ROBERT SIMS, ASSISTANT EDITOR

The Electronic Café

Somewhere in New York there's a tablecloth with the bleached-out vestiges of poetic lines scrawled by Allen Ginsberg as he sat in a Greenwich Village cafe and argued with Jack Kerouac about the meaning of life.

As long as there have been cities, there have been inns, coffeehouses, and cafés where creative people congregated to pass the time and try out new ideas on old friends. And from all the chaos and seemingly rambling conversation in cafes in Vienna, Paris, Moscow, San Francisco, and Baghdad, have come many of the artistic movements and revolutions that have defined contemporary cultures.

Espresso And Computers

Telecommunications is beginning to play a role in this creative ferment. Originally conceived as a practical, if expensive, means by which business data could be transferred and manipulated over long distances, telecommunications is being transformed by a few innovative individuals and companies into a multifaceted creative and social tool for the home.

With the advent of home telecommunications, an artist's forum for the exchange of ideas no longer is confined to the neighborhood café. Given access to all the bulletin boards, newsletters, and conference lines on the information networks, a home computer owner becomes a patron in a vast electronic café in which time and distance no longer matter.

Computer owners can meet and talk in the same electronic café, regardless of whether they live in small towns or in major metropolitan areas. And the words which pass between them need not evaporate as they are spoken, as happens in an ordinary café.

In the electronic café, words have a life span. You no longer have to be sitting at the table to hear an interesting conversation. You can log on to a bulletin board and read the thoughts left there days or even weeks before. And you don't have to rely on your memory to recall some idea that excited your imagination. You can go back and read the words again, or even download them and make them a permanent part of your personal library.

A Writer's Bulletin Board

Home telecommunications is broadening artistic horizons in other, more practical, ways. For example, if you're a professional writer looking for new and different markets, there's a bulletin board in West Palm Beach, Florida, which serves just that function.

Called The Notebook, the bulletin board contains queries from publishers looking for articles on specific subjects, notices from writers who have articles or ideas to sell, and occasional tips on how to write and sell for the hottest markets, such as romance novels or fantasy.

You won't find any games or programs in the download section, and the bulletin board items are by and for professionals only. Callers have to pass the sysop's scrutiny before they are admitted, but, once approved, writers have one more resource in the struggle to make a living in a volatile field.

Electronic Publishing

It's not just in the exchange of ideas and information that the electronic café is serving artists. Literary, visual, and musical works are being published in these electronic media with increasing frequency.

On-line newsletters and special interest groups—such as the ARTSIG on CompuServe and Writer's Corner on Delphi—are offering artists a new way to get their work before the public.

Traditionally, this function of publishing and displaying the work of new writers and visual artists has been inadequately handled through small literary journals and storefront galleries. Financed on a shoestring, these enterprises have had very limited circulation and very short lives.

But the electronic café has hundreds of thousands of patrons, and the cost of publishing a work (apart from the initial cost of a computer and modem) is a matter of the time it takes to upload it. With no store rent to pay, no printers' costs, no problem trying to convince a distributor to handle a work by an unknown artist, the electronic café is a godsend to aspiring artists and publishers both.

A Wide-Open Market

Although the networks are adding services every day, the field is still wide open. An entrepreneur can found a publishing empire on-line with little more than a fresh idea and a user ID number. The procedure is simple. Just write up a proposal outlining your idea and goals for a special interest group or newsletter, and mail it (or Email it) to the Information Provider Department or Customer Services Department of your favorite network. If the network thinks your idea will appeal to a reasonably large group of subscribers (or potential subscribers), they'll send you guidelines and a contract, and you're in business.

It's even possible for the proprietors of these on-line publishing enterprises to count the number of times an article is accessed, and so pay a contributor royalties based on the number of users who read or view a work, instead of the usual method of paying a set amount for each story or article. In electronic publishing, an artist's monetary reward can be exactly proportional to his or her popularity.

An Uneven Advantage

Writers are more fortunate than visual artists and musicians, because they work with words, which can be transmitted in a form compatible with almost any computer. (Most home telecommunications services have been adapted from software written to transmit business documents.) But businesses have little need to transmit animated graphics and music, and business applications so far have defined the scope of telecommunications software.

This is because home telecommunications began as a "poor relation," as an afterthought. Some of the information networks like The Source and CompuServe, that were providing telecommunications services for business, saw a way to profitably use their computers from 6 p.m. to 6 a.m. when business traffic was at a virtual standstill.

Since their computers were sitting idle, the networks began to offer bulletin boards and conference lines to computer hobbyists. The response has been phenomenal, and growth has come so quickly that research can't keep pace. The numbers of new home telecommunications users can safely be measured in the thousands every month. As the income from home users grows to a more significant percentage of the networks' profits, we can expect to see software written specifically for games, entertainment, and the visual arts.

Until that happens, musicians and visual artists will not be able to fully use the electronic café for direct delivery of their work. They must depend on programmers to package their work within programs which must be downloaded and run off-line before the work can be enjoyed.

Artistic Bottlenecks

While current technology is capable of transmitting high-resolution screens and music, the lack of standards for different brands of computers makes the matter a practical impossibility. For example, the CompuServe Information Service offers color graphics to its subscribers, but, in designing a format which is compatible with all computers, the network's software fails to make use of the best graphics features of any. As a result, transmitted graphics are inferior to what owners are used to seeing on their screens.

The answer to this quality problem would be to write custom network software which could take advantage of each computer's special features. However, given the rampant changes in the home computer industry, no network is likely to invest the time and money necessary to fully exploit the features of a computer which may be out of production before the software is written.

Another bottleneck which hampers the growth of on-line art is transmission speed. Ordinary phone lines, over which most home telecommunications travel, do not transfer data reliably at transmission speeds greater than 1200 bits per second.

The most common transmission speed available to home users is 300 bits (roughly 30 characters) per second, which is slower than most printers in home computer systems.

At 30 characters per second, a VIC screen can be transmitted in 15–30 seconds, and for the 64, in 30 seconds to a full minute. And a detailed high-resolution screen takes eight times longer. Even at 1200 bps, graphics transmissions using ordinary phone lines are far too slow to appeal to

a generation brought up on the fast visual pace of television.

New Developments

Fortunately for computer artists, the broadcast media are interested in the home computer market. Whereas the information networks are primarily business-oriented, broadcasters deal in consumer entertainment, and so are possibly more sensitive to consumers' special needs. FM radio stations are experimenting with services which broadcast software, news, and games to home computer owners.

Also under development are systems which will allow television transmission of digital signals for computers simultaneously with "The A-Team." Other companies are trying cable hookups for computers, so that on channel 19 you get Home Box Office and on channel 18 you get a computer network like CompuServe, Delphi, or The Source.

With a much faster transmission speed and a predisposition toward the consumer, the broadcast media may well replace phone-based services as the primary resources for home telecommunications.

But whenever and however the shakeout comes, consumers are sure to be well-served. After all, the home computer industry was born when a few legendary hobbyists in a few legendary garages decided to transform a dull and mysterious business machine—the computer—into an exciting and entertaining personal tool.

If a few hackers in garages can create a billion-dollar industry in their spare time, then giving them access to continental communications networks should yield even more astonishing results.

If you have questions or ideas about subjects you'd like to see covered in this column, write to: Home Telecommunications, COMPUTE!'s GAZETTE, *P.O. Box 5406, Greensboro, NC 27403. Or you can send me electronic mail. My CompuServe ID is 75005,1553. For Delphi, it's BOZART.*

The Notebook
West Palm Beach, Florida
J.D. Pitt and Karl Meyer, Sysops
Voice Phone (305)684-8751
Modem (305)686-4862

User Group Update

When writing to a user group for information, please remember to include a self-addressed, stamped envelope. Send additions, corrections, and deletions for this list to:

COMPUTE! Publications
P.O. Box 5406
Greensboro, NC 27403
attn: Commodore User Groups

Gila Hackers
c/o Paul R. Machula
Rt. 1, Box 34
Globe, AZ 85501
(602) 425-7260

Thunder Mountain Commodore User Group
P.O. Box 1796
Sierra Vista, AZ 85636

Auburn Commodore Computer Club
Pat Strub
11210 Mira Loma Dr.
Auburn, CA 95603
(916) 823-7095

San Luis Obispo Commodore Computer Club (SLOCCC)
Joan Rinehart
1766 9th St.
Los Osos, CA 93402
(805) 528-3371

CIVIC64
Nathan Okun
120 W. Magnolia Ave.
Oxnard, CA 93030
(805) 985-8150

San Diego Commodore (PET) User Group
Jane Campbell
Box 86531
San Diego, CA 92138-6531
(619) 277-7214

Aurora Market Users Group (AMUG)
Roger Oberdier
c/o Computer Market Place
15200 E. 6th Ave.
Aurora, CO 80012
(303) 367-0901

Vicdore Users Group
Wayne Sundstrom
326 Emery Dr.
Longmont, CO 80501
(303) 772-2821

Commodore Brooksville User Group (C-BUG)
Eleanor Hott
414 W. Broad St.
Brooksville, FL 33526
(904) 796-2909
(904) 799-5292

Fred & Kent's 64 User's Group
Fred Brock
Kent Lawson
P.O. Box 1298
Fort Pierce, FL 33450
(305) 464-5792

Suncoast 64s
Curtis J. Miller
2419 US 19 North
Palm Harbor, FL 33563
(813) 785-1036

Charlotte County Commodore Club (C.C.C.C.)
Lee Truax
567 N. Ellicott Cir.
Port Charlotte, FL 33952
(813) 625-1277

Titusville Commodore Club
Bob Murray
890 Alford St.
Titusville, FL 36780

Commodore Computer Club of Columbus
Nosh Sethna
6618 Foxdale Drive
Columbus, GA 31907
(404) 563-0828

Eagle Rock Commodore Computer Club
P.O. Box 3884
Idaho Falls, ID 83403-3884
(208) 529-4738

Caribou Commodore Club
P.O. Box 535
Soda Springs, ID 83276
(208) 547-3921
(208) 547-4143

Commodore SIG Cache
Herb Swanson
Box C-176
323 S. Franklin, #804
Chicago, IL 60606
(312) 685-0994

McHenry County Commodore Club
John B. Katkus
227 East Terra Cotta Ave.
Crystal Lake, IL 60014
(815) 455-3942 after 6 p.m.

Fox Valley 64 Users Group
Frank Christensen
P.O. Box 28
No. Aurora, IL 60542
(312) 898-2779

Salt City Commodore Club
Wendell Hinkson
P.O. Box 2644
Hutchinson, KS 67501

Compumania
Richard L. Nadeau
81 North St.
Saco, ME 04072
(207) 282-7418

Your Computer Users Group
Mike Procise
P.O. Box 611
Westbrook, ME 04092
(207) 854-4579
Don Miller
So. Windham, ME
(207) 892-9533

Soft-Type Users Group
Al Southern II
20231 Westmoreland
Detroit, MI 48219
(313) 535-4549

Commodore Users Club of the Ozarks
Morris Williams
211 N. Aurora
Eldon, MO 65026
(314) 392-4248

Commodore User Group of Springfield (C.U.G.O.S)
Keith Masavage
Box 607 HSJ
Springfield, MO 65801
(417) 831-6403

Ewing Commodore Users Group
John C. Jones
11 Van Saun Dr.
Trenton, NJ 08628
(609) 882-4826

Jersey Shore Commodore Users Group
Bob McKinley
George Decker
Wall Township First Aid Building
1905 Monmouth Blvd.
Wall Township, NJ 07724
542-2113
223-1387

The Northern New Mexico VIC-20 Commodore Users Group
David Martinez
2725 Camino Cimarron
Santa Fe, NM 87501
(505) 471-3110 evenings

Highland Falls Commodore Users Group
Kevin R. Mandoske
8 Knox Rd.
Highland Falls, NY 10928
(914) 446-2802

Empire State User Group
Mark Dixon
52 Underhill Rd.
Ossining, NY 10562

Commodore User's Group of Rockland
Glenn Goldberg
13 Orchard Ct.
Spring Valley, NY 10977
(914) 356-5914

64 Southtowns User Group
Michael Leskow
504 Sommerville Rd.
Tonawanda, NY 14150
(716) 837-5643

Akron Area Commodore Club (AACC)
Dan Shibley
P.O. Box 9243
Akron, OH 44305
(216) 762-1934

Akron Area C-64 Users Group
Paul M. Hardy
2453 Second St.
Cuyahoga Falls, OH 44221
(216) 923-4396

B.A.S.I.C.
Dimitri N. Dantos
1433 13th Ave.
Altoona, PA 16601
(814) 942-9565

Worldwide Commodore Users Group
Dave Walter
P.O. Box 337
Blue Bell, PA 19422
(215) 584-4483

Castle Commodores Computer Club
c/o Dean G. Thomas
R.D. #1, Box 210A
Edinburg, PA 16116
(412) 652-3352

Republic User Group
Shandy Casteel
Box 186
Republic, PA 15475

Upper Buxmont C64 Users
Don Roques
655 Bergey Rd.
Telford, PA 18969
(215) 723-7039

Main Line Commodore Users Group (MLCUG)
c/o Emil Volcheck
1046 General Allen La.
West Chester, PA 19382
(215) 388-1581

Greenville-Spartanburg User Group (Commodore 64)
Robert Tolbert
Rte. 1, Box 231
Travelers Rest, SC 29690
(803) 895-4481

South Plains 64 User Group
John N. Bottoms Sr.
7709 Avenue W.
Lubbock, TX 79423
(806) 745-4381

Uintah Basin Commodore Users Club
Terry Hall
P.O. Box 1102
Roosevelt, UT 84066

Commodore Home User's Group (C.H.U.G.)
Alice Shipley
David Hoskins
81 Lynwood Ave.
Wheeling, WV 26003
(304) 242-8362
(304) 242-2605

Outside The U.S.

Geelong Commodore Computer Club
c/o 15 Jacaranda Pl.
Belmont, 3216
Geelong, Australia

Juan De Fuca 64 Users Group
Mike Watson
578 Langholme Drive
Victoria, British Columbia
Canada V9C 1L8
(604) 478-5851

Club 20
Steven Benoit
7145 Cannes #2
Montreal
Canada H1S 2P9

Pro Byte-64 Computer Club
Jim Hutchinson
R.R. #4 Queenston Rd.
N.O.T.L., Ontario
Canada L0S 1J0

Tuesday User Group (T.U.G.)
Andy Baca
Box 970
Port Perry, Ontario
Canada L0B 1N0
(416) 985-7686

Club Commodore Champlain
Claude Hebert
P.O. Box 522
Boucherville, Quebec
Canada J4B 6Y2
(514) 655-9232

Regina Personal Computer Users Group
Bill Swan
98 Pappas Crescent
Regina, Saskatchewan
Canada S4R 7G8
(306) 545-9596
(306) 347-3474

The POKEr's Club
Kris Finnestad
Box 75
St. Louis, Saskatchewan
Canada S0J 2C0

C-64 Milano Club
Claudio Cerroni
Via Sorrento 24
20153 MILANO
Italy

C-64 Anwender Club
Martin Sommer
Rotbuchstr.22
8037 ZURICH
Switzerland

Ferrari Rudi
Kettenberg 24
D 5880 Luedenscheid
West Germany

Trinidad Association of Commodore Owners (TACO)
Mark Mahannah
91 Cherry Crescent
Westmoorings/Carenage
Trinidad, West Indies
(809) 637-8091

Bremerhaven Users Group (C-64 B*U*G)
Ronald Debell
John Ford
626 TCF Box 119
APO, NY 09069

Commodore Computer Users Group Heidelberg
Robert H. Jacquot
P.O. Box, Gen. Del.
A.P.O. New York, NY 09102

ALL THE BEST PRICES

commodore

M-801 Dot MatrixParallel.....$219.00
MCS 801 Color Printer.....$499.00
1520 Color Printer/Plotter.....$129.00
1530 Datasette.....$69.99
1541 Disk Drive.....$249.00
1600 VIC Modem.....$59.99
1610 Vic Term 40.....$59.99
1650 Auto Modem.....$89.99
1702 Color Monitor.....$249.00
DPS Daisywheel Printer.....$459.00
Magic Voice Speech Module.....$54.99
Desk Organizer Lock.....$49.99
1311 Joystick...each.....$4.99
1312 Paddles.....$11.99
1110 VIC 8K.....$42.99
1111 VIC 16K.....$69.99
IEEE/RS-232 Interface.....$42.99
1211 Super Expander.....$53.99

MSD

SD 1 Disk Drive.....$359.00
SD 2 Disk Drive.....$589.00

CARDCO.

Light Pen.....$32.99
3 Slot VIC Expansion Interface.$32.99
6 Slot Expansion Interface.....$79.99
Cassette Interface.....$29.99
Parallel Printer Interface.....$49.99
Parallel Interface w/Graphics....$69.99

PRINTERS

C.ITOH

Gorilla Banana.....$149.00
Prowriter 8510P.....$379.00
Prowriter 1550P.....$599.00
A10 (18 cps) Son of Starwriter.....$569.00
F10-40 Starwriter.....$999.00
F10-55 Printmaster.....$1349.00

SX-64 PORTABLE

$839

Call for CMB 4 plus

Commodore 64.....$199
VIC 20.....CALL

COMREX

ComWriter II Letter Quality.....$449.00

EPSON

RX-80, RX-80FT, RX-100 FX-80, FX-100 CAll

JUKI

6100.....$469.00

MANNESMAN TALLY

160L.....$589.00
190L.....$799.00
Spirit 80.....$289.00

NEC

8027 Dot Matrix.....$389.00

OKIDATA

82,83,84,92,93,2350,2410.....CALL

OLYMPIA

Compact 2.....$479.00
Compact RO.....$509.00
ESW 3000.....$1449.00

SMITH CORONA

TP-1000.....$449.00

SILVER REED

500 Letter Quality.....$449.00
550 Letter Quality.....$549.00
770 Letter Quality.....$869.00

STAR

Gemini 10X.....$279.00
Gemini 15X.....$389.00
Radix 10.....$599.00
Radix 15.....$699.00

TOSHIBA

1340.....$829.00
1351.....$1499.00

TRANSTAR

120P.....$469.00
130P.....$649.00
315 Color.....$459.00

MONITORS

AMDEK

300 Green.....$149.00
300 Amber.....$159.00
Color 1.....$269.00
Color 1 Plus.....$289.00

BMC

1201 (12" Green).....$88.99
1201 Plus (12" Green Hi-Res).....$98.99

GORILLA

12" Green.....$88.99
12" Amber.....$95.99

NEC

JB 1201 Green.....$149.99
JB 1205 Amber.....$159.99
JB 1215 Color.....$259.00

SAKATA

SC-100 Color.....$269.00
SG-1000 Green.....$129.00
SA-1000 Amber.....$139.00

TAXAN

100 12" Green.....$125.00
105 12" Amber.....$135.00

USI

Pi 1, 9" Green.....$99.99
Pi 2, 12"Green.....$119.99
Pi 3, 12" Amber.....$129.99
Pi 4, 9" Amber.....$119.99
1400 Color.....$269.99

ZENITH

ZVM122 Amber.....$99.99
ZVM123 Green.....$89.99

SOFTWARE

ATARISOFT [C-64/VIC]

Dig Dug.....$34.99
Defender.....$34.99
Robotron 2084.....$34.99
PacMan.....$34.99
Donkey Kong.....$34.99
Centipede.....$34.99
Stargate.....$34.99

BRODERBUND [64]

Choplifter.....$29.99
David's Midnight Magic.....$22.99
Bank Street Writer.....$49.99

COMMODORE [64]

C-64 Reference Guide.....$18.99
EasyCalc.....$65.99
Easy Finance I,II,III,IV.....$19.99
Easy Mail.....$19.99
EasyScript.....$39.99

COMMODORE VIC

Jupitar Lander [R].....$9.99
Pinball Spectacular [R].....$9.99
VIC Reference Guide.....$15.99

CREATIVE SOFTWARE [VIC]

Apple Panic.....$19.99
Black Hole.....$29.99
Home Inventory [C-64/VIC].....$11.99

CBS [64]

Addition & Subtraction.....$16.99
Linear Equation.....$16.99
Multiplication & Division.....$16.99
Quadratic Equations.....$16.99

CONTINENTAL SOFTWARE [64]

The Home Accountant.....$49.99

CSA [64]

PractiCalc.....$39.99
PractiFile.....$39.99

DESIGNWARE [64]

Cryto Club.....$29.99
Trap-a-Zoid.....$29.99

DYNATECH [64]

Codewriter.....$75.99

ELECTRONIC ARTS [64]

Pinball Construction.....$29.99
Cut & Paste.....$39.99
Hard Hat Mack.....$29.99

EPYX [C-64/VIC]

Temple of Apshai.....$29.99
Upper Reaches of Apshai.....$16.99
Jumpman Junior.....$29.99

HES [64]

Tri Math.....$22.99
The Pit.....$27.99
Ghost Manor.....$15.99
Pool Challenge.....$19.99
Hes Mon.....$29.99
Hes Writer.....$35.99

INFOCOM [64]

Zork I,II,III.....$27.99
Deadline.....$29.99
Witness.....$29.99

PROFESSIONAL SOFTWARE [64]

Word Pro 64 plus Spell.....$59.99

SEGA [64]

Star Trek.....$29.99
Congo Bongo.....$29.99

SPINNAKER [64]

Snooper Troops 1 or 2.....$29.99
Delta Drawing.....$29.99
Kids on Keys.....$29.99

SCREENPLAY [64]

Wylde.....$22.99
Kalv.....$22.99
Pogo Joe.....$20.99

SUB LOGIC [64]

Flight Simulator II.....$40.99

SYNAPSE [64]

Zaxxon.....$29.99
Protector II.....$23.99
Blue Max.....$24.99

TIMEWORKS [64]

Wall Street Manager.....$19.99
Word Writer.....$39.99
Data Manager.....$19.99

COMPUTER MAIL ORDER

VISA

east
800-233-8950
In PA call (717)327-9575, Dept. 115
Order Status Number: 327-9576
Customer Service Number: 327-1450
477 E. 3rd St., Williamsport, PA 17701

canada
Ontario/Quebec 800-268-3974
Other Provinces800-268-4559
In Toronto call (416)828-0866, Dept. 115
Order Status Number: 828-0866
2505 Dunwin Drive, Unit 3B
Mississauga, Ontario, Canada L5L1T1

west
800-648-3311
In NV call (702)588-5654, Dept. 115
Order Status Number: 588-5654
P.O.Box 6689
Stateline, NV 89449

MasterCard

No risk, no deposit on C.O.D. orders and no waiting period for certified checks or money orders. Add 3% (minimum $5) shipping and handling on all orders. Larger shipments may require additional charges. NV and PA residents add sales tax. All items subject to availability and price change. Call today for our catalog. CANADIAN ORDERS: All prices are subject to shipping, tax and currency fluctuations. Call for exact pricing in Canada.

MACHINE LANGUAGE FOR BEGINNERS

Richard Mansfield, Senior Editor

First Questions

Although the difficulty of learning and programming machine language (ML) is often exaggerated, there *are* some confusing things about it when you first try to understand it. This month let's answer some of the basic questions, some of the questions which everyone seems to have when they first dare to leave BASIC.

What exactly is *assembling*? How is an ML program written?

Just as a BASIC program is written under the control and with the assistance of the BASIC language, an ML program is written with an *assembler*. BASIC comes with the VIC and 64. It's sitting in ROM memory and available whenever you turn on the machine. An assembler, however, must be loaded in from disk or tape, or plugged in as a cartridge. An assembler is a program and can be written in either BASIC or machine language.

Perhaps the main reason that ML has a reputation for being difficult to write is the poor quality of many commercially available assemblers. They are often slow or awkward to use. Many have few error messages, limited features, or cumbersome editing features.

When you are first learning ML, you'll be writing short, simple programs. For that, a simple assembler written in BASIC can suffice. But when you progress to more sophisticated programming, you'll want as comfortable a programming environment as you can get.

An ML program is usually written as *source code*. You write instructions much the way a BASIC program is written:

```
10 LDA #65;    LOAD THE ASCII CODE FOR
               THE LETTER "A"
20 STA $0400;  STORE IT ON SCREEN
30 RTS;        RETURN TO BASIC
```

The best assemblers allow you to write this source code exactly the way you write your BASIC programs—with line numbers, with the normal Commodore full-screen editing features, with remarks, multiple-statement lines (divided by colons), etc. Writing ML in this way has the added benefit that all your BASIC programming aids will work. You can use automatic line numbering, search and replace, mass line deletion, renumbering, and so forth.

When you want to try out your source code, you tell the assembler to assemble it. The assembler will then turn all your commands (like LDA #65) into numbers that the 6502 can execute as an ML program. While it's assembling, the assembler should point out any errors you might have made and print the line where the error occurred. It should also provide you the option of having the results of the assembly (called *object code*) POKEd into RAM for testing, SAVEd to tape or disk, and printed on screen and a printer. You should also be able to turn these options on and off at will.

After the source code has been assembled into object code, it is a finished ML program. If you assembled it to disk, you can load it in and SYS to its starting address. If you assembled directly into RAM, just SYS and test it right after assembly.

How long does assembly take?

Speed is also an essential quality of a good assembler. This is because you'll want to write sophisticated programs—perhaps an all-ML arcade game—and you'll need to reassemble every time you make some adjustments. If your assembler takes 15 minutes to assemble 5K, that will become a significant burden. You'll find yourself

trying to debug the program in your head, trying to figure out a cure by staring at the source code rather than testing reassembled object code. If you have to wait a long time for every assembly, your programming style will become distorted as you do everything possible to avoid another long wait.

How fast is fast enough? The best disk based assemblers will do roughly 1K a minute. This is about the fastest that assembly can be accomplished on larger ML programs because it's the speed at which a disk drive will deliver source code to an assembler. Larger programs are generally written in sections, each section then saved to disk and linked to the next file by an assembler command as the last instruction in the file. Composing programs of these linked modules is an efficient way to divide a complex job into manageable tasks.

RAM, too, can be too small to contain all the source code at once. Heavily commented source code can take up much more memory than its assembled object code. Here's a commented line of source code:

```
100 INY; RAISE THE COUNTER UNTIL IT
    REACHES 45
```

As source code, this line takes up 46 bytes. When assembled, the INY will be translated into the number 200. So, the object code for this line, the number 200, will be stored in one byte.

What do you do with an ML program after you assemble it? RUN? SYS?

BASIC programs are always located at the same place in a computer's memory, so there is a predictable starting address. The BASIC command RUN knows right where to go to find the first instruction in a BASIC program. RUN doesn't need an *argument* (a number or address following the command). You never need to specify where the RUN should start. You never need to say RUN 1024.

