PLAY"XN$;XM$;"
2183 A=USR0(0)
2185 IF INKEY$=" " OR PEEK(339)=
254THEN10 ELSE2180
2200 PRINTSTRING$(2,127+16*V);:R
ETURN
2210 PRINTSTRING$(2,127+16*(9-V)
)TAB(30)STRING$(2,127+16*V);:RET
URN
2220 PRINTSTRING$(2,127+16*(9-V)
);:RETURN
2300 POKE&H3F26,&H87:POKE&H3F08,
&H5B
2302 EXEC&H3F3C:EXEC&H3F00
2304 FORX=1TO30:NEXT
2305 EXEC&H3F3C
2306 EXEC&H3F1E
2307 GOSUB130:EXEC&H4130:IF PEEK
(&H415A)=100 THEN 480
2308 EXEC&H3F3C
2310 IF PEEK(&H3F59)=&H5C THEN19
0
2312 POKE&H3F59,Y
2314 Y=Y+1
2315 EXEC&H3F00
2316 RETURN
2330 POKE&H3F26,&HEF:POKE&H3F08,
&HC3
2500 EXEC&H3F3C:EXEC&H3F00
2510 FORX=1TO30:NEXT
2515 EXEC&H3F3C
2520 POKE&H3F59,Y
2530 EXEC&H3F1E
2535 EXEC&H4130:IF PEEK(&H415A)=
100 THEN 480
2540 GOSUB130
2550 EXEC&H3F3C
2555 IF PEEK(&H3F59)=&H40 THEN19
0
2595 Y=Y-1
2598 EXEC&H3F00
2600 RETURN
2650 C1=210:D1=225
2655 GOTO2700
2660 C1=35:D1=48
2700 IF C1=210 AND P=10 THEN RET
URN
2702 IF C1=210 AND P=30 THEN RET
URN

```

```
2705 POKE&H3F07,&H40:POKE&H3F08,
&H2B
2710 EXEC&H3F00
2715 GET(C1,C)-(D1,D),R,G
2730 POKE&H3F08,&H57
2740 EXEC&H3F00
2745 GET(C1,C)-(D1,D),S,G
2751 PUT(C1,C)-(D1,D),R,PSET
2752 GOSUB130:C=C-2:D=D-2:EXEC&H
4099
2753 LINE(C1,C+2)-(D1-2,D+2),PRE
SET,BF
2754 PUT(C1,C)-(D1,D),S,PSET
2755 FORX=1TO40:NEXT:C=C-2:D=D-2
2756 LINE(C1,C+2)-(D1,D+2),PRESE
T,BF
2757 DRAW"BM30,104;R22"
2758 DRAW"BM206,64;R22"
2759 DRAW"BM206,144;R22"
2760 P=P+1:IF P=10 THEN 2765
2762 IF P=20 THEN 2765
2763 IF P=30 THEN 2765
2764 GOTO2751
2765 POKE&H3F07,&H3F:POKE&H3F08,
&H5B
2767 T=T-5
2768 POKE&H3F58,T
2770 RETURN
2780 XX=220
2782 IF KL=9 THEN 2900
2783 EXEC&H4099
2784 A=JOYSTK(0):KL=KL+1
2785 IF A>50 THEN GOSUB2300
2786 IF A<10 THEN GOSUB2330
2787 IF PEEK(&H3F59)>&H5A THEN G
OSUB 2850
2788 IF PEEK(&H3F59)<82 AND KL<1
6 THEN GOSUB480
2790 IF PEEK(&H3F59)=&H48 THEN 2
930
2840 IF XX<200 THEN 2783
2845 GOTO 2782
2850 IF PEEK(339)=254 THEN 2852
ELSE RETURN
2852 XX=220:YY=57
2853 CIRCLE(XX,YY),4,8:CIRCLE(XX
,YY),4,1
2854 XX=XX-3
2855 IF PPOINT(XX-4,YY)=6 THEN 2
857 ELSE 2853
2857 LINE(70,63)-(136,44),PRESET
,BF
2858 RETURN
2900 XS=136:YS=57
2905 CIRCLE(XS,YS),4,6:CIRCLE(XS
,YS),4,1
2907 XS=XS+3
2910 IF XS=220 THEN GOSUB480 ELS
E GOTO 2905
2920 GOTO2780
2930 T=T-1
2932 EXEC&H3F3C
2935 POKE&H3F58,T
2936 GOSUB2330
2947 FORX=1TO100:NEXTX
2948 IF PEEK(&H3F58)=&H08 THEN 2
955
2950 GOTO2930
2955 GOSUB2330
2956 TR=5
2957 FORE=1TO1000:NEXTE
2960 GOTO9
3000 PRINT @232,"ONE MOMENT PLEA
SE"
3001 FOR A=16128 TO 16730
3005 READ D*
3010 V=VAL("&H"+D*)
3015 POKE A,V
3020 NEXT
3025 RETURN
3030 DATA 86,16,B7,3F,56,10,8E,3
F
3035 DATA 5B,8E,3F,58,C6,02,A6,A
0
3040 DATA A7,80,5A,26,F9,30,88,1
E
3045 DATA 7A,3F,56,26,EF,39,86,1
4
3050 DATA B7,3F,5A,10,8E,3F,87,B
E
3055 DATA 3F,58,C6,03,A6,A0,A7,8
0
3060 DATA 5A,26,F9,30,88,1D,7A,3
F
3065 DATA 5A,26,EF,39,86,16,B7,3
F
3070 DATA 57,8E,3F,58,86,00,C6,0
3
3075 DATA A7,80,5A,26,FB,30,88,1
D
3080 DATA 7A,3F,57,26,F1,39,FF,0
0
3085 DATA 1A,5E,FF,0A,80,0A,80,2
A
3090 DATA AB,0F,C0,2B,70,2B,FC,A
B
3095 DATA FC,AB,C0,0F,F0,0F,F0,3
E
3100 DATA 80,3E,80,3E,80,3E,A0,3
F
3105 DATA 88,3F,08,0F,08,0A,A0,0
A
3110 DATA 80,0A,80,0F,F0,0F,FC,0
0
3115 DATA AB,00,00,AB,00,02,AA,8
0
3120 DATA 00,FC,00,02,B7,00,02,B
F
3125 DATA C0,0A,BF,C0,0A,BC,00,0
0
```

```

3130 DATA FF,00,00,FF,00,3F,AF,0
C
3135 DATA 3F,AF,FC,00,AF,FC,00,A
A
3140 DATA 00,00,AA,0C,0E,AA,BC,0
E
3145 DATA AA,AC,0E,A0,AC,0C,00,A
C
3150 DATA 0C,00,00,02,A0,02,A0,2
A
3155 DATA AB,03,F0,0D,EB,3F,EB,3
F
3160 DATA EA,03,EA,0F,F0,0F,F0,0
2
3165 DATA BC,02,BC,02,BC,0A,BC,2
2
3170 DATA FC,20,FC,20,F0,0A,A0,0
2
3175 DATA A0,02,A0,0F,F0,3F,F0,0
0
3180 DATA 2A,00,00,2A,00,02,AA,8
0
3185 DATA 00,3F,00,00,DE,80,03,F
E
3190 DATA 80,03,FE,A0,00,3E,A0,0
0
3195 DATA FF,00,00,FF,00,30,FA,F
C
3200 DATA 3F,FA,FC,3F,FA,00,00,A
A
3205 DATA 00,30,AA,00,32,AA,B0,3
A
3210 DATA AA,B0,3A,0A,B0,3A,00,3
0
3215 DATA 00,00,30,02,A0,02,A0,0
A
3220 DATA AB,03,F0,02,A0,02,A0,3
A
3225 DATA AB,3A,AB,33,F0,3F,BC,3
F
3230 DATA BC,02,AC,02,AC,02,AC,0
2
3235 DATA A0,02,A0,02,A0,03,20,0
F
3240 DATA 20,00,20,00,30,00,3C,0
2
3245 DATA A0,02,A0,0A,AB,03,F0,0
2
3250 DATA A0,02,A0,0A,AB,0A,AB,0
3
3255 DATA F3,0F,BF,0F,BF,0E,A0,0
E
3260 DATA A0,0E,A0,02,A0,02,A0,0
2
3265 DATA A0,02,30,02,3C,02,00,0
3
3270 DATA 00,0F,00
3280 DATA BE,03,FF,30,01,A6,84,2
C,04
3290 DATA 8B,10,8A,80,A7,80,8C

```

```

3300 DATA 06,01,2F,F1,39
3310 DATA 12,BE,41,22,30,01,BF,4
1,28,8D,56,BE,41,24,30,1F,BF,41
3320 DATA 28,8D,4C,BE,41,26,30,0
1,BF,41,28,8D,42,BE,41,22,8C,12
3330 DATA 41,27,4D,8D,1A,30,1F,B
F,41,22,BE,41,24,8D,10,30,01,BF
3340 DATA 41,24,BE,41,26,8D,06,3
0,1F,BF,41,26,39,10,8E,41,2A,BF
3350 DATA 41,28,C6,06,A6,A0,A7,8
0,30,88,1F,5A,26,F6,A6,9F,41,22
3360 DATA B7,41,21,BE,41,28,39,B
E,41,28,86,0,C6,06,A7,80,30,88
3370 DATA 1F,5A,26,F8,BE,41,28,3
9,8E,12,59,BF,41,22,8E,17,46,BF
3380 DATA 41,24,8E,1C,59,BF,41,2
6,7E,40,99,0,12,59,17,46,1C,59
3390 DATA 12,5D,41,41,14,14,55,5
5
3400 DATA BE,41,22,30,01,8D,0F,B
E,41,24,30,1F,8D,08,BE,41,26
3410 DATA 30,01,8D,01,39,A6,84,8
1,41,26,07,39,86,64,B7,41,5A
3420 DATA 39,A6,84,81,00,26,F4,3
9,0

```

Hint . . .

Simplified Saves

For disk users:

```

3 GOTO 10
5 KILL "PROGRAM/BAS" : SAVE"PROGRAM" :
END
10 ' THIS IS FIRST LINE OF PROGRAM

```

To use, first *SAVE "PROGRAM"* (your program name) in the usual fashion. Thereafter, just type *RUN 5 ENTER*. This is especially handy during debugging.

For cassette users, change Line 5 to read:

```

5 FOR S = 1 TO 3 : CSAVE"PROGRAM" : MOTORON
: FOR DL* 1 TO 460 *3 : NEXT DL : MOTOROFF :
SOUND 200,1 : NEXT S : END

```

To use, set the recorder to record mode, then type *GOTO5 ENTER* or *RUN5 ENTER*. Line 5 will then *CSAVE* the program three times, putting a three second "rewind gap" between each of the saves and after the last save. The *SOUND* command may be deleted, but it's a handy signal which reminds you to jot down the tape counter number.

A special note to *Worksaver* users, cassette or disk: store the applicable Line 5 from the above as a key definition, then to do a whole save routine it takes only two (at most three) keystrokes!

Chris W. Brown
Siloam Springs, AR

GOCO

Did you read The Bulletin of August 7?

Batman has discovered the Model 100! Congratulations mate - enjoyed the article!

Like you, thousands are discovering that they can use the Model 100 for serious word processing, communication with other computers, games and a wide range of other applications.

And have you seen the Model 2000? The 2000 is out IBMing the IBM. The machine

is brilliant.

Next month we hope to start a series of articles on this computer if we can make enough sense of it to write something worthwhile!

In the meantime PCM for last month and this month arrived and we've crammed in as much of each magazine as we possibly could.

If you need bar codes let us know.

Mastering MS-DOS

Part II

More basics on using MS-DOS and an introduction to MS-DOS commands

By Danny Humphress
PCM Technical Editor

Congratulations are in order for you brave souls who joined me on my trek through the outer barriers of MS-DOSdom. Perhaps some of you were eager to forge on into the unknown without my wise guidance. Don't be so anxious, my friends, that you will go on ahead of the party and find yourselves lost in the abyss of directories, files and commands. We're going to take it slowly and carefully, shining our lamp beams down each dark corridor before we proceed, and taking care not to overlook any precious finds along the way.

If you'll remember from our previous explorations, we learned about what MS-DOS does and how it communicates with the components of the computer. We also uncovered some of the mysteries of files and directories. As you learn MS-DOS experience through our travels, you will gain wonderful insights as to how to put what you've learned to use.

In the excursion into the world of MS-DOS, we're going to finish going over basic MS-DOS survival tactics and begin to discover some of the many commands that our operating system provides.

Entering The World Of MS-DOS

You enter the World of MS-DOS each time you turn on your Tandy 2000 and put a disk in. Tandy 2000HD users have MS-DOS stored on their hard disk, so they don't even need a floppy disk. After the computer briefly checks itself and makes sure that there are no obvious equipment problems, it starts reading MS-DOS from the floppy or hard disk. It is at this point that MS-DOS takes control and displays its name, version number, and a lot of copyright notices to give credit where credit's due.

MS-DOS is so disoriented when you

wake it that it doesn't even know what day or time it is. It will ask you first for the date and then for the time. It always thinks it's January 1, 1980 when you turn it on - talk about disorientation! Unlike most computer programs though, MS-DOS understands if you don't like to use leading zeros and slashes and the like. To enter a date of June 9, 1984 for example, you would type any of the following:

6 9 84 06 9 84 06 09 1984
6-09-84 6-9-84

You can also just bypass entering the date and let the computer think that it really is January 1, 1980, by pressing ENTER without entering a date. I don't suggest this, however. As you will see, MS-DOS needs to know the correct date in order to give you correct information and for certain commands to work properly.

MS-DOS is a bit more particular about how you enter the time. It requires you to use European (military) time conventions (2:15 p.m. is 14:15). It also wants you to use colons between the hour, minutes and seconds and a decimal point between the seconds and hundredths of seconds if you want to be that accurate. The general format for entering the time is HH:MM:SS.n where HH is hours, MM is minutes, SS is seconds and nn is hundredths of seconds. You need not bother with entering the seconds or fractions of seconds if you don't want - just entering as much of the time as will suit your purposes. The following are examples of properly entered times:

13:01:57.90 (1:01 p.m. 57.9 seconds)
 11:05:41 (11:05 a.m. 41 seconds)
 14:48 (2:48 p.m.)
 16 (4:00 p.m.)

As with the date, you can just press ENTER here without entering a time. MS-DOS will start with 00:00:00.01 and count from there. Entering the time is not nearly as important as entering the date (I usually skip it), but it can be useful if you want to keep track of when during a day files are updated — more about that later.

MS-DOS Is At Your Command

Once it knows the date and time, MS-DOS stops asking questions. It is up to you to tell it — in its own language, of course — where you want to go and what you want to do. It just displays a "prompt" on the screen and patiently awaits your command.

One of two prompts may appear on the screen depending on whether you are using a floppy disk or not. If you are using a floppy disk system, or you have a disk in the floppy drive of your Tandy 2000HD, you will get a "A>" prompt. Hard disk booters are greeted with a "C>" prompt.

Remember in May when we talked about device names. Do 'A' and 'C' look familiar to you? They are device names for disk drives. 'A' is the name of the first (bottom) floppy disk drive and 'C' is the name of the hard disk drive.

What MS-DOS is telling you here is that any command you enter now will, unless you specify otherwise, take place on this particular disk drive. This is called the *default* drive. If you do not tell MS-DOS on which drive to perform a command, it will use the default drive. Likewise, if you want to access a file on drive B; and A; is your default, you must either change the default or specify drive B; when you access the file. This is important to remember.

Because the hard disk drive C; is the most often used drive on a hard disk system, the default drive if you have a hard disk is drive C;. Any commands you enter will default to drive C;.

If you want to change the default drive to another drive, you simply type the drive letter followed by a colon and press ENTER. For example, to change the default drive to B;, enter B;. That's simple enough.

Communicating With MS-DOS

As with most computers, we communicate with our Tandy 2000 through its operating system (MS-DOS) by way of the standard human interface (the keyboard). MS-DOS does not care whether we enter our requests in UPPER-CASE or lower case or any CoMbinAtIoN of upper and lower. It gets our message either way.

You let MS-DOS know your wishes by using specialized commands that it understands. Most of these commands involve "parameters" that give the specifics of how the command is to work and "path names" that tell MS-DOS which files and which disks or devices to use. A parameter usually follows the command. Each command has its own special syntax that we must learn in order to use them properly. Fortunately, most commands are similar enough for us to figure them out once we know the basic rules.

Some commands have required and optional parameters. For instance, we may not need to enter a disk drive name because we want the computer to use the default drive.

In general, there are only a few basic types of command parameters. There are *filespecs* (file specifications) which include any or all of a drive name, directory names, and a file name. There are *arguments* which are a set of parameters from which you choose such as "ON" and "OFF." And we have *switches* that tell the command to act in a certain way depending upon the specific command.

As we learn about each individual

command in MS-DOS, we'll explore the parameters that the command uses and how it changes the results. Once you begin using some of the "everyday" commands, you will be able to use your common sense to figure out how to use the others without ever having to crack the MS-DOS manual.

Learning The Native Tongue

MS-DOS has a language all its own. While it's very similar to our familiar English in many ways, it's more similar to Orwell's Newspeak with its rigid rules and sterile, efficient syntax. Fortunately for us, though, learning the language of MS-DOS is not nearly as difficult as learning a human language. There are only a few dozen words and only a handful of those will be used in everyday communication with your computer.

Unlike the *MS-DOS Reference Manual* that was packed with your new Tandy 2000, we're not going to go over each command in alphabetical order from BACKUP to VOLUME. Instead, we'll begin with the fundamental commands and work our way up to the bells and whistles.

It's now time to hit the power switch on your Tandy 2000 (turn it on — not off) and get ready to do some exploring!

Format

Before MS-DOS can begin to put data on a floppy disk, the disk needs to be prepared to receive the data. This is called "formatting" a disk.

Have you ever tried to write a lengthy letter on a piece of blank, unruled paper? Of course you have. It's not easy to make nice neat lines across that paper, is it! It's even more difficult for a computer to write data on a blank floppy disk. Orderly as it is, the computer needs to be able to write the data on the disk in a neatly organized fashion. It needs to have those little lines to guide it along the disk. While formatting does not physically put lines on a disk, the effect is very similar.

The first thing you must do to a new disk is to format it. And this is the first command we're going to learn about.

Let's get started by "booting" (getting everything up and running) our system with the disk labeled *MS-DOS/BASIC* that came with your Tandy 2000 in the bottom drive. Enter the current date and time when the computer asks. If you have a Tandy 2000HD with a hard disk, just turn on the computer — there is no need for a diskette (you should have already followed the directions that came with your 2000HD that explained how to initialize your hard disk).

After entering the date and time, you'll be greeted with an "A>" prompt ("C>" if you have a hard disk).

Get a blank disk. If you don't have one, go out right now and buy a box — you'll need them. Put this disk in the top drive of your computer (drive B;). What! No top drive? If you have no top drive, you have a Tandy 2000HD with a hard disk. Put the blank diskette in the only disk drive (drive A;).

Now, are we ready? Type the following command:

```
FORMAT B:/S or
FORMAT A:/S if you don't have
a hard drive
```

We are telling MS-DOS to prepare the disk in drive B; (or A;). The "/S" is a "switch parameter" that tells the format command to make room, and copy MS-DOS to this disk. When a disk has MS-DOS on it, it is called a "system" disk. A disk with the system on it should always be in drive A; when you're using the computer. If you have a hard disk, the system is stored on it and you don't need to have a system disk in drive A;. More on this later.

MS-DOS's FORMAT command will display:

Insert new diskette for drive B;

and strike any key when ready

Put the disk in the specified drive, if it is not already there, and press any key on the keyboard.

Important: Format *completely erases* anything that is on a disk, so be sure that you are formatting a new disk or that you really want to erase the disk in the format drive!

FORMAT will display "Formatting tracks" and a line of eighty dashes across the screen. As format works, each dash will change to a period. If a dash changes to a question mark, there may be problems with the disk (or your disk drive). Try the process again with the same disk and, if you get the same results, try another new disk.

When the format is finished, you will be asked if you want to format another disk. You can go on formatting as many disks as you like. For now, press 'N' for no.

We will again have an "A>" or "C>" prompt telling us that MS-DOS is awaiting our next orders. The disk in the drive is newly formatted and is now ready for computer use.

There are a couple more things that FORMAT can do, but we'll save that for a little later.

The Most Important Commands

It is amazing how much we trust to the whim of a machine. We entrust this unthinking box with some of our most valuable possessions — time and money. When we store our precious data on a computer's floppy or hard disk, we assume that it will be safe and sound and that only a natural disaster could bring harm to it. Not so, my friends! What would happen if, for instance, the computer that handles PCM's mailing list decides to delete all the names and addresses — it did happen. Can you imagine what it would take for us to rebuild this mailing list! Our only savior was that we religiously make "backup" copies of the data for just such an emergency. What could have been weeks of work and thousands of dollars in expense turned out to be only a minor inconvenience.

When you purchased your Tandy 2000, you received a single disk entitled *MS-DOS/BASIC*. This is your "master" MS-DOS disk. You should use it for only one thing — making a copy of itself. This is true of any software package that you purchase. Use the original to make a copy and put it away in a safe place. Note, however, that some software cannot be copied. The software publishing company usually gives you a spare or offers to replace it for a nominal fee.

Once you start using a program, the information on the disk or disks becomes even more valuable than the program itself because you have added to it what cannot be replaced by a software publisher — your own data. It is imperative that you copy this important data on a regular basis (and keep several copies) to avoid a "data disaster."

The same holds true, even more so, if you are keeping your programs and data on a hard disk. There is so much to lose if something goes haywire with your hard disk drive.

MS-DOS provides several ways of copying entire disks. To copy a floppy disk, the most common method is to use COMPDUPE. This command performs several functions. It will format a blank disk, copy the entire contents of the disk in drive A; to drive B;, and compare the two copies to make sure that there were no errors in copying. If you are using a Tandy 2000HD, you may want to just skim over the discussion of COMPDUPE; there is another copy command made just for your hard disk.

COMPDUPE is short for "compare duplicate," which really sums up what this command is all about. It makes a

mirror-image copy of the floppy disk in drive A; onto the disk in drive B;.

Put your MS-DOS master disk in drive A; and a blank (or formatted) disk in drive B;. Type the following command:

```
COMPDUPE /D
```

The "/D" is a switch parameter telling COMPDUPE to work in the "duplicate disk mode." If you don't use the "/D," COMPDUPE will simply compare the two disks and not copy them.

The screen will clear and a copyright message will appear. You are told to press the space bar to continue or to press CONTROL 'C' to abort. Press the space bar and let's get things moving.

As with FORMAT, there will be a line of 80 dashes across the screen. During the process, each dash should turn into a period. If not, there may be problems with one or the other disks or the computer. Try again.

After successfully copying and comparing the disks, you will be asked if you want to copy another. Like FORMAT, you can do this till the power goes off or you run out of disks. Let's just press 'N' for no and return to the "A>" prompt.

For those of you who are using Tandy 2000HDs with hard disks, there is a special command for copying data from the hard disk to floppy disks.

The reason we need a special command is that a hard disk can store many times the amount of data that can fit on a floppy disk. The hard disk's BACKUP command allows you to copy all or a portion of the hard disk onto multiple formatted floppy disks.

BACKUP is a very powerful command with many parameters. For today, we'll use it in its simplest form. I could spend a whole month's column just on the different uses of BACKUP (and I will later).

Unlike COMPDUPE, BACKUP will not format disks for you. You must use the FORMAT command to get enough disks ready before you use BACKUP. It will take about 14 disks if your hard disk is completely full and you are copying the entire disk. It is a good idea to have a bunch of formatted disks on hand, because once you start your backup and you run out of disks, you'll have to quit, format more disks, and start the backup process from square one.

If you just got your computer, chances are that your hard disk will be almost empty. Going from that assumption, format two or three disks to get ready to make a backup.

With formatted disks in hand, type the following command at the "C>" prompt:

```
BACKUP C: A: S
```

This tells MS-DOS to copy the contents of drive C; to drive A;. The "S" is a switch parameter telling BACKUP to copy all the files in all the directories on the hard disk.

You will be warned by BACKUP that you will erase all the files on the destination disk (A;) and asked to insert a disk to receive the backup data in drive A;.

When you strike any key to continue, the backup process will begin. Data will be read from the hard disk (C;) and copied to the floppy disk (A;). If the floppy disk fills, you will be prompted to insert another disk in drive A; and the process will continue.

BACKUP has a sister command, RESTORE, that moves data from backup disks to the hard disk drive. We'll explore this command a little later.

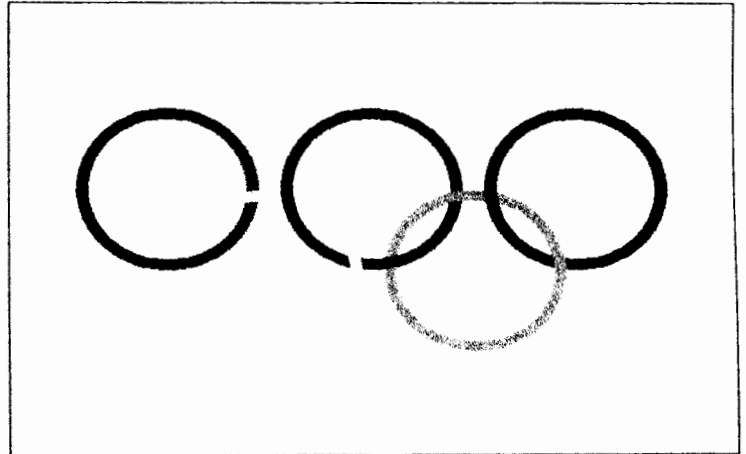
Next Month

In July, we'll explore some of the other fundamental commands of MS-DOS. Of course, you will by now be using some of these commands on your own — good. After we've introduced ourselves to them, we'll take a closer look at what makes them tick. Until then

The 2000 Gallery

Flag of the Month Club

By Wayne Sanders



1984 will be remembered as the year of Michael Jackson, Indiana Jones, Boy George, Gremlins, the Model 2000, George Orwell, a presidential election, break dancing and the 1984 Olympic Games. This month's flag

honors the Winter Games in Sarajevo and the Summer Games beginning this month in Los Angeles.

This program beautifully reproduces the Olympic Flag on the monitor of a color-equipped 256K Model 2000. It can be printed on a CGP-220 ink jet printer

by changing the first line to read:

1000 SP=1

You must have *CGPDMP.BIN* on your disk and enter BASIC using the following MS-DOS command line:

BASIC /M:&HFF00

The listing:

```

1000 SP=0
1010 IF SP THEN CGPDMP=&HFF00:BLOAD"CGPDMP.BIN",CGPDMP
1020 SCREEN 3:PALETTE 1,4:PALETTE 2,2:PALETTE 3,6:PALETTE 4,1:PALETTE 5,5:PALETTE 6,3:PALETTE 7,7
1030 CLS:KEY OFF
1040 DEGREE=1.745329E-02
1050 LINE (1,1)-(638,398),7,BF
1060 CIRCLE (140,164),70,4
1070 CIRCLE (140,164),80,4
1080 PAINT ( 65,164),4,4
1090 CIRCLE (320,164),70,0
1100 CIRCLE (320,164),80,0
1110 FAINT (245,164),0,0
1120 CIRCLE (500,164),70,1
1130 CIRCLE (500,164),80,1
1140 PAINT (425,164),1,1
1150 CIRCLE (230,236),70,3
1160 CIRCLE (230,236),80,3
1170 PAINT (155,236),3,3
1180 CIRCLE (410,236),70,2
1190 CIRCLE (410,236),80,2
1200 PAINT (335,236),2,2
1210 CIRCLE (140,164),70,4,225*DEGREE,315*DEGREE
1220 CIRCLE (140,164),80,4,225*DEGREE,315*DEGREE
1230 PAINT (155,230),4,4
1240 CIRCLE (320,164),70,0,135*DEGREE,225*DEGREE
1250 CIRCLE (320,164),80,0,135*DEGREE,225*DEGREE
1260 PAINT (245,170),0,0
1270 CIRCLE (320,164),70,0,270*DEGREE,315*DEGREE
1280 CIRCLE (320,164),80,0,270*DEGREE,315*DEGREE
1290 PAINT (335,230),0,0
1300 CIRCLE (500,164),70,1,135*DEGREE,225*DEGREE
1310 CIRCLE (500,164),80,1,135*DEGREE,225*DEGREE
1320 PAINT (425,170),1,1
1330 IF SP THEN CALL CGPDMP(ER%)
1340 GOTO 1340

```

Introducing dBASE II

This month we begin a second major tutorial series for the Model 2000 — 'Introducing dBASE II.' As a database system, dBASE towers

Part I

By Danny Humphress
PCM Technical Editor

above the rest, we feel this series by PCM Technical Editor Danny Humphress will help you rise to its challenge and potential.

Today's business world is turning away from the filing cabinets of the past and looking toward computers as the filing system of the '80s. Electronic database software systems for microcomputers are as plentiful today as blades of grass on a golf course. Towering above all the rest in popularity is Ashton-Tate's dBASE II. While it is not the most feature loaded or easiest to use, dBASE is a pioneering piece of software. It introduced the computer world to new concepts in database management. dBASE is the *de-facto* standard to which all other database systems are compared.

Perhaps the most innovative thing about dBASE is that it goes beyond conventional database programs in that it allows one to actually write programs in its specialized language. dBASE as a language is nothing to laugh at. It is a nicely structured language with a wealth of features. Programs that may take weeks to write in BASIC, COBOL, or another application languages, can usually be done in a day or so with dBASE.

That's what this series of tutorials is going to lead to — using dBASE as a programming language. We'll assume that you have no knowledge of dBASE or any other programming language, and start at the first rung of the dBASE ladder. You'll have the opportunity for a lot of hands-on experience, so having access to dBASE II, while not necessary, will be very helpful.

Before we can learn *how* dBASE works, let's talk about

what dBASE actually does.

dBASE is a *database manager*. This simply means that it is designed for storing and retrieving records. You can think of dBASE as a smart Rolodex file. You have a file box full of cards. Each card may contain "fields" such as name, street, city, state, ZIP code, and telephone number.

When you want to find the address for, say Marla Smith, dBASE gets it for you. You can do that with a \$10 Rolodex, you say. Maybe so, but suppose you want to send a letter to all the people in your file who live in Kentucky or Indiana. You would have to go through all the cards in your file box, pulling out only those that meet your criteria. This could turn into a major job, but it's a piece of cake for dBASE. This is just the beginning of the capabilities of dBASE.

Using this example, our Rolodex box is a "file" to dBASE. There can be an unlimited number of files on a disk, depending upon the size of the files and the storage capacity of the disk. dBASE can even work with two files at the same time. Each card in the Rolodex is a "record." dBASE can handle over 65,000 records in a single file — that's a pretty big file box!

On each of the file cards (records), we have one or more "fields." Fields define the specific information that we want to keep in each record. Name, street, city, state, ZIP code, and telephone number are the fields in this example. Each record in a dBASE file can have up to 32 fields. Our sample database would look like this:

Creating A Database File

Before you begin using dBASE as your filing system, you have to tell it what information you want to keep in your files. You give the file a name (which can be up to eight characters), and tell it the names, types, and sizes of each of the fields. This is done with dBASE's CREATE command.

Let's convert our address Rolodex file to dBASE. Put the dBASE disk in your computer and type "dBASE" from MS-DOS. There will be a short copy-right message on the screen followed by a period and a cursor. The period, or "dot prompt," tells you that dBASE is awaiting your command.

We will name this file "MAIL." To create the file, type the following:

```
CREATE MAIL Press ENTER
```

If you just typed "CREATE," without the file name, dBASE would ask you for a file name.

dBASE prompts you to enter four pieces of information for each field in your database: field name, type, length, decimal places.

The field name can be up to 10 alphanumeric characters, you can also use colons (:) in your field names — handy for separating a two-word field name.

A field can have one of three types: character, numeric, or logical. Character fields can store letters, numbers, punctuation marks — anything. Numeric fields can store only digits, a decimal point, and a minus sign. Logical fields can either store a 'T' for true or an 'F' for false.

The length of a field determines how much room you'll have for entering data into this field. You would want to make the name field in our example large enough for most names, yet not overly large. If made too big, you'll be wasting storage on your disk. If you made the length 100 characters, each record would reserve 100 spaces for the name. Even if a name were very short, it would still take the same amount of space as a 100-character name.

Decimal place tells dBASE where to put a decimal, if any, in a numeric field. You would usually use a decimal place of two for dollar amounts.

We'll use the following table for our mailing list database:

FIELD NAME	TYPE	LENGTH	DECIMAL
NAME	C	35	
STREET	C	35	
CITY	C	20	
STATE	C	2	
ZIP CODE	C	10	
TELEPHONE	C	13	
YTD SALES	N	10	2

I've added a year-to-date sales field to our database to show you an example of a numeric field with two decimal places. Since we've allocated ten spaces for this field including two decimal places, the largest number that we could have here would be 9999999.99 (the decimal place counts as one space).

When you tell dBASE about your fields, use this format:

```
FIELD NAME, TYPE, LENGTH, DECIMAL
```

Separate each with a comma. If you don't need the decimal places, just enter the field name, type, and length — forget the last comma and decimal. After we've told dBASE to CREATE a file named "MAIL," we need to enter the field information. Enter it as follows:

ing the fields.

Once you've completed this, you have created a file that is ready to receive our data and dBASE will ask you if you want to add records at this time for now say 'N' for no and you'll see another dot prompt telling you, once again, that dBASE is awaiting your next command.

Using A Database

OK, we've set up our database, now let's enter some names into it. First, we must tell dBASE which file we want to work with, I know — we just created a new database — doesn't dBASE remember? No, it does not. We have to remind it. Once we've told it, though, it will remember which file we're using until we either leave dBASE or use another file.

This brings us to a new dBASE command, USE. Type this (at the dot prompt, of course)

```
USE MAIL Press ENTER
```

We've just told dBASE that all our subsequent commands will deal with the "MAIL" file. After pressing ENTER, we'll see our familiar dot prompt. If dBASE could not find the file on the disk (perhaps we typed it wrong), it would tell us so and give us a chance to correct it.

Appending Records

We're all ready to start adding records to our database file. This is done with the APPEND command. Type:

```
APPEND Press ENTER
```

Voila! A blank form is displayed on the screen with our field names in a column followed by white (or green) boxes for our data. Notice "Record 00001" at the top of the screen. Each record is given a number, starting with one. dBASE is telling us that this is the

DATABASE						
Records →	NAME	STREET	CITY	STATE	ZIP	TELEPHONE
	NAME	STREET	CITY	STATE	ZIP	TELEPHONE
	NAME	STREET	CITY	STATE	ZIP	TELEPHONE
	NAME	STREET	CITY	STATE	ZIP	TELEPHONE
	NAME	STREET	CITY	STATE	ZIP	TELEPHONE
	NAME	STREET	CITY	STATE	ZIP	TELEPHONE

Fields

first record in the database.

Let's go ahead and enter a record. If you make a mistake, just use the arrow keys to move around within the white boxes and type over the hoo hoo. Enter the following data:

```
NAME,C,35      Press ENTER
STREET,C,35   Press ENTER
CITY,C,20     Press ENTER
STATE,C,2     Press ENTER
ZIP CODE,C,10 Press ENTER
TELEPHONE,C,13 Press ENTER
YTD SALES,N,10,2" Press ENTER
```

Notice the '2' after YTD SALES. This tells dBASE that we want two decimal places in this field.

Pressing ENTER for the seventh field instead of entering another field name tells dBASE that you are finished enter-

NAME	PCM
STREET	9529 U.S. Highway 42
CITY	Prospect
STATE	KY
ZIP CODE	40659
TELEPHONE	(502) 228-4492
YTD SALES	5100.50

When you press ENTER after typing the last field (telephone), the record is stored on the disk and a new blank form for the next record (record 2) is displayed. If we were finished entering records at this point, we would press CTRL 'Q' (hold down CTRL, and press 'Q'). Telling BASE to "quit" this operation. We don't want to do that now, though; we want to enter a few more records. Enter the following records on your own.

Radio Shack 300 One Tandy Center Fort Worth, TX 76102 (no telephone number) 123456.78	Portable Computer Support Group 11035 Harry Hines Blvd. Dallas, TX 75229 (214) 351-0564 1000.00
Computer Plus 480 King Street Lynnfield, MA 01460 (617) 456-1191 2000.00	B.T. Enterprises 10 Carlough Road Bohemia, NY 11716-2996 (516) 583-8155 3000.00
Chattanooga Choo Choo P.O. Box 15892 Chattanooga, TN 37415 (615) 815-8586 4000.00	Dr. Preble's Programs 6540 Outer Loop Louisville, KY 40228 (502) 966-8281 5000.00
Computer Solutions Co. 901 Embassy Square Blvd Louisville, KY 40299-1814 (502) 491-6122 6000.00	Prickly-Pear Software 9234 E. 30th Street Tucson, AZ 85710 (606) 886-1505 7000.00
Skyline Marketing Corp. 4510 W. Irving Park Road Chicago, IL 60641 (312) 286-0762 8000.00	Purple Computing 2068 Ventura Blvd. Camarillo, CA 93610 (805) 987-4788 9000.00
Spectrum Projects 4283 Payne Avenue #9866 San Jose, CA 95117 (212) 441-2807 10000.00	Dennison 82 Calvary Street Waltham, MA 02154 (800) 343-8413 11000.00
Traveling Software, Inc. 11050 Fifth Avenue NE Seattle, WA 98125 (800) 343-8080 12000.00	

Wow! All that typing should give you a little experience with entering data into a dBASE file. Press CTRL 'Q' to quit appending.

Listing A Database
Now that you've done all that work, let's take a look at what we have. The LIST command lists the contents of a database. You can be specific about which records you want to see and which fields of those records you want displayed, but for now just type:

```
LIST
Press ENTER
The records in your database are listed in the record number order. The number on the far left of the screen is the record number. There is too much in each record to display it on a single line on the screen, so dBASE continues the record on a second line.
```

See Figure 1.

Suppose we just want to list the names and the year-to-date sales. Enter:

```
LIST NAME,YTD:SALES
Press ENTER
```

This tells dBASE to display only the name and year-to-date sales fields for each record.

See Figure 2.

Now suppose that we want a list of names that have year-to-date sales greater than 5000. Type the following on one line:

```
LIST NAME,YTD:SALES
FOR YTD:
SALES > 5000
Press ENTER
```

We've told dBASE to list the name and amount only for those that are greater than (>) 5000.

See Figure 3.

Finally, we want a list of those located in Kentucky with year-to-date sales of less than 7000. Type the following on one line:

```
LIST NAME,YTD:SALES
FOR YTD:SALES < 7000
AND STATE="KY"
Press ENTER
```

The ".AND." is an example of a "boolean" or "logical" operator. The record will only be displayed if the first and the

GoCo

second conditions are true. Another example of a boolean operator is ".OR." As I am sure you can logically figure, ".OR." would work only if either the first or the second conditions were true. See what happens if you substitute ".OR." for ".AND." in the previous command. Figure 4 shows the results of the ".AND." example.

Finally for this month, a very important command, QUIT. When we're through dBASEing, we need to exit it properly to insure that our database file is properly closed. dBASE will return you to MS-DOS. At the dopt prompt, type