But ML programs can be located anywhere you want to put them. You can make them reside wherever you've got some free RAM. That's why an assembler will always want to know what starting address you've chosen. For example:

```
10 *= 12000
```

might be the first line of an ML program, the *= symbol signifying that address 12000 will be where the ML program begins. After it's assembled and is sitting in RAM from address 12000 on up in memory, you must SYS to execute the program. In this case, a SYS 12000 will turn control of the computer over to your ML program.

There is an ML instruction, RTS, which will return you to the normal BASIC environment after your ML program is finished.

How do you pass a number from ML to BASIC?

Just before you RTS back to BASIC, store the number (if it's less than 256) in address 251. Larger numbers are stored in ML by using additional bytes, as multiples of 256. So, you could use address 252 to hold a multiplier. For example, if you wanted to store 1024, you would stick a 4 into 252 and a 0 into 251. To store 1025, put 4 into 252 and 1 into 251.

Then, when you return to BASIC, you can get the number into a BASIC variable in this way:

```
10 X = PEEK (251) + PEEK (252) * 256
```

You can use address 251 and 252 for this since the computer leaves those addresses alone. Any address between 251–254 is safe to use. You can also use addresses 163–177 and 828–1019. In fact, if you know where your BASIC program ends and where your ML program resides, you could use any unused RAM to pass messages between BASIC and ML. You just want to avoid storing things on top of the resident programs.

There are other ways to pass numbers, but this is an easy and effective method.

IF-THEN, Logic, And Flags

John Michael Lane

If you've discovered a clever timesaving technique or a brief but effective programming shortcut, send it to "Hints & Tips," c/o COMPUTE!'s GAZETTE. If we use it, we'll pay you $35. Due to the volume of items submitted, we regret that we cannot always reply individually to submissions.

If it weren't for the IF-THEN statement, BASIC would be like a highway with no exit or entrance ramps. You could travel only one direction, from the beginning of the program to the end. IF-THEN, like its cousin ON-GOTO, is a decision-maker, a fork in the road. The ability to make logical decisions is what gives BASIC much of its programming power. It allows you to build entrance and exit ramps wherever you wish.

Truth And Falsity

Have you ever wondered what happens when your computer executes an IF-THEN statement? For example,

```
100 IF A=7 THEN B=C+5
```

In ordinary English, we'd say something like, "If A is seven then we'll make B equal to C plus five."

When your Commodore 64 or VIC-20 runs into an IF (condition) THEN (action) statement, it does something similar. It first checks the condition (Is A=7 true?). If it is true, it takes the action (make B=C+5). Otherwise, it goes to the next program line, skipping over anything else on the current line.

The letter A in the example above is a numeric variable, which acts as a number when you add, subtract, or do any other mathematical operations. But when you put an equals sign and another variable or number after it, the whole thing becomes an expression (A=7). When necessary, the BASIC interpreter evaluates these expressions and decides if they are true or false.

It may seem to be a simple task, figuring out if A=7, but BASIC has to be ready for almost anything. An IF expression may contain floating point, integer, or string variables. It can contain logical operators (NOT, AND, OR). There might be parentheses, to signal the order of evaluation. And any extra spaces have to be ignored, unless they are inside quotation marks.

Once the expression has been evaluated, a number is returned to the IF-THEN part of BASIC. If the expression is false, the number returned is zero. If it's true, the result is negative one (−1). It's not coincidental that the REMark section of the interpreter follows IF-THEN. When a false expression is found, your computer drops into REM mode, ignoring anything after the THEN, and looks for the next line of real BASIC.

To see how this logic works, enter the following line:

```
Q=9:PRINT "Q=9":PRINT Q=9
```

Your computer should respond by printing the string (Q=9), followed by a −1 (which means the expression is true, because you assigned the value of nine to the variable Q). If you try to PRINT Q=15, you should see a zero

(because the expression is false).

The three equals signs do three completely different things in the line above. The first one is an assignment-equals. It assigns the value of nine to the variable Q. You could also say LET Q=9, although LET is optional; it's rarely used anymore. After the PRINT, it's inside quotation marks and is simply a character-equals. The final time it is a comparison-equals, used to compare the numbers or variables on either side.

The difference between assignment and comparison is illustrated in this unusual-looking line:

```
R=5:S=R=5:PRINT S
```

First, we assign five to variable R. Next, the computer wants to assign a value to S. It decides that R=5 is an expression and does an evaluation (using a comparison-equals). The expression is true, which gives a value of minus one. That value is assigned to S, and a −1 is printed on the screen.

The greater-than (>) and less-than (<) symbols are also valid within an evaluation, although they cannot be used to assign values. LET A<9 doesn't make much sense, anyway; how would you assign a range of numbers to a single variable?

Why Minus One?

It's not hard to understand that zero means false. But why minus one for true? Why not ten, or one-half, or sixteen million?

Actually, you can use any number (except zero) to signal a true expression. It is fairly common to use a statement like this in a program:

```
10 IF A<>0 THEN PRINT "MESSAGE"
```

Knowing that zero always turns out to be false, and nonzero numbers are evaluated as nonfalse, you can make a modification to the above line. You want to find out if A is not equal to zero. Another way of interpreting it is, if A is not false (in other words, if A is true) then print the message. You can substitute this:

```
10 IF A THEN PRINT "MESSAGE"
```

Leaving off the <>0 saves some memory, and can be a valuable programming technique. You just have to remember that zero means false, and anything else counts as true.

What is a variable and what is an expression? In the example above, the variable A is evaluated for truth or falsity as if it were an expression. To turn it around the other way, you can use an expression as if it were a variable. If false, the expression is equivalent to a zero. If true, it's equivalent to minus one.

Let's say your bank charges a fee of 15 cents per check when your balance falls below $400. Otherwise, checks are free. In your checkbook balancing program, you might have these two lines:

```
130 BAL=BAL−CHK
132 IF BAL<400 THEN BAL=BAL−.15
```

You input the check's amount, and the program subtracts it from the balance and checks if the new balance is below $400 and subtracts the fee if necessary. Now look at this variation:

```
130 BAL = BAL−CHK +(BAL<400)*.15
```

First the check is subtracted from the old balance. Next, the expression BAL<400 is evaluated. If the balance is $400 or more, the expression is false, giving you a zero. Zero times 15 cents is zero and the new balance remains as is. But if the balance is below $400, the expression is true and fifteen cents is subtracted (or more accurately, minus fifteen is added).

But do you notice the bug in this line? The balance which is compared to $400 is the old balance. The bank will be looking at the new balance. To fix this, change the line to read:

```
130 BAL = BAL−CHK +((BAL−CHK)<400)*.15
```

We still haven't seen why a true statement is worth minus one.

For one thing, it makes certain situations work out nicely. Like subtracting fifteen cents when your balance goes below $400.

Specifically, however, in twos complement arithmetic, minus one is the logical opposite of zero. At the bit level, you flip the bits and add one. In BASIC, this is the equivalent of adding one and changing the sign. Ask your computer to PRINT NOT 8. You should see a −9 on the screen.

An interesting corollary to this is that if you are using a logical AND as a mask, zero masks everything and negative one masks nothing. In other words, for any number X, X AND 0 always result in zero, while X AND −1 always return X. It's similar to multiplication, where zero times any number yields zero, and one times any number gives back the number.

Waving A Logical Flag

Knowing how to use variables as expressions (IF A THEN xxx) and how to use expressions as variables (A = (B<15)*2) offers a lot of flexibility in BASIC programming.

Flags, for example, can be useful in almost any type of program. When you first type RUN, all variables are set to zero. So, if you have a variable called FLAG, you know it starts out being false. The flag is down. By assigning a value to FLAG, it is set, and you test it with a simple IF FLAG THEN (action), rather than the bulkier IF FLAG <> 0 THEN (action).

Disk Tricks

Gerald E. Sanders

Many operations with your 1540 or 1541 disk drive can be tedious and difficult. This article discusses how your drive works and then demonstrates some nifty tricks to help you get the most out of it. Included are programs which allow you to change a disk name, change a disk ID, unscratch, and scratch disk files.

Have you ever needed to unscratch a program or file on a Commodore 1540/1541 disk? Did you ever want to rename an old disk or give it a new disk ID without erasing the other files? Have you ever saved a program to disk and then seen a funny-looking title when you listed the directory? Or found you couldn't determine the right combination of characters to scratch the unwanted file? And then, did you search the disk manual in vain to find the commands to rescue you from your predicament?

While there are no neat, one-word commands to solve these types of problems, all the necessary information is there in the manual, although it's somewhat scattered and cryptic. All that's really necessary to do some effective tricks with disks is a rudimentary knowledge of the hexadecimal number system, the disk drive manual, a chart of ASCII (CHR$) codes, and the "Display T&S" program from the *TEST/DEMO DISK* which comes with the drive.

DOS Knows Where To Look

The 1540 and 1541 are called intelligent devices because each has its own microprocessor, RAM, and ROM. Like the VIC and 64, the drives contain an *operating system* program in ROM. For the drives the program is called, simply enough, the *disk operating system*, or DOS for short. The DOS controls all the operations of the drive.

To understand the operation of the drives, we first need to understand how the DOS knows where to look for a particular program. Data is stored on the disk in a series of concentric circular paths called *tracks*. These tracks are referred to by number, starting with track 1 at the outside edge of the disk, to track 35 near the inside edge. The tracks are further subdivided into *sectors*, or areas for storage. Sectors are synonymous with *blocks*, which you see on the left when you list a directory. Each sector can store 256 characters, or bytes, of data. A track can have 17–21 sectors, with tracks near the outside of the disk having the most sectors and tracks near the inside of the disk the fewest.

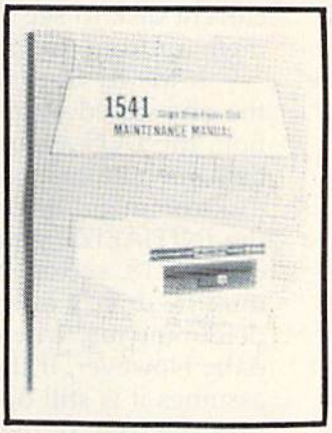
DOS reserves all of track 18 (the center track) to scribble notes to itself. Track 18 consists of 19 sectors (numbered 0–18), but let's look at the really significant sections of this track.

Sector 0 of this track contains the Block Availability Map, or BAM, which the DOS uses to keep track of the status of all the other sectors. Among other things stored in that sector are the 16-character disk name and the 2-character disk ID. These are stored in bytes 144–159 and bytes 162–163, respectively, as a series of binary numbers. Each number corresponds to the ASCII (CHR$) value of a character of the name or ID. If the name you gave the disk when it was formatted was shorter than 16 characters, the DOS added shifted spaces, CHR$(160), to get the total to 16. If your name was longer than 16 characters, it was truncated after the 16th one.

To see this, LOAD and RUN the Display T&S program from the *TEST/DEMO* disk. (For the VIC, this program requires at least 8K of memory expansion.) The program will ask if you

GOSUB

How to do your own maintenance, troubleshooting, schematics, theory of operation, cleaning hints, conversion from one power source to another and calibration. These topics and many more will make this manual a valued addition to your reference shelf. Whether you are an amateur electronics technician or a seasoned professional, you will be able to realize the full potential of your VIC-1541 by using this manual. Step-by-step instructions will lead you through the proper methods to get your VIC-1541 up and going in a hurry. The manual is 170 pages long, has two foldouts and over 100 illustrations, including:



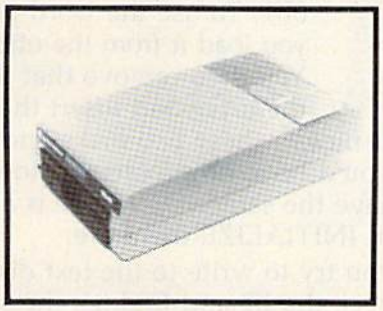
Block Diagrams
Schematics
Waveforms
Isometric (Pictorial) views
Test point locators

With all these illustrations and the detailed theory for each circuit involved, along with step-by-step procedures to follow, the manual is a great time and money saver.

CONTENTS OF MANUAL

Front Matter
Section 1 Introduction
Section 2 Theory of Operation
Section 3 Initial Configuration
Section 4 Performance Test
Section 5 Calibration
Section 6 Disassembly/Reassembly
Section 7 Preventive Maintenance
Section 8 Troubleshooting
Section 9 Schematics and Parts Layout
Appendices

Suggested list price: $39.95



HEAT DISSIPATING KIT
For VIC-1541 and 1541 Disk Drives

FEATURES:
Reduces internal temperatures to safe operating levels.
Does not promote dust migration.
No added noise.
Easy to install.
Increases life expectancy and reliability of disk drive.
Increases operating time and life of disk drive.
Installs on both VIC-1541 and 1541 Disk Drives.

The heat Dissipating kit cools the internal components of the Disk Drive by transferring internal heat to an external heat sink, where the heat is then dissipated into the surrounding air. The kit will lower operating temperatures of the IC's by as much as 20 degrees C (36 degrees F), and thus allow all the IC's to operate within their absolute maximum temperature ratings.

Suggested list price: $24.95

GRIDIRON STRATEGY '64

AND YOU THOUGHT FOOTBALL SEASON WAS OVER GRIDIRON STRATEGY '64 and the Commodore 64 now give you a year-round seat on the 50-yard line. GRIDIRON STRATEGY '64 is a highly realistic simulation of football instincts NOT "Joystick Reflexes". Most football games let you control a few players on a scrolling field. NOT GRIDIRON. In GRIDIRON, you coach the entire team and the colorful field and the stadium styled scoreboard are completely visible at all times. Also, with the use of TEAM DATA DISK '84, the teams you corftrol are the actual pro teams, based on their performances in the '84-'85 season. Disk can be updated every year, so you can constantly keep up with the rise and fall of each team. Finally, compare these features with any other football game on the market, for any other computer:

- Real time game and 30-second play clocks?
- Colorful Graphics, and Sprite animation?
- Realistic sounds of a packed stadium?
- Optional printout copy of plays and statistics?
- Individuaized teams, based on actual performances?
- 96 possible play combinations, infinite results?
- Does not require and charts or dice for results?
- In-depth playbook and strategy sections?

GRIDIRON STRATEGY '64 offers all of these qualities.

ORDER NOW!!!

Suggested retail price:
GRIDIRON STRATEGY '64 - $27.95
TEAM DATA DISK '84 - $14.95

FOR COMMODORE 64 OWNERS —

The Aventure Situation You've Waited For !

WIZARDS, WARLOCKS AND WARRIORS

Outfit a party of up to six adventurers, hand chosen from the characters guild, descend into the depths of a true 3-D dungeon, matching wits with dozens of orcs, wraiths, and other adversaries you've learned to hate. The only difference ... no more dice charts, or pleading for mercy with a ruthless dungeon master!

The first scenario is "Quest of the Dark Orb."; use it to learn, experiment, and increase the stregnth of your characters. 100% machine language programming, Hi-Res graphics, character print out sheets & a book on the nature of the adventure are included.

Suggested list price: $39.95

ORDER FROM:

GOSUB of Slidell, Inc.
P. O. Box 1781
Slidell, LA 70459
(504) 641-8307
MasterCard and VISA
Shipping & Handling $2.00
C. O. D. add $2.00

Dealer and Distributor inquires welcome

want results printed to the screen or printer. Although printing to the screen works fine for most needs, sending the results to the printer makes studying the process much easier.

The program then asks you to enter a track and sector number. Enter 18 for the track and 0 for the sector, and the program will begin to dump the contents of that sector to the chosen device. If dumped to the screen, the display will scroll. To slow down the scrolling, hold down the CTRL key. Pressing RUN/STOP will stop the dump, but it will also take you out of the program and you'll have to start all over to get the complete dump.

The contents of the sector (see the accompanying figure) are displayed as hexadecimal (hex) numbers, a common way of representing binary values. If you're not yet familiar with the hexadecimal numbering system, see "Hexed By Numbers," which accompanies this article. Hex numbers are usually prefixed with a dollar sign ($) to distinguish them from decimal numbers, but dollar signs aren't used in the dumps.

```
TRACK 18  SECTOR 0

00 :12 01 41 00 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F :  A  ππ  ππ  ππ
10 :15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F : ππ  ππ  ππ  ππ
20 :15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 14 FF FE 1F : ππ  ππ  ππ  π■
30 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
40 :00 00 00 00 00 00 00 00 0E 6C FB 07 00 00 00 00 :          └■
50 :00 00 00 00 00 00 00 00 00 00 00 00 0A 3E FB 00 :              >■
60 :13 FF FF 07 0D CF 3D 03 02 80 01 00 00 00 00 00 : ππ  ┌=  ■
70 :04 80 03 02 12 FF FF 03 12 FF FF 03 11 FF FF 01 : ■   ππ  ππ  ππ
80 :11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01 : ππ  ππ  ππ  ππ
90 :55 54 49 4C 49 54 49 45 53 20 30 30 31 A0 A0 A0 :UTILITIES 001
A0 :A0 A0 41 43 A0 32 41 A0 A0 A0 A0 00 00 00 00 00 :  AC 2A
B0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
C0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
D0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
E0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
F0 :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
```

Printed to the right of each line of the dump is the ASCII equivalent of the characters in that line. Here you'll see both graphics symbols and alphanumeric characters. The first two bytes of any block (sector) tell DOS where to look for the next block to read. On track 18, sector 0, these bytes should always be $12 (18) and $01 (1) to direct DOS to find the disk directory beginning at track 18, sector 1.

Name And ID

Continuing into the last 128 bytes (beginning at hex $80) of track 18, sector 0, we find other important information. Bytes 144–159 ($90–9F) hold the name we gave the disk when it was formatted. If the title contained less than 16 characters, you'll see shifted spaces, $A0, filling in the remaining characters. At bytes 162 and 163 ($A2 and $A3), you'll see the two alphanumeric characters used as the disk ID. The disk name is for your convenience and is not used by the DOS, unlike the ID, which is extremely important to the DOS.

After any read or write operation, including a SAVE or LOAD, the DOS places the current disk ID in the working storage area of its built-in memory. Whenever a new write command is sent to the drive, the DOS checks the ID on the current disk to see if it is the same as the one in memory from the previous operation.

If the ID from the current disk doesn't match the one stored in memory from the last operation, the DOS assumes it is working with a new disk and automatically executes an INITIALIZE disk command before it stores the data. During the INITIALIZE process, the BAM (bytes 4–143 of track 18, sector 0) of the new disk is loaded into the drive's internal memory so DOS can determine just where on the new disk it can put data. However, if the ID is the same, the DOS assumes it is still operating on the same disk and uses the copy of the BAM currently stored in its memory to determine which blocks are available to store the incoming data. This is part of the intelligence of the drive, but it works correctly only if you understand the process, and take advantage of it by giving every disk a unique ID.

Suppose you've saved a word processing program on a disk titled UTILITIES 001, and you store the text files you create with this word processor on another disk called TEXT FILES 001. To use the word processor, you load it from the utilities disk. You then remove that disk from the drive and insert the disk with the text files. If these two disks had different IDs, you'd have no problems. However, since they have the same ID, trouble is ahead if you forgot to INITIALIZE the drive.

When you try to write to the text disk, the DOS will check the ID and find it's the same as the one currently in its memory. Since the IDs are the same, the DOS thinks it is dealing with the same disk. It will use the BAM currently in memory to determine which blocks are available for text storage. As a result, DOS is likely to overwrite some or all of your text files, irretrievably destroying that data. Unique disk IDs prevent this kind of mistake.

Using all possible combinations of the letters A–Z and numbers 0–9 in a systematic way will give you 1,296 possible combinations. I started mine at AA and will go to AZ, then A0 to A9, then start over with BA. This works well, since Commodore's *TEST/DEMO DISK* has an ID of

ZZ. You can also use the graphics characters available when you press SHIFT or the Commodore key, which gives you an even greater number of possibilities.

Handling Files

The remaining sectors of track 18 list the programs by name and indicate where they are located. Use the Display T&S program to dump the contents of track 18, sector 1, the first block of the directory. Here's an example arrangement of a directory sector.

```
TRACK 18  SECTOR 1

00 :12 04 82 14 01 54 45 53 54 43 41 52 44 A0 A0 A0 :  ▮ TESTCARD
10 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 07 00 :
20 :00 00 82 11 01 44 49 53 50 4C 41 59 20 54 26 53 :  ▮ DISPLAY T&S
30 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 0E 00 :
40 :00 00 82 19 04 56 49 43 20 57 45 44 47 45 A0 A0 :  ▮ VIC WEDGE
50 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 04 00 :
60 :00 00 82 11 06 48 4F 4D 45 20 49 4E 56 20 31 20 :  ▮ HOME INV 1
70 :54 41 50 45 A0 00 00 00 00 00 00 00 00 00 09 00 :TAPE
80 :00 00 82 13 00 48 4F 4D 45 20 49 4E 56 20 31 20 :  ▮ HOME INV 1
90 :44 49 53 4B A0 00 00 00 00 00 00 00 00 00 09 00 :DISK
A0 :00 00 82 13 05 48 4F 4D 45 20 49 4E 56 20 32 20 :  ▮ HOME INV 2
B0 :54 41 50 45 A0 00 00 00 00 00 00 00 00 00 09 00 :TAPE
C0 :00 00 82 13 0E 48 4F 4D 45 20 49 4E 56 20 32 20 :  ▮ HOME INV 2
D0 :44 49 53 4B A0 00 00 00 00 00 00 00 00 00 0A 00 :DISK
E0 :00 00 82 10 00 53 50 45 45 44 54 59 50 45 A0 A0 :  ▮ SPEEDTYPE
F0 :A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 0D 00 :
```

As with track 18, sector 0, the first two bytes of the block point to the next block to be read. Notice that the directory does not use the sectors of track 18 sequentially. The first two bytes are $12 and $04, indicating that track 18, sector 4 is the next directory block. This staggered arrangement allows the drive to read the directory more swiftly. The last block of the directory will have $00 as the first byte in the block. Since there is no track 0 on the disk, this signals DOS that the end of the directory has been reached, and no more directory blocks will be read.

DOS scans the next byte to determine if the directory entry is valid, and if so, what kind of file the entry is. An $81 indicates a sequential (SEQ) data file, while $82, $83, $84 indicate program (PRG), user (USR), and relative (REL) files, respectively. A value of $80 or less signals a deleted or improperly closed file, and the DOS will treat this entry as if it doesn't exist.

Each directory sector can hold information on eight files. The other entries start at bytes 34, 66, 98, 130, 162, 194, and 226 ($22, $42, $62, $82, $A2, $C2, and $E2) in the block. In the previous figure, all these bytes are $82, so all the files in the example sector are programs. Storing eight entries per block in each of the 18 directory sectors allows a maximum of 144 entries in the directory.

The next two bytes (the fourth and fifth) tell the DOS where to look for this file. These two numbers give the track and sector of the first data block of the file. For the first entry in the previous figure, these bytes are $14 and $01, so the first file in the example directory, TESTCARD, is stored on the disk beginning at track 20, sector 1. The first two bytes of that block will, in turn, point to the second data block, and so on. As with the directory sectors, the file will be stored in staggered rather than sequentially numbered sectors, and the last block in the file will have $00 as the first byte of the sector.

Occasionally, these pointers may be disturbed, by turning off the drive while a disk is still inside, for example, resulting in spurious data being written to the disk. Knowing how the DOS finds its way around the disk, you may be able to locate and relink the program, thereby saving a precious original which otherwise might be lost forever.

The next 16 bytes of each directory entry contain the characters for the name of the file, padded with shifted spaces ($A0) if the name is less than 16 characters. The rest of the information contained in the directory entry is important for other applications, but space limits a full discussion.

Altering Disk Tracks

Now that we know how and where all the important information is stored, we can begin to perform the tricks mentioned earlier. Program 1, "Change Disk Name," allows you to change the name of a disk without erasing the disk as a NEW command will. Program 2 changes the disk ID without affecting disk contents. Program 3 helps you recover any previously scratched files, provided the DOS has not already overwritten them with another file. This program is 100% effective if used immediately after an erroneous SCRATCH command, but the likelihood of success diminishes rapidly as the number of files written to the disk after the SCRATCH increases.

When a file is scratched, the DOS updates the BAM to show that the blocks which were previously allocated to that file are now free to be overwritten. With luck, those blocks will not be used for some time; without luck, the next write may destroy part or all of the old file. And, finally, Program 4 allows you to scratch a file which cannot be deleted from BASIC. You have the option of permanently scratching the file and dropping it from the disk directory, or deleting the file, but leaving the name listed on the directory.

Hexed By Numbers

J. Blake Lambert, Assistant Editor

Reading and writing bytes, and directly modifying disk tracks and sectors is something like PEEKing and POKEing memory locations. You are advancing beyond simple BASIC programs into the realm of machine language. Sooner or later, you face the challenge of learning new, more convenient, numbering systems. Since computers only understand binary (base two) numbers and most people understand decimal (base ten) numbers, it's advantageous to learn how to translate from one to the other. To understand these number systems, let's think a minute about our normal counting method.

Our decimal (base ten) system evolved simply because humans have ten fingers. When we count in decimal, we start with zero and add ones until we get to nine, at which point we're out of numbers. To get to ten, we place a zero in the "ones" column and add a new column to the left, called the "tens" column. The next column to the left is named "hundreds," which is ten times ten (or 10↑2).

Binary counting is the same, but we run out of digits faster, because the computer has only two "fingers." That is, the electronic "switches" with which it counts are either off (0) or on (1). We count zero then one, and that's it. We start a new column called "twos," place a one in it, and clear the "ones" column. The next number is 11 and we're out of digits again. The "fours" column is created (two times two, or 2↑2), and we have the number 100, then 101, 110, 111 and 1000 (which equals eight in decimal, 2↑3). The problem with binary is that it takes too much room to write even small numbers. For example, the decimal number 255 written in binary is 11111111.

It requires 16 ones and zeros to describe a memory location on a VIC or 64. (Decimal 49152 translates to the unwieldy binary number 1100000000000000.) This is why hexadecimal, called hex by most computerists, is useful.

Hex is a simple shorthand for binary, but it requires learning a few new numbers. Just as the decimal number system has ten digits available (0 through 9), hex requires 16. These 16 digits are 0 through 9, then A, B, C, D, E, F. After counting to $F (the dollar sign represents "hex," the F stands for decimal 15), we next count $10, by placing a one in the "sixteens" column. The hex number $FF in decimal, then, equals 15 times 16, plus 15. Conveniently, this turns out to be decimal 255, so the eight-column binary representation (1111 1111) can be written using just two hex numerals ($FF). It's much easier to remember shorter numbers, so hex is usually preferred.

Here's a counting table that should help you understand how the systems work.

Binary	Decimal	Hexadecimal
0	0	$0
1	1	$1
10	2	$2
11	3	$3
100	4	$4
101	5	$5
110	6	$6
111	7	$7
1000	8	$8
1001	9	$9
1010	10	$A
1011	11	$B
1100	12	$C
1101	13	$D
1110	14	$E
1111	15	$F
10000	16	$10

These programs operate directly on data stored on the disk. They can do no permanent harm to the drive, DOS, or the disk itself; but, if the programs are not typed in carefully, they may destroy data stored on the disk. Such damage is usually repairable if you make a printout of the block on which you are working before you make any changes. If necessary, you can use these techniques to completely rewrite an entire block; I've done it before. But some mistakes can't be corrected, especially errors created on track 18, sector 0.

You don't have to understand how these programs work to use them, but let's look at a brief explanation. The computer sends information to the drive as fast as the drive can accept it. This means the computer may send one or two characters at a time, or it may send several thousand, depending on the situation. However, the DOS writes information to the disk only in whole 256-byte blocks. This means the drive must store incoming information in a buffer. The DOS maintains eight buffers in its built-in RAM. You can read a sector from the disk into a buffer

PRINTERS

Alphacom 40C/Int 99.95
Alphacom 80C/Int 189.95
Epson ..... Call
Gemini 10X 269.00
Okidata ..... Call
Silver Reed ..... Call
Prowriter 8510 ..... Call

MODEMS

Hayes Smart Modem 300 . Call
Mark VII/Auto Ans/
Auto Dial ..... Call
Mark XII/1200 Baud .... Call
Novation ..... Call

COMMODORE 64

Concorde Third Party Disk Drive for Commodore 64 parallel & serial models available ..... Call

TOUCH TABLETS

Koala Touch Tablet-D . 69.95
Koala Touch Tablet-Cart 74.95

CBM 64 ..... Call
SX-64 ..... Call
1541 Disk Drive ..... Call
1526 Printer 279.00
1530 Datasette 66.00
1702 Monitor ..... Call
1650 AD/AA Modem 89.00
RS 232 Interface ..... Call

Call for Special Package 64 System Price

SUPER PRINTER PACKAGES

Gemini 10X and
Cardco + G ... 349.00
Prowriter and
Cardco + G ... 419.00
No additional shipping charges on Printer Packages in Continental USA

MONITORS

USI ..... Call
AMDEK ..... Call

C O M M O D O R E 6 4 S O F T W A R E

ACCESS

Neutral Zone-D/T 23.95
Spritemaster-D/T 23.95
Beachhead-D/T 23.95
Master Composer-D ... 27.95

ACCESSORIES

WICO Joystick ..... Call
Flip'n'File-D 20.95
Flip'n'File Cart 20.95
Joysensor 24.95
Elephant Disks
(Box of 10) 20.00
WICO Trakball 37.95
KRAFT Joystick 15.95

ATARISOFT

Battlezone-Cart 34.95
Centipede-Cart 34.95
Defender-Cart 34.95
Dig Dug-Cart 34.95
Donkey Kong-Cart 34.95
Galaxian-Cart 34.95
Joust-Cart 34.95
Jungle Hunt-Cart 34.95
Moon Patrol-Cart 34.95
Ms. Pac-Man-Cart 34.95
Pac-Man-Cart 34.95
Pole Position-Cart 34.95
Robotron: 2084-Cart .. 34.95

BATTERIES INCLUDED

Consultant-D 69.95
Paperclip w/Spellpak-D 84.95
Super Busscard II ..... Call
Home Inventory-D 23.95
Recipe-D 23.95
Audio/Video Cat-D 23.95
Mail List-D 23.95
Stamps-D 23.95

BOOKS

Compute's Basic
Source Book 12.95
Compute's Machine
Lang/Beg 14.95
Compute's 1st Bk/64
Games 12.95
Com. 64 Program
Ref. Guide 19.95
Guide to Your Com. 64 . 14.95
Elementary Com. 64 ... 14.95
Power of Multiplan 14.95
Compute's 1st Bk/64
Sound/Graphics 12.95
Compute's 64 Ref Guide 12.95
Compute's 1st Book
of Com. 64 ..... 12.95

BRODERBUND

AE-D 23.95
Bank Street Writer-D ... 49.95
Choplifter-D 23.95
Drol-D 23.95
Loderunner-D 23.95
Matchboxes-D 20.95
Midnight Magic-D 23.95
Operation Whirlwind-D 27.95
Sea Fox-Cart 27.95
Serpentine-Cart 27.95
Spare Change-D 23.95
Mask of the Sun-D 27.95

CARDCO

Cardprint/B 47.95
Cardco+G 69.95
Cardboard/5 ..... 59.95
Cardkey 39.95
Cassette Recorder 47.95
Printer Utility-D/T 19.95
Write Now-Cart 34.95
Mail Now-D 29.95

CBS SOFTWARE

Argos Expedition-D 29.95
Charles Goren's Bridge-D 54.95
Coco Notes-D 24.95
Ducks Ahoy-D 24.95
Ernie's Magic Shapes-D . 24.95
Mastering the SAT-D . 104.95
Movie Musical
Madness-D 24.95
Murder by the Dozen-D . 23.95
Peanut Butter Panic-D . 24.95
Sea Horse Hide'n Seek-D . 24.95
Success Decimals
(Add/Subt)-D/T 19.95
Success Decimals
(Mult/Div)-D/T 19.95
Success Fractions
(Add/Subt)-D/T 19.95
Success Fractions
(Mult/Div)-D/T 19.95
Timebound-D 24.95
Webster Word Game-D . 24.95

COMMODORE

Program Ref. Guide 19.95
Assembler-D 39.95
Easy Finance I,II,III,IV-D . 19.95
Easy Calc-D 64.95
Easy Mail-D 19.95
Easy Script-D 44.95
Easy Spell-D 19.95
Logo-D 57.95
The Manager-D 39.95
General Ledger-D 39.95
Accts. Rec.-D 39.95
Accts. Pay.-D 39.95
Magic Desk-D 52.95
Zork I, II or III-D 32.95
Suspended-D 32.95
Starcross-D 32.95
Deadline-D 32.95
Soccer-Cart 29.95

CYBERIA

Farm Mgr. Vol I
General-D 37.95
Farm Mgr. Vol II Beef-D . 37.95
Farm Mgr. Vol III Pork-D 37.95
Farm Mgr. Vol IV Grain-D 37.95

CYMBAL

Accounts Payable-D 52.95
Accounts Receivable-D . 52.95
Inventory Control-D 52.95
Invoice Writer-D 52.95
Trivia -D 27.95

DYNATECH

Adventure Writer-D 37.95
Codewriter-D 69.95
Dialog-D 37.95
Elf System-D 37.95
Home File Writer-D 49.95

EPYX

Construction Crew-D .. 23.95
Dragons/Pern-D/T 27.95
Fax-D 20.95
Fire!-D 23.95
Fun With Art-Cart 27.95
Fun With Music-Cart .. 27.95
Fun With Words-Cart .. 27.95
Gateway to Apshai-Cart . 27.95
Jumpman Jr.-Cart 27.95
Jumpman-D/T 27.95
Lunar Outpost-D/T 23.95
Mission Impossible-D .. 23.95
Oil Barons-D 37.95
Pitstop-Cart 27.95
Puzzlemania-D 23.95
Robots of Dawn-D 27.95
Summer Games-D 27.95
Temple of Apshai-D/T .. 27.95

HANDIC

64 Forth-Cart 29.95
64 Graf-Cart 23.95
Stat 64-Cart 23.95
Calc Result Easy-Cart . 34.95
Calc Result Adv.-Cart 69.95
The Diary-Cart 23.95
The Tool-Cart 29.95

HESWARE

64 Forth-Cart 41.95
6502 Profess Dev Sys-T . 20.95
Coco-D/T 27.95
Factory-D 23.95
Finance Manager-D 49.95
Ghost Manor/Spike Pk-D . 19.95
Graphics Basic-Cart 34.95
HES Cat-D 19.95
HES Font-Cart 16.95
HES Games '84-D 27.95
HES Kit-Cart 34.95
HES Mon-Cart 27.95
HES Writer-Cart 30.95
Microsoft Multiplan-D . 69.95
Minnesota Fats' Pool-Cart . 20.95
Missing Links-D 20.95
Mr. TNT-Cart 20.95
Omniwriter/
Omnispell-D 49.95
Root n' Tootin-Cart 23.95
Synthesound-D 16.95
The Pit-Cart 20.95
Time Money Manager-D . 49.95
Turtle Graphics II-Cart ... 41.95
Turtle Toyland Jr.-D/T .. 23.95
Type n' Writer-D 20.95

INFOCOM

Enchanter-D 34.95
Infidel-D 34.95
Planetfall-D 34.95
Sea Stalker-D 34.95
Sorcerer-D 34.95
Witness-D 34.95

INSTA (CIMMARON)

Insta-Writer-Cart 39.95
Insta-Mail-D 24.95
Insta-File-D 49.95
Management Combo 64.95
Insta-Calc-Cart/D 31.95
Insta-Graph-D 24.95
Insta-Vestor-D 31.95
Insta-Speed-D 99.95
Insta-Music-Cart/D 79.95
Invest Combo 74.95

MICROFUN

Death in the Caribbean-D . 27.95
Dino Eggs-D 27.95
English
SAT I, II, or III-D 20.95
Globe Grabber-D 20.95
Highrise-D 20.95
Homewriter-D 34.95
Math
SAT I, II, or III-D 20.95
Personal Banker-D 34.95
The Heist-D 23.95
U.S. Constitution-D 20.95

MICROPROSE

Floyd/Jungle-D 23.95
Hellcat Ace-D/T 23.95
NATO Commander-D 23.95
Solo Flight-D/T 23.95
Spitfire Ace-D/T 23.95

MISCELLANEOUS

Ken Uston's
Blackjack-D 49.95
Quick Brown Fox-D/Cart . 34.95
Ultima III-D 41.95
Flight Simulator II-D 37.95
Night Mission/
Pinball-D/T 20.95
Praticalc PS-D 59.95
M-File-D 64.95
Word Pro 3+/Spell-D 74.95
Home Accountant-D 52.95
Step By Step-D/T 44.95
Barron's Sat.-D 67.95
Bristles-D/T 20.95
Telestar 64-Cart 37.95
Star League
Baseball-D/T 20.95
Castle Wolfenstein-D 20.95
Mastertype-D/Cart 27.95
Vic Switch 124.95
First Class Mail-D 34.95
Aztec-D 27.95

MISC. (cont'd.)

Miner 2049er-Cart 27.95
Sea Dragon-D/T 23.95
Diskey-D 34.95
Hodge Podge-D/T 19.95
Strip Poker-D 23.95
Mr. Robot-D 23.95
Paint Magic-D 34.95
Pooyan-D/T 20.95
Astro Chase-D/T 20.95
Flip Flop-D/T 20.95
Basic Building Blks-D . 54.95
Critical Mass-D 27.95
Rescue Squad-D 20.95
Super Text Word Pro.-D . 69.95
Beyond Wolfenstein-D . 23.95
Sam-D 41.95
Chatterbee-D 27.95

PARKER BROTHERS

Frogger-Cart 34.95
Gyruss-Cart 34.95
James Bond-Cart 34.95
Popeye-Cart 34.95
Q*Bert-Cart 34.95
Star Wars-Cart 34.95

SIERRA ON-LINE

Apple Cider Spider-D .. 20.95
Aquatron-D 20.95
Championship Boxing-D . 20.95
Dark Crystal-D 27.95
Frogger-D/T 23.95
Homeword Speller-D 34.95
Homeword-D 49.95
Learning With Leeper-D . 20.95
Lunar Leeper-D 20.95
Mission:Asteroid-D 20.95
Oil's Well-D 23.95
Prisoner-D 23.95
Quest For Tires-D 23.95
Threshold-D 27.95
Time Zone-D 74.95
Ultima II-D 41.95
Ultima I-D 23.95
Ulysses-D 27.95
Wizard/Princess-D 22.95
Wiztype-D 23.95

SPINNAKER

Adventure Creator-Cart . 27.95
Aerobics-D 30.95
Aegean Voyage-Cart 27.95
Alf in the Color Caves-C . 27.95
Alphabet Zoo-Cart 23.95
Bubble Burst-Cart 27.95
Cosmic Life-Cart 23.95
Delta Drawing-Cart 27.95
Facemaker-Cart 23.95
Fraction Fever-Cart 23.95
Grandma's House-D 23.95
Jukebox-Cart 27.95
Kids on Keys-Cart 23.95
Kidwriter-D 23.95
Kindercomp-Cart 20.95

SPINNAKER (cont'd.)

Ranch-Cart 27.95
Rhymes/Riddles-D 20.95
Search/
Amazing Thing-D 27.95
Snooper #1-D 30.95
Snooper #2-D 30.95
Story Machine-Cart 27.95
Trains-D 27.95
Up For Grabs-Cart 27.95

SSI

50 Million Crush-D 27.95
Battle/Normandy-D/T 27.95
Combat Leader-D/T 27.95
Computer Baseball-D 27.95
Cosmic Balance-D 27.95
Eagles-D 27.95
Fortress-D 23.95
Germany 1985-D 41.95
Knight/Desert-D/T 27.95
Professional Golf-D 27.95
RDF 1985-D 23.95
Ringside Seat-D 27.95
Tigers in the Snow-D .. 27.95

SYNAPSE

Blue Max-D/T 23.95
Drelbs-D/T 23.95
Fort Apocalypse-D/T .. 23.95
Necromancer-D/T 23.95
New York City-D/T 23.95
Pharoah's Curse-D/T .. 23.95
Protector II-D/T 23.95
Quasimodo-D/T 23.95
Rainbow Walker-D/T .. 23.95
Relax Stress
Reduction Sys. 64.95
Shamus Case II-D/T .. 23.95
Shamus-D/T 23.95
Slam-Ball-D/T 23.95
Survivor-D/T 23.95
Zaxxon-D/T 27.95
Zepplin-D/T 23.95

TIMEWORKS

Accounts Payable/
Checkwriter-D 41.95
Accounts Receivable/
Invoice-D 41.95
Cash Flow
Management-D 41.95
Cave/Word
Wizards-D/T 19.95
Data Manager 2-D 34.95
Data Manager-D/T 19.95
Dietron-D/T 19.95
Dungeon Algebra
Dragon-D/T 19.95
Electronic
Checkbook-D/T 19.95
General Ledger-D 41.95
Inventory Management-D . 41.95
Money Manager-D/T .. 19.95
Payroll Management-D ... 41.95

D-Disk T-Cassette Cart-Cartridge

To Order Call Toll Free
800-558-0003

For Technical Info, Order Inquiries, or for Wisc. Orders ·
414-351-2007

Hundreds of items available for the CBM 64 . please call

NEW LOWER PRICES

NEW PRODUCTS

ComputAbility™

NO SURCHARGE FOR MASTERCARD OR VISA

ORDERING INFORMATION. Please specify system. For fast delivery send cashier's check, money order or direct bank transfers. Personal and company checks allow 2 weeks to clear. Charges for COD are $3.00. School Purchase Orders welcome. In CONTINENTAL USA, include $3.00 shipping per software order. Include 3% shipping on all Hardware orders, minimum $3.00. Mastercard & Visa please include card # and expiration date. WI residents please add 5% sales tax. HI, AK, FPO, APO, Canadian orders — add 5% shipping, minimum $5.00. All other foreign orders, please add 15% shipping, minimum $10.00. All goods are new and include factory warranty. Due to our low prices, all sales are final. All defective returns must have a return authorization number. Please call 414-351-2007 to obtain an RA# or your return will NOT be accepted for replacement or repair. Prices and availability are subject to change without notice.

COMPUTABILITY
P.O. Box 17882
Milwaukee, WI 53217

ORDER LINES OPEN
Mon-Fri 11 AM - 7 PM CST
Sat 12 PM - 5 PM CST

using the block read (U1) command (see line 1050 of Program 1), and you can write the contents of the buffer back to the disk with the block write (U2) command (line 1080).

For a direct access operation, you gain access to a buffer by adding "#" after the OPEN statement for the data channel (see line 1040 of Program 1). The DOS will keep track of which buffer to use. Once a direct access channel is open to the disk, you can directly manipulate the contents of the buffer. Data may be read from or written to any byte of the buffer. The key to selecting a particular byte is the buffer pointer (B–P) command (line 1060). By setting the buffer pointer to the byte you wish to change, or the starting byte of a series, you can determine where in the buffer the PRINT# statement will place the data.

The programs are written so that they may be used alone. However, they will be much more useful if appended to the Display T&S program. If you chose to do this, simply remove the REM that begins the first and last line of each program (just the first occurrence of the word REM, not the whole line). The special features can then be accessed by RUN *xxxx* or GOTO *xxxx*, where *xxxx* is the second line of the appended program. For example, RUN 1000 would start the change disk name feature, and GOTO 3000 would start the unscratch feature. The programs are numbered so that you can add all four together to the Display T&S program to provide a versatile disk editing program.

Programs 3 and 4 require that you know the number of the byte where the directory entry starts. You can obtain the address of the target byte using the DISPLAY T&S program, convert that hex value into the correct decimal value, and supply this to the programs whenever they prompt for a buffer pointer value.

See program listings on page 149.

Dan Carmichael, Assistant Editor

Simulating Hi-Res Animation Part 1

The VIC-20 offers a lot of computing power for a very low cost. A major limitation, however, is its small amount of memory, only 3.5K (3583) bytes.

If you've ever tried to program hi-resolution graphics on the unexpanded VIC, you know just how restrictive this lack of memory can be. For example, setting up the entire screen for hi-res graphics on the VIC uses 4048 bytes, more memory than is available.

You can program a small hi-res "window" in the middle of the screen. The one recommended by the *Programmer's Reference Guide* is 64 × 64 pixels, and uses only 512 bytes. But this window is small, only 8 characters wide. So what do you do if you want smooth animation on the entire screen, and all you have is 3.5K?

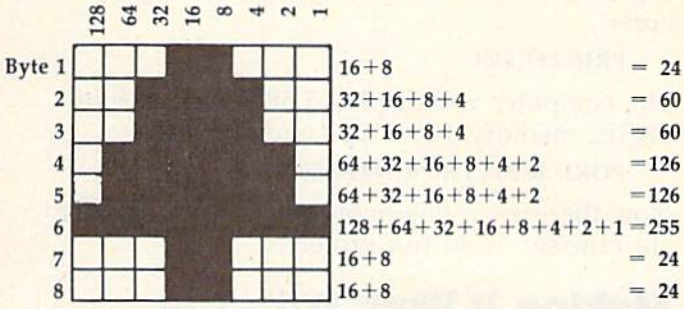
With custom characters, you can simulate hi-res animation. This simulation technique has an additional benefit: You can write your program in BASIC. Not that you can't program hi-res using BASIC. It's just that true hi-resolution animation should be programmed in machine language because it can be excruciatingly slow in BASIC.

Creating Custom Characters

With the VIC, you can redefine any character in the standard character set to display anything from a happy face to a flying saucer. Try this program, which changes the letter A to a pine tree.

```
10 POKE56,28:CLR:POKE36869,255:PRINT"
   {CLR}A"
20 FORA=7176TO7183:READB:POKEA,B:NEXT
30 DATA 24,60,60,126,126,255,24,24
40 GETA$:IFA$=""THEN40
```

Graphics characters in the VIC are composed of little dots (pixels) on an 8 × 8 grid. Each pixel is like a light switch—it's either on or off. When a pixel is on, it prints on the monitor screen as a dot. When it's off, it's blank. Characters are created by turning on and off various pixels to form a pattern:



DATA FOR PINE TREE IS:
24,60,60,126,126,255,24,24

Pine tree custom character as depicted on an 8 × 8 grid. Shaded areas are pixels that are turned on.

The data for each 8 × 8 grid is stored in eight bytes, each one corresponding to one of the horizontal rows on the grid. The bytes are consecutive, byte 1 corresponding to row 1 of the grid, byte 2 to row 2, and so on. Each of the eight bits in a byte corresponds to one of the eight vertical columns on the grid.

The value of each of the eight bytes is determined by which pixels are on. The values of the "on" pixels are then added together.

As an example, look at the first byte (row 1) in the figure. The pixels that are turned on correspond to the bit positions with the respective values of 16 and 8. So the total value of the first byte is 16 + 8 = 24. Using this formula to tally the totals for each of the eight bytes of the pine tree character, we get bytes with the values of 24, 60, 60, 126, 126, 255, 24, and 24.

A Safe Location

The data for the standard VIC character set can be found in a 4K block of memory in locations 32768 to 36863. Because this area of memory is ROM (Read Only Memory), you cannot POKE new values here to create the custom character set. The character set has to be moved to an area of RAM where the new values can be placed. A safe area in the unexpanded VIC is locations 7168–7679, the top of BASIC memory.

The first thing to do is to protect the character data so it cannot be overwritten by BASIC. This can be done with POKEs to locations 55 and 56, which point to the *top of BASIC memory.* Moving the top of memory down two pages (a page is 256 bytes), will reserve 512 bytes for the custom characters. This will be enough for 64 characters, because a complete character needs eight bytes. The normal value of byte 56 in the

unexpanded VIC is 30. So to set the custom character memory aside, POKE 56,28 then type and enter the BASIC command CLR (not the CLR/HOME key).

To see this demonstrated, turn your unexpanded VIC off and on again, then type and enter:

```
PRINTFRE(0)
```

The computer will display 3,581 free bytes of BASIC memory. Now type and enter:

```
POKE 56,28:CLR:PRINTFRE(0)
```

Now there are 3,069 bytes free. The 512 bytes are now set aside and protected.

Making It Work With The Expanded VIC

In the expanded (8K or more) VIC, reserving the memory for the custom characters is a little more complicated. The VIC chip, which points to the character set (more about this later), cannot "see" expansion memory. Because of this, you cannot reserve a few pages at the top of memory for the character set.

Not to worry, though—there's a way around this. If you move the start of BASIC up, you can POKE the custom characters into the RAM below BASIC.

Before performing any of the following POKEs, turn off your VIC, plug in an 8K or larger expander, then turn it back on.

When you use an 8K or more expander, BASIC memory starts at 4608. To make room for the custom characters, we'll move the start of BASIC to 5632. The area of memory running from 4608 to 5631 can then be used to safely store the custom characters. With this setup, the 64 custom characters will go into the 512 bytes of memory from 5120 to 5631.

Memory locations 43 and 44 control the *start of BASIC memory.* To change the beginning of BASIC, you have to POKE new values here. To move BASIC to 5632, POKE 44,22. A word of caution before performing any of these POKEs: Don't do so with a BASIC program in memory or you could lose part of your program.

The next set of pointers that has to be changed to move BASIC are bytes 45 and 46. These two bytes control the *start of variables.* As you write your BASIC programs, the values in these pointers continually change, always keeping the variables just three bytes past the end of your BASIC program. To move the start of variables, POKE 46,22.

One last pointer to change: bytes 641 and 642. These two bytes control the *start of the operating system.* Here we'll POKE 642,22.

The next and last step is to POKE a zero where the new start of BASIC is. To do this POKE 5632, 0 then type and enter NEW.

Following is a short machine language program that will automatically set up the expanded VIC as mentioned above. RUN the program, then LOAD your BASIC program.

```
10 FORA=8192TO8224:READB:POKEA,B:NEXT:SYS
   8192:CLR
20 DATA 169,0,141,129,2,141,0,22,141,1,22
   ,141,2,22,169,1,133,43,169,3,133,45
30 DATA 169,22,133,44,133,46,141,130,2,96
   ,234
```

Tell The VIC Where To Look

The VIC (Video Interface Chip) chip in your VIC-20 controls sound, video, modes of color operation, and more. It also tells the operating system where to find the character set. In order to use the custom characters, you have to tell the computer where to find them.

Memory location 36869 (on the VIC chip) is a pointer that tells the VIC where to get its character set information. To get the VIC to look at the custom character set, we have to POKE new values here. When using the areas of memory we've reserved, use one of the following POKEs:

POKE 36869,205 (For the 8K or more expanded VIC).
POKE 36869,255 (For the unexpanded VIC).

To return to the standard character set, POKE 36869,192 (for the expanded VIC), or POKE 36869,240 (for the unexpanded VIC).

Using The Standard Character Set

The 64 custom characters you create and store in the reserved area will correspond to the first 64 *screen POKE codes.* That is, the first eight bytes will correspond to screen *POKE* character 0 (@), the next eight bytes character 1 (A), and so forth.

If you wish to use part of the standard character set, it can be copied and placed into the area reserved for the custom characters. Use one of the following lines in your program to do so:

(for unexpanded VIC)

```
10 B=7168:C=7679:D=32768:FORA=BTOC:POKEA,
   PEEK(D):D=D+1:NEXT
```

(for expanded VIC)

```
10 B=5120:C=5631:D=32768:FORA=BTOC:POKEA,
   PEEK(D):D=D+1:NEXT
```

Creating your custom characters is up to you. There are many good reference materials available including the *Programmer's Reference Guide* and past issues of COMPUTE!'s GAZETTE.

Next month we'll design some custom characters, and look at how to use them to simulate smooth, high-resolution animation.

Animating The VIC

Mike Scharland

Fast animation in a BASIC program might seem to be a contradiction. The useful technique presented here can spice up almost any game that depends on fast action.

How many times have you wished you could add some blinking stars, whirring planets, or flashing explosions to your favorite game? If you've written any arcade-style games in BASIC, you may have tried this a few times. There are a variety of ways to speed up BASIC, using variables rather than ASCII numbers, for example.

But there comes a point when you can't add any more speed. The more characters there are to move around, the slower the game gets. And as the pace slows, the game loses its appeal.

The Multicolor Blinker

A fairly simple method (and it doesn't even involve machine language) is to use multicolor characters. To get an idea of how it looks, enter this short program:

```
10 POKE36878,15:POKE36879,11:PRINT"{CLR}"
20 POKE36876,INT(RND(1)*127+128):POKE3687
   8,100-SQR(100)
25 IFRND(1)*11>4THENPOKE36878,15:GOTO20
30 Y=INT(RND(1)*512+1):POKE38400+Y,INT(RN
   D(1)*9+8)
35 POKE7680+Y,42+RND(1)*2:POKE36878,15:GO
   TO20
```

Your screen should quickly fill up with several hundred flashing, blinking characters—a lot of action for a five-line program.

In a larger program, the flashing will be a little slower, depending on how many lines are executed in the loop.

The first line sets the volume of the sound and the screen color. Experimenting with different colors will give you different effects. The second line POKEs a random note into one of the sound registers. POKE 36878,100−SQR(100) is responsible for the blinking of the characters.

You could use 90 instead of 100, as in SQR(90), but in this case the time it takes to calculate a square root slows the program a little, to make the blinking more obvious.

Memory location 36878 does two things. The first four bits control the volume of sound (fifteen is the loudest). The last four bits control the auxiliary color in multicolor mode. If you set this color to match the screen, parts of the characters will seem to become invisible.