```
QUIT
Press ENTER
```

You're just beginning to see the power and versatility of dBASE II with only a few commands. There are many more commands that do things like edit a record, print a report, sort a file, globally update a file, and even a command to let you change the color of the screen and characters. And this is merely the beginning. When you incorporate these commands into a "command file," which we will be doing in future encounters with dBASE, you can create programs that really make your "Rolodex" come to life.

Until next month, do some experimenting on your own with our sample database file. Try doing some fancy LISTS using ".AND." and ".OR." (try combinations of both on the same line). Add some more records, if you like, but don't erase what you've entered today.

We'll want to use the same file next month when we'll learn about how to edit records and two different ways of sorting our database.

PCOM

Figure 2

LIST NAME,YTD:SALES		
00001	PCM	5100.50
00002	Radio Shack	123456.78
00003	Portable Computer Support Group	1000.00
00004	Computer Plus	2000.00
00005	B.T. Enterprises	3000.00
00006	Chattanooga Choo Choo	4000.00
00007	Dr. Preble's Programs	5000.00
00008	Computer Solutions Company	6000.00
00009	Prickly-Pear Software	7000.00
00010	Skyline Marketing Corp.	8000.00
00011	Purple Computing	9000.00
00012	Spectrum Projects	10000.00
00013	Dennison	11000.00
00014	Traveling Software, Inc.	12000.00

Figure 3

LIST NAME,YTD:SALES FOR YTD:SALES>5000		
00001	PCM	5100.50
00002	Radio Shack	123456.78
00008	Computer Solutions Company	5000.00
00009	Prickly-Pear Software	7000.00
00010	Skyline Marketing Corp.	8000.00
00011	Purple Computing	9000.00
00012	Spectrum Projects	10000.00
00013	Dennison	11000.00
00014	Traveling Software, Inc.	12000.00

Figure 4

LIST NAME,YTD:SALES FOR YTD:SALES<7000 .AND. STATE="KY"		
00001	PCM	5100.50
00007	Dr. Preble's Programs	5000.00
00008	Computer Solutions Company	6000.00

Figure 1

LIST				
00001	PCM	9529 U.S. Highway 42		P
rospect	KY 40059	(502)228-4492	5100.50	
00002	Radio Shack	300 One Tandy Center		F
ort Worth	TX 76102		123456.78	
00003	Portable Computer Support Group	11035 Harry Hines Blvd.		D
alias	TX 75229	(214)351-0564	1000.00	
00004	Computer Plus	480 King Street		L
ittleton	MA 01460		2000.00	
00005	B.T. Enterprises	10 Carlough Road		B
ohemia	NY 11716-2996		3000.00	
00006	Chattanooga Choo Choo	P.O. Box 15892		C
hattanooga	TN 37415		4000.00	
00007	Dr. Preble's Programs	6540 Outer Loop		L
ouisville	KY 40228		5000.00	
00008	Computer Solutions Company	901 Embassy Square Blvd.		L
ouisville	KY 40299-1814	(502)491-6122	6000.00	
00009	Prickly-Pear Software	9234 E. 30th Street		T
ucson	AZ 85710	(606)886-1505	7000.00	
00010	Skyline Marketing Corp.	4510 W. Irving Park Road		C
hicago	IL 60641	(312)286-0762	8000.00	
00011	Purple Computing	2068 Ventura Blvd.		C
amarillo	CA 93010	(805)987-4788	9000.00	
00012	Spectrum Projects	4285 Payne Avenue #9866		S
an Jose	CA 95117	(805)987-4788	10000.00	
00013	Dennison	82 Calvary Street		W
altham	MA 02154	(800)343-8413	11000.00	
00014	Traveling Software, Inc.	11050 Fifth Avenue NE		S
eatle	WA 98125	(800)343-8080	12000.00	

MS-DOSsier

Mastering MS-DOS

Part III — Exploring MS-DOS Commands

By Danny Humphress
PCM Technical Editor

Welcome, explorers, to the third day of our trek through the world of MS-DOS. We've been exploring the outer reaches of this new realm, working our way inward uncovering its many riches. Today's journey will take us even further.

In Part II, we began using MS-DOS's commands to format and duplicate disks and to backup the hard disk. Today we're going to explore many more MS-DOS commands to do everything from clear the screen to printing the list of files on a disk.

Let's Get Started

Get (or make) a backup copy of the MS-DOS disk that came with your Model 2000 and boot your system with it ("boot" is computer talk for "start up"). If you have a 2000HD hard disk system which automatically boots from the hard disk, boot from the floppy anyway.

Don't forget to enter the correct date and time when MS-DOS asks.

As we have already learned, a disk is a collection of files. MS-DOS provides a command to allow us to get a list of the files on the disk. With the backup of your MS-DOS disk in drive A:, type:

DIR Press ENTER

You should get something like what's in Figure 1. DIR displays five columns of information. The first column is the filename, the second is a three-character extension, the third shows the size of the file in bytes, the fourth and fifth columns show the date and time that the file was last updated. We talked about filenames in MS-DOSsier Part I if you would like to refresh your memory at this time.

Suppose you don't care about the size or "change date" of the files — you just want to know which files are on the disk. The "/W" wide display switch of the DIR command gives you that option. Try:

DIR /W Press ENTER

The filenames will be displayed across the screen as in Figure 2.

Often there are more files on a disk than can be displayed on the screen at a time — especially when not using "/W." DIR normally just makes the directory fly by. When it is finished, only the last 20 or so files are still on the screen, the others having scrolled off the top. Another DIR switch solves this problem. "/P" causes DIR to pause until you strike a key after it displays a full screen of directory listings; it then proceeds to display the directory one page at a time. This format of the DIR command is:

DIR /P

Go ahead and try it.

DIR also allows you to get information on a single file. This extended form of DIR is:

DIR path-name

I bet you were wondering when we were going to get to those pathnames you tried so hard to understand in MS-DOSsier Part I. Remember, a path-name is the entire path through levels of directories to a file. Our MS-DOS disk only has a single directory now, so the

Volume in drive A has no label
Directory of A:\

COMMAND	COM	15480	1-01-80	12:00a
DEBUG	COM	11764	2-01-83	10:13a
EXE2BIN	EXE	1649	2-01-83	9:19a
CHKDSK	COM	6330	2-01-83	9:16a
EDLIN	COM	4389	2-01-83	9:31a
PRINT	COM	3808	2-01-83	12:39p
RECOVER	COM	2277	2-01-83	2:22p
SYS	COM	850	2-01-83	2:26p
MORE	COM	4364	1-14-83	6:42p
DISKCOPY	COM	1419	2-14-83	4:39p
LINK	EXE	42330	4-01-83	2:21p
SORT	EXE	1216	2-08-83	7:04p
FIND	EXE	5796	1-14-83	6:35p
FC	EXE	2553	2-01-83	9:36a
COMPDUPE	COM	1704	1-01-80	12:29a
FORMAT	COM	5795	1-01-80	1:44a
ANSI	SYS	2138	1-01-80	1:47a
MAILLIST	BAS	13056	1-01-80	12:09a
GRAPHICS	BAS	9216	11-02-83	2:37p
CGPDMP	BIN	180	11-14-83	12:02p
BASIC	EXE	52064	1-01-80	12:04a
HFORMAT	COM	6291	1-19-84	12:10a
CONFIGHD	BAT	34	1-01-80	12:00a
		23 File(s)	458752 bytes free	

Volume in drive A has no label
Directory of A:\

COMMAND	COM	15480	1-01-80	12:00a
DEBUG	COM	11764	2-01-83	10:13a
CHKDSK	COM	6330	2-01-83	9:16a
EDLIN	COM	4389	2-01-83	9:31a
PRINT	COM	3808	2-01-83	12:39p
RECOVER	COM	2277	2-01-83	2:22p
SYS	COM	850	2-01-83	2:26p
MORE	COM	4364	1-14-83	6:42p
DISKCOPY	COM	1419	2-14-83	4:39p
COMPDUPE	COM	1704	1-01-80	12:29a
FORMAT	COM	5795	1-01-80	1:44a
HFORMAT	COM	6291	1-19-84	12:10a
		12 File(s)	458752 bytes free	

Volume in drive A has no label
Directory of A:\

COMMAND	COM	DEBUG	COM	EXE2BIN	EXE	CHKDSK	COM	EDLIN	COM
PRINT	COM	RECOVER	COM	SYS	COM	MORE	COM	DISKCOPY	COM
LINK	EXE	SORT	EXE	FIND	EXE	FC	EXE	COMPDUPE	COM
FORMAT	COM	ANSI	SYS	MAILLIST	BAS	GRAPHICS	BAS	CGPDMP	BIN
BASIC	EXE	HFORMAT	COM	CONFIGHD	BAT				
		23 File(s)	458752 bytes free						

pathname need only be a filespec.

One of the files on every MS-DOS disk is "COMMAND.COM." Don't worry yourself with what it is now — for our purposes, it's just another file. To find out the size (how much disk space used) of this file and the last change date and time, we would enter:

DIR COM:MAND.COM Press ENTER

DIR will only display information about the file you specified instead of all the files on the disk.

As with all MS-DOS commands, DIR will automatically act upon the default drive unless you specify otherwise. When you see "A>" as your MS-DOS prompt, the default drive is drive A:. To obtain a directory of drive B:, you would put the drive designation directly after the DIR command as in the following examples:

DIR B:
DIR B:\W
DIR B:COMMAND.COM P

Remember, to change the default drive, simply type the drive letter followed by a colon and press ENTER. As an example, to change the default drive to B:, type:

B: Press ENTER

You can try it if you like. The new MS-DOS prompt will be "B>." To use our upcoming examples work properly, change the default drive to B:, type:

The Joker's Wild

This is a good time to bring up the subject of "wild cards." No, not the kind that has won (or lost) so many poker games for you, but similar in nature. Wildcards are used to replace all or part of a filename. The best way to explain wild cards is by a demonstration. Type the following command:

DIR *.COM Press ENTER
(Don't forget the period [:])

You should get a listing similar to that in Figure 3. Only the files with extensions of ".COM" are displayed. The asterisk (*) tells MS-DOS "I don't care what goes here."

Try this:

DIR D:.* Press ENTER

As you can see, you can use the asterisks either in the filename or in the extension or both. The above example will give you a list of all files beginning with "D" such as "DISKCOPY.COM," and "DEBUG.COM."

The asterisks can take the place of many characters. To represent a single or certain number of "mystery" characters, use the question mark (?). Try this example:

DIR ?*X Press ENTER

This gives you a list of all files with any filename and an extension having "X" as the second character. This really isn't a practical example of using the question mark wildcard — and you very well may not see any practical uses of this wildcard now. You'll find, though, that as you use MS-DOS on a daily basis, you'll one day say to yourself "I wish I could..." and this wildcard will be the answer!

COPYING

In the second installment of MS-DOSsier, we made a copy of an entire disk. There are times when you'll need to copy a single file or a group of files to another disk. COPY is the MS-DOS command for this purpose. With COPY, you can copy everything from a single file to an entire disk (it's a very slow way to copy an entire disk).

We'll need to format a fresh disk to use for these examples. You should be an old pro at formatting by this time. If you need a little reassurance, refer to last month's MS-DOSsier.

Once formatted, put the new disk in drive B. If you have only a single floppy disk drive, hold on to the new disk for now and leave your MS-DOS backup in the drive.

Type this command:

```
COPY A:FORMAT.COM B: Press ENTER
```

If you have a two-drive system, you'll see the drive-active lights flashing as the file named "FORMAT.COM" on drive A is copied to drive B. If you are using a single-drive system, MS-DOS will act as though you have a disk in drive A and drive B (even though you have no drive B) and tell you to swap the disks periodically. At one moment, your floppy disk is drive A; — the next, it's B. Doing enough of this disk swapping makes you realize the benefits of having more than one floppy disk drive.

The "A:" preceding "FORMAT.COM" could have been omitted in the above example and MS-DOS would have defaulted to A: (A: is the default drive.) The "B:" however, is mandatory because we are copying to a drive other than the default.

Wildcards can be very effectively used with COPY. Say, for instance, that you want to copy all your BASIC programs (which all have extensions of ".BAS") from drive A: to Drive B:. You would type:

```
COPY *.BAS B: Press ENTER
```

Go ahead and try it. There should be a few BASIC programs on your MS-DOS disk to copy.

To see the results of what we just did, do a directory of drive B:. You should see "FORMAT.COM" along with a few ".BAS" programs.

COPY works in both directions. See if you can evaluate what this command will do and then try it:

```
COPY B:*.* A: Press ENTER
```

Were you right? This command copies all the files on drive B: (*.*) is a "total wildcard meaning all files with any filename and any extension) to drive A:. Of course, the files were already on drive A:, but it copied over them.

Again, we could have omitted the "A:" in the above example, and COPY would have defaulted to drive A: for the same results.

In the examples so far, the copies of the files had the same names as the originals. It is possible and sometimes desirable to give the copy a different name. For example:

```
COPY A:DISKCOPY.COM B:
COPYDISK.COM Press ENTER
```

The file, "COPYDISK.COM" on B: will be an exact copy of "DISKCOPY.COM" on A: with a different name. Using this, it is possible to have more than one copy of a file on a disk. Try this:

```
COPY A:DISKCOPY.COM A:
COPYDISK.COM Press ENTER
```

Take a look at the directory of drive A:. You'll have a new file, "COPYDISK.COM" that is an exact copy of "DISKCOPY.COM." MS-DOS will not let you copy a file onto itself by using the

same "source" and "destination" names there is no need to.

Get This Junk Off The Screen

No, that's not a valid MS-DOS command, but there is a counterpart: C.L.S. C.L.S. is one of the simplest MS-DOS commands. There are no strange parameters to remember and no drive names to worry about. It just means "clear screen," and it does just that. It's really handy to make the screen a bit less confusing to view. Try it:

```
C.L.S. Press ENTER
```

You may need to practice with this command for a while — it's pretty complicated!

You have enough ammunition here to begin seriously using MS-DOS, but we've only begun! There are many more useful commands that we'll be discovering next month. Even the very basic commands that we worked with today have other more advanced uses that we'll be taking a look at later in our journey through MS-DOSdom.

Square Deal

Rick Rothstein

Games make computers fun to use. Although an argument can be made that the Model 100 is already a fun computer to use, it is not an exception to this rule. However, the small screen size and directional viewing requirement of its LCD display generally limit the types of games that can be played to either the operator-vs-computer or the solitaire kind. *Square Deal* is a game that belongs to this latter category.

Square Deal is modeled after a challenging solitaire card game called, simply enough, Poker Solitaire. In that game a deck of cards is thoroughly shuffled and 25 cards are turned up from the pack one at a time, and placed to best advantage into a square grid. The object is to score as many points as possible with the 10 poker hands formed by these five rows and five columns.

Scoring

There are two recognized methods of scoring this game. One is called the American system, which evaluates the hands based on their odds of occurring in a regular poker game. The other method is called the English system and it evaluates hands based on their difficulty in being formed while playing this particular solitaire game. (See Table 1 for a breakdown of these two scoring methods.)

If you score more than 200 points in the American system, or more than 70 points in the English system, then you can consider yourself as having "won" the game. It is no easy feat to score this many points, but you shouldn't mind, in as much as the game is very addicting.

Playing The Game

Of course, the Model 100 will take care of shuffling, dealing and scoring for you. When you first RUN the program, the screen will clear, a five-by-five grid of empty squares (actually, they are rectangles) will be displayed on the left while the first card to be played is shown on the right. Simply move the blinking cursor to the empty space into which you wish to play this card and press ENTER to lock it into place. (Once played, a card cannot be removed from the grid — so plan carefully.) A new card will now be displayed on the right

for insertion into the grid. After the 24th card is played, the 25th will automatically be entered into the remaining empty square for you.

In addition to being able to move the cursor around the grid with the four dedicated arrow keys, *Square Deal* has implemented two additional, diamond-patterned directional key controls. The keys 'S', 'D', 'E' and 'X', keys 'K', 'L' and 'O', will also move the cursor left, right, up and down respectively. You may use any one of these keys for moving the cursor — your choice.

All cursor movements are auto-repeating, that is, holding down any of the directional control keys will continue to move the cursor until released. In addition, the cursor is designed to skip over any cards that have already been played. Because of this, the cursor's movements are implemented as follows:

- 1) RIGHT DIRECTIONAL KEY — moves the cursor from left to right, dropping to the beginning of the next row down when the end of the current row is reached;
- 2) DOWN DIRECTIONAL KEY — immediately drops cursor down to the beginning of the next row;
- 3) LEFT DIRECTIONAL KEY — moves the cursor from right to left, moving up to the end of the preceding row when the beginning of the current row is reached;
- 4) UP DIRECTIONAL KEY — immediately raises cursor to the end of the preceding row.

Once the last card has been entered into the grid, the program will signal that it is scoring the game. At the end of approximately six seconds, both the American and English version scores will be displayed. Pressing any key will then replace the displayed scores with an option menu. (The filled-in grid will remain visible throughout.) Your choices will be as follows:

- 1) Start a new game with a freshly shuffled deck;
- 2) Replay the same 25 cards in the same order of presentation — this will allow you to challenge a friend to beat your score, or allow you to replay the last game for a better score;
- 3) Redisplay the American and English versions of your score;
- 4) Display the names of the 10 poker hands that have been formed.

You may end the game at any time by pressing DEL.

About This Program

Since it would require a "book" to describe the circuitous — yes, even tortuous — combination of commands that serve as my program logic, I will not even attempt to explain *Square Deal* in detail. However, there are a few points about the program which should be discussed.

First, the listing is shown with blank spaces separating most keywords. This was done to make it easier for you to read the listing. I would suggest, however, that these spaces be omitted when you actually type the program in — they take up valuable RAM, slow down program execution and are simply not needed to make the program run. (If your Model 100 only has 8K of memory, then the blank spaces *must* be left out in order to fit the program in.)

I would also like to draw special attention to Line 832 in the program listing. A rather long string constant is included as part of the MIDS command. There are four spaces after the word PAIR, one space after the number 2, one after the word PAIRS, two after 3/KIND, three after STRAIGHT-FLUSH, and two more after 4/KIND.

These spaces are important if you wish the option which displays the names of the poker hands to work properly.

In Lines 530 and 6100 of the program listing, you will find the command POKF 65450.0. The purpose of this poke is to nullify the type-ahead feature which is always active on the Model 100. Poking a zero into this memory location tells the computer that there are no more keystrokes stored in the type-ahead buffer — even if there are. Hence, any spurious or unintended keystrokes entered prior to this poke will be ignored by the subsequent INKEY\$ function. (This poke should *not* be made part of the direct keyboard reading loop, but rather be performed once prior to each keystroke being read — see Lines 6100 and 6101.)

The final thing I wish to discuss concerns random numbers. Line 119 implements the "standard" method of randomizing the Model 100's random number generator. This line, in and by itself, is not sufficient for a game like *Square Deal*. That's because the sequence might still be predictable. For example, if the number of seconds determined by Line 110 is "two" the first time the program is RUN and, say, "10" the second time it is RUN, then these two games will share 17 of the *same* cards, dealt in exactly the same order. After a while, you might start to remember and recognize the various patterns. So, in Line 550, I have added an additional randomizer. Line 550 is a more complex implementation of the standard keyboard input routine which is generally written as 100 KS=INKEY\$: IF KS="" THEN 100 where the line number was selected for example purposes only. As long as no key is pressed, this line would continue to loop back onto itself. Only when a key is pressed will the loop be broken. If we simply add a statement to read a random number for each loop, say, 100 W=RND(1): KS=INKEY\$: IF KS="" THEN 100 then the next random number to be used by the program will depend upon when the user presses a key on the keyboard — a time lapse that will vary from game to game and from user to user.

Table 1. Scoring For *Square Deal*

HAND	English	American
Royal Flush	30	100
Straight Flush	30	75
Four Of A Kind	16	50
Full House	10	25
Flush	5	20
Straight	12	15
Three Of A Kind	3	10
Two Pairs	3	5
One Pair	1	2



The listing:

```

1 REM SQUARE DEAL
2 REM BY
3 REM RICK ROTHSTEIN
4 REM TRENTON, NEW JERSEY
100 DEFINT A-Z: DIM D(52), P(25), S(25), G$(
2), H$(2,5)
110 FOR N=1 TO VAL(RIGHT$(TIME$,2)): W=RN
D(1): NEXT
120 CLS: FOR N=0 TO 2: G$(N)=STRING$(25,48
): NEXT
130 FOR X=14 TO 98 STEP 21: FOR Y=0 TO 52
STEP 13: LINE(X,Y)-(X+19,Y+11),1,B: NEXT:
NEXT
140 IF F=0 THEN FOR N=1 TO 52: D(N)=N-1: N
EXT: D=53
150 PRINT @62,"PLEASE PLAY": PRINT @103,"
THIS CARD": LINE(153,27)-(176,40),1,B: LIN
E(154,28)-(175,39),1,B
500 FOR N=1 TO 25: X=157: Y=30: IF F<>0 THE
N 520
510 D=0-1: M=D*RND(1)+1
515 P(N)=1+(D(M) MOD 13): S(N)=D(M)/13: D(
M)=D(M)
520 C$=CHR$(13*S(N)+P(N)+64): GOSUB 50000
: GOSUB 60000: Z=1: G=0: GOSUB 39000
530 V=1: GOSUB 40000: POKE 65450,0
540 B=0: LINE(X,Y)-(X+15,Y+7),V,BF
550 B=B+1: IF B=12 THEN V=1-V: GOTO 540 EL
SE K$=INKEY$: IF N=25 THEN K$=CHR$(13): GO
TO 600 ELSE IF K$="" THEN W=RND(1): GOTO
550
600 IF K$=CHR$(13) THEN GOSUB 40000: GOSUB
B 50000: GOSUB 60000: MID$(G$(0),31-C-5)*R
=C$: MID$(G$(1),C+5)*R+5=C$: MID$(G$(2),R+
5)*C-5=C$: NEXT: GOTO 700
610 IF INSTR(CHR$(28)+"dDlL",K$)=0 THEN
620
615 GOSUB 50000: Z=1: GOSUB 39000: IF G=0 T
HEN GOSUB 39000: GOTO 530 ELSE 530
620 IF INSTR(CHR$(29)+"sSkK",K$)=0 THEN
640
625 GOSUB 50000: Z=0: G=26-G: GOSUB 39000: I
F G=0 THEN GOSUB 39000
630 G=26-G: R=6-R: C=6-C: Z=1: GOTO 530
640 IF INSTR(CHR$(30)+"eEoO",K$) THEN G=
G-(G MOD 5)-(G MOD 5)<>4*(G MOD 5=0): G
OTO 625
645 IF INSTR(CHR$(31)+"xX",K$) THEN G=G
-(G MOD 5)-5*(G MOD 5)<>0: GOTO 615
650 IF K$=CHR$(127) THEN 10000 ELSE BEEP
: GOTO 540
700 F=0: PRINT @61,SPACE$(12): PRINT @102,
"SCORING..."
710 U=0: T=0: FOR N=1 TO 21 STEP 5: FOR L=1
TO 2: F$="" : H$=STRING$(13,48): S=0: J=0
720 FOR K=0 TO 4: P=ASC(MID$(G$(L),N+K,1)
)-65: F$=F$+RIGHT$(STR$(INT(P/13)),1)
725 P=(P+1) MOD 13: P=13*(P=0): MID$(H$,
P)=MID$(STR$(1+VAL(MID$(H$,P,1))),2,2): N
EXT
730 IF F$=STRING$(5,ASC(F$)) THEN J=1
740 H=INSTR(H$,"1"): IF H>0 AND INSTR(H+1
,H$,"1111")=H+1 THEN S=1
750 IF INSTR(10,H$,"1111")>0 AND ASC(H$)
=49 THEN S=2
760 IF J=1 AND S>0 THEN E=30: A=75-25*(S=
2): GOTO 830
770 IF J=1 OR S>0 THEN E=-5*(J=1)-12*(S>
0): A=-20*(J=1)-15*(S>0): GOTO 830
780 IF INSTR(H$,"4") THEN E=16: A=50: GOTO
830
790 IF INSTR(H$,"3")>0 AND INSTR(H$,"2")
>0 THEN E=10: A=25: GOTO 830
800 IF INSTR(H$,"3") THEN E=6: A=10: GOTO
830
810 P=INSTR(H$,"2"): IF P>0 AND INSTR(P+1
,H$,"2")>0 THEN E=3: A=5: GOTO 830
820 E=-P>0: A=-2*(P>0)
830 IF A<25 THEN 0=A/5+1-9*(A=0) ELSE 0=
A/25+5
832 H$(L,(N+4)/5)=MID$( "PAIR 2 PAIRS
3/KIND STRAIGHTFLUSH FL/HOUSE4/KIND
ST/FLUSHROYAL/FLNOTHING",8*Q-7,8)
835 U=U+E: T=T+A: NEXT: NEXT

```

GoCo