Line 25 checks the random number generator, in effect slowing down the program a little more.

Line 30 picks a random screen location and POKEs color memory with a number from eight to fifteen. Normally, you would use a number from zero to seven for character colors. Adding eight tells the VIC to switch from regular characters to multicolor characters.

Finally, an asterisk (*) or plus sign (+) is put on the screen, in line 35. Notice that lines 25 and 35 both POKE 36878,15. Rapidly alternating the value of the auxiliary color (the high nybble of 36878) gives the blinking effect.

Animating A Program

Following is a simple game which uses the technique described above. It's called "Pop Up" and runs on an unexpanded VIC with a joystick.

The object is to move the pi character around the screen without hitting any of the characters which keep popping up. You get one point for each space you move through. The game is fairly easy, and you should be able to survive a long time. The only danger is if a character pops up right on top of you.

Lines 57–70 illustrate the blinking character technique. Note that because of the time used to read the joystick and move the pi character, the blinking is slower in the game than in the example program.

Now that you have a new technique to add to your bag of BASIC tricks, you might want to experiment with custom characters in multicolor mode. Some of the effects with this technique are quite nice.

See program listing on page 171.

Screen Headliner

Todd Heimarck, Assistant Editor

This short machine language routine expands a letter to four times its normal size. The large character can then be used in a headline or for a variety of other purposes. The program is also compatible with Commodore printers. For the VIC and 64.

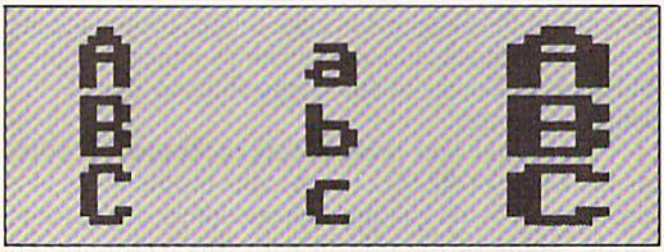
Oversized characters can be useful—on a title screen, in a children's alphabet or math program, or for visually impaired computer users. Finding the right combination of graphics characters usually takes time; you have to experiment. And creating a whole alphabet can use up a lot of memory.

The simplest method for displaying huge letters without experimenting or wasting memory is to PEEK the character generator in ROM and print a solid block (reverse space) for each bit that is on. And if the bit is off, you print a space. The one major disadvantage to this method is that each character expands to eight times its normal size. Very little space remains on the screen. But keeping in mind the idea of reading character ROM, we can sidestep this problem with some special Commodore characters.

The Quarter Square Solution

Hold down the Commodore key and type IKBVDCF. These seven characters, plus a blank space, make up half of the quarter square graphics set. The other half is accessed by typing the same keys while reverse is turned on. There are 16 different characters, one for each combination of quarter squares turned on or off.

Quarter squares enable you to set up what amounts to a medium-resolution screen. It's less complicated to program than a high-res screen, and has better resolution than the usual low-resolution character set. Instead of making characters turn on and off, you control big pixels (each of which is one fourth of a character). A VIC-20 suddenly has a 44 × 46 grid available; a 64 has the capability to address 80 × 50 big pixels.



Headliner is compatible with Commodore printers. Uppercase, lowercase, and enhanced print are illustrated here.

The 16 characters are the starting point for the "Screen Headliner." The basic idea is to read the character ROM, translate each bit into a big pixel, and print the equivalent quarter square graphics character. You can do it in BASIC with a lot of PEEKs and POKEs, but machine language is faster and more elegant.

The program is easy to use. After entering and SAVEing the program, type RUN. A short machine language program is POKEd into memory. To make it work, you need two POKEs and a SYS:

POKE 249,0: POKE 250,1: SYS 828

You should see a large capital "A," four characters wide and four deep. Now simultaneously press Commodore and SHIFT to switch to the upper/lowercase set. Cursor up to the POKEs, press RETURN, and you will see a large lowercase "a." Now try putting a 129 into location 250; the result is the same character printed in reverse.

If you've SAVEd a copy of Headliner, type NEW to erase the BASIC loader program. (It won't affect the ML program, which is safely tucked into the cassette buffer.) Now type this in:

```
2 MK=7:REM FOR VIC-20, USE 3
5 PRINT"{CLR}";
10 FORX=0TO255
20 Y=(XANDMK)*4:POKE249,Y
25 IFXANDMKTHENPRINT"{4 UP}";
30 POKE250,X:SYS828
40 NEXT
```

(Note: Tape users should not SAVE this example program; tape operations erase Headliner from the cassette buffer.) Type RUN and the whole Commodore character set will parade down the screen.

The top of the large character is printed wherever the cursor happens to be when you SYS. The POKE to 249 determines how far the cursor spaces over before it begins. The number

! WHY PAY MORE !

TWO SOPHISTICATED PROGRAMS FOR YOUR **COMMODORE 64™ & 1541**

THE WORD PROCESSOR!
TYPEWRITE©

- 51K machine language.
- Over 70 commands including:
- Right and left justify,
- Word wrap, • Pagination,
- Horizontal and vertical scrolling,
- Alpha-numeric sorting,
- Column manipulation,
- Global research and replace.
- Works on virtually any printer using utility program.

THIS IS A PROFESSIONAL WORD PROCESSOR AT A BUDGET PRICE!

INTRODUCTORY PRICE **$39.95**

NEW

LOST ANOTHER DISK!

GET *CARBON COPY*© NOW.

THE COPY UTILITY!
CARBON COPY©

- Guaranteed to make backup copies of 90% of all programs on the market.
- Includes "Error Maker" and "examine".
- Find, start and ending addresses.
- Copy "protected" disks.
- Copies entire disk in 3 swaps.
- Change heading and ID's.

*DON'T LOSE YOUR DISK! GET **CARBON COPY** TODAY!*

INTRODUCTORY PRICE **$39.95**

CALL TOLL FREE 1-800-663-4355 (USA OR CANADA)

OR SEND CERTIFIED CHEQUE OR MONEY ORDER TO:

SMART SOFTWARE — P.O. Box 526, Kelowna, B.C., Canada V1Y 7P1

SMART SOFTWARE LTD.

BOTH FOR $69.95

VISA (USE YOUR VISA OR MASTERCARD) MasterCard

THE BANNER MACHINE™

Menu-driven program works like a word processor. Great for businesses, schools or organizations. Produces large signs up to 13" tall by any length. Make borders of widths up to ¾". Eight sizes of letters from ¾" to 8" high. Proportional spacing; automatic centering; right and left justification. Use with Gemini 10 or 10X; Epson MX with Graftrax, or the RX or FX; Commodore 1525E or MPS 801; and the Banana. Four extra fonts available ($19.95 each). Tape or disk $49.95

Flex File 2.1 By Michael Riley. Save up to 1500 typical records on a 1541 disk drive. Print information on labels or in report format. Select records 9 ways. Sort on up to 3 keys. Calculate report columns. 1541·4040·2031. Disk $59.95

Disk Organizer Need to make a backup of your word processing files? No need to copy entire disks. Make a backup copy of a single file, copy a file, put the programs on the disk directory, in alphabetical order, maintain a library of the directories on all of your disks. Print a library listing for reference. Disk $24.95

Chessmate 64 Analyze your own games, master games, book games, and openings. Save, print, and watch your games in a unique "chess movie." Memorize any board position and recall it after you have played through variations. Disk $29.95

Grade Organizer Teachers— store grades for 6 classes, up to 40 students each, 680 grades per student. Print interim and final reports, class rosters and more! Disk $39.95

Cardinal Software™

13646 Jeff Davis Hwy.
Woodbridge, VA 22191
Order Toll Free: 800-762-5645
Information: 703-491-6509

Catalogs available. Specify: Business/ Utilities, Educational, or Games/Simulations

Commodore 64 and VIC-20 are registered trademarks of Commodore Electronics Ltd.

ENTER THE INFORMATION AGE WITH

VERSATERM II™
TERMINAL SOFTWARE FOR C-64

IT'S BEEN SAID THAT INFORMATION IS POWER, AND THE GROWING WEALTH OF INFORMATION AVAILABLE FROM INFORMATION SERVICES, BBS'S, AND ONLINE DATA BASES IS STAGGERING, NOT TO MENTION THE

FREE SOFTWARE

LOTS OF IT! AVAILABLE FROM BBS'S AND INFORMATION UTILITIES.

You can have all this power at your fingertips with your C 64, MODEM, and VERSATERM II™. Our extensive documentation will show you **how** to access this information.

GROWS WITH YOU

You don't need a disk drive, Printer, or Automodem to run VERSATERM II™, but as you expand your system, you will also expand the program's capabilities. Telecommunications is a fast growing field and we will continue to upgrade VERSATERM II™ as new technological advances are made. And we will be available to lend technical assistance after the sale.

TECHNICAL FEATURES

- **LARGE BUFFER** - 43,000 bytes of data storage.
- **UPLOAD** - text or programs using **ASCII** or **XMODEM** Protocol.
- **DOWNLOAD** - send data to buffer, disk or printer while on line. Convert **IMAGE FILES**. Transfer data using **XMODEM** file transfer protocol.
- **AUTODIAL** - store, retrieve, and automatically dial phone numbers using 1650 Automodem.
- **PROGRAM KEYBOARD** - program up to 25 keys with I.D. #'s, passwords, and often used commands, then send them with a single keystroke. Store the file on disk for later use.
- **PRINTER DUMP** - dump contents of buffer to printer.
- **LOAD** - Load the buffer from disk, tape, or keyboard for uploading.
- **SAVE** - Save the contents of the buffer to disk or tape.
- **CHANGE PARAMETERS** - BAUD RATE, DUPLEX, etc. menu selectable.

$34.95 DISK OR TAPE

STILL AVAILABLE - VERSATERM I™ FOR THE VIC-20

Upload/Download/Printer dump/Tape or Disk save/Sequential & Image file conversion. $24.95 Tape, $27.95 Disk

ALSO - EASY-BYTER™ EPROM PROGRAMMER. $99.95

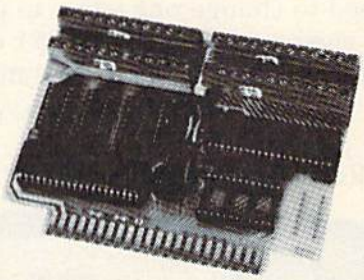
NEW RE-PORT™ ADAPTER CABLE TO ALLOW SX-64 OWNERS TO USE THE 1650 MODEM. CAN ALSO BE USED WITH THE C-64 OR VIC-20 TO "REMOTE" THE MODEM. 18" LONG. CALL FOR DETAILS & PRICE

NEW IMPORTED SOFTWARE FOR C-64. Spreadsheet, Home Manager, Arcade & Adventure games.

TERMS: Check, Money Order, C.O.D.

ELECTROSHARP
1981 Sandalwood Dr.
Santa Maria, CA 93455
(805) 922-4095 or (805) 736-0288

NEW!
Universal Input/Output Board for VIC-20/64



- 16 channel 8-bit A/D converter with 100 microsecond sampling time.
- 1 D/A output.
- 16 high voltage/high current discrete outputs.
- 1 EROM socket.
- Use multiple boards for additional channels up to 6 boards.

VIC-20 uses MW-311V $205.00
CBM-64 uses MW-311C $225.00

VISA MasterCard

Dealer inquiries invited.

Micro World Electronix, Inc.
3333 S. Wadsworth Blvd. #C105,
Lakewood, CO 80227
(303) 987-9532 or 987-2671

must be between 0 and 17 on a VIC, or between 0 and 35 on a 64.

Next, POKE the letter's screen code into 250. Ignore the ASCII value, you want the screen code—the number you use when POKEing a character to the screen. Numbers 1 through 26 are the letters A-Z, 48-57 are the characters zero through nine, and so on. To get a reversed character, add 128 to the screen code.

After you've POKEd into 249 and 250, enter SYS 828. The oversize character appears almost instantly.

Four Bonuses And A Drawback

The original version of this routine (used in "Aardvark Attack" a year ago, and more recently in "Campaign Manager") figured out the shape of the large character and POKEd the appropriate quarter square graphics to the screen. But Headliner now PRINTs (using the Kernal PRINT routine at $FFD2) instead of POKEing. It's necessary to turn reverse on and off repeatedly to get all the quarter squares, which is a little cumbersome. But there are some major advantages to sending everything through $FFD2.

The first advantage is that the VIC version doesn't care what kind of memory expansion is plugged in. You don't have to use different versions for different memory sizes: One program fits all. In fact, the Kernal PRINT vector is common to the VIC and 64.

Another bonus is that you can send large characters to a Commodore printer, although you need to change one value to print spaces instead of cursor-rights (see line 951 of the 64 version, VIC line 923). Enter this to make a printout:

```
OPEN 4,4: CMD4: POKE 249,xx: POKE 250,yy:
  SYS 828
```

Remember to replace *xx* with the location where you want to print, and substitute the screen code for *yy*. If you can, adjust your printer's line spacing to zero—so there is no extra space between the characters. When you're finished printing, PRINT#4:CLOSE4 properly closes the file to the printer. (See the figure for an example.) Unfortunately, printers do not allow cursor up movements; you are limited to one large character per line. To get around this limitation, you could manually move the paper back, or use a screen dump program, or (if you're feeling ambitious) use CMD to send output to a tape or disk file and then read the data back into an array for dumping to the printer.

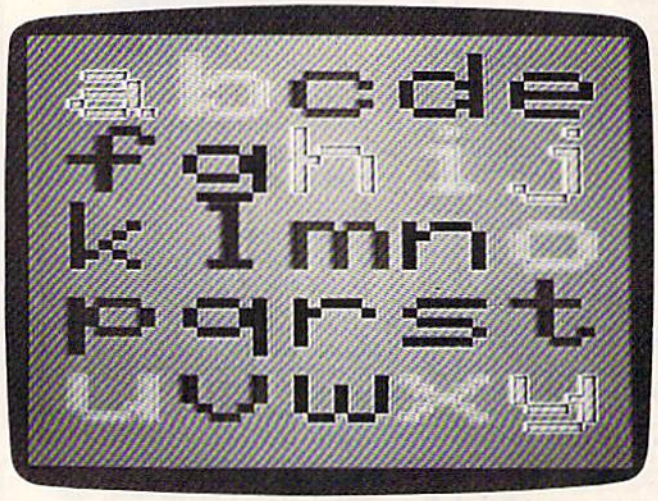
A third bonus of PRINTing rather than POKEing is that Screen Headliner is completely compatible with "Screen-80" (elsewhere in this issue); you can use large letters (up to 19 per line) in combination with 80-column text on your Commodore 64.

Finally, the flexibility of the PRINT command is at your fingertips: You can print almost anywhere on the screen, in any color you like (just change the cursor color). You can even mix large uppercase, lowercase, and graphics characters on the same screen.

A slight drawback is that each line has to be followed by a carriage return, which means you cannot put a character at the right edge of the screen.

How It Works

There are two sets of POKEs in the BASIC loader program. The first loop (688 to 703) contains the modified ASCII values of the quarter square graphics characters. Since there is no such thing as an ASCII value of a reversed character, the reverse flag has to be turned on and off. Bit 6 of each character is used to signal whether or not



The VIC version prints up to 25 large letters in any color.

When used with Screen-80, you get large condensed characters. (64 version—also works in full-color 40 columns).

the character is reversed; the number is then ANDed with $BF (191) to turn off bit 6 before the character is printed.

The second loop (828 on) is the machine language routine. It goes into the cassette buffer, but is written to be relocatable—if you need the cassette buffer for another ML program, or if you are using a Datassette, you can move the routine anywhere else in memory (the first loop has to stay where it is, however). If you put it in BASIC RAM, you'll have to protect it from being overwritten.

If you're interested in machine language, here's a brief explanation of how Headliner works. The main routine first checks which character set is being used and sets a zero page pointer accordingly. The screen code number is then multiplied by eight and added to the pointer. Once the pointer is set, the bytes from character ROM are loaded in two by two. By alternately shifting left the bytes (ASL) and rotating left the accumulator (ROL), a number from zero to fifteen is generated. This is used as an offset to look up the appropriate quarter square graphics character in the table at 688. Bit six is checked (if set, reverse is turned on) and finally, a JSR to $FFD2 prints the character. The program then loops back to get the next set of bits.

See program listings on page 147.

HORIZONS: 64

Charles Brannon
Program Editor

Several readers have found that some programs won't run on Commodore's SX-64 portable computer, first covered here in the July column. At the time, we had no problems with the portable, and found no incompatible software. After nearly 90 days of transporting the machine to and from the office, we found some problems using SpeedScript with it. The directory listed very slowly and would sometimes freeze up the machine. Also, disk loads and saves were unreliable, took too long, and would sometimes lock up the system (although the RUN/STOP key would break out of the lock-up). Printer output worked just fine.

SpeedScript was carefully written to avoid illegal programming techniques which might cause a program to be incompatible with future revisions of the 64. The copy files program on the 1541 test/demo disk will also not run properly on the SX-64, but the same program on the SX-64 test/demo disk will work, indicating some change has been made. As far as we can tell, the problem seems to be in the disk hardware, or the DOS used in the SX-64's disk drive. Several readers have had this problem.

Some of the exciting market entries announced at CES are the new thermal transfer printers, which have made high-quality printing

very affordable. Unlike some thermal printers, which use heat to change the color of special thermal-sensitive paper, thermal transfer printers can print on ordinary paper. The ribbon uses a waxlike ink. The print head heats and melts the ink onto the paper. The technology is relatively simple, so some *color* thermal transfer printers cost under $200. These printers are extremely quiet, and the quality is actually better than impact dot matrix printers, with a raised type you can actually feel. Thermal transfer printers are a little slower (around 60 characters per second), and the ribbon only lasts for about 50 pages. We're currently working with Commodore's new MCS-801 color dot matrix printer. This is not a thermal transfer printer, but prints across a four-color ribbon. It's similar to the MPS-801, but has a few features of the 1526. Unfortunately, the codes and modes used to program the MCS-801 are not fully compatible with those used on either the MPS-801, 1525, or 1526 printers.

The case is charcoal gray, Commodore's new favorite color, and it looks strikingly similar to Commodore's newest computer, the Plus/4. When setting up the printer, you'll need a lot of patience. I spent 15 minutes trying to follow the poorly translated (from Japanese, obviously) manual to install the ribbon. Obviously, this is only a preliminary manual, which, despite its claim to be a friendly guide to using your MCS-801, harks back to the dark ages of computer documentation.

Most ribbons rest horizontally, parallel to the carriage, and I struggled with the ribbon, trying in vain to see how it could possibly fit. The protrusions on the bottom of the ribbon couldn't match any sockets behind the print head. Finally, it all fell into place—literally. The crazy ribbon clicked in securely at a 45-degree angle. Then I realized that at a 45-degree angle, the print head can print across all four colors in a single pass.

The ribbon has four reservoirs of ink, which paint the ribbon to keep the ink fresh and wet. Each reservoir can be turned on and off to prevent drying out if you are not using certain colors. It's probably best to turn off the colors when you shut down the printer.

By combining colors, the MCS-801 can print text and graphics in the eight primary Commodore colors: white, black, red, cyan, purple, green, blue, and yellow. You open a channel to the printer as usual, OPEN 4,4. You can then print text with PRINT#4. To change colors, you use PRINT#4,CHR$(20);CHR$(*n*). The second value, *n*, is a number from 0–7. The printer also has two graphics modes. One prints much like Epson or Prowriter graphics; you send a code designating how many columns of graphics dots

you want to print, then send the graphics bytes. Another way of printing is similar to how TV sets are refreshed, one pixel row at a time, sweeping left to right. You tell the printer how many columns and rows you want to send, then give one byte for each dot. It appears to be the simplest mode to use.

Other features include elongated characters, a single programmable character, full support of keyboard graphics and reverse video, and a listing mode that prints program listings as they appear on the screen. The printer is fairly fast, clipping along at 50 characters per second.

To our knowledge, there is no software that supports this printer. Unfortunately, there are only a few sample programs in the manual, and no program to perform a full-color high-resolution screen dump. You have to write your own to support this printer (some word processors, including SpeedScript, can change printing colors with embedded codes). Amazingly, the manual states that the primary, most important use of the printer is to print program listings.

As you read in our CES feature, there is an overwhelming amount of new hardware and software for the 64. The flow of software has swelled from a trickle to a torrent, and now the dam has burst. We'll try to review the best and the brightest here, with an emphasis on uniqueness. Write and tell us what you want to see: more hardware reviews, more game reviews, or reviews of programming languages and utilities. If you want to see fewer reviews and more programming tips and tutorials, let us know. Any other ideas and suggestions are also greatly appreciated. Send your comments to the attention of Horizons 64.

Cursor GET For VIC And 64

David Mills

This practical subroutine lets you create a cursor for use during GET routines.

It sometimes makes more sense to use GET rather than INPUT when asking a user for information. The GET command is a little more flexible and gives you more control over the characters entered.

But there is a drawback. INPUT gives you a blinking cursor, which you don't have with GET.

Because the cursor is often convenient (sometimes essential), let's look at a subroutine to provide a cursor while using the GET statement for input.

First, to see why the cursor is significant, type the following program:

```
10 FOR K=1 TO 30
20 GET A$:IF A$="" GOTO 20
30 PRINT A$;:NEXT K
```

When you run this program, the computer will GET and PRINT 30 keystrokes. Notice that the cursor has vanished.

The vanishing cursor can be a big problem if you include keystrokes such as cursor movement, RETURNs, DELETEs, etc., in your input. In these cases it's very easy to forget where the cursor is. The only way to find out is to start typing and see where the letters appear on the screen.

Creating A Cursor

The short subroutine provided with this article will provide the standard blinking cursor format during GET routines.

It is invisible to the host program even if it is embedded, because the first statement sends the host program around the subroutine. Also, to minimize the chance of interference with any other program, I've used variables starting with X. This makes it harder to follow, but safer to use as long as you avoid such variables in the main program.

It's called by using GOSUB 1102, and on return A$ will have the character from the GET statement.

In lines 1102–1104, XL% is set to the memory address of the screen cursor. In line 1104,

XC is set to the color memory address and the program compensates for different memory sizes on the VIC. In line 1105 the current color (PEEK(646)) is put in the color memory, XO% is set to the character at the cursor, and XT% and XQ% are set up for the blink process. In line 1106, XT% is reversed and POKEd into the cursor position on the screen. Then XQ% is reset for the next blink, and a FOR/NEXT loop that actually gets the character is started.

If something has been typed in, line 1107 resets the screen and RETURNs; otherwise, the FOR/NEXT loop continues in line 1108. When the loop is complete, the screen character is reversed again and the process repeats.

You can remove the few REMarks in this program except line 1109. Line 1101 directs the host program to line 1109. If you remove the REM statement, line 1109 vanishes and you'll get an execution error.

Refer to the "Automatic Proofreader" article before typing these programs.

Program 1: Cursor GET For VIC-20

```
10 GOSUB1102:PRINTA$;:GOTO10:REM THIS IS
   {SPACE}THE "HOST" PROGRAM        :rem 77
1101 GOTO1109:REM GET WITH CURSOR BLINK
                                    :rem 79
1102 XL%=PEEK(211)                  :rem 212
1103 IFXL%>21THENXL%=XL%-22:GOTO1103
                                    :rem 133
1104 XL%=XL%+PEEK(214)*22+4096:XC=33792+X
     L%:IFPEEK(210)>20THENXC=XC+512:XL%=X
     L%+3584                        :rem 70
1105 POKEXC,PEEK(646):XO%=PEEK(XL%):XT%=X
     O%:XQ%=128:IFXO%>127THENXQ%=-XQ%
                                    :rem 235
1106 XT%=XT%+XQ%:POKEXL%,XT%:XQ%=-XQ%:FOR
     XR=1TO60:REM CHANGING 60 CHANGES BLI
     NK SPEED                       :rem 238
1107 GETA$:IFA$<>""THENPOKEXL%,XO%:RETURN
                                    :rem 51
1108 NEXT XR:GOTO1106               :rem 238
1109 REM                            :rem 175
```

Program 2: Cursor GET For The 64

```
10 GOSUB1102:PRINTA$;:GOTO10:REM THIS IS
   {SPACE}THE "HOST" PROGRAM        :rem 77
1101 GOTO1109:REM GET WITH CURSOR BLINK
                                    :rem 79
1102 XL%=PEEK(211)                  :rem 212
1103 IFXL%>39THENXL%=XL%-40:GOTO1103
                                    :rem 142
1104 XL%=XL%+PEEK(214)*40+1024:XC=54272+X
     L%                             :rem 87
1105 POKEXC,PEEK(646):XO%=PEEK(XL%):XT%=X
     O%:XQ%=128:IFXO%>127THENXQ%=-XQ%
                                    :rem 235
1106 XT%=XT%+XQ%:POKEXL%,XT%:XQ%=-XQ%:FOR
     XR=1TO60                       :rem 81
1107 GETA$:IFA$<>""THENPOKEXL%,XO%:RETURN
                                    :rem 51
1108 NEXT XR:GOTO1106               :rem 238
1109 REM                            :rem 175
```

MLX Machine Language Entry Program

For Commodore 64

Charles Brannon, Program Editor

MLX is a labor-saving utility that allows almost failsafe entry of machine language programs published in COMPUTE!'s GAZETTE. You need to know nothing about machine language to use MLX—it was designed for everyone. There are separate versions for the Commodore 64.

MLX is a new way to enter long machine language (ML) programs with a minimum of fuss. MLX lets you enter the numbers from a special list that looks similar to BASIC DATA statements. It checks your typing on a line-by-line basis. It won't let you enter illegal characters when you should be typing numbers. It won't let you enter numbers greater than 255 (forbidden in ML). It won't let you enter the wrong numbers on the wrong line. In addition, MLX creates a ready-to-use tape or disk file. You can then use the LOAD command to read the program into the computer, as with any program:

```
LOAD "filename",1,1   (for tape)
LOAD "filename",8,1   (for disk)
```

To start the program, you enter a SYS command that transfers control from BASIC to machine language. The starting SYS number always appears in the appropriate article.

Using MLX

Type in and save the correct version of MLX for your computer (you'll want to use it in the future). When you're ready to type in an ML program, run MLX. MLX asks you for two numbers: the starting address and the ending address. These numbers are given in the article accompanying the ML program.

You'll see a prompt corresponding to the starting address. The prompt is the current line you are entering from the listing. It increases by six each time you enter a line. That's because each line has seven numbers—six actual data numbers plus a *checksum number*. The checksum verifies that you typed the previous six numbers correctly. If you enter any of the six numbers wrong, or enter the checksum wrong, the computer rings a buzzer and prompts you to reenter the line. If you enter it correctly, a bell tone sounds and you continue to the next line.

MLX accepts only numbers as input. If you make a typing error, press the INST/DEL key; the entire number is deleted. You can press it as many times as necessary back to the start of the line. If you enter three-digit numbers as listed, the computer automatically prints the comma and goes on to accept the next number. If you enter less than three digits, you can press either the SPACE bar or RETURN key to advance to the next number. The checksum automatically appears in inverse video for emphasis.

To simplify your typing, MLX redefines part of the keyboard as a numeric keypad (lines 581–584):

```
  U I O                 7 8 9
H J K L   become 0      4 5 6
  M , .                 1 2 3
```

MLX Commands

When you finish typing an ML listing (assuming you type it all in one session) you can then save the completed program on tape or disk. Follow the screen instructions. If you get any errors while saving, you probably have a bad disk, or the disk is full, or you've made a typo when entering the MLX program itself.

You don't have to enter the whole ML program in one sitting. MLX lets you enter as much as you want, save it, and then reload the file from tape or disk later.

MLX recognizes these commands:

SHIFT-S: Save	**SHIFT-N: New Address**
SHIFT-L: Load	**SHIFT-D: Display**

When you enter a command, MLX jumps out of the line you've been typing, so we recommend you do it at a new prompt. Use the Save command to save what you've been working on. It will save on tape or disk as if you've finished, but the tape or disk won't work, of course, until you finish the typing. Remember what address you stop at. The next time you run MLX, answer all the prompts as you did before, then insert the disk or tape. When you get to the entry prompt, press SHIFT-L to reload the partly completed file into memory. Then use the New Address command to resume typing.

To use the New Address command, press SHIFT-N and enter the address where you previously stopped. The prompt will change, and you can then continue typing. Always enter a New Address that matches up with one of the line numbers in the special listing, or else the checksum won't work. The Display command lets you display a section of your typing. After you press SHIFT-D, enter two addresses within the line number range of the listing. You can abort the listing by pressing any key.

What if you forgot where you stopped typing? Use the Display command to scan memory from the beginning to the end of the program. When you reach the end of your typing, the lines will contain a random pattern of numbers. When you see the end of your typing, press any key to stop the listing. Use the New Address command to continue typing from the proper location.

See program listing on page 170.

The Automatic Proofreader

"The Automatic Proofreader" will help you type in program listings from COMPUTE!'s Gazette without typing mistakes. It is a short error-checking program that hides itself in memory. When activated, it lets you know immediately after typing a line from a program listing if you have made a mistake. Please read these instructions carefully before typing any programs in COMPUTE!'s Gazette.

Preparing The Proofreader

1. Using the listing below, type in the Proofreader. The same program works on both the VIC-20 and Commodore 64. Be very careful when entering the DATA statements — don't type an l instead of a 1, an O instead of a 0, extra commas, etc.

2. SAVE the Proofreader on tape or disk at least twice *before running it for the first time.* This is very important because the Proofreader erases this part of itself when you first type RUN.

3. After the Proofreader is SAVEd, type RUN. It will check itself for typing errors in the DATA statements and warn you if there's a mistake. Correct any errors and SAVE the corrected version. Keep a copy in a safe place — you'll need it again and again, every time you enter a program from COMPUTE!'s Gazette.

4. When a correct version of the Proofreader is RUN, it activates itself. You are now ready to enter a program listing. If you press RUN/STOP-RESTORE, the Proofreader is disabled. To reactivate it, just type the command SYS 886 and press RETURN.

Using The Proofreader

All VIC and 64 listings in COMPUTE!'s Gazette now have a *checksum number* appended to the end of each line, for example *":rem 123". Don't enter this statement when typing in a program.* It is just for your information. The rem makes the number harmless if someone does type it in. It will, however use up memory if you enter it, and it will confuse the Proofreader, even if you entered the rest of the line correctly.

When you type in a line from a program listing and press RETURN, the Proofreader displays a number at the top of your screen. *This checksum number must match the checksum number in the printed listing.* If it doesn't, it means you typed the line differently than the way it is listed. Immediately recheck your typing. Remember, don't type the rem statement with the checksum number; it is published only so you can check it against the number which appears on your screen.

The Proofreader is not picky with spaces. It will not notice extra spaces or missing ones. This is for your convenience, since spacing is generally not important. But occasionally proper spacing *is* important, so be extra careful with spaces, since the Proofreader will catch practically everything else that can go wrong.

There's another thing to watch out for: if you enter the line by using abbreviations for commands, the checksum will not match up. But there is a way to make the Proofreader check it. After entering the line, LIST it. This eliminates the abbreviations. Then move the cursor up to the line and press RETURN. It should now match the checksum. You can check whole groups of lines this way.

Special Tape SAVE Instructions

When you're done typing a listing, you must disable the Proofreader before SAVEing the program on tape. Disable the Proofreader by pressing RUN/STOP-RESTORE (hold down the RUN/STOP key and sharply hit the RESTORE key). This procedure is not necessary for disk SAVEs, *but you must disable the Proofreader this way before a tape SAVE.*

SAVE to tape erases the Proofreader from memory, so you'll have to LOAD and RUN it again if you want to type another listing. SAVE to disk does not erase the Proofreader.

Since the Proofreader is a machine language program stored in the cassette buffer, it will be erased during a tape SAVE or LOAD. If you intend to type in a program in more than one sitting or wish to make a safety SAVE, follow this procedure:

1. LOAD and RUN the Proofreader.
2. Disable it by pressing RUN/STOP–RESTORE.
3. Type the following three lines in direct mode (without line numbers):

```
A$="PROOFREADER.T":B$="{10 SPACES}":FO
RX=1TO4:A$=A$+B$:NEXTX
FORX=886 TO 1018:A$=A$+CHR$(PEEK(X)):N
EXTX
OPEN1,1,1,A$:CLOSE1
```

After you type the last line, you will be asked to press RECORD and PLAY. We recommend you start at the beginning of a new tape.

You now have a new version of the Proofreader (PROOFREADER.T, as renamed in the above code). Turn your computer off and on, then LOAD the program you were working on. Put the cassette containing PROOFREADER.T into the tape unit and type:

OPEN1:CLOSE1

You can now get into the Proofreader by typing SYS 886. To test this, PRINT PEEK (886) should return the number 173. If it does not, repeat the steps above, making sure that A$ (PROOFREADER.T) contains 13 characters and that B$ contains 10 spaces.

The new version of Automatic Proofreader will load itself into the cassette buffer whenever you type OPEN1:CLOSE1 and PROOFREADER.T is the next program on your tape. It will not disturb the contents of BASIC memory.

Automatic Proofreader For VIC And 64

```
100 PRINT"{CLR}PLEASE WAIT...":FORI=886TO
    1018:READA:CK=CK+A:POKEI,A:NEXT
110 IF CK<>17539 THEN PRINT"{DOWN}YOU MAD
    E AN ERROR":PRINT"IN DATA STATEMENTS.
    ":END
120 SYS886:PRINT"{CLR}{2 DOWN}PROOFREADER
     ACTIVATED.":NEW
886 DATA 173,036,003,201,150,208
892 DATA 001,096,141,151,003,173
898 DATA 037,003,141,152,003,169
904 DATA 150,141,036,003,169,003
910 DATA 141,037,003,169,000,133
916 DATA 254,096,032,087,241,133
922 DATA 251,134,252,132,253,008
928 DATA 201,013,240,017,201,032
934 DATA 240,005,024,101,254,133
940 DATA 254,165,251,166,252,164
946 DATA 253,040,096,169,013,032
952 DATA 210,255,165,214,141,251
958 DATA 003,206,251,003,169,000
964 DATA 133,216,169,019,032,210
970 DATA 255,169,018,032,210,255
976 DATA 169,058,032,210,255,166
982 DATA 254,169,000,133,254,172
988 DATA 151,003,192,087,208,006
994 DATA 032,205,189,076,235,003
1000 DATA 032,205,221,169,032,032
1006 DATA 210,255,032,210,255,173
1012 DATA 251,003,133,214,076,173
1018 DATA 003
```

Power Basic

(Article on page 136.)

> BEFORE TYPING...
>
> Before typing in programs, please refer to "How To Type COMPUTE!'s Gazette Programs," "A Beginner's Guide To Typing In Programs," and "The Automatic Proofreader" that appear before the Program Listings.

Program 1:

Screen Headliner—64 Version

```
5 PRINT"{CLR}PLEASE WAIT A MOMENT"
                                        :rem 153
10 T=0:FORJ=688TO703:READK:T=T+K:POKEJ,K:
   NEXT                                 :rem 134
15 IFT<>3078THENPRINT"ERROR IN DATA STATE
   MENTS":STOP                          :rem 88
20 T=0:FORJ=828TO1006:READK:T=T+K:POKEJ,K
   :NEXT                                :rem 176
25 IFT<>20306THENPRINT"ERROR IN DATA STAT
   EMENTS":STOP                         :rem 130
30 POKE249,0                            :rem 141
688 DATA32,188,190,226,172,225,191,251
                                        :rem 148
696 DATA187,255,161,236,162,254,252,96
                                        :rem 158
828 DATA 169,208,133,004,173,024  :rem 46
834 DATA 208,041,002,240,004,169  :rem 32
840 DATA 216,133,004,169,000,162  :rem 31
846 DATA 003,006,250,042,202,208  :rem 28
852 DATA 250,024,101,004,133,004  :rem 19
858 DATA 165,250,133,003,173,014  :rem 40
864 DATA 220,041,254,141,014,220  :rem 27
870 DATA 165,001,041,251,133,001  :rem 23
876 DATA 169,000,133,250,169,005  :rem 46
882 DATA 133,002,160,000,177,003  :rem 26
888 DATA 133,005,230,003,177,003  :rem 36
894 DATA 133,006,230,003,198,002  :rem 36
900 DATA 240,028,162,004,169,000  :rem 28
906 DATA 006,006,042,006,006,042  :rem 25
912 DATA 006,005,042,006,005,042  :rem 20
918 DATA 164,250,153,048,002,230  :rem 38
924 DATA 250,202,208,232,240,210  :rem 26
930 DATA 165,001,009,004,133,001  :rem 20
936 DATA 173,014,220,009,001,141  :rem 28
942 DATA 014,220,160,000,166,249  :rem 33
948 DATA 240,008,169              :rem 229
951 DATA 029:REM 032 IF USING A PRINTER
                                        :rem 129
952 DATA 032,210                  :rem 14
954 DATA 255,202,208,250,169,004  :rem 45
960 DATA 133,006,185,048,002,170  :rem 38
966 DATA 189,176,002,133,005,041  :rem 46
972 DATA 064,240,005,169,018,032  :rem 43
978 DATA 210,255,165,005,041,191  :rem 46
984 DATA 032,210,255,169,146,032  :rem 47
990 DATA 210,255,200,198,006,208  :rem 43
996 DATA 221,169,013,032,210,255  :rem 43
1002 DATA 192,016,208,196,096     :rem 153
```

Program 2:

Screen Headliner—VIC Version

```
5 PRINT"{CLR}PLEASE WAIT A MOMENT"
                                        :rem 153
10 T=0:FORJ=688TO703:READK:T=T+K:POKEJ,K:
   NEXT                                 :rem 134
15 IFT<>3078THENPRINT"ERROR IN DATA STATE
   MENTS":STOP                          :rem 88
20 T=0:FORJ=828TO978:READK:T=T+K:POKEJ,K:
   NEXT                                 :rem 145
25 IFT<>17289THENPRINT"ERROR IN DATA STAT
   EMENTS":STOP                         :rem 146
30 POKE249,0                            :rem 141
688 DATA32,188,190,226,172,225,191,251
                                        :rem 148
696 DATA187,255,161,236,162,254,252,96
                                        :rem 158
828 DATA 169,128,133,004,173,005  :rem 46
834 DATA 144,041,002,240,004,169  :rem 31
840 DATA 136,133,004,169,000,162  :rem 32
846 DATA 003,006,250,042,202,208  :rem 28
852 DATA 250,024,101,004,133,004  :rem 19
858 DATA 165,250,133,003,169,000  :rem 40
864 DATA 133,250,169,005,133,002  :rem 36
870 DATA 160,000,177,003,133,005  :rem 26
876 DATA 230,003,177,003,133,006  :rem 34
882 DATA 230,003,198,002,240,028  :rem 36
888 DATA 162,004,169,000,006,006  :rem 39
894 DATA 042,006,006,042,006,005  :rem 30
900 DATA 042,006,005,042,164,250  :rem 24
906 DATA 153,048,002,230,250,202  :rem 28
912 DATA 208,232,240,210,160,000  :rem 19
918 DATA 166,249,240,008,169      :rem 118
923 DATA 029:REM 032 IF USING PRINTER
                                        :rem 63
924 DATA 032,210,255,202,208,250  :rem 30
930 DATA 169,004,133,006,185,048  :rem 45
936 DATA 002,170,189,176,002,133  :rem 43
942 DATA 005,041,064,240,005,169  :rem 36
948 DATA 018,032,210,255,165,005  :rem 41
954 DATA 041,191,032,210,255,169  :rem 44
960 DATA 146,032,210,255,200,198  :rem 40
966 DATA 006,208,221,169,013,032  :rem 41
972 DATA 210,255,192,016,208,196  :rem 52
978 DATA 096                      :rem 97
```

Learning To Count

(Article on page 80.)

Program 1:

Learning To Count—VIC Version

```
28 POKE808,114                          :rem 251
29 POKE36879,250                        :rem 110
30 PRINT"{CLR}"SPC(244)"LEARNING TO COUNT
   "                                    :rem 88
40 Z=7680:POKE36878,15:S1=36876:COL=30720
   :BC=36879                            :rem 66
50 FORI=0TO21:READA,B,C:POKEZ+I,A:POKEZ+I
   +COL,B:POKES1,C:FORT=1TO75:NEXT:POKES1
   ,0                                   :rem 222
52 IFC=236THENGOSUB4000                 :rem 132
53 NEXT                                 :rem 167
54 FORI=0TO21:READA,B,C:POKEZ+484+I,A:POK
   EZ+484+I+COL,B:POKES1,C:FORT=1TO75:NEX
   T                                    :rem 47
55 POKES1,0:IFC=236THENGOSUB4000        :rem 208
56 NEXT                                 :rem 170
57 FORI=0TO22:READA,B,C:POKEZ+22*I,A:POKE
   Z+22*I+COL,B:POKES1,C:FORT=1TO75:NEXT:
   POKES1,0                             :rem 2
58 IFC=236THENGOSUB4000                 :rem 138
59 NEXT                                 :rem 173
```

```
60 FORI=0TO22:READA,B,C:POKEZ+21+22*I,A:P
   OKEZ+21+22*I+COLL,B:POKES1,C:FORT=1TO7
   5:NEXT                          :rem 27
61 POKES1,0:IFC=236THENGOSUB4000 :rem 205
62 NEXT:FORT=1TO2500:NEXT          :rem 109
70 PRINTCHR$(147):POKEBC,30:PRINT"
   {2 DOWN}{2 SPACES}{RVS}LEARNING TO COU
   NT{OFF}"                        :rem 172
71 PRINT"{DOWN} CAN HELP YOU LEARN"
                                   :rem 180
72 PRINT"{DOWN} TO COUNT UP TO 50.":PRINT
   "{3 DOWN} ENTER {RVS}1{OFF}, {RVS}2
   {OFF}, {RVS}3{OFF}, OR {RVS}4{OFF}."
                                   :rem 187
75 PRINT"{2 DOWN}{3 SPACES}UP TO 10---
   {RVS}1{OFF}":PRINT"{DOWN}{3 SPACES}UP
   {SPACE}TO 25---{RVS}2{OFF}":PRINT"
   {DOWN}{3 SPACES}UP TO 35---{RVS}3{OFF}
   "                               :rem 182
76 PRINT"{DOWN}{3 SPACES}UP TO 50---{RVS}
   4{OFF}{2 RIGHT}{3 UP}";         :rem 131
77 INPUTD$                         :rem 102
80 D=VAL(D$)                       :rem 133
81 IFD=1THENDL=10:GOTO100          :rem 11
82 IFD=2THENDL=25:GOTO100          :rem 19
83 IFD=3THENDL=35:GOTO100          :rem 22
84 IFD=4THENDL=50:GOTO100          :rem 21
90 GOTO70                          :rem 9
100 POKE36878,15:S1=36876:COL=30720:R=0:W
    =0:N=0:POKEBC,27:X=DL          :rem 138
205 PRINTCHR$(147)                 :rem 18
206 A=(INT(X*RND(1)))*2            :rem 231
207 IFA/2+1=1THEN206               :rem 91
210 N=N+1:RESTORE:SCR=7834         :rem 91
220 FORH=0TOASTEP2:C=0             :rem 108
225 READL:READM:READK              :rem 69
226 IFK=236THENGOSUB4000           :rem 191
230 POKESCR+H,L:POKESCR+COL+H,M:POKES1,K:
    FORT=1TO75:NEXT:POKES1,0:FORT=1TO350:
    NEXT                           :rem 140
232 IFH=20ANDL=38THENSCR=SCR+22    :rem 159
233 IFH=42THENSCR=SCR+22           :rem 221
234 IFH=64THENSCR=SCR+22           :rem 226
235 IFH=86THENSCR=SCR+22           :rem 231
236 IFH=108THENSCR=SCR+22          :rem 19
238 NEXTH                          :rem 36
239 PRINT"{19 DOWN}ENTER {RVS}0{OFF} TO S
    TART OVER.":POKE198,0          :rem 6
240 PRINT"{HOME}{2 DOWN}{2 SPACES}HOW MAN
    Y?{10 SPACES}";:PRINTSPC(22);:PRINT"
    {2 UP}{11 RIGHT}";:INPUTY$     :rem 196
245 IFY$="0"THENN=N-1:GOTO3000     :rem 244
250 Y=VAL(Y$)                      :rem 222
260 IFY=A/2+1THENGOSUB2000:R=R+1:PRINTCHR
    $(147):GOTO206                 :rem 99
270 C=C+1:IFC=3THENGOTO2500        :rem 99
280 PRINT"{HOME}{3 DOWN}{2 SPACES}{RVS}SO
    RRY! TRY AGAIN.":FORT=1TO1300:NEXT
                                   :rem 211
281 PRINT"{HOME}{3 DOWN}{20 SPACES}":GOTO
    240                            :rem 187
2000 PRINT"{CLR}{DOWN}{2 SPACES}QQQQQ
     {8 SPACES}QQQQQ":PRINT" Q{5 SPACES}Q
     {6 SPACES}Q{5 SPACES}Q"       :rem 176
2001 PRINT"{3 SPACES}£{3 +£}{10 SPACES}
     £{3 +£}":PRINT"{3 SPACES}£{3 +£}
     {10 SPACES}£{3 +£}":PRINT"{3 SPACES}
     £{3 +£}{10 SPACES}£{3 +£}"    :rem 86
2002 PRINT"{4 DOWN}{9 SPACES}£{4 +£}
     {18 SPACES}£{4 +£}{18 SPACES}£{4 +£}"
                                   :rem 161
2003 PRINT" £{+£}{18 SPACES}£{+£}{2 SPACES}
     £{+£}{18 SPACES}£{+£}{3 SPACES}£{+£}
     {16 SPACES}£{+£}{5 SPACES}£{+£}";:rem 91
2004 PRINT"{14 SPACES}£{+£}{7 SPACES}£{+£}
     {DOWN}£{+£}{DOWN}£{+£}{DOWN}£8 +£}{UP}£{+£}
     {UP}£{+£}{UP}£{+£}"           :rem 55
2006 FORH=235TO241:POKES1,H:FORT=1TO175:N
     EXT:NEXTH                     :rem 198
2007 FORH=241TO235STEP-1:POKES1,H:FORT=1T
     O175:NEXT:NEXTH:POKES1,0:RETURN
                                   :rem 196
2500 PRINT"{CLR}{3 DOWN}"SPC(8)"WRONG!"SP
     C(8):PRINT"{3 DOWN} THERE WERE";A/2+
     1;"{LEFT} OBJECTS "           :rem 118
2501 POKE36876,159                 :rem 208
2502 FORT=1TO800:NEXT:POKE36876,0:FORT=1T
     O3500:NEXT:PRINTCHR$(147):W=W+1:GOTO
     206                           :rem 1
3000 PRINT"{CLR}{8 DOWN}{2 SPACES}YOU HAD
     :";N;"TRYS"                   :rem 13
3001 PRINT"{DOWN}"SPC(10);R;"RIGHT":PRINT
     "{DOWN}{2 RIGHT}{8 SPACES}";W;"WRONG
     "                             :rem 188
3005 FORT=1TO4000:NEXT:GOTO50      :rem 41
4000 RESTORE:RETURN                :rem 2
9000 DATA81,0,219,65,2,221,83,3,223,90,4,
     225,88,5,227,90,6,228,102,7,229,42,0
     ,231,35,2                     :rem 169
9001 DATA232,36,3,233,38,4,235,0,5,236
                                   :rem 116
```

Program 2:
Learning To Count—64 Version

```
28 POKE788,52:POKE53281,0:POKE53280,0:S=5
   4272                            :rem 48
30 PRINT"{CLR}{12 DOWN}"TAB(11)"{WHT}LEAR
   NING TO COUNT"                  :rem 226
35 FORL=STOS+24:POKEL,O:NEXT:POKES+5,14:P
   OKES+9,240:POKES+24,15:HF=S+1:LF=S
                                   :rem 217
40 Z=1024:C=0:COL=S:DL(1)=10:DL(2)=25:DL(
   3)=35:DL(4)=50                  :rem 19
50 FORI=0TO39:READA:READB:POKEZ+I,A:POKEZ
   +I+COL,B:GOSUB5000:FORT=1TO75:NEXT
                                   :rem 124
52 IFB=5THENRESTORE                :rem 253
53 NEXT                            :rem 167
54 FORI=0TO39:READA:READB:POKEZ+960+I,A:P
   OKEZ+960+I+COL,B:GOSUB5000      :rem 169
55 FORT=1TO75:NEXT:IFB=5THENRESTORE
                                   :rem 107
56 NEXT                            :rem 170
57 FORI=0TO24:READA:READB:POKEZ+40*I,A:PO
   KEZ+40*I+COL,B:GOSUB5000        :rem 46
58 FORT=1TO75:NEXT:IFB=5THENRESTORE
                                   :rem 110
59 NEXT                            :rem 173
60 FORI=0TO24:READA:READB:POKEZ+39+40*I,A
   :POKEZ+39+40*I+COLL,B:GOSUB5000
                                   :rem 162
61 FORT=1TO75:NEXT:IFB=5THENRESTORE
                                   :rem 104
62 NEXT:FORT=1TO2500:NEXT          :rem 109
70 PRINTCHR$(147):PRINT"{2 DOWN}
   {4 SPACES}{RVS}LEARNING TO COUNT{OFF}
   {SPACE}CAN HELP YOU"            :rem 39
72 PRINT"{DOWN}{4 SPACES}LEARN TO COUNT U
   P TO 50."                       :rem 196
```

```
73 PRINT"{3 DOWN}{4 SPACES}ENTER {RVS}1
   {OFF}, {RVS}2{OFF}, {RVS}3{OFF}, OR
   {RVS}4{OFF}."                    :rem 153
75 PRINT"{4 DOWN}{5 SPACES}UP TO 10---
   {RVS}1{OFF}":PRINT"{DOWN}{5 SPACES}UP
   {SPACE}TO 25---{RVS}2{OFF}"      :rem 174
76 PRINT"{DOWN}{5 SPACES}UP TO 35---{RVS}
   3{OFF}":PRINT"{DOWN}{5 SPACES}UP TO 50
   ---{RVS}4{OFF}{2 RIGHT}{3 UP}";
                                    :rem 173
80 INPUTD$:D=VAL(D$):IFD<1ORD>4THEN70
                                    :rem 30
100 R=0:W=0:N=0:X=DL(D):PRINT"{CLR}"
                                    :rem 213
206 A=(INT(X*RND(1)))*2:IFA/2+1=1THEN206
                                    :rem 227
210 N=N+1:RESTORE:SCR=1304          :rem 77
220 FORH=0TOASTEP2:C=0              :rem 108
225 READL:M=INT(RND(0)*15)+1        :rem 82
226 IFL=5THENRESTORE                :rem 58
230 POKESCR+H,L:POKESCR+COL+H,M:FORT=1TO7
    5:NEXT:GOSUB5000:FORT=1TO350:NEXT
                                    :rem 94
232 IFH=39THENSCR=SCR+80            :rem 230
238 NEXT                            :rem 220
239 PRINT"{19 DOWN}{10 SPACES}ENTER {RVS}
    0{OFF} TO START OVER.":POKE198,0
                                    :rem 6
240 PRINT"{HOME}{2 DOWN}{16 SPACES}";:INP
    UT"{HOME}{2 DOWN}{2 SPACES}HOW MANY";
    Y$                              :rem 245
245 IFY$="0"THENN=N-1:GOTO3000      :rem 244
250 Y=VAL(Y$)                       :rem 222
260 IFY=H/2THENGOSUB2000:R=R+1:PRINTCHR$(
    147):GOTO206                    :rem 14
270 C=C+1:IFC=3THENGOTO2500         :rem 99
280 PRINT"{HOME}{3 DOWN}{RVS}SORRY! TRY A
    GAIN.":FORT=1TO1700:NEXT:GOSUB4000:GO
    TO240                           :rem 94
2000 PRINT"{CLR}{4 DOWN}{WHT}"TAB(6)"
     {4 SPACES}QQQQQ{8 SPACES}QQQQQ "
                                    :rem 59
2001 PRINTTAB(6)"{3 SPACES}Q{5 SPACES}Q
     {6 SPACES}Q{5 SPACES}Q"        :rem 54
2002 PRINTTAB(6)"{BLU}{5 SPACES}[3 +]
     {10 SPACES}[3 +]{2 SPACES}"    :rem 246
2003 PRINTTAB(6)"{5 SPACES}[3 +]
     {10 SPACES}[3 +] {6 DOWN}"     :rem 62
2004 PRINTTAB(6)"{RED}{11 SPACES}[4 +]
     {8 SPACES}"                    :rem 169
2005 PRINTTAB(6)"{CYN}{2 SPACES}[+]
     {7 SPACES}{RED} [4 +]{8 SPACES}{CYN}
     [+]"                           :rem 52
2006 PRINTTAB(6)"{2 SPACES}[+]{20 SPACES}
     [+]"                           :rem 67
2007 PRINTTAB(6)"{3 SPACES}[+]{18 SPACES}
     [+] "                          :rem 68
2008 PRINTTAB(6)"{4 SPACES}[+]{16 SPACES}
     [+]{2 SPACES}"                 :rem 69
2009 PRINTTAB(6)"{5 SPACES}[+]{14 SPACES}
     [+]{3 SPACES}"                 :rem 70
2010 PRINTTAB(6)"{6 SPACES}[+]{12 SPACES}
     [+]{4 SPACES}"                 :rem 62
2011 PRINTTAB(6)"{7 SPACES}[12 +]
     {4 SPACES}{WHT} "              :rem 192
2020 GOSUB5010:RETURN               :rem 36
2500 PRINT"{CLR}{10 DOWN}"TAB(16)"{RVS}WR
     ONG!{OFF}"                     :rem 250
2510 PRINT"{2 DOWN}"TAB(9)"{RVS}THERE WER
     E";H/2;"{LEFT} OBJECTS{OFF}":rem 185
2520 FORT=1TO800:NEXT:FORT=1TO3500:NEXT:P
     RINTCHR$(147):W=W+1:GOTO206    :rem 46
3000 PRINT"{CLR}{10 DOWN}"TAB(10)"YOU HAD
     :";N;"TRYS":PRINT"{DOWN}"TAB(18);R;"
     RIGHT"                         :rem 51
3010 PRINT"{DOWN}{2 RIGHT}"TAB(18);W;"WRO
     NG":FORT=1TO4000:NEXT:RESTORE:GOTO50
                                    :rem 8
4000 PRINT"{HOME}{3 DOWN}{20 SPACES}";:RE
     TURN                           :rem 48
5000 POKES+4,17:POKEHF,INT(RND(0)*50)+80:
     POKELF,250:POKES+4,16:RETURN:rem 166
5010 POKES+4,17:FORM=70TO116STEP2:POKEHF,
     M:POKELF,INT(M/2):FORDL=1TO40:NEXT
                                    :rem 22
5020 NEXT:POKES+4,16:RETURN         :rem 206
9000 DATA81,1,65,2,83,3,90,4,88,5,90,6,10
     2,7,42,1,35,2,36,3,38,4,1,5    :rem 41
```

Disk Tricks

(Article on page 126.)