```

940 GOSUB 15000: PRINT @66,"SCORES": LINE(
153,5)-(193,17),1,B: LINE(152,4)-(194,18)
,1,B
850 PRINT @143,"AMERICAN =": T: PRINT @223
,"ENGLISH =": U: GOSUB 61000: GOSUB 15000
870 PRINT @62,"PRESS FOR": PRINT @144,"1
NEW DEAL": LINE(131,17)-(161,17): LINE(1
67,17)-(185,17)
880 PRINT @184,"2 SAME DEAL": PRINT @22
4,"3 SCORES": PRINT @264,"4 HANDS"
890 GOSUB 61000: IF K$<"1" OR K$>"4" THEN
BEEP: GOTO 870
900 ON ASC(K$)-48 GOTO 120,1000,840,4000
1000 F=1: GOTO 120
4000 GOSUB 15000: PRINT @63,"ROW COL
UMN": PRINT @101,"-----"
4010 FOR N=1 TO 5: PRINT @101+40*N,H$(1,N
): PRINT @111+40*N,H$(2,N): NEXT
4020 GOSUB 61000: GOSUB 15000: GOTO 870
10000 CLS: PRINT @135,"GAME OVER": PRINT:
PRINT: END
15000 FOR N=20 TO 300 STEP 40: PRINT @N,S
PACE$(19): NEXT: RETURN
20000 LINE(X,Y+7)-(X,Y+3): LINE-(X+2,Y+1)
: LINE-(X+4,Y+3): LINE-(X+4,Y+7): LINE(X+1,
Y+5)-(X+3,Y+5): RETURN
20010 GOSUB 25000: LINE-(X+2,Y+5): LINE-(X
+1,Y+5): LINE-(X,Y+6): LINE-(X,Y+7): LINE-(
X+4,Y+7): RETURN
20020 GOSUB 25000: LINE(X+2,Y+4)-(X+3,Y+4
): LINE-(X+4,Y+5): LINE-(X+4,Y+6): LINE-(X+
3,Y+7): LINE-(X+1,Y+7): PSET(X,Y+6): RETURN
20030 LINE(X+3,Y+7)-(X+3,Y+1): LINE-(X,Y+
4): LINE-(X,Y+5): LINE-(X+4,Y+5): RETURN
20040 LINE(X+4,Y+1)-(X,Y+1): LINE-(Y,Y+3)
: LINE-(X+3,Y+3): LINE-(X+4,Y+4): LINE-(X+4
,Y+6): LINE-(X+3,Y+7): LINE-(X+1,Y+7): PSET
(X,Y+6): RETURN
20050 LINE(X+3,Y+1)-(X+2,Y+1): LINE-(X,Y+
3): LINE-(X,Y+6): LINE-(X+1,Y+7): LINE-(X+3
,Y+7): LINE-(X+4,Y+6): LINE-(X+4,Y+5): LINE
-(X+3,Y+4): LINE-(X+1,Y+4): RETURN
20060 PSET(X,Y+2): LINE(X,Y+1)-(X+4,Y+1)
: LINE-(X+4,Y+2): LINE-(X+2,Y+4): LINE-(X+2,
Y+7): RETURN
20070 PSET(X,Y+3): GOSUB 25000: LINE-(X+3,
Y+4): LINE-(X+1,Y+4): LINE-(X,Y+5): LINE-(X
,Y+6): LINE-(X+1,Y+7): LINE-(X+3,Y+7): LINE
-(X+4,Y+6): PSET(X+4,Y+5): RETURN
20080 LINE(X+1,Y+4)-(X+3,Y+4): PSET(X,Y+3
): GOSUB 25000: LINE-(X+4,Y+5): LINE-(X+2,Y
+7): PSET(X+1,Y+7): RETURN
20090 FOR XA=0 TO 4 STEP 2: LINE(X+XA,Y+1)
-(X+XA,Y+7): NEXT: PSET(X+3,Y+1): PSET(X+3
,Y+7): RETURN
20100 LINE(X+2,Y+1)-(X+4,Y+1): LINE(X+3,Y
+2)-(X+3,Y+6): LINE-(X+2,Y+7): LINE-(X+1,Y
+7): LINE(X,Y+6)-(X,Y+5): RETURN
20110 GOSUB 25000: LINE-(X+4,Y+5): LINE-(X
+2,Y+7): LINE-(X+1,Y+7): LINE-(X,Y+6): LINE
-(X,Y+3): LINE-(X+2,Y+5)-(X+4,Y+7): RETURN
20120 LINE(X,Y+1)-(X,Y+7): LINE(X+4,Y+1)-(
X+1,Y+4): LINE-(X+4,Y+7): RETURN
25000 LINE(X,Y+2)-(X+1,Y+1): LINE-(X+3,Y+
1): LINE-(X+4,Y+2): LINE-(X+4,Y+3): RETURN
30000 DATA 4,5,3,6,2,6,1,5,0,7,1,5,2,6,3
,6,4,5
30010 DATA 1,3,0,4,0,5,1,6,2,7,1,6,0,5,0
,4,1,3
30020 DATA -2,-2,4,4,3,5,2,6,1,7,2,6,3,5
,4,4,-2,-2
30030 DATA 4,5,3,6,1,6,0,5,0,7,0,5,1,6,3
,6,4,5
39000 G=INSTR(G+1,G$(2),"0")
39010 C=(G-1) MOD 5+1: R=(G-1)/5+1: RETU
RN
40000 X=21*C-5: Y=13*R-11: RETURN
50000 LINE(X,Y)-(X+15,Y+7),0,BF: RETURN
60000 ON P(N) GOSUB 20000,20010,20020,20
030,20040,20050,20060,20070,20080,20090,
20100,20110,20120
60010 IF S(N)=0 THEN RESTORE 30000 ELSE
IF S(N)=1 THEN RESTORE 30010 ELSE IF S(N)
=2 THEN RESTORE 30020 ELSE RESTORE 3003
0
60020 FOR XA=7 TO 16: READ YA,YB: LINE(X+X
A,Y+YA)-(X+XA,Y+YB): NEXT: IF S(N)=3 THEN
PRESET(X+9,Y+2): PRESET(X+10,Y+3): PRESET(
X+12,Y+3): PRESET(X+13,Y+2)
60030 RETURN

```

```

61000 POKE 65450,0
61010 K$=INKEY$: IF K$="" THEN 61010 ELSE
IF K$=CHR$(127) THEN 10000 ELSE RETURN

```

ZOID PATROL

Ben Firschein

You can turn your Model 100 into the arcade game of your dreams. This article describes *Zoid Patrol*, a video game that features graphics, inverse video and music, and makes use of some interesting memory locations. I will discuss how to use these features in your own video games. *Zoid Patrol* runs in 8K. You can remove the REMs following the colon in the program lines to conserve memory.

The Program

In *Zoid Patrol*, you must use the arrow keys to navigate a maze that is patrolled by the dreaded Zoids. The Zoids try to track you down and catch you. If one gets you, it will decimate you: the program will then play a funeral dirge. You can choose the number of Zoids that patrol the maze from one to six where one equals easy and six equals pro.

You gain points by staying alive and hitting targets. Number targets are worth 1,000 times the number on the target, and targets with a "C" on them are worth 500,000 points. Scoring is accompanied by sound effects. You lose 300 points if you hit a non-target barrier, and 10 points if you touch a key other than an arrow key. The program also features "wrap around" — if you go off the screen, you will end up on the other side if there is not a barrier in the way.

The program maintains a RAM file to keep track of the high score. When you run the program it displays the high score and scorer if it has been played at least once since it was loaded. If you beat the high score, the program will inform you (while playing the Marine Hymn) that you have broken the record. It will then ask you for your name. Your name and your score will be stored in the RAM file.

How It Works

The Constants Section (Lines 16-88):

This section shows the memory locations and graphics characters the program uses. Location 65446 stores the code (not in ASCII) of the last key pressed. I use this location instead of INKEY\$, because INKEY\$ has an annoying auto-repeat feature; if you hold down a key for more than a few seconds, the Model 100 will think you pressed it several times. Location 65024 is the start of the screen memory. I PEEK at the screen memory to see if the player has hit a barrier or a target, and if a Zoid has hit a barrier or the player. Since there are 40 characters per line, if "loc" is the location of the player, the positions left, right, above and below the player are respectively: loc-1, loc+1, loc-40, loc+40. Location 63791 is a timer (with values 0-125). I use the timer to randomize the random number generator (see Lines 2000-2060). Line 75 shows how to use inverse video: printing "ESC p" turns on inverse video and printing "ESC q" turns it off. Lines 80-87 are the ASCII codes for the barrier, targets, Zoids and player. Note that inverse video numbers have the same ASCII codes as standard video numbers.

Initialization (Lines 89-120): The program calls subroutines that load music data, print instructions, randomize the number generator, set up the barriers, and set up the Zoids. It then plays a tune (Line 108), and waits for the user to hit a

key. Tunes are stored in a two-dimensional array, T%. The designation T%(4,7) refers to the 7th note of the 4th tune.

Main Program (Lines 150-560): Here is where the program displays the player, and shows the score at the bottom of the screen. The PRINT USING command (Line 165) causes the score to be displayed in a field width of six. Line 200 determines the most recent key pressed (not in ASCII). To find the codes for the keys you want to use in your own video game, write the one-line program: 10 PRINT PEEK(65446):GOTO 10 and then hit the keys with the one-line program running. The codes for the left, right, up and down arrow keys are respectively, 44, 45, 46, 47. Lines 505-530 test for targets and barriers. Lines 540-550 cause the player to wrap around if he or she goes off the screen.

Set Up Barriers (Lines 1000-1042): There are up to 75 barriers or targets on the screen. The chance of getting a bonus barrier (500,000 points) depends on the number of Zoids chosen (2*number/50). If a player selects one Zoid there is a four percent chance of a bonus barrier; if an intrepid (and suicidal) player selects six Zoids there is a 24 percent chance of a bonus barrier.

Randomize (Lines 2000-2060): RND always generates the same random number series. If you use memory location 63791, you can access a timer with 126 possible values, and thus start the generator at one of 126 possible locations. This subroutine uses a FOR loop with an RND statement in it to move the number generator forward.

Set Up Enemy (Lines 3000-3090): These lines use the random number generator to pick a position on the screen to place a Zoid.

Move Enemy (Lines 4000-4070): The strategy the Zoids use to catch the player is quite simple and can be used in other video games that require pursuit. The strategy is two-fold:

1) Convert the location of the player to X,Y coordinates. For each Zoid, convert its location to X,Y coordinates. Take the sign (+ or -) of the difference of the X coordinates to find out which direction to move to decrease the horizontal distance between the Zoid and the player. Do the same with the Y coordinates to determine which direction to move to decrease the vertical distance.

2) If the Zoid will not go off the screen and there is not a barrier in the way, move the Zoid. Otherwise, move the Zoid in a random direction.

This simple strategy makes the Zoids very efficient at tracking; down the player and at extracting themselves from dead ends in the maze. One advantage the player has, however, is that he or she can use "wrap around" to move to the opposite side when in imminent danger of destruction. Instead of wrapping around, the Zoids will start moving over to the other side.

Dead (Lines 5000-5190): All players will eventually succumb to the impulse to risk themselves in the quest for a 500,000 target dangerously close to some Zoids. This section of the program provides the appropriate sound effects and graphics for their demise. After the word "MUNCH" appears on the screen, the Zoid will blink on and off to the accompaniment of random tones (Lines 5020-5035). The words "YOU ARE DEAD" then blink on and off in inverse video, synchronized with a bit of "funeral" music (Lines 5050-5070). Line 5060 plays the odd notes, and Line 5067 plays the even notes. If you have broken the high score, then it is recorded in Lines 5107-5147.

Load Music Data (Lines 6000-6240): The notes for the music are stored here.

You can incorporate this subroutine into your own video game to add sound effects to it.

Instructions And Specs (Lines 7000-7300): Line 7012 opens the RAM file that stores the high score and scorer. If the file is not present, the "error interrupt" declared in Line 10 of the program causes the program to continue running. If there is a high scorer, the score and the scorer are displayed. The program then displays instructions and asks for the number of Zoids.

With the techniques used in *Zoid Patrol* you should be able to write your own video game. Space Wars! Computerized Ping Pong! Maybe even the definitive *Porta-Kong!* The possibilities are endless.

The listing:

```

1  ZOID PATROL
2  BEN FIRSCHEIN
3  DECEMBER 1983
4
10 ON ERROR GOTO 7030
12
16 REM -----CONSTANTS----
17 REM
20 DIM T%(4,15):REM stores tunes
30 D(1)=-1:D(2)=1:D(3)=-40:D(4)=40:REM d
irections
40 MS=279:REM maximum allowed screen pos
ition
50 BOARD=65446:REM memory loc stores cod
e (not in ascii) of last key pressed
60 SCRN=65024:REM start of screen memory
70 TIMER=63791:REM memory location of ti
mer, values 0-125
75 ESC=27:REM ascii code for ESC. Printi
ng ESC p turns on reverse video.
77 REM printing ESC q turns off reverse
video
80 BARRIER=239:REM ascii code for grph x
82 EXTRA=171:REM extra points
84 EMPTY=32:REM ascii code of space
85 ENEMY=144:REM ascii code for grph y
87 PLAYER=147:REM graph q
88
89 REM ----initialization----
90
91 GOSUB 6000:REM load music data
92 GOSUB 7000:REM print instructions and
ask user for set up
102 GOSUB 2000:REM randomize
103 GOSUB 1000:REM set up barriers
104 GOSUB 3000:REM set up enemies
105 L=160:PRINT@L,CHR$(PLAYER);
106 PS=100:REM player's score
107 PRINT@MS+1,"HIT ANY ARROW KEY TO STA
RT";
108 FOR Z=1 TO 11:SOUND T%(4,Z),10:NEXT
Z:REM play tune #4
109 K$=INKEY$:IF LEN(K$)=0 THEN 109:REM
wait for a key to be pressed
110 PRINT@ MS+1,SPACE$(30);
120
150 ----Main Program----
152
160 PRINT@L,CHR$(PLAYER);
165 PRINT@MS," SCORE: ";PRINT USING"###
####";PS;
170 GOSUB 4000:REM move enemy
200 P=PEEK(BOARD):REM most recent key pr
essed
205 PRINT@ L, " ";
206 IF P<44 OR P> 47 THEN PS=PS-10:GOTO
160:REM not an arrow key
208 PS=PS+100:REM player gets 100 pts
for staying alive
210 D=D-(P-43):REM direction to move. 44
is code (not ascii) for left arrow
505 PK=PEEK(SCRN+L+D):REM see what is on
screen where player wants to move
507 IF PK<>EXTRA THEN 510:REM hit a bonu
s target?
508 PRINT@MS+1,"BONUS";PRINT@L+D,CHR$(B
ARRIER):D=0:PS=PS+50000
509 FOR Z=1 TO 9:SOUND T%(3,Z),3:NEXT Z:

```