BEFORE TYPING...

Before typing in programs, please refer to "How To Type COMPUTE!'s Gazette Programs," "A Beginner's Guide To Typing In Programs," and "The Automatic Proofreader" that appear before the Program Listings.

Program 1: Change Disk Name

```
999 REM END: REM PROGRAM 1: CHANGE DISK N
    AME                             :rem 249
1000 INPUT "{CLR}NEW DISK NAME";DN$
                                    :rem 79
1010 IF LEN(DN$)<16 THEN DN$=DN$+CHR$(160
     ):GOTO 1010:REM STRETCH TO 16 CHARS
                                    :rem 177
1020 IF LEN(DN$) > 16 THEN DN$=LEFT$(DN$,
     16): REM SHORTEN NAME TO 16 CHARACTE
     RS                             :rem 52
1030 OPEN 15,8,15,"I": REM OPEN DISK COMM
     AND CHANNEL                    :rem 126
1040 OPEN 8,8,8,"#": REM OPEN DIRECT ACCE
     SS CHANNEL                     :rem 64
1050 PRINT#15, "U1:"8;0;18;0: REM READ TR
     ACK 18, SECTOR 0 INTO CHANNEL 8 BUFF
     ER                             :rem 39
1060 PRINT#15, "B-P:"8;144: REM MOVE BUFF
     ER-POINTER TO FIRST BYTE OF DISK NAM
     E                              :rem 239
1070 PRINT#8, DN$;: REM PUT NEW NAME IN C
     HANNEL 8 BUFFER, REPLACING OLD NAME
                                    :rem 50
1080 PRINT#15,"U2:"8;0;18;0:REM WRITE BUF
     FER WITH NAME CHANGED          :rem 108
1090 CLOSE 8: REM CLOSE DIRECT ACCESS CHA
     NNEL                           :rem 114
1100 CLOSE15: REM CLOSE COMMAND CHANNEL
                                    :rem 42
1110 REM GOTO 100: REM RESTART DISPLAY T&
     S PROGRAM IF APPENDED          :rem 127
```

Program 2: Change Disk ID

```
1999 REM END: REM PROGRAM 2: CHANGE DISK
     {SPACE}ID                      :rem 151
2000 INPUT "{CLR}NEW DISK ID";ID$:rem 183
```

```
2010 IF LEN(ID$) <> 2 THEN 2000: REM REJE
     CT IMPROPER LENGTH ID          :rem 104
2020 OPEN 15,8,15,"I": REM OPEN DISK COMM
     AND CHANNEL                    :rem 126
2030 OPEN 8,8,8,"#": REM OPEN DIRECT ACCE
     SS CHANNEL                     :rem 64
2040 PRINT#15, "U1:"8;0;18;0: REM READ TR
     ACK 18, SECTOR 0 INTO CHANNEL 8 BUFF
     ER                             :rem 39
2050 PRINT#15, "B-P:"8;162: REM MOVE BUFF
     ER-POINTER TO FIRST BYTE OF DISK ID
                                    :rem 91
2060 PRINT#8, ID$;: REM PUT NEW ID IN CHA
     NNEL 8 BUFFER, REPLACING OLD ID
                                    :rem 5
2070 PRINT#15, "U2:"8;0;18;0: REM STORE B
     UFFER TO DISK                  :rem 245
2080 CLOSE 8: REM CLOSE DIRECT ACCESS CHA
     NNEL                           :rem 114
2090 CLOSE15: REM CLOSE COMMAND CHANNEL
                                    :rem 51
2100 REM GOTO 100: REM RESTART DISPLAY T&
     S PROGRAM IF APPENDED          :rem 127
```

Program 3: Unscratch

```
2999 REM END: REM PROGRAM 3, UNSCRATCH FI
     LES                            :rem 75
3000 INPUT "{CLR}WHICH SECTOR";S$: S=VAL(
     S$): IF S<0 OR S>19 THEN 3000
                                    :rem 170
3010 PRINT "{2 DOWN}WHAT IS THE FIRST BYT
     E":PRINT"OF THE FILE YOU WISH"
                                    :rem 253
3011 PRINT"TO UNSCRATCH?"           :rem 35
3020 INPUT BP$: BP=VAL(BP$): REM INPUT FI
     LE TARGET BYTE FOR UNSCRATCH :rem 89
3030 BS=(BP=2)+(BP=34)+(BP=66)+(BP=98)+(P
     P=130)+(BP=162)+(BP=194)+(BP=226)
                                    :rem 160
3035 IFBS<>-1THEN3020:REM REJECT INVALID
     {SPACE}INPUT                   :rem 45
3040 PRINT "{2 DOWN}SELECT FILE TYPE:"
                                    :rem 22
3050 PRINT "{DOWN}{2 SPACES}1. SEQUENTIAL
     "                              :rem 4
3060 PRINT "{2 SPACES}2. PROGRAM" :rem 18
3070 PRINT "{2 SPACES}3. USER"    :rem 59
3080 PRINT "{2 SPACES}4. RELATIVE":rem 90
3090 PRINT "{2 DOWN}WHICH ONE?"   :rem 83
3100 GET A$: IF A$="" THEN 3100   :rem 171
3110 A=VAL(A$): IF A<1 OR A>4 THEN 3100:
     {SPACE}REM REJECT INVALID CHOICE
                                    :rem 39
3120 B=A+128: REM SET INPUT BYTE TO MATCH
      DOS FILE CODES                :rem 158
3130 OPEN 15,8,15,"I": REM OPEN COMMAND C
     HANNEL TO DISK                 :rem 36
3140 OPEN8,8,8,"#": REM OPEN DIRECT ACCES
     S CHANNEL TO DISK              :rem 17
3150 PRINT#15, "U1:"8;0;18;S: REM LOAD SE
     CTOR CONTAINING FILE TO BE UNSCRATCH
     ED                             :rem 90
3160 PRINT#15, "B-P:"8;BP: REM SET BUFFER
      POINTER TO TARGET ADDRESS     :rem 163
3170 PRINT#8, CHR$(B);: REM CHANGE TARGET
      FILE CODE IN CHANNEL 8 BUFFER
                                    :rem 246
3180 PRINT#15, "U2:"8;0;18;S: REM RETURN
     {SPACE}CHANGED CONTENTS TO TARGET SE
     CTOR                           :rem 120
3190 CLOSE 8: REM CLOSE DIRECT ACCESS CHA
     NNEL                           :rem 117
3200 CLOSE 15: REM CLOSE COMMAND CHANNEL
                                    :rem 45
3210 REM GOTO 100: REM RESTART DISPLAY T&
     S PROGRAM WHEN FIRST REM REMOVED
                                    :rem 98
```

Program 4: Scratch

```
3999 REM END: REM PROGRAM 4, SCRATCH OR S
     CRATCH AND LEAVE ON DIRECTORY
                                    :rem 114
4000 PRINT"{CLR}SELECT OPTION:"  :rem 251
4010 PRINT"{DOWN}1. COMPLETE SCRATCH"
                                    :rem 103
4020 PRINT"2. SCRATCH, BUT LEAVE":rem 131
4021 PRINT"{3 SPACES}ON DIRECTORY"
                                    :rem 234
4030 PRINT"{2 DOWN}WHICH ONE?"    :rem 78
4040 GET A$: IF A$="" THEN 4040   :rem 179
4050 A=VAL(A$): IF A<1 OR A>2 THEN 4040:
     {SPACE}REM REJECT INVALID INPUT
                                    :rem 18
4060 IF A=1 THEN B=0: REM SET TO PERMANEN
     TLY DELETE                     :rem 149
4070 IF A=2 THEN B=128: REM SET TO LEAVE
     {SPACE}ON DIRECTORY            :rem 191
4080 INPUT"{2 DOWN}WHICH SECTOR";S$: S=VA
     L(S$): IF S<0 OR S>19 THEN 4080
                                    :rem 75
4090 PRINT "{2 DOWN}WHAT IS THE FIRST BYT
     E":PRINT"OF THE FILE YOU WISH":rem 6
4091 PRINT "TO SCRATCH?"          :rem 137
4100 INPUT BP$: BP=VAL(BP$): REM INPUT FI
     LE TARGET BYTE FOR SCRATCH  :rem 182
4110 BS=(BP=2)+(BP=34)+(BP=66)+(BP=98)+(B
     P=130)+(BP=162)+(BP=194)+(BP=226)
                                    :rem 160
4120 IFBS<>-1THEN4100:REM REJECT INVALID
     {SPACE}INPUT                   :rem 41
4130 OPEN 15,8,15,"I": REM OPEN COMMAND C
     HANNEL TO DISK                 :rem 37
4140 OPEN8,8,8,"#": REM OPEN DIRECT ACCES
     S CHANNEL TO DISK              :rem 18
4150 PRINT#15, "U1:"8;0;18;S: REM LOAD SE
     CTOR CONTAINING FILE TO BE SCRATCHED
                                    :rem 184
4160 PRINT#15, "B-P:"8;BP: REM SET BUFFER
      POINTER TO TARGET ADDRESS     :rem 164
4170 PRINT#8, CHR$(B);: REM CHANGE TARGET
      FILE CODE IN CHANNEL 8 BUFFER
                                    :rem 247
4180 PRINT#15,"U2:"8;0;18;S:REM RETURN CH
     ANGED CONTENTS TO TARGET SECTOR
                                    :rem 121
4190 CLOSE 8: REM CLOSE DIRECT ACCESS CHA
     NNEL                           :rem 118
4200 CLOSE 15: REM CLOSE COMMAND CHANNEL
                                    :rem 46
4210 REM GOTO 100: REM RESTART DISPLAY T&
     S PROGRAM WHEN FIRST REM REMOVED
                                    :rem 99
```

BEFORE TYPING...

Before typing in programs, please refer to "How To Type COMPUTE!'s Gazette Programs," "A Beginner's Guide To Typing In Programs," and "The Automatic Proofreader" that appear before the Program Listings.

The Beginner's Corner

(Article on page 112.)

Program 1:

Household Inventory—64 Version

```
10 REM HOUSEHOLD INVENTORY          :rem 190
20 FOR I=1 TO 9:READ R$(I):NEXT     :rem 187
30 DATA LIVING ROOM,KITCHEN,BEDROOMS
                                    :rem 60
40 DATA BATHROOMS,UTILITY ROOM,FAMILY ROO
   M                                :rem 245
50 DATA DEN,COMPUTER ROOM,STORAGE ROOMS
                                    :rem 255
60 PRINT "{CLR}"                    :rem 202
70 PRINT TAB(12)"** INVENTORY **"   :rem 57
80 PRINT "{2 DOWN}CHOOSE:{DOWN}"    :rem 103
90 FOR I=1 TO 9:PRINT TAB(4)I;R$(I):NEXT
                                    :rem 19
100 PRINT TAB(5)"0 WHOLE HOUSE"     :rem 242
110 GET A$:IF A$<"0" OR A$>"9" THEN 110
                                    :rem 55
120 PRINT "{CLR}"                   :rem 247
130 A=VAL(A$)                       :rem 171
140 TT=0                            :rem 170
150 PRINT TAB(7)"INVENTORY--";R$(A)"
    {DOWN}"                         :rem 134
160 RESTORE:FOR I=1 TO 9:READ D$:NEXT
                                    :rem 166
170 READ ROOM,ITEM$,C               :rem 223
180 IF ROOM=10 THEN 250             :rem 201
190 IF A=0 THEN 210                 :rem 153
200 IF ROOM<>A THEN 170             :rem 224
210 C$=STR$(C)                      :rem 232
220 PRINT ITEM$;TAB(36-LEN(C$));C$
                                    :rem 166
230 TT=TT+C                         :rem 144
240 GOTO 170                        :rem 103
250 T$=STR$(TT)                     :rem 98
260 PRINT "{DOWN}TOTAL";TAB(36-LEN(T$));T
    $                               :rem 82
270 PRINT "{DOWN}DIFFERENT ROOM? (Y/N)";
                                    :rem 240
280 GET A$:IF A$="Y" THEN 60        :rem 124
290 IF A$="N" THEN 520              :rem 36
300 GOTO 280                        :rem 102
310 REM INVENTORY ITEMS             :rem 200
320 DATA 3,BED--BUNK,200            :rem 33
330 DATA 3,BED--DOUBLE,250          :rem 178
340 DATA 3,BED--KING,725            :rem 40
350 DATA 8,COMPUTER,300             :rem 68
360 DATA 7,DESK,130                 :rem 253
370 DATA 6,DINING TABLE,325         :rem 253
380 DATA 5,DRYER,350                :rem 96
390 DATA 1,LOVESEAT,375             :rem 65
400 DATA 2,MICRO OVEN,450           :rem 131
410 DATA 1,PIANO,9800               :rem 128
420 DATA 8,PRINTER,255              :rem 0
430 DATA 2,REFRIGERATOR,425         :rem 98
440 DATA 1,SOFA,425                 :rem 255
450 DATA 7,STEREO,875               :rem 184
460 DATA 2,STOVE,525                :rem 107
470 DATA 8,TELEVISION--13,225       :rem 158
480 DATA 6,TELEVISION--19,475       :rem 170
490 DATA 7,TYPEWRITER,300           :rem 248
500 DATA 5,WASHING MACHINE,560      :rem 221
510 DATA 10,ZZZ,0                   :rem 167
520 END                             :rem 110
```

Program 2:

Household Inventory—VIC Version

```
10 REM HOUSEHOLD INVENTORY          :rem 190
20 FOR I=1 TO 9:READ R$(I):NEXT     :rem 187
30 DATA LIVING ROOM,KITCHEN,BEDROOMS
                                    :rem 60
40 DATA BATHROOMS,UTILITY ROOM,FAMILY ROO
   M                                :rem 245
50 DATA DEN,COMPUTER ROOM,STORAGE ROOMS
                                    :rem 255
60 PRINT "{CLR}"                    :rem 202
70 PRINT "{3 SPACES}** INVENTORY **"
                                    :rem 174
80 PRINT "{2 DOWN}CHOOSE:{DOWN}"    :rem 103
90 FOR I=1 TO 9:PRINT TAB(4)I;R$(I):NEXT
                                    :rem 19
100 PRINT TAB(5)"0 WHOLE HOUSE"     :rem 242
110 GET A$:IF A$<"0" OR A$>"9" THEN 110
                                    :rem 55
120 PRINT "{CLR}"                   :rem 247
130 A=VAL(A$)                       :rem 171
140 TT=0                            :rem 170
150 PRINT R$(A)"{DOWN}"             :rem 128
160 RESTORE:FOR I=1 TO 9:READ D$:NEXT
                                    :rem 166
170 READ ROOM,ITEM$,C               :rem 223
180 IF ROOM=10 THEN 250             :rem 201
190 IF A=0 THEN 210                 :rem 153
200 IF ROOM<>A THEN 170             :rem 224
210 C$=STR$(C)                      :rem 232
220 PRINT ITEM$;TAB(20-LEN(C$));C$
                                    :rem 159
230 TT=TT+C                         :rem 144
240 GOTO 170                        :rem 103
250 T$=STR$(TT)                     :rem 98
260 PRINT "{DOWN}TOTAL";TAB(20-LEN(T$));T
    $                               :rem 75
270 PRINT "{DOWN}DIFFERENT ROOM? (Y/N)";
                                    :rem 240
280 GET A$:IF A$="Y" THEN 60        :rem 124
290 IF A$="N" THEN 520              :rem 36
300 GOTO 280                        :rem 102
310 REM INVENTORY ITEMS             :rem 200
320 DATA 3,BED--BUNK,200            :rem 33
330 DATA 3,BED--DOUBLE,250          :rem 178
340 DATA 3,BED--KING,725            :rem 40
350 DATA 8,COMPUTER,300             :rem 68
360 DATA 7,DESK,130                 :rem 253
370 DATA 6,DINING TABLE,325         :rem 253
380 DATA 5,DRYER,350                :rem 96
390 DATA 1,LOVESEAT,375             :rem 65
400 DATA 2,MICRO OVEN,450           :rem 131
410 DATA 1,PIANO,9800               :rem 128
420 DATA 8,PRINTER,255              :rem 0
430 DATA 2,REFRIGERATOR,425         :rem 98
440 DATA 1,SOFA,425                 :rem 255
450 DATA 7,STEREO,875               :rem 184
460 DATA 2,STOVE,525                :rem 107
470 DATA 8,TELEVISION--13,225       :rem 158
480 DATA 6,TELEVISION--19,475       :rem 170
490 DATA 7,TYPEWRITER,300           :rem 248
500 DATA 5,WASHING MACHINE,560      :rem 221
510 DATA 10,ZZZ,0                   :rem 167
520 END                             :rem 110
```

Program 3:

Computer Inventory—64 Version

```
10 REM COMPUTER INVENTORY           :rem 130
```

```
20 C$(1)="COMMODORE":C$(2)="RADIO SHACK"
                                    :rem 239
30 C$(3)="TEXAS INSTRUMENTS"        :rem 192
40 D$(1)="COMPUTERS":D$(2)="PERIPHERALS":
   D$(3)="SOFTWARE"                 :rem 136
50 PRINT "{CLR}"                    :rem 201
60 PRINT TAB(12)"** INVENTORY **" :rem 56
70 PRINT "{2 DOWN}CHOOSE:{DOWN}" :rem 102
80 FOR I=1 TO 3:PRINT TAB(4)I;C$(I):NEXT
                                    :rem 253
90 PRINT TAB(5)"0 EVERYTHING"       :rem 204
100 GET A$:IF A$<"0" OR A$>"3" THEN 100
                                    :rem 47
110 CC=VAL(A$)                      :rem 238
120 PRINT "{2 DOWN}CHOOSE:{DOWN}":rem 146
130 FOR I=1 TO 3:PRINT TAB(4)I;D$(I):NEXT
                                    :rem 42
140 PRINT TAB(5)"0 EVERYTHING":POKE 198,0
                                    :rem 95
150 GET A$:IF A$<"0" OR A$>"3" THEN 150
                                    :rem 57
160 DD=VAL(A$)                      :rem 245
170 PRINT "{CLR}"                   :rem 252
180 TT=0                            :rem 174
190 PRINT C$(CC),D$(DD);"{DOWN}"    :rem 98
200 RESTORE                         :rem 182
210 READ C,D,ITEM$,SN$,DATE$,CST    :rem 86
220 IF C=10 THEN 310                :rem 199
230 IF CC=0 THEN 250                :rem 221
240 IF CC<>C THEN 210               :rem 42
250 IF DD=0 THEN 270                :rem 227
260 IF DD<>D THEN 210               :rem 47
270 C$=STR$(CST)                    :rem 149
280 PRINT ITEM$;TAB(17);SN$;TAB(27);DATE$
    ;TAB(39-LEN(C$));C$             :rem 195
290 TT=TT+CST                       :rem 61
300 GOTO 210                        :rem 95
310 T$=STR$(TT)                     :rem 95
320 PRINT "{DOWN}TOTAL";TAB(39-LEN(T$));T
    $                               :rem 82
330 PRINT "{DOWN}DIFFERENT CATEGORY? (Y/N
    )";                             :rem 14
340 GET A$:IF A$="Y" THEN 50        :rem 120
350 IF A$="N" THEN 550              :rem 36
360 GOTO 340                        :rem 105
370 REM INVENTORY ITEMS             :rem 206
380 DATA 1,1,VIC-20,V029972,1982,225
                                    :rem 100
390 DATA 1,2,DATASETTE,282754,1982,70
                                    :rem 11
400 DATA 1,3,VICMON,,1982,60        :rem 244
410 DATA 1,1,COMMODORE 64,P00144607,1983,
    345                             :rem 83
420 DATA 1,2,1541 DISK DRIVE,K0177958,198
    4,250                           :rem 140
430 DATA 1,2,1525 PRINTER,5016223,1984,25
    0                               :rem 177
440 DATA 1,3,RADAR RAT RACE,,1983,30
                                    :rem 150
450 DATA 2,1,16K COLOR,0024023,1982,620
                                    :rem 234
460 DATA 2,2,CTR-80A RECORDER,,1982,56
                                    :rem 72
470 DATA 2,3,COLOR LOGO,,1983,55 :rem 229
480 DATA 2,3,VIDEOTEX,,1983,40   :rem 152
490 DATA 3,1,TI-9/4,61205,1980,635
                                    :rem 203
500 DATA 3,1,TI-99/4A,40102545,1982,425
                                    :rem 211
510 DATA 3,2,TI RECORDER,025426,1983,60
                                    :rem 82
520 DATA 3,3,ATTACK,5159992,1983,35
                                    :rem 96
530 DATA 3,3,MUNCHMAN,9976273,1982,35
                                    :rem 2
540 DATA 10,0,,,,0                  :rem 80
550 END                             :rem 113
```

Program 4:
Computer Inventory—VIC Version

```
10 REM COMPUTER INVENTORY           :rem 130
20 C$(1)="COMMODORE":C$(2)="RADIO SHACK"
                                    :rem 239
30 C$(3)="TEXAS INSTRUMENTS"        :rem 192
40 D$(1)="COMPUTERS":D$(2)="PERIPHERALS":
   D$(3)="SOFTWARE"                 :rem 136
50 PRINT "{CLR}"                    :rem 201
60 PRINT "{4 SPACES}** INVENTORY **"
                                    :rem 173
70 PRINT "{2 DOWN}CHOOSE:{DOWN}" :rem 102
80 FOR I=1 TO 3:PRINT I;C$(I):NEXT
                                    :rem 161
90 PRINT " 0 EVERYTHING"            :rem 111
100 GET A$:IF A$<"0" OR A$>"3" THEN 100
                                    :rem 47
110 CC=VAL(A$)                      :rem 238
120 PRINT "{2 DOWN}CHOOSE:{DOWN}":rem 146
130 FOR I=1 TO 3:PRINT I;D$(I):NEXT
                                    :rem 206
140 PRINT " 0 EVERYTHING":POKE 198,0
                                    :rem 2
150 GET A$:IF A$<"0" OR A$>"3" THEN 150
                                    :rem 57
160 DD=VAL(A$)                      :rem 245
170 PRINT "{CLR}"                   :rem 252
180 TT=0                            :rem 174
190 PRINT C$(CC),D$(DD);"{DOWN}"    :rem 98
200 RESTORE                         :rem 182
210 READ C,D,ITEM$,SN$,DATE$,CST    :rem 86
220 IF C=10 THEN 310                :rem 199
230 IF CC=0 THEN 250                :rem 221
240 IF CC<>C THEN 210               :rem 42
250 IF DD=0 THEN 270                :rem 227
260 IF DD<>D THEN 210               :rem 47
270 C$=STR$(CST)                    :rem 149
280 PRINT ITEM$,SN$,DATE$;TAB(20-LEN(C$))
    ;C$"{DOWN}"                     :rem 89
290 TT=TT+CST                       :rem 61
300 GOTO 210                        :rem 95
310 T$=STR$(TT)                     :rem 95
320 PRINT "{DOWN}TOTAL";TAB(20-LEN(T$));T
    $                               :rem 72
330 PRINT "{DOWN}DIFFERENT CATEGORY?
    {3 SPACES}(Y/N)";               :rem 14
340 GET A$:IF A$="Y" THEN 50        :rem 120
350 IF A$="N" THEN 550              :rem 36
360 GOTO 340                        :rem 105
370 REM INVENTORY ITEMS             :rem 206
380 DATA 1,1,VIC-20,V029972,1982,225
                                    :rem 100
390 DATA 1,2,DATASETTE,282754,1982,70
                                    :rem 11
400 DATA 1,3,VICMON,,1982,60        :rem 244
410 DATA 1,1,C-64,P00144607,1983,345
                                    :rem 30
420 DATA 1,2,1541 DRIVE,K0177958,1984,250
                                    :rem 97
430 DATA 1,2,1525 PRINTER,5016223,1984,25
    0                               :rem 177
440 DATA 1,3,RADAR RAT,,1983,30     :rem 123
```

```
450 DATA 2,1,16K COLOR,0024023,1982,620
                                    :rem 234
460 DATA 2,2,RECORDER,,1982,56      :rem 137
470 DATA 2,3,COLOR LOGO,,1983,55    :rem 229
480 DATA 2,3,VIDEOTEX,,1983,40      :rem 152
490 DATA 3,1,TI-9/4,61205,1980,635
                                    :rem 203
500 DATA 3,1,TI-99/4A,40102545,1982,425
                                    :rem 211
510 DATA 3,2,TI RECORDER,025426,1983,60
                                    :rem 82
520 DATA 3,3,ATTACK,5159992,1983,35
                                    :rem 96
530 DATA 3,3,MUNCHMAN,9976273,1982,35
                                    :rem 2
540 DATA 10,0,,,,0                  :rem 80
550 END                             :rem 113
```

SpeedScript Customizer

(Article on page 54.)

BEFORE TYPING...

Before typing in programs, please refer to "How To Type COMPUTE!'s Gazette Programs," "A Beginner's Guide To Typing In Programs," and "The Automatic Proofreader" that appear before the Program Listings.