```

REM play tune #3
510 IF PK<49 OR PK>57 THEN 523:REM 49 is
ascii code of 'I'
511 REM player hit a number barrier
512 FOR ZZ=1 TO 4:SOUND T%(4,ZZ),5:NEXT
ZZ:REM tune#4
515 PRINT@L+D,CHR$(BARRIER);D=0:PS=PS+1
000*(PK-47):REM hit numb
523 IF PEEK(SCRN+L+D)<>EMPTY THEN SOUND
500*(P-43)+500,4;D=0:PS=PS-300: barrier
530 L=L+D:REM move
540 IF L>MS THEN L=L-MS:REM offscreen
550 IF L<1 THEN L=L+L:REM offscreen
560 GOTO 160
570
5700 REM ----set up barriers
1002
1010 FOR K=1 TO 75:L=INT(RND(1)*MS+1):RE
M random location of barrier
1015 T=INT(RND(1)*5):REM type of barrier
1020 IF T<>0 THEN PRINT@L,CHR$(BARRIER);
1025 IF T=0 THEN PRINT@L,CHR$(ESC);"p";O
HR$(INT(RND(1)*9+49));CHR$(ESC);"q";
1026 " a number target
1027 T=INT(RND(1)*50):IF T<NUMBER*2 THEN
PRINT@L,CHR$(EXTRA):REM bonus target
1030 NEXT K
1040 RETURN
1042
2000 REM ----randomize---
2002
2010 R=PEEK(TIMER)+1:REM ACCESS TIMER FO
R VALUE BETWEEN 1 AND 126
2020 FOR I=1 TO R
2030 DUMMY=RND(1)
2040 NEXT J
2060 RETURN
2062
3000 REM --set up enemy---
3002
3020 FOR E=1 TO NUMBER
3040 L=INT(RND(1)*MS+1):REM random locat
ion
3050 PRINT@ L,CHR$(ENEMY);
3060 L(E)=L:REM store
3070 NEXT E
3090 RETURN
3095
4000 REM ----move enemy---
4002
4010 Y=INT(L/40):X=L-Y*40:REM convert to
x y coordinates
4020 FOR E=1 TO NUMBER
4030 Y2=INT(L(E)/40):X2=L(E)-Y2*40:REM c
onvert to x,y
4040 X2=X2+SGN(X-X2):Y2=Y2+SGN(Y-Y2)
4042 NL=Y2*40+X2:REM new location
4047 IF PEEK(SCRN+NL)=PLAYER THEN 5000:R
EM player has been caught
4048 IF PEEK(SCRN+NL)<>EMPTY THEN NL=L(E
)+D(INT(RND(1)*4+1)):move randomly
4049 IF NL>MS OR NL<1 THEN 4060:REM dont
go off screen
4055 IF PEEK(SCRN+NL)=EMPTY THEN PRINT@
L(E)," ";L(E)=NL:PRINT@ NL,CHR$(ENEMY);
4057 " move enemy if it will not hit
a barrier
4060 NEXT E
4070 RETURN
4072
5000 REM ----dead---
5002
5005 PRINT@L(E)," ";
5010 PRINT@NL,CHR$(ENEMY);
5015 PRINT@MS+20,"MUNCH";
5020 FOR I=1 TO 10
5025 PRINT@NL," ";
5030 SOUND INT(RND(1)*1000+500),3
5032 PRINT@NL,CHR$(ENEMY);
5033 SOUND INT(RND(1)*1000+500),3
5035 NEXT I
5036 FOR ZZ=1 TO 200:NEXT ZZ
5037 FOR K=1 TO 6
5050 PRINT@MS+20,"YOU ARE DEAD";
5060 SOUND T%(1,K*2-1),10:REM play a not
e of tune #2
5065 PRINT@MS+20,CHR$(ESC);"pYOU ARE DEA
D";CHR$(ESC);"q";
5067 SOUND T%(1,K*2),10:REM play a note
of tune #2
5070 NEXT K

```

```

5100 IF HS=HS THEN 5172
5102
5105 FOR Z=1 TO 11: SOUND TX(4,Z),10:NEXT
Z
5107 CLS:PRINT:PRINT"CONTRATULATIONS!!!"
5110 PRINT YOU HAVE BROKEN THE HIGH SCORE
RE
5120 FOR Z=1 TO 11: SOUND TX(3,Z),10:NEXT
Z
5125 PRINT
5130 INPUT please enter your name ";NM$
5140 OPEN "SCORES.do" FOR OUTPUT AS 1:RE
M stores high score & scorer
5142 PRINT#1,PS:REM store player's score
5144 PRINT#1,NM$:REM store player's name
5146 CLOSE 1:CLS
5147 PRINT@161,"THE HISTORIC DEED HAS BE
EN RECORDED"
5150
5172 PRINT@1," GAME OVER ";
5174 PRINT@MS+1,"HIT SPACE BAR TO PLAY,
ENTER TO STOP";
5178 K$=INKEY$:IF LEN(K$)=0 THEN 5178
5180 IF K$=" " THEN 92:REM play
5185 IF ASC(K$)=13 THEN CLS:END:REM 13 i
s ascii for ENTER key
5190 GOTO 5178
5192
6000 REM ---load music data---
6002
6010 DATA 4697,4697,4697,4697,3950,4184
,4184,4697,4697,4976,4697,4697
6012 funeral march
6020 DATA 6266,6269,6269,7900
6022 beethoven's fifth
6030 DATA 3134,4976,4184,4184,4184,4184,
4184,3134,4184
6032 halls of montezuma
6040 DATA 1567,2348,2092,1864,2092,2348,
2092,1864,2348,2348
6042 video game tune
5100 RESTORE
6200 FOR K=1 TO 12:READ TX(1,K):NEXT K
6210 FOR K=1 TO 4:READ TX(2,K):NEXT K
6215 FOR K=1 TO 9:READ TX(3,K):NEXT K
6230 FOR K=1 TO 11:READ TX(4,K):NEXT K
6240 RETURN
6242
7000 REM ---instructions and specs--
7002
7012 OPEN "scores.do" FOR INPUT AS 1:REM
stores high score and scorer
7016 INPUT #1,HS:INPUT #1,SC$:REM READ T
HE HIGH SCORE AND THE SCORER
7018 CLOSE 1
7020 GOTO 7090
7030 IF ERR<>52 THEN PRINT "error ";ERR;
" in line ";ERR:END:ELSE RESUME 7090
7032 52 is error code for file not
found
7090 CLS
7092 PRINT CHR$(ESC);"pZOID PATROL";CHR$
(ESC);"q";
7094 IF HS<>0 THEN PRINT " HIGH SCORE:";
:PRINT USING"*****";HS:PRINT " ";
7095 IF HS=0 THEN PRINT
7096 PRINT USING"\
";SC$:REM s:
spaces between slashes
7100 PRINT
7105 PRINT"USE ARROW KEYS (";CHR$(152);C
HR$(153);CHR$(154);CHR$(155);")";
7107 PRINT" TO MOVE, TO SCORE,";
7108 PRINT "HIT: ";CHR$(ESC);"p1";CHR$(E
SC);"q = 1000 pts ";CHR$(ESC);"p9";
7110 PRINT CHR$(ESC);"q = 9000 ";CHR$(EX
TRA);" = 50000
7115 PRINT "IF HIT: ";CHR$(BARRIER);" LD
SE 300 PTS. ";
7120 PRINT "THE ZOIDS ( ";CHR$(ENEMY);"
";
7125 PRINT "WILL KILL YOU ( ";CHR$(PLAYE
R);" ) IF THEY CATCH YOU!";
7196 PRINT@280,"number of ZOIDS (1-6) 1=
easy 6=pro ? ";
7197 N$=INKEY$:IF LEN(N$)=0 THEN 7197:EL
SE NUMBER=VAL(N$)
7198 IF NUMBER<1 OR NUMBER >6 THEN 7197
ELSE PRINT NUMBER;
7250 CLS
7300 RETURN

```

PCM

GoCo

Creating Categorical Directories

By Lawrence C. Falk

One of the more interesting functions which you can perform with the Tandy 2000 is to place individual programs in separate directories to make it easier for you to access and use them.

Sound a bit complicated? Perhaps. Yet, all in all it is not only easy, but makes things easier for you.

Let's take an example of a program which you may have purchased — or may wish to purchase — for your 2000. The example I will use is *Multiplan* by Microsoft, a spreadsheet program which I happen to think is excellent.

Now, let us further suppose that you have either a hard disk on your 2000 or a number of additional files on your floppy (with the capacity of the 2000's drives, that is not hard to do).

One of the easiest ways to call up a spreadsheet that you have already created is to have *Multiplan* "look up" the filenames for you. This is especially important when you have a lot of files — as I do — and cannot remember the names of all of them. "Now," you may think, "was that file named 'SUBS' or 'SUBSCRIP?'" With *Multiplan*, you need only press the cursor keys once you have told the program to load in a spreadsheet and it will display all the files on the drive.

Oh, oh. There are a lot of files on the drive. Sorting through all of them (especially with a hard disk that will show all your system files as well) is almost as much trouble as remembering the name of the file in the first place.

How can I get all those extra files off the display?

The most simple way is to create a special directory for the application that you will be using and simply move all of the files that are used with that application into that directory. In the case of *Multiplan*, there are several of them.

Since *Multiplan* deals primarily with numbers and dollars, I created a directory called "\$" for its files. You do this by simply using the following command:

MKDIR \$

This creates a directory called "\$" and, if you will simply run a directory of your main disk, you will find a new entry for "\$" that shows as a (DIR) in the list.

Your next step is to copy all of the files which pertain directly to that program to the new directory. One of those files is called *MP.EXE*, so you just move that file to the new directory with this command:

MV MP.EXE \S

The MV means "move," the "MP.EXE" is the name of your file and the backslash and dollar sign are how you designate the new directory. You then perform this same function for all of the files which have anything to do with *Multiplan*.

Once you have done that, you can ask the 2000 to list the main directory with the DIR command, and you will note that all of the files which you just moved are gone.

Where did they go? To the new directory called "\$" that you just created. In order to see them, simply execute the following command:

CD \S

You will then see the directory for "\$" and note that all the files you moved are listed there

If you will then go into *Multiplan* and

work the cursor keys to load a file, you will see that only your spreadsheet files are listed. This eliminates clutter and makes things much easier for you to see what is going on.

This procedure will work with any programs which read the directory and display files for you. Examples of other programs are *WordStar* and *MultiMate*, both word processing programs. It makes things much easier for you to have only the files which pertain to the particular programs showing on the screen.

There is another advantage, too. Someone cannot just "run" your programs by typing in a name that is to be found on the main directory. Things are a little more complicated, because they don't even see your application programs in the first place. And, they don't see all the files, either. Who, if his name were Jim, could resist trying to see what is in "FIRE-JIM" if he accidentally saw it on the main directory?

Since "FIRE-JIM" would never appear on the main directory, it would take some understanding of your own file system to "get" to the directory where the file — and its application program — are located.

For you to get there, all you do is change the directory, using the CD \S command. And, incidentally, to get back to your main directory, all you need to do is type in CD \ with no directory specified. You will be right back in the main directory.

If changing all these directories seems to be a bit cumbersome to you, we'll explore a way that you can do it automatically in next month's issue of PCM. And, while we're at it, we'll make an automatic backup of your data in the process.

PCM

Meet Professor PoCo

By Mel Perkins



The listing:

```

10 MATH MASTER... rev 3/31/84
20 MAXFILES=3:CLS
30 GOTO:0000
100 CLS:OPEN"M.DO"FOROUTPUTAS2
110 T$(1)=TIME$:ST=1
120 IFEOF(1)THEN8000
130 INPUT#1,A,B:GOSUB600
140 IF VAL(TL$)<C THEN PRINT@210-N,SPACE
$(15):NR=NR+1:GOTO120ELSE150
150 PRINT#2,A;" ";B;" ";PRINT@210-N,SPA
CE$(15):NW=NW+1:GOTO120
600 LA=LEN(STR$(A))
610 LB=LEN(STR$(B))
620 PRINT@100-LA+1,A
630 PRINT@140-LB+1,B
640 IF LAL>LB THEN LL=LA ELSE LL=LB
650 LINE (127,35)-(127-LL*8,35)
660 IFAS#"1"THENC=A+B:PRINT@140-LB-2,"+
"
670 IFAS#"2"THENC=A-B:PRINT@140-LB-2,"-
"

```

```

675 IFAS#="3" THEN C=A*B:PRINT@140-LR-2,"x
680 LC=LEN(STR$(C))-1
690 N=0
700 PRINT@220-N,"?"
710 AN$(N)=INKEY$
720 IFST=1 THEN T$(2)=TIME$:GOSUB7000
730 IFAN$(N)="" THEN 710
740 IFAN$(0)=CHR$(13) THEN 710
750 IFAN$(N)=CHR$(8) THEN PRINT@220-N," ":
N=N-1:GOTO700
760 PRINT@220-N,AN$(N)
770 IFAN$(N)<>CHR$(13) THEN N=N+1:GOTO700
900 TL$=""
910 FOR NN=N-1 TO 0 STEP -1
920 TL$=TL$+AN$(NN)
930 NEXT NN
935 RETURN
7000 ' ELAPSED TIME
7005 S(3)=0:M(3)=0:H(3)=0
7010 FORCT=1 TO 2
7020 H(CT)=VAL(LEFT$(T$(CT),2))
7030 M(CT)=VAL(MID$(T$(CT),4,2))
7040 S(CT)=VAL(RIGHT$(T$(CT),2))
7050 NEXTCT
7060 IFS(1)=S(2) THEN 7070 ELSE IFS(2)>S(1) T
HENS(3)=S(2)-S(1) ELSE S(3)=(60-S(1))+S(2)
:M(3)=1
7070 IFM(2)=M(1) THEN 7080 ELSE IFM(2)>M(1) T
HENH(3)=M(3)+(M(2)-M(1)) ELSE H(3)=M(3)+(6
0-M(1))+M(2):H(3)=1
7080 IFH(2)=H(1) THEN 7090 ELSE IFH(2)>H(1) T
HENH(3)=H(3)+(H(2)-H(1)) ELSE H(3)=H(3)+(2
4-H(1))+H(2)
7095 PRINT@241,;
7090 IFH(3)>0 THEN PRINTH(3)"hour";:IFH(3)
>1 THEN PRINT"s ";
7100 IFM(3)>0 THEN PRINTM(3)"minute";:IFM(3)
>1 THEN PRINT"s ";
7110 IF S(3)>0 THEN PRINTS(3)"second";:IFS
(3)>1 THEN PRINT"s "
7120 RETURN
8000 RESULTS
8010 CLS:CLOSE
8020 PRINT@43,"NUMBER NUMBER"
8030 PRINT@83,"RIGHT WRONG"
8040 PRINT@164,NR:PRINT@172,NW
8050 PRINT@201,"TIME USED:"
8060 PRINT@241,M(3);" minutes ";S(3);" s
econds"
8070 OPEN"RESULT.DO"FORAPPENDAS3
8080 IFLEN(STR$(S(3)))=2 THEN SP#=" " ELSE
SP#=" "
8090 PRINT#S,DATE$;"("AS$;")";M(3);";";
S(3);SP#;NR;"RIGHT";" ";NW;"WRONG"
8100 IFNW=0 THEN PRINT@100,"NONE MISSED !"
:PRINT@140,"GOOD WORK !!":KILL"M.DO":PRI
NT@0,;:END
8110 PRINT@100,"PRESS ENTER TO WORK"
8120 PRINT@140,"PROBLEMS MISSED";
8130 INPUTO
9000 'REDO PROBLEMS MISSED
9010 CLS:OPEN"M.DO"FORINPUTAS1:ST=2
9020 PRINT@10,"PROBLEMS MISSED"
9030 LINE(50,8)-(150,8)
9040 IFEOF(1) THEN CLOSE:KILL"RAM:M.DO":CL
S:MAXFILES=1:END
9050 INPUT#1,A,B:GOSUB600
9060 IF VAL(TL$)=C THEN PRINT@82,"GOOD!":
PRINT@210-N,SPACE$(15):GOTO9040
9070 PRINT@82,"WRONG":PRINT@210-N,SPACE$
(15):GOSUB600:GOTO9060
9080 GOSUB900
10000 'ADD/SUB SELECT
10010 CLS
10020 PRINT@52,"MATH MASTER"
10030 PRINT@132,"1. ADDITION"
10040 PRINT@172,"2. SUBTRACTION"
10045 PRINT@212,"3. MULTIPLICATION"
10050 PRINT@292,"Select";
10060 AS#=INKEY$:IFAS#="" THEN 10060
10080 IFAS#="1" THEN OPEN"NUMS+.DO"FORINPU
TAS1:GOTO100
10090 IFAS#="2" THEN OPEN"NUMS-.DO"FORINPU
TAS1:GOTO100
10100 IFAS#="3" THEN OPEN"NUMSX.DO"FORINPU
TAS1:GOTO100
10200 GOTO10060

```

Those Model 100 owners having elementary school-aged children, as I have, should be able to put this math tutor program to good use. It makes possible repetitive calculations in addition, subtraction, and multiplication from predetermined sets of numbers, with elapsed time.

To ready the program is very simple. For addition, you must create a *TEXT* file named *NUMS+.DO*. Similarly, *NUMS-.DO* is needed for subtraction; and *NUMSX.DO* is required for multiplication. All three, or any one or two, may be placed in RAM. These files may then, of course, be saved to tape or disk for future use, or to create additional files with different sets of numbers.

TEXT file setup is easy. Simply enter all numbers separated by commas. The first number entered is the top number of the problem, the second is the bottom number, the third is the top number of the second problem, the fourth is the bottom number of the second problem, etc. Needless to say, there must be an even number of numbers in each file. In the multiplication mode, no numbers larger than nine are allowed as the multiplier (bottom number).

Upon running the program, a menu is displayed from which you are asked to select addition, subtraction, or multiplication. If the text file corresponding to the selection is not in RAM, a "file not found" (FF) error results. When the selection is made, the first problem is displayed with the respective operating sign. Throughout the problem set, elapsed time is continuously displayed.

Entering the answer is done as you would actually do it on paper. That is, from right to left. Digits are entered over a question mark and displayed as the number is depressed on the keyboard. Backspacing will erase the digit just entered. ENTER will terminate the answer entry. No error correction is done at this time.

Upon completion of all problems, the screen will display the number of problems worked correctly, the number worked incorrectly, and the total time used. At this time you are congratulated for no errors, or you are asked to press ENTER to rework problems missed. Working missed problems is done in a similar manner as above with two exceptions: no elapsed time is shown and there is immediate error identification.

A file titled *RESULT.DO* is created and/or appended to monitor results. The file contains the dates the program was used, operation used, time used in minutes and seconds, and the number right and wrong. GEM

CHEAP FRILLS

Editor:

I recently purchased a Model 100 and have found it be a truly versatile computer, due mainly to its portability.

The usefulness could be enhanced even further by the availability of an [inexpensive] option which would allow an interface with a monitor or TV. At present it is necessary to purchase the Disk Video Interface which sells for \$699. Do you have information regarding any third-party vendors who are presently marketing (or who plan to market) such a hook-up capability?

M.N. Folkening, M.D.
Holland, MI

Editor's Note: To our knowledge, there is no product on the market offering such a capability.

A USEFUL VARIETY

Editor:

I was amazed at the usefulness and variety of the programs I found in the nine back-issues of PCM that I recently received. You have certainly succeeded in filling the needs of many of us out here.

I would like to suggest a few things in the same direction you have so admirably taken in your publication:

- 1) Could we subscribe to a cassette edition of machine-readable programs at some future date, possibly including back issues?
- 2) Could you consider bar code listing of programs?
- 3) Could you expand your readership interest by including articles on the PC8201A?
- 4) Could you include a cumulative index in some future issue?

Perhaps some of these are major considerations for you at this time, but as you can imagine, they would be major attractions for those of us who would benefit.

Chuck Olson
Cupertino, CA

Editor's Note: The obvious answer to your question about bar code is yes—we have been publishing bar code listings for several months now. We believe this offers several advantages over a cassette system. As for your other questions, they will certainly be taken into consideration.

FIRST, THE GOOD NEWS

Editor:

As a novice at this computing hobby, I found Richard White's "BASIC Bytes" for March '84 very frustrating.

First of all, let me say that I enjoy PCM very much and have copied several programs out of it and, after correcting my stupid mistakes, they run very well. I did experience some difficulties with this particular article, though.

After being unable to get the first little program to run (middle column of Page 9), my son pointed out that the statement in Line 20 that read *:LPRINT STS(X)* should read *STR(X)*.

The next little program would not work correctly either (top right-hand column on Page 9). I figured this one out myself when I read a couple of paragraphs later that removing the ";" would make the program go to pot. The semicolon that should have been in the statement *:PRINT@X.X::NEXT* just wasn't there.

Again, near the bottom of the first column on Page 11, the statement *CLS:PRINT@125,USINGS;D;* should be *USINGS\$;D;*. And in the second column of the same page, the program line *SS="##,##,## PAID"* should have a "\$" after the first quote in order for it to print as indicated.

MEET CONTACTS



ADELAIDE	JOHN HAINES	08 278 3560	MAFFRA	MAX HUCKERBY	051 45 4315
ADELAIDE NTH	STVN EISENBERG	08 250 6214	MAITLAND	LYN DAWSON	049 49 8144
ALBURY	RON DUNCAN	060 43 1031	MELBOURNE	JEFF SHEEN	03 528 3724
BAIRNSDALE	COLIN LEHMANN	051 57 1545	MELTON	MARIO GERADA	03 743 1323
BALLARAT	MARK BEVELANDER	053 32 6733	MILDURA	DOUG MATTHEWS	056 29 5701
BANKSTOWN	KEN HAYWARD	02 759 2227	MCF	STEPHEN SEMPLE	031 27 6841
BLACKTOWN	KEITH GALLAGHER	02-627-4627	MORWELL	GEORGE FRANCIS	051 34 5175
BLAXLAND	BRUCE SULLIVAN	047 39 3903	MT ISA PAUL	BOUCKLEY-SIMONS	077 43 6280
BOWEN	TONY EVANS	077 86 2220	MUDGE	BRIAN STONE	063-72-1958
BOWRAL	MAX BETTRIDGE	048 83 9203	NAMBUCCA HOS	WENDY PETERSON	065 68 6723
BRISBANE EAST	BOB THOMPSON	07 848 5512	NEWCASTLE	LYN DAWSON	049 49 8144
BRISBANE NTH	JACK FRICKER	07 262 8869	NOWRA	ROY LOPEZ	044 48 7031
BRISBANE STH	PATRIC SIMONIS	07 209 3177	PARKES	DAVID SMALL	068 62 2682
BRISBANE SW	GRAHAM BUTCHER	07 376 3400	PENRITH	TOM LEHANE	047-31-5303
BRISBANE WEST	BRIAN DOUGAN	07 30 2072	PERTH	JOHN CHRISTOU	09 344 6745
BUNDABERG	JIM MCPHERSON	071 72 8329	PORT MacQUARIE	RON LALOR	065 83 8223
CAMBERWELL	TONY BALDWIN	03 728 3676	PORT NOARLUNGA	ROB DALZELL	08 386 1647
CAMPBELLTOWN	LEO BINLEY	02 605 4572	PORT PIRIE	KEVIN GREEN	044 32 1368
CANBERRA	SHAUN WILSON	062 51 2339	RINGWOOD	ANDREW FAWLINGS	03 726 6521
CAULFIELD	JEFF SHEEN	03 526 3724	ROCKHAMPTON	KEIRAN SIMPSON	079 28 6162
CHATSWOOD	BILL O'DONNELL	02 411 3336	ROCKHAMPTON MICO	TIM SHANK	079 28 1946
CHURCHILL	GEOFF SPOWART	051 22 1389	ROSEVILLE	KEN UZZELL	02 467 1619
COLYTON TEENS	DWAYNE MANSON	02 623 5805	SALE	BRYAN McHUGH	051 44 4792
COOMA	SHEILA HAMMILL	064.82.3905	SINGLETON	DAVID NICHOLS	065-73-1222
DANDENONG	BRETT CRUICKSHANK	03 798 5408	SPRINGWOOD	DAVID SEAMONS	047 51 2107
DARWIN	BRENTON PRIOR	089.81.7766	STURT	MARY DAVIS	08 296 7477
DENILIQUIN	WAYNE PATTERSON	058 81 3014	SUNBURY	JACK SMIT	03.744.1355
DUBBO	GRAEME CLARKE	068 89 2095	SUTHERLAND	IAN ANNABEL	02 528 3391
EMERALD	CAROL CATHCART	059 68 3026	SWAN HILL	BARRIE GERRAND	050.32.3538
FORSTER	GARY BAILEY	065 54 5029	SYDNEY EAST	BOB JONES	02-331-4621
FRANKSTON	BOB HAYTER	03.783.9748	SYDNEY TEENS	ROD HOSKINSON	02 48 5948
GIPPSLAND STH	PAT KERMODE	056 74 4583	TAMWORTH	ROBERT WEBB	067 65 7256
GOLD COAST	SHERYL BENTICK	075-39-2003	TONGALLA	TONY HILLIS	058 59 2251
GOSFORD	PETER SEIFERT	043 32 7874	TOOWOOMBA		
GRAFTON	DAVID HULME	066.42.0627	✓ BEGIN NTH	DAVID PROUT	076.32.7533
GREENACRES	BETTY LITTLE	08 261 4083	✓ BEGIN STH	LEW GERSEKOWSKI	076 35 8264
HOBART	BOB DELBOURGO	002 25 3896	✓ ADVANCED	GRAHAM BURGESS	076 30 4259
IPSWICH	MILTON ROWE	07 281 4059	TOWNSVILLE	JOHN O'CALLAGHAN	077 73 2064
KENMORE	GRAHAM BUTCHER	07 376 3400	TRARALGON	MORRIS GRADY	051 66 1331
LITHGOW	STUART RAYNER	063 51 4214	UPPER HUNTER	TERRY BRAVDLIN	065 45 1698
LIVERPOOL	LEONIE DUGGAN	02-607-3791	WAGGA WAGGA	BRUCE KING	069 25 3091
MACKAY	LEN MALONEY	079511333x782	WESTLEIGH	ATHALIE SMART	02 848 8830
MACLEOD	ROBIN ZIUKELIS	03 450211x465	WHYALLA NORRIE	CHRIS HUNTER	086 45 3395
MacQUARIEFIELDS	KIETH ROACH	02 618 2858	WOLLONGONG	BRIAN McCAULEY	042 71 4265
			WONTHAGGI	PAT KERMODE	056 74 4583

Registered by Australia Post-
Nos. NBG 5033, 6279 & 6280X,
Graham Morphett, P.O. Box 1742, Southport, QLD. 4215.

Remember . . .

Address all mail to:

AUSTRALIAN RAINBOW

P.O. Box 1742, SOUTHPORT. QLD. 4215.

or PHONE GRAHAM MORPHETT, on 075-51-0015.