Program 1: The Customizer Boot

```
2 DN=8:PRINT"{CLR} BOOT: {RVS}D{OFF}ISK O
  R {RVS}T{OFF}APE"                 :rem 131
4 GETZ$:IFZ$="T"THENDN=1:GOTO8      :rem 136
6 IFZ$<>"D"THEN4                    :rem 168
8 SYS65517:IFPEEK(781)=22THENPRINT"{CLR}P
  OKE44,39:POKE256*39,0:NEW":GOTO12
                                    :rem 115
10 IFPEEK(781)=40THENPRINT"{CLR}POKE44,48
   :POKE256*48,0:NEW                :rem 99
12 PRINT"{2 DOWN}LOAD"CHR$(34)"CUST.SS"CH
   R$(34)","DN                      :rem 65
14 FORI=631TO635:READN:POKEI,N:NEXT:POKE1
   98,5                             :rem 98
16 DATA19,13,13,33,131              :rem 94
```

Program 2: The Customizer

```
1 IFFL=1THEN52                      :rem 86
10 GOTO16                           :rem 1
12 PRINT"{CLR}"X$"{RVS}SPEEDSCRIPT CUSTOM
   IZER{OFF}"                       :rem 6
14 RETURN                           :rem 69
16 LC=0:BC=1:C$="VIC":P0=129:P1=4674:P2=1
   32:P3=145:SYS65517               :rem 216
18 IFPEEK(781)=40THENC$="C64":X=1:P1=2062
   :P2=103:P3=106                   :rem 211
20 DIMV(20):PRINTCHR$(14):PRINTCHR$(8):IF
   XTHENX$="{9 SPACES}"             :rem 97
22 GOSUB12                          :rem 71
24 PRINTX$"{2 DOWN}CUSTOMIZER WILL NOT"
                                    :rem 147
26 PRINTX$"DESTROY THE SOURCE "     :rem 145
28 PRINTX$"COPY OF SPEEDSCRIPT."    :rem 251
30 PRINTX$"{DOWN}INSERT SOURCE DISK
   {2 SPACES}"                      :rem 18
32 PRINTX$"OR TAPE VERSION OF       :rem 22
34 PRINTX$"SPEEDSCRIPT 1.0 OR"      :rem 42
36 PRINTX$"2.0 FOR {RVS}"C$"{OFF}.
                                    :rem 136
38 PRINTX$"{DOWN}ENTER SOURCE FILENAME."
                                    :rem 7
40 PRINTX$;:INPUT"NAME";NF$         :rem 74
42 PRINTX$"{DOWN}{RVS}D{OFF}ISK OR {RVS}T
   {OFF}APE?"                       :rem 65
44 GETZ$:IFZ$<>"D"ANDZ$<>"T"THEN44 :rem 9
46 IFZ$="T"THEND$="{RVS}TAPE{OFF}":DN=1
                                    :rem 108
48 IFZ$="D"THEND$="{RVS}DISK{OFF}":DN=8
                                    :rem 102
50 FL=1:LOADNF$,DN,1                :rem 146
52 IFXTHENLC=11:BC=12:POKE53280,BC:POKE53
   281,BC                           :rem 67
54 POKE646,LC:IFX=0THENPOKE36879,25
                                    :rem 131
56 PE=PEEK(P1):IFPE=P2ORPE=P0THENV$="V1":
   V=1:IFP1=2062THENPOKE5755,133    :rem 59
58 IFPE=P3THENV$="V2":V=2           :rem 2
60 IFPE<>P0ANDPE<>P2ANDPE<>P3THENPRINT"RE
   AD ERROR.":FORI=1TO2000:NEXT:RUN
                                    :rem 134
62 IFP1=4674THENV=3                 :rem 127
64 PRINTX$"{DOWN}{RVS}"C$" SPEEDSCRIPT
   {RVS}"V$:FORI=1TO2000:NEXTI:PRINT"
   {CLR}"                           :rem 244
66 PRINT"{HOME}"X$"{RVS}SPEEDSCRIPT CUSTO
   MIZER{OFF}"                      :rem 143
68 PRINTX$"{DOWN}{RVS}F1{OFF} CHANGES BAC
   KGROUND"                         :rem 192
70 PRINTX$"{DOWN}{RVS}F3{OFF} CHANGES LET
   TERS"                            :rem 254
72 PRINTX$"{DOWN}{RVS}RETURN{OFF} SETS TH
   E COLORS                         :rem 27
74 PRINTX$"AS DEFAULT COLORS.       :rem 47
76 GETZ$:IFZ$=CHR$(133)THENBC=BC+1AND15:I
   FX=1THENPOKE53281,BC:POKE53280,BC
                                    :rem 0
78 IFZ$=CHR$(133)ANDX=0THENBP=(BCAND15)*1
   6+(BCAND7)+8:POKE36879,BP        :rem 126
80 IFZ$=CHR$(133)THENFORI=0TO200:NEXT:GOT
   O76                              :rem 245
82 IFZ$=CHR$(134)THENLC=LC+1AND15:IFX=0TH
   ENLC=LCAND7                      :rem 223
84 IFZ$=CHR$(134)THENPOKE646,LC:GOTO66
                                    :rem 56
86 IFZ$<>CHR$(13)THEN76             :rem 71
88 GOSUB12                          :rem 83
90 PRINTX$"ORIGINAL DEFAULT{3 SPACES}"
                                    :rem 144
92 PRINTX$"SETTINGS ARE LISTED "    :rem 198
94 PRINTX$"BELOW:"                  :rem 109
96 PRINTX$;:INPUT"LEFT MARGIN{7 SPACES}5
   {3 LEFT}";V(0)                   :rem 200
98 PRINTX$;:INPUT"RIGHT MARGIN{6 SPACES}7
   5{4 LEFT}";V(1)                  :rem 242
100 PRINTX$;:INPUT"PAGE LENGTH{7 SPACES}6
    6{4 LEFT}";V(2)                 :rem 182
102 PRINTX$;:INPUT"TOP MARGIN{8 SPACES}5
    {3 LEFT}";V(3)                  :rem 183
104 PRINTX$;:INPUT"BOTTOM MARGIN
    {5 SPACES}58{4 LEFT}";V(4)      :rem 113
106 PRINTX$;:INPUT"SPACING{11 SPACES}2
    {3 LEFT}";V(5)                  :rem 14
108 PRINTX$;:INPUT"FANFOLD(N=0/Y=1)
    {2 SPACES}1{3 LEFT}";V(6)       :rem 7
110 FORI=1TO9:READJ$:K=I+6          :rem 60
112 PRINTX$"[CTRL] £ "I" ={4 SPACES}"J$;
    :INPUT"{4 LEFT}";V(K)           :rem 76
114 NEXTI                           :rem 30
```

```
116 DATA27,14,15,18,00,00,00,00,00
                                  :rem 143
118 PRINTX$"{RVS}C{OFF}ONTINUE OR {RVS}R
    {OFF}ERUN."                   :rem 239
120 GETZ$:IFZ$="R"THENRUN          :rem 47
122 IFZ$<>"C"THEN120              :rem 101
124 GOSUB12                       :rem 122
126 PRINTX$;:INPUT"{DOWN}NEW FILENAME";NF
    $                             :rem 154
128 IFV=1THENBL=2408:LL=2417:DT=5200
                                  :rem 105
130 IFV=2THENBL=2411:LL=2425:DT=5275
                                  :rem 104
132 IFV=3THENBL=4979:LL=5031:DT=7750
                                  :rem 124
134 POKEBL,BC:POKELL,LC:FORI=0TO15:POKEDT
    +I,V(I):NEXTI                 :rem 245
136 IFDN=1THENPRINTX$"{DOWN}{RVS}PRESS ST
    OP ON TAPE{OFF}"              :rem 116
138 PRINTX$"{DOWN}INSERT DESTINATION
    {2 SPACES}"                   :rem 145
140 PRINTX$D$" TO HOLD"            :rem 20
142 PRINTX$"MODIFIED SPEEDSCRIPT":rem 107
144 PRINTX$"AND PRESS {RVS}RETURN{OFF}."
                                  :rem 248
146 GETZ$:IFZ$<>CHR$(13)ANDZ$<>CHR$(141)T
    HEN146                        :rem 237
148 IFX=0ANDV$="V2"THENV=4         :rem 61
150 ONVGOSUB152,154,156,158:GOTO160:rem 3
152 HS=8:LE=162:HE=27:RETURN      :rem 208
154 HS=8:LE=0:HE=40:RETURN        :rem 100
156 HS=18:LE=108:HE=37:RETURN       :rem 6
158 HS=18:LE=8:HE=38:RETURN       :rem 168
160 PRINT"{CLR}PO43,1:PO44,"HS"   :rem 136
162 PRINT"{2 DOWN}PO45,"LE":PO46,"HE"
                                  :rem 179
164 PRINT"{2 DOWN}SAVE"CHR$(34)NF$CHR$(34
    )","DN                        :rem 233
166 DATA19,13,13,13,33,131         :rem 36
168 POKE198,6:FORI=631TO636:READN:POKEI,N
    :NEXT                         :rem 158
```

Mystery At Marple Manor

(Article on page 104.)

BEFORE TYPING...

Before typing in programs, please refer to "How To Type COMPUTE!'s Gazette Programs," "A Beginner's Guide To Typing In Programs," and "The Automatic Proofreader" that appear before the Program Listings.

Program 1: 64 Version

```
9 POKE53280,1:POKE53281,0:S=54272:FORJ=0T
  O24:POKES+J,0:NEXT:POKES+24,15   :rem 35
12 PRINT"{CLR}{6 DOWN}"TAB(7)"[8]{RVS}[*]
   {4 RIGHT}£"                    :rem 121
13 PRINTTAB(7)"{RVS} [*]{2 RIGHT}£ ":PRI
   NTTAB(7)"{RVS}{2 SPACES}[*]£
   {2 SPACES}{OFF} YSTERY"        :rem 238
15 PRINTTAB(7)"{RVS} B{2 SPACES}B ":PRINT
   TAB(7)"{RVS} B{2 SPACES}B ":PRINT"
   {3 UP}"TAB(21)CHR$(142);       :rem 153
24 GOSUB1713:PRINT"AT"            :rem 82
27 PRINT"{DOWN}"TAB(12)"[5]{RVS}[*]
   {4 RIGHT}£":PRINTTAB(12)"{RVS} [*]
   {2 RIGHT}£ "                   :rem 42
28 PRINTTAB(12)"{RVS}{2 SPACES}[*]£
   {2 SPACES}{OFF} ARPLE"         :rem 102
30 PRINTTAB(12)"{RVS} B{2 SPACES}B ":PRIN
   TTAB(12)"{RVS} B{2 SPACES}B " :rem 129
33 PRINT"{UP}"TAB(17)"[4]{RVS}[*]
   {4 RIGHT}£":PRINTTAB(17)"{RVS} [*]
   {2 RIGHT}£ ":PRINTTAB(17)"{RVS}
   {2 SPACES}[*]£{2 SPACES}{OFF} ANOR"
                                  :rem 167
36 PRINTTAB(17)"{RVS} B{2 SPACES}B ":PRIN
   TTAB(17)"{RVS} B{2 SPACES}B " :rem 145
39 GOSUB1713                      :rem 184
42 FORJ=1TO1000:NEXT              :rem 226
45 POKES+5,15:POKES+6,0:POKES+4,129
                                   :rem 57
50 J=1:FORI=1TO15:POKE53281,J:POKE53280,1
   -J                             :rem 31
51 POKES+1,INT(RND(1)*60)+5        :rem 2
53 J=1-J:FORP=1TO30:NEXT:NEXT     :rem 110
56 POKES+4,0                      :rem 168
100 DEFFNR(X)=INT(RND(1)*X)+1:J=RND(-TI)
                                  :rem 108
103 DIMP%(50),S$(22),R$(14),C$(6),V$(3),V
    (3),D%(10,2)                   :rem 56
112 FORJ=1TO10:P%(J)=FNR(11)+3:NEXT
                                   :rem 46
115 FORJ=11TO22:P%(J)=FNR(13)+1:NEXT
                                  :rem 101
118 FORJ=24TO31:P%(J)=4:NEXT      :rem 169
121 P%(23)=FNR(8)+6               :rem 204
124 J=FNR(10):P%(35)=J:P%(34)=P%(J):P%(J)
    =0                             :rem 16
127 J=FNR(10):IFP%(J)=0THEN127    :rem 200
130 P%(32)=J:P%(J)=0:J=FNR(12):P%(33)=J:P
    %(J+10)=0                     :rem 136
133 FORJ=1TO22:IFRND(1)<=.75THENP%(J)=-P%
    (J)                            :rem 56
136 READS$(J):NEXT                 :rem 65
139 FORJ=1TO14:READR$(J):NEXT      :rem 36
142 FORJ=0TO10:READD%(J,1),D%(J,2):IFRND(
    1)<.9THEND%(J,0)=-1           :rem 122
143 NEXT                          :rem 215
145 FORJ=0TO3:READV$(J):NEXT      :rem 242
148 P=2049:I=0:FORJ=4000TO7000STEP1000
                                  :rem 188
151 IFJ=PEEK(P+2)+PEEK(P+3)*256THENDA(I)=
    P:I=I+1:GOTO157                :rem 54
154 P=PEEK(P)+PEEK(P+1)*256:GOTO151
                                   :rem 11
157 NEXT                          :rem 220
172 PRINT"{HOME}{21 DOWN}{BLK}{6 SPACES}H
    OW MANY PLAYERS (1-6) ?"      :rem 218
175 GETA$:IFA$<"1"ORA$>"6"THEN175 :rem 75
178 I=VAL(A$):P%(49)=I            :rem 178
181 FORJ=1TOI:P%(35+J)=1:NEXT     :rem 233
190 PRINT"{CLR}{2 DOWN}[4]ALL PLAYERS EXC
    EPT PLAYER #1 MUST LEAVE"CHR$(14)
                                   :rem 7
192 PRINT"THE ROOM AT THIS POINT.":PRINT"
    {DOWN}{3 SPACES}PLAYER # 1: PRESS
    {RVS} RETURN {OFF}"           :rem 152
193 PRINT"{7 SPACES}TO BEGIN THE GAME!"
                                  :rem 146
194 GETA$:IFA$<>CHR$(13)THEN194    :rem 14
196 POKE53280,12:POKE53281,15:Q=1 :rem 87
200 PRINT"{CLR}{2 DOWN}{BLK}PLAYER #"Q"--
    -------------[4]{DOWN}"       :rem 120
203 IFC$(Q)<>""THEN212            :rem 175
```

```
206 PRINT"PRESS ANY TWO KEYS TO ESTABLISH
     YOUR"                          :rem 37
207 PRINT"SECRET CODE. WITH THIS CODE, NO
     OTHER"                         :rem 211
209 PRINT"PLAYER CAN STEAL YOUR TURN!":PR
    INT"{DOWN}ENTER YOUR CODE NOW!"
                                    :rem 214
210 GOSUB1700:C$(Q)=A$:GOTO218      :rem 206
212 PRINT"{DOWN}ENTER YOUR SECRET CODE!":
    GOSUB1700                       :rem 72
215 IFC$(Q)<>A$THENI=0:GOSUB1710:GOTO200
                                    :rem 124
218 PRINT"{CLR}{2 DOWN}{BLK}PLAYER #"Q"--
    -------------{4}{DOWN}"         :rem 129
221 R=P%(35+Q):PRINT"YOU ARE IN THE "R$(R
    )"."                            :rem 49
224 PRINT"DO YOU WISH TO LEAVE THIS ROOM
    {SPACE}[Y/N] ?"                 :rem 6
227 GETA$:IFA$="N"THENPRINT"NO":GOTO330
                                    :rem 3
230 IFA$<>"Y"THEN227                :rem 106
233 I=1:J=R:GOSUB1730:FORJ=0TO3:READV(J):
    NEXT                            :rem 85
236 PRINT"YES":PRINT"{DOWN}DOORS FROM THI
    S ROOM ARE FOUND TO THE:"       :rem 187
239 FORJ=0TO3:IFV(J)<>0THENPRINTTAB(4);V$
    (J)                             :rem 222
242 NEXT:PRINT"{DOWN}TYPE {RVS}{BLK} N
    {OFF} , {RVS} S {OFF} , {RVS} E {OFF}
     ,{4}OR {RVS}{BLK} W {OFF}{4} TO MOVE
    !":I=3                          :rem 227
245 GETA$:IFA$=""THEN245            :rem 89
248 A=ASC(A$)OR128:I=0:IFA<197ORA>215THEN
    245                             :rem 62
251 IFA=ASC(V$(I))THEN260           :rem 168
254 I=I+1:IFI<4THEN251              :rem 15
257 GOTO245                         :rem 114
260 PRINT"GO "V$(I)                 :rem 147
261 IF V(I)<1THENPRINT"NO DOOR THIS WAY.
    {SPACE}YOU CAN'T MOVE.":GOTO1910
                                    :rem 154
263 IFV(I)<100THENR=V(I):PRINT"MOVING TO
    {SPACE}NEW ROOM.":FORI=1TO1000:NEXT:G
    OTO330                          :rem 166
266 Z=V(I)-100:IFD%(Z,0)=0THEN300 :rem 75
269 PRINT"THAT DOOR IS LOCKED":GOSUB1760
                                    :rem 45
270 IFA=0THENPRINT"YOU DON'T HAVE A MATCH
    ING KEY.":PRINT"NO MOVE.":GOTO1910
                                    :rem 65
272 PRINT"YOUR KEY OPENS THE DOOR.":GOSUB
    1770:PRINT"MOVING TO NEW ROOM."
                                    :rem 200
300 I=D%(Z,1):IFI=RTHENI=D%(Z,2)    :rem 82
303 R=I:GOSUB1760:IFA<>1THEN330     :rem 112
306 PRINT"DO YOU WANT TO LOCK THIS DOOR B
    EHIND{4 SPACES}YOU{2 SPACES}[Y / N] ?
    "                               :rem 96
309 GETA$:IFA$="N"THENPRINT"NO":GOTO330
                                    :rem 4
312 IFA$<>"Y"THEN309                :rem 108
315 PRINT"YES":GOSUB1770:PRINT"DOOR LOCKE
    D."                             :rem 3
330 P%(Q+35)=R:PRINT"{DOWN}{CLR}{5 DOWN}Y
    OU ARE IN THE "R$(R)"."         :rem 43
333 PRINT"YOU CARRY ";:I=P%(Q+41):GOSUB17
    80:PRINT"."                     :rem 205
336 J=0:PRINT"YOU SEE THE FOLLOWING HERE:
    "                               :rem 168
339 FORI=1TO31:IFP%(I)=RTHENJ=J+1:PRINT"
    {3 SPACES}";:GOSUB1780:PRINT"."
                                    :rem 16
342 NEXT:FORI=1TO6:IFI<>QANDP%(35+I)=RTHE
    NPRINT"{3 SPACES}PLAYER #"I".":J=J+1
                                    :rem 252
345 NEXT:IFP%(34)=RTHENPRINT"{3 SPACES}TH
    E BODY OF THE "S$(P%(35))".":J=J+1
                                    :rem 180
348 IFJ=0THENPRINT"NOTHING OF INTEREST."
                                    :rem 173
351 PRINT"{DOWN}PRESS {RVS}{BLK} RETURN
    {SHIFT-SPACE}{OFF}{4} FOR OPTIONS....
    "                               :rem 158
354 GETA$:IFA$<>CHR$(13)THEN354     :rem 10
375 PRINT"{CLR}"                    :rem 3
376 PRINT"{4 DOWN}{BLK}{3 SPACES}>>>>> TU
    RN{SHIFT-SPACE}OPTIONS <<<<<{2 DOWN}"
    :PRINT" {4}{RVS}A{OFF} ACCUSE THE MUR
    DERER!"                         :rem 129
377 PRINT"{SHIFT-SPACE}{RVS}D{OFF} DROP A
    N ITEM.":PRINT" {RVS}H{OFF} HIDE AN I
    TEM OR SUSPECT."                :rem 224
379 PRINT" {RVS}N{OFF} NO ACTION.":PRINT"
     {RVS}P{OFF} PILFER FROM ANOTHER PLAY
    ER."                            :rem 240
381 PRINT" {RVS}S{OFF} SEARCH THE ROOM FO
    R HIDDEN ITEMS.":PRINT" {RVS}T{OFF} T
    AKE AN ITEM."                   :rem 143
384 PRINT"{2 DOWN}ENTER LETTER FOR ACTION
     DESIRED!{3 DOWN}"              :rem 89
387 GETA$:IFA$<"A"ORA$>"T"THEN387:rem 131
390 PRINT"{CLR}":A=ASC(A$):ONA-64GOTO700,
    387,387,800                     :rem 36
393 IFA$="H"THEN970                 :rem 43
396 IFA$<"N"THEN387                 :rem 53
400 ONA-77GOTO450,387,880,398,387,930,820
                                    :rem 165
450 PRINT"{2 DOWN}PRESS {RVS}{BLK} RETURN
     {OFF}{4} TO END YOUR TURN!" :rem 119
453 GETA$:IFA$<>CHR$(13)THEN453     :rem 10
456 I=0:PRINT"{BLK}{CLR}{4 DOWN}PLAYER #"
    Q"========== END TURN":GOSUB1710
                                    :rem 142
459 Q=Q+1:IFQ>P%(49)THENQ=1         :rem 86
462 IFP%(Q+35)=0THEN459             :rem 19
465 GOTO200                         :rem 106
700 PRINT"{CLR}{DOWN}{BLK}{3 SPACES}*****
     MAKE AN ACCUSATION *****{DOWN}{4}":I
    =1                              :rem 112
703 FORJ=1TO10:PRINTJ"{LEFT}:"TAB(5)"THE
    {SPACE}";S$(J)".":NEXT          :rem 163
706 PRINT"{3 DOWN}ENTER NUMBER OF MURDER
    {SPACE}VICTIM ";:INPUTJ         :rem 231
709 IFJ<>P%(35)THENI=0              :rem 6
712 GOSUB1900                       :rem 228
715 FORJ=1TO10:PRINTJ"{LEFT}:"TAB(5)"THE
    {SPACE}";S$(J)".":NEXT          :rem 166
718 PRINT"{3 DOWN}ENTER NUMBER OF MURDERE
    R ";:INPUTJ                     :rem 53
721 IFJ<>P%(32)THENI=0              :rem 253
724 GOSUB1900                       :rem 231
727 FORJ=1TO12:PRINTJ"{LEFT}:"TAB(5)"THE
    {SPACE}"S$(J+10)".":NEXT        :rem 252
730 PRINT"{3 DOWN}ENTER NUMBER OF MURDER
    {SPACE}WEAPON ";:INPUTJ         :rem 226
733 IFJ<>P%(33)THENI=0              :rem 1
736 GOSUB1900                       :rem 234
739 FORJ=1TO14:PRINTJ"{LEFT}:"TAB(5)"THE
    {SPACE}"R$(J)".":NEXT           :rem 116
742 PRINT"{3 DOWN}ENTER NUMBER OF MURDER
    {SPACE}ROOM ";:INPUTJ           :rem 88
745 IFJ<>ABS(P%(34))THENI=0         :rem 44
746 PRINT"{CLR}{5 DOWN}SUMMONING THE POLI
```

```
CE TO MAKE AN":PRINT"ARREST....."
                                    :rem 244
748 POKES+14,5:POKES+18,16:POKES+3,1:POKE
    S+24,143:POKES+6,240:POKES+4,65:A=538
    9                               :rem 163
751 FORJ=1TO200:R=A+PEEK(S+27)*3.5:POKES,
    RAND255:POKES+1,INT(R/256):NEXT
                                    :rem 131
754 FORJ=0TO24:POKES+J,0:NEXT:POKES+24,15
                                    :rem 44
757 FORJ=1TO2500:NEXT               :rem 37
760 IFI=0THEN772                    :rem 177
763 I=3:PRINT"YOUR SOLUTION IS CORRECT!":
    GOSUB1710                       :rem 2
769 PRINT"{2 DOWN}PLAYER #"Q"HAS CRACKED
    {SPACE}THE CASE!":GOTO787       :rem 158
772 I=2:PRINT"NO!...THAT WAS A FALSE ARRE
    ST!":GOSUB1710                  :rem 232
775 GOSUB1800:P%(35+Q)=0:P%(50)=P%(50)+1:
    PRINT"YOU'RE OUT OF THE GAME!":rem 85
778 IFP%(50)<P%(49)THEN450          :rem 151
781 RESTORE:GOSUB1713:FORJ=1TO500:NEXT:GO
    SUB1713                         :rem 90
784 PRINT"{DOWN}ALL PLAYERS HAVE GIVEN IN
    CORRECT":PRINT"SOLUTIONS TO THE CRIME
    !!"                             :rem 85
785 PRINT"{DOWN}NOBODY WINS !"      :rem 51
787 PRINT"HERE IS THE CORRECT SOLUTION:":
    PRINT"THE "S$(P%(32))           :rem 192
789 PRINT"KILLED THE "S$(P%(35)):PRINT"IN
     THE "R$(ABS(P%(34)))","        :rem 19
791 PRINT"USING THE "S$(P%(33)+10)".
    {2 DOWN}":END                   :rem 254
800 PRINT"{2 DOWN}{BLK}{3 SPACES}*** DROP
     AN ITEM ***{4}":GOSUB1800      :rem 36
803 IFI=0THENPRINT"{DOWN}YOU WEREN'T CARR
    YING ANYTHING !":GOTO450        :rem 88
806 PRINT"{DOWN}YOU DROP ";:GOSUB1780:PRI
    NT".":GOTO450                   :rem 60
820 PRINT"{2 DOWN}{BLK}{3 SPACES}*** TAKE
     AN ITEM ***{4}":J=1:PRINT"{DOWN}THES
    E ITEMS ARE AVAILABLE:"         :rem 175
823 FORI=11TO31:IFP%(I)<>RTHEN829:rem 233
826 PRINTJ": ";:GOSUB1780:PRINT".":POKE90
    0+J,I:J=J+1                     :rem 70
829 NEXT:IFJ=1THENPRINT"NO ITEMS.":GOTO45
    0                               :rem 60
832 PRINT"{DOWN}ENTER NUMBER TO TAKE AN I
    TEM, OR":PRINT"ENTER ZERO TO TAKE NOT
    HING."                          :rem 111
835 INPUT"WHAT ITEM DO YOU WANT";A:IFA<0O
    RA>=JTHEN835                    :rem 137
838 IFA=0THENPRINT"{DOWN}NO ITEM TAKEN.":
    GOTO450                         :rem 234
841 GOSUB1800:IFI<>0THENPRINT"YOU DROP ";
    :GOSUB1780:PRINT"."             :rem 82
844 I=PEEK(900+A):P%(I)=100+Q:P%(Q+41)=I
                                    :rem 155
845 PRINT"YOU TAKE ";:GOSUB1780:PRINT".":
    GOTO450                         :rem 30
880 PRINT"{2 DOWN}{BLK}{3 SPACES}*** PILF
    ER FROM ANOTHER ***{4}":J=0     :rem 46
881 PRINT"{DOWN}THESE PLAYERS ARE ALSO IN
     THE ROOM..."                   :rem 226
883 FORI=1TO6:IFP%(35+I)=RANDI<>QTHENPRIN
    T"{3 SPACES}PLAYER #"I".":J=J+1
                                    :rem 141
886 NEXT:IFJ=0THENPRINT"NO OTHER PLAYERS
    {SPACE}ARE IN THE ROOM!":GOTO450
                                    :rem 222
889 PRINT"{DOWN}WHICH PLAYER WILL YOU STE
    AL FROM ?"                      :rem 108
890 PRINT"ENTER NUMBER, OR PRESS ZERO."
                                    :rem 1
892 INPUT"PILFER FROM PLAYER #";A:IFA<0OR
    A>P%(49)THEN889                 :rem 250
893 IFA=0THENPRINT"NO THEFT.":GOTO450
                                    :rem 179
895 IFA=QTHENPRINT"YOU CAN'T STEAL FROM Y
    OURSELF!":GOTO892               :rem 43
898 IFP%(35+A)<>RTHENPRINT"PLAYER #"A"IS
    {SPACE}NOT HERE !":GOTO889      :rem 129
901 GOSUB1800:IFI<>0THENPRINT"YOU DROP ";
    :GOSUB1780:PRINT"."             :rem 79
904 I=P%(A+41):IFI=0THENPRINT"PLAYER #"A"
    CARRIED NO ITEM!":GOTO450       :rem 33
907 P%(Q+41)=I:P%(A+41)=0:P%(I)=100+Q
                                    :rem 158
908 PRINT"YOU TAKE ";:GOSUB1780:PRINT".":
    GOTO450                         :rem 30
930 PRINT"{2 DOWN}{BLK}{3 SPACES}*** SEAR
    CH THE ROOM ***{4}":J=0:PRINT"{DOWN}Y
    OU FIND THE FOLLOWING:"         :rem 125
933 FORI=1TO31:IFP%(I)<>-RTHEN942:rem 227
936 IFRND(1)>.5THEN942              :rem 6
939 J=J+1:PRINTTAB(4);:GOSUB1780:PRINT".
    :P%(I)=R                        :rem 203
942 NEXT:IFP%(34)<>-RORRND(1)>.5THEN948
                                    :rem 73
945 J=1:PRINT"{4 SPACES}THE BODY OF THE "
    S$(P%(35))".":P%(34)=R          :rem 200
948 IFJ=0THENPRINT"{2 SPACES}------NOTHIN
    G !"                            :rem 177
951 GOTO450                         :rem 113
970 PRINT"{2 DOWN}{BLK}{3 SPACES}*** HIDE
     ITEM OR SUSPECT ***{4}":J=1    :rem 57
971 PRINT"{DOWN}THESE CAN BE HIDDEN:"
                                    :rem 187
973 FORI=1TO31:IFP%(I)<>RTHEN979 :rem 196
976 PRINTJ": ";:GOSUB1780:PRINT".":POKE90
    0+J,I:J=J+1                     :rem 76
979 NEXT:I=P%(Q+41):IFI=0THEN985 :rem 163
982 PRINTJ": ";:GOSUB1780:PRINT" (YOU CAR
    RY IT).":POKE900+J,Q+41:J=J+1 :rem 77
985 IFP%(34)=RTHENPRINTJ": THE BODY OF TH
    E "S$(P%(35))".":POKE900+J,34:J=J+1
                                    :rem 211
988 IFJ=1THENPRINT"NOTHING HERE CAN BE HI
    DDEN!":GOTO450                  :rem 221
991 PRINT"{DOWN}ENTER NUMBER OF ITEM TO H
    IDE, OR":PRINT"ENTER ZERO TO HIDE NOT
    HING."                          :rem 101
994 INPUT"WHAT WILL YOU HIDE";A:IFA<0ORA>
    =JTHEN994                       :rem 235
997 IFA=0THENPRINT"NOTHING HIDDEN.":GOTO4
    50                              :rem 99
1000 I=PEEK(900+A):IFI>34THEN1009:rem 114
1003 P%(I)=-R:IFI=34THENPRINT"YOU HIDE TH
     E BODY.":GOTO450               :rem 37
1006 PRINT"YOU HIDE ";:GOSUB1780:PRINT"."
     :GOTO450                       :rem 57
1009 I=P%(Q+41):PRINT"YOU HIDE THE OBJECT
      YOU CARRY....":GOSUB1780:PRINT"."
                                    :rem 42
1012 P%(Q+41)=0:P%(I)=-R:GOTO450 :rem 233
1700 GETA$:IFA$=""THEN1700          :rem 179
1703 GETB$:IFB$=""THEN1703          :rem 187
1706 A$=A$+B$:RETURN                :rem 128
1710 J=1:GOSUB1730                  :rem 6
1713 READW,I,J:POKES+2,I:POKES+3,J:READI,
     J:POKES+5,I:POKES+6,J          :rem 129
1716 READZ:IFZ<0THENRETURN          :rem 227
```

```
1719 POKES+1,INT(Z/256):POKES,ZAND255:REA
     DZ:POKES+4,W                    :rem 61
1722 FORJ=1TOZ*100:NEXT:POKES+4,0:GOTO171
     6                               :rem 85
1730 P=DA(I):IFJ=1THEN1736           :rem 248
1733 FORI=1TOJ-1:P=PEEK(P)+PEEK(P+1)*256:
     NEXT                            :rem 209
1736 P=P-1:POKE66,INT(P/256):POKE65,PAND2
     55:RETURN                       :rem 62
1760 A=0:I=P%(41+Q):IFI<23ORI>31THENRETUR
     N                               :rem 49
1763 IFI=23THENA=-1:RETURN           :rem 112
1766 I=I-17:IFI=D%(Z,1)ORI=D%(Z,2)THENA=1
                                     :rem 111
1769 RETURN                          :rem 183
1770 IFD%(Z,0)=0THEND%(Z,0)=-1:RETURN
                                     :rem 143
1773 D%(Z,0)=0:RETURN                :rem 201
1780 IFI=0THENPRINT"NO ITEM";:RETURN
                                     :rem 54
1783 IFI<23THENPRINT"THE "S$(I);:RETURN
                                     :rem 147
1786 IFI=23THENPRINT"THE SKELETON KEY";:R
     ETURN                           :rem 212
1789 PRINT"THE "R$(I-17)" KEY";:RETURN
                                     :rem 50
1800 I=P%(Q+41):IFI=0THENRETURN      :rem 132
1803 R=P%(Q+35):P%(I)=R:P%(Q+41)=0:RETURN
                                     :rem 69
1900 PRINT"{CLR}{DOWN}{BLK}{3 SPACES}****
     * MAKE AN ACCUSATION *****{DOWN}{4}"
     :RETURN                         :rem 204
1910 FORI=1TO2200:NEXT:GOTO330       :rem 82
2000 DATA17,0,0,0,240,14435,1,12860,1,144
     35,7,0,4                        :rem 122
2005 DATA12860,1,11457,1,10814,1,9634,1,9
     094,6,9634,8,0,8,-1             :rem 196
2020 DATA17,0,0,0,240,7217,1,6430,1,7217,
     8,0,7                           :rem 236
2025 DATA5407,6,5728,6,4547,6,4817,24,-1
                                     :rem 247
3000 DATA"COOK","BUTLER","GARDENER","CHAU
     FFER","DUKE","DUCHESS","NANNY"
                                     :rem 131
3005 DATA"OPERA STAR","AMBASSADOR","PRIME
      MINISTER","CARVING KNIFE","ROPE"
                                     :rem 9
3010 DATA"BOX OF WEED KILLER","ANTIQUE MA
     CE","DUELLING PISTOL","FENCING FOIL"
                                     :rem 216
3015 DATA"ICE PICK","PLASTIC BAG","CHAIN
     {SPACE}SAW","HEDGE TRIMMERS","POLO M
     ALLET"                          :rem 208
3020 DATA"GARDEN SPADE","ENTRY FOYER","CO
     RRIDOR","HALL","PANTRY","DINING ROOM
     "                               :rem 97
3025 DATA"KITCHEN","STUDY","BEDROOM","BAT
     HROOM","CLOSET","GREENHOUSE","GARDEN
     "                               :rem 187
3030 DATA"POOL","GARAGE",2,13,2,14,3,7,3,
     8,3,11,7,9,8,9,8,10,11,12,12,13,13,1
     4                               :rem 163
3035 DATA"NORTH","EAST","SOUTH","WEST"
                                     :rem 7
4000 DATA33,0,0,88,89,1804,6,2025,3,2145,
     6,2703,3                        :rem 149
4005 DATA2408,1,2551,1,2408,1,2551,1,2408
     ,1,2551,1,2408,1,2551,1,2703,8,-1
                                     :rem 81
5000 DATA5,3,0,2                     :rem 45
5005 DATA4,1,101,100                 :rem 240
5010 DATA104,103,102,1               :rem 81
5015 DATA0,6,2,0                     :rem 49
5020 DATA6,0,1,0                     :rem 44
5025 DATA0,0,5,4                     :rem 51
5030 DATA102,105,0,0                 :rem 239
5035 DATA0,107,106,103               :rem 94
5040 DATA106,0,0,105                 :rem 244
5045 DATA0,0,0,107                   :rem 148
5050 DATA0,0,104,108                 :rem 246
5055 DATA0,108,109,0                 :rem 0
5060 DATA109,100,0,110               :rem 86
5065 DATA101,110,0,0                 :rem 242
6000 DATA65,255,0,9,0,1804,6,1804,4.4,180
     4,1.5,1804,6,2145,4.5,2025,1.5
                                     :rem 202
6005 DATA2025,4.5,1804,1.5,1804,4.5,1804,
     1.5,1804,12,-1                  :rem 177
7000 DATA33,0,0,88,89,2408,4,3215,12,3608
     ,1.33,2408,1.33,3608,1.33       :rem 223
7005 DATA4050,4,4050,4,4050,4,4050,1.33,4
     291,1.33,3215,1.33              :rem 116
7010 DATA4050,6,3608,2,3215,8,-1     :rem 77
```

Program 2: Mystery At Marple Manor —VIC Version

```
5 POKE36879,29:PRINT"{CLR}{BLK}{5 DOWN}
  {2 RIGHT}MYSTERY AT"CHR$(14):PRINTTAB(7
  )"{2 DOWN}MARPLE"                  :rem 159
10 PRINTTAB(12)"{2 DOWN}MANOR"       :rem 220
100 D$="NESW":T$="EC@BDANMKHGA@FB@F@A@@@E
    DCI@@@JICH@@G@@@H@@CL@KM@LB@NBM@@"
                                     :rem 210
105 DEFFNR(X)=INT(RND(1)*X)+1:J=RND(-TI):
    DIMP%(39)                        :rem 183
110 FORJ=0TO21:P%(J)=FNR(13)+1:NEXT:J=FNR
    (10)-1                           :rem 228
115 P%(25)=J+1:P%(24)=P%(J):P%(J)=0
                                     :rem 17
120 I=FNR(10)-1:IFI=JTHEN120         :rem 106
125 P%(22)=I+1:P%(I)=0:J=FNR(12):P%(23)=J
    :P%(9+J)=0                       :rem 188
130 FORJ=0TO21:IFRND(1)<.8THENP%(J)=-P%(J
    )                                :rem 194
135 NEXT:PRINT"{DOWN}{3 RIGHT}{2 DOWN}
    {BLU}PLAYERS (1-6)?"             :rem 80
140 GETA$:J=VAL(A$):IFJ<1ORJ>6THEN140
                                     :rem 215
145 P%(38)=J:FORI=1TOJ:P%(25+I)=1:NEXT:Q=
    1                                :rem 210
200 PRINT"{CLR}{BLK}{3 DOWN}PLAYER #"Q:PR
    INTCHR$(142)"---PRESS {RVS}RETURN"
                                     :rem 244
205 GETA$:IFA$<>CHR$(13)THEN205      :rem 0
210 R=P%(Q+25):PRINT"{DOWN}YOU ARE IN":PR
    INT"THE ";:X=R+22:GOSUB3000      :rem 167
215 PRINT"LEAVE? [Y/N]"              :rem 163
220 GETA$:IFA$="N"THEN275            :rem 163
225 IFA$<>"Y"THEN220                 :rem 103
230 C$=MID$(T$,4*R-3,4):PRINT"EXITS ARE T
    O THE:{DOWN}"                    :rem 219
235 FORI=1TO4:X=I+36:IFMID$(C$,I,1)>"@"TH
    ENGOSUB3000                      :rem 64
240 NEXT:PRINT"{DOWN}TYPE {RVS}N{RIGHT}S
    {RIGHT}E{OFF} OR {RVS}W":I=0     :rem 241
245 GETA$:IFA$<"E"THEN245            :rem 157
250 FORJ=1TO4:IFMID$(D$,J,1)=A$THENI=J
                                     :rem 1
255 NEXT:IFI=0THEN245                :rem 36
260 X=36+I:PRINT"MOVING ";:GOSUB3000:A$=M
    ID$(C$,I,1)                      :rem 1
```

```
265 I=ASC(A$)-64:IFI<1THENPRINT"NO DOOR":
    PRINT"NO MOVE":GOSUB4050:GOTO275
                                    :rem 10
270 R=I:P%(Q+25)=I                  :rem 218
275 GOSUB4050                       :rem 231
276 PRINT"{CLR}{2 DOWN}YOU ARE IN":PRINT"
    THE ";:X=R+22:GOSUB3000         :rem 229
280 PRINT"YOU HAVE";:X=P%(31+Q)+1:IFX=1TH
    ENPRINT" NO ITEM":GOTO290       :rem 1
285 PRINT" THE ";:GOSUB3000         :rem 9
290 P=0:PRINT"YOU SEE:":FORI=0TO21:rem 84
295 IFP%(I)=RTHENX=I+1:PRINT"THE ";:GOSUB
    3000:P=1                        :rem 210
300 NEXT:IFP%(24)=RTHENGOSUB3100:P=1
                                    :rem 193
305 FORJ=1TO6:IFJ<>QANDP%(J+25)=RTHENPRIN
    T"PLAYER"J:P=1                  :rem 1
310 NEXT:IFP=0THENPRINT"NOTHING!":rem 145
315 INPUT"{DOWN}PRESS {RVS}RETURN{WHT}";A
    $                               :rem 162
400 PRINT"{CLR}{DOWN}{BLK}{3 SPACES}OPTIO
    NS:{2 SPACES}"                  :rem 255
401 PRINT"{DOWN}{BLU}1 ACCUSE":PRINT"2 DR
    OP":PRINT"3 TAKE":PRINT"4 SEARCH"
                                    :rem 69
405 PRINT"5 PILFER":PRINT"6 NO ACTION":PR
    INT"{DOWN}{BLK}CHOOSE NOW!"     :rem 127
415 GETA$:I=VAL(A$):ONIGOTO600,700,750,80
    0,850,500                       :rem 0
420 GOTO415                         :rem 105
500 INPUT"{DOWN}PRESS {RVS}RETURN{WHT}";A
    $                               :rem 158
505 Q=Q+1:IFQ>P%(38)THENQ=1         :rem 76
510 IFP%(Q+25)=0THEN505             :rem 4
515 GOTO200                         :rem 102
600 P=1:W=0:Z=10:GOSUB3200:INPUT"{DOWN}VI
    CTIM";J:IFJ<>P%(25)THENP=0      :rem 25
605 GOSUB3200:INPUT"{DOWN}MURDERER";J:IFJ
    <>P%(22)THENP=0                 :rem 141
610 W=10:Z=12:GOSUB3200:INPUT"{DOWN}WEAPO
    N";J:IFJ<>P%(23)THENP=0         :rem 81
615 W=22:Z=14:GOSUB3200:INPUT"{DOWN}SCENE
    ";J:IFJ<>ABS(P%(24))THENP=0     :rem 39
650 IFPTHENPRINT"{CLR}{3 DOWN}THAT'S RIGH
    T!":PRINT"YOU WIN!":GOTO670     :rem 177
655 PRINT"{CLR}{2 DOWN}WRONG SOLUTION!":P
    RINT"{DOWN}YOU LOSE!"           :rem 190
660 P%(25+Q)=0:P%(39)=P%(39)+1:GOSUB3300:
    IFP%(39)<P%(38)THEN500          :rem 149
665 PRINT"{DOWN}ALL PLAYERS HAVE LOST!"
                                    :rem 3
670 PRINT"{DOWN}{RED}THE SOLUTION:":PRINT
    "{BLK}{DOWN}THE ";:X=P%(22):GOSUB3000
                                    :rem 113
673 PRINT"KILLED THE ";:X=P%(25):GOSUB300
    0                               :rem 187
675 PRINT"IN THE ";:X=22+ABS(P%(24)):GOSU
    B3000                           :rem 84
678 PRINT"WITH THE ";:X=P%(23)+10:GOSUB30
    00:END                          :rem 226
700 GOSUB3300:IFX=0THENPRINT"DROP NO ITEM
    ":GOTO500                       :rem 116
705 PRINT"YOU DROP THE ";:GOSUB3000:GOTO5
    00                              :rem 64
750 P=1:PRINT"AVAILABLE:":FORI=10TO21:IFP
    %(I)=RTHENGOSUB3400             :rem 11
760 NEXT:IFP=1THENPRINT"NO ITEMS":GOTO500
                                    :rem 138
765 INPUT"CHOOSE";Z:IFZ<0ORZ>=PTHEN765
                                    :rem 242
770 IFZ=0THENPRINT"TAKE NOTHING":GOTO500
                                    :rem 56
775 GOSUB3300:IFX>0THENPRINT"DROP THE ";:
    GOSUB3000                       :rem 70
780 I=PEEK(900+Z):P%(I)=100+Q:P%(Q+31)=I:
    X=I+1:PRINT"YOU TAKE THE ";:GOSUB3000
    :GOTO500                        :rem 244
800 P=0:PRINT"{DOWN}YOU FIND:":FORI=0TO21
    :IFP%(I)<>-RORRND(1)>.6THEN810
                                    :rem 232
805 P=1:PRINT"THE ";:X=I+1:GOSUB3000:P%(I
    )=R                             :rem 75
810 NEXT:IFP%(24)=-RANDRND(1)<.6THENP=1:G
    OSUB3100:P%(24)=R               :rem 194
820 IFP=0THENPRINT"--NOTHING"       :rem 87
825 GOTO500                         :rem 109
850 P=0:PRINT"{DOWN}NOW HERE:":FORI=1TO6:
    IFP%(25+I)=RANDI<>QTHENPRINT"PLAYER #
    "I:P=1                          :rem 138
860 NEXT:IFP=0THENPRINT"NOBODY!":GOTO500
                                    :rem 87
865 INPUT"STEAL FROM WHOM";W:IFW<0ORW>P%(
    38)THEN865                      :rem 182
870 IFP%(25+W)<>RTHENPRINT"NOT HERE!":GOT
    O865                            :rem 245
875 GOSUB3300:IFX>0THENPRINT"DROP THE ";:
    GOSUB3000                       :rem 71
880 P=P%(W+31):IFP=0THENPRINT"NOTHING TAK
    EN":GOTO500                     :rem 241
885 P%(Q+31)=P:P%(W+31)=0:P%(P)=100+Q:X=P
    +1:PRINT"YOU TAKE THE ";:GOSUB3000:GO
    TO500                           :rem 14
3000 FORJ=1TOX:READX$:NEXT:PRINTX$:RESTOR
     E:RETURN                       :rem 99
3100 X=P%(25):PRINT"THE BODY OF":PRINT"TH
     E ";:GOSUB3000:RETURN          :rem 243
3200 PRINT"{CLR}{BLK}":FORI=1TOZ:X=I+W:PR
     INTITAB(4)": THE ";:GOSUB3000:NEXT:R
     ETURN                          :rem 62
3300 I=P%(Q+31):X=0:IFI=0THENRETURN
                                    :rem 127
3305 P%(I)=R:P%(Q+31)=0:X=I+1:RETURN
                                    :rem 68
3400 PRINTPTAB(4)":THE ";:X=I+1:GOSUB3000
     :POKE900+P,I:P=P+1:RETURN      :rem 10
4000 DATACOOK,BUTLER,GARDENER,DOCTOR,DUKE
     ,DUCHESS,NANNY,"FILM STAR"     :rem 129
4005 DATASENATOR,JUDGE,KNIFE,ROPE,POISON,
     MACE,PISTOL,SWORD,"ICE PICK"   :rem 5
4010 DATABOMB,RAZOR,LAMP,CLUB,SHOVEL,FOYE
     R,CORRIDOR,HALL,PANTRY,"DINING ROOM"
                                    :rem 92
4015 DATAKITCHEN,STUDY,BEDROOM,BATHROOM,C
     LOSET,GREENHOUSE,GARDEN,POOL:rem 197
4020 DATAGARAGE,NORTH,EAST,SOUTH,WEST
                                    :rem 197
4050 FORI=1TO1200:NEXT:RETURN       :rem 96
```

80-Columns

(Article on page 48.)

BEFORE TYPING...

Before typing in programs, please refer to "How To Type COMPUTE!'s Gazette Programs," "A Beginner's Guide To Typing In Programs," and "The Automatic Proofreader" that appear before the Program Listings.

Program 1: Screen-80

```
49152 :011,008,000,000,158,050,227
49158 :048,054,049,000,000,000,157
```

```
49164 :160,044,185,065,008,153,115
49170 :198,002,136,192,255,208,241
49176 :245,160,000,169,160,133,123
49182 :252,132,251,169,008,133,207
49188 :254,169,109,133,253,177,107
49194 :253,145,251,200,208,249,068
49200 :165,252,201,173,240,007,062
49206 :230,254,230,252,076,042,114
49212 :008,076,198,002,169,054,055
49218 :133,001,032,000,160,169,049
49224 :055,133,001,096,072,169,086
49230 :054,133,001,104,032,028,174
49236 :162,072,169,055,133,001,164
49242 :104,096,072,169,054,133,206
49248 :001,104,032,148,161,072,102
49254 :169,055,133,001,104,096,148
49260 :169,090,141,250,255,169,158
49266 :169,141,251,255,173,002,081
49272 :221,009,003,141,002,221,205
49278 :169,252,045,000,221,141,186
49284 :000,221,169,032,013,017,072
49290 :208,141,017,208,169,072,185
49296 :141,024,208,169,000,141,059
49302 :244,173,169,011,141,134,254
49308 :002,169,000,141,243,173,116
49314 :133,212,141,236,173,169,202
49320 :015,141,033,208,169,015,237
49326 :141,032,208,032,244,160,223
49332 :032,003,164,169,210,141,131
49338 :038,003,169,002,141,039,066
49344 :003,169,226,141,036,003,002
49350 :169,002;141,037,003,032,070
49356 :099,160,096,160,000,185,136
49362 :116,160,141,227,173,032,035
49368 :042,162,200,192,129,208,125
49374 :242,096,147,013,029,029,010
49380 :029,029,029,029,029,029,146
49386 :029,029,029,029,029,029,152
49392 :029,029,029,029,029,029,158
49398 :029,029,029,029,056,048,210
49404 :032,067,079,076,085,077,156
49410 :078,083,032,070,079,082,170
49416 :032,084,072,069,032,067,108
49422 :079,077,077,079,068,079,217
49428 :082,069,032,054,052,017,070
49434 :017,157,157,157,157,157,060
49440 :157,157,157,157,157,157,206
49446 :157,157,157,157,157,157,212
49452 :157,157,157,157,157,066,127
49458 :089,032,071,082,069,071,208
49464 :071,032,080,069,069,076,197
49470 :069,017,017,157,157,157,124
49476 :157,157,157,157,157,157,242
49482 :157,157,157,157,157,157,248
49488 :065,078,068,032,075,069,211
49494 :086,073,078,032,077,065,241
49500 :082,084,073,078,160,000,057
49506 :173,033,208,041,015,170,226
49512 :173,134,002,010,010,010,187
49518 :010,141,237,173,138,013,054
49524 :237,173,032,058,169,153,170
49530 :000,208,153,000,209,153,077
49536 :000,210,200,008,032,074,140
49542 :169,040,208,216,160,231,134
49548 :032,058,169,153,000,211,251
49554 :032,074,169,136,192,255,236
49560 :208,242,096,072,169,001,172
49566 :141,244,173,104,032,042,126
49572 :162,169,000,141,244,173,029
49578 :165,198,240,252,120,032,153
49584 :180,229,201,131,208,016,117
49590 :162,009,120,134,198,189,226
49596 :230,236,157,118,002,202,109
49602 :208,247,240,228,201,013,051
49608 :208,209,160,007,032,058,106
49614 :169,177,251,077,223,173,252
49620 :145,251,032,074,169,160,019
49626 :079,132,208,032,058,169,128
49632 :177,209,201,032,208,003,030
49638 :136,208,244,200,132,200,070
49644 :160,000,132,211,132,212,059
49650 :165,202,048,060,165,202,060
49656 :133,211,197,200,144,052,161
49662 :176,094,165,153,208,014,040
49668 :165,009,133,202,173,222,140
49674 :173,133,201,133,214,076,172
49680 :190,161,032,074,169,076,206
49686 :102,241,160,007,032,058,110
49692 :169,177,251,077,223,173,074
49698 :145,251,032,074,169,076,013
49704 :062,161,152,072,138,072,185
49710 :165,208,240,230,032,058,211
49716 :169,164,211,177,209,133,091
49722 :215,032,074,169,041,063,140
49728 :006,215,036,215,016,002,042
49734 :009,128,144,004,166,212,221
49740 :208,004,112,002,009,064,219
49746 :032,074,169,230,211,032,062
49752 :132,230,196,200,208,026,056
49758 :169,000,133,208,169,013,018
49764 :166,153,224,003,240,006,124
49770 :166,154,224,003,240,003,128
49776 :032,042,162,169,013,032,050
49782 :074,169,133,215,104,170,215
49788 :104,168,165,215,201,222,175
49794 :208,002,169,255,024,096,116
49800 :072,165,154,201,003,208,171
49806 :004,104,076,042,162,076,094
49812 :213,241,072,141,227,173,191
49818 :152,072,138,072,169,000,245
49824 :141,235,173,032,070,162,205
49830 :032,068,168,032,146,168,012
49836 :104,170,104,168,104,096,150
49842 :173,227,173,032,132,230,121
49848 :208,006,169,001,141,235,176
49854 :173,096,173,227,173,201,209
49860 :032,144,003,076,097,162,198
49866 :076,194,162,201,096,176,083
49872 :023,201,064,176,003,076,239
49878 :174,162,201,128,240,082,177
49884 :056,173,227,173,233,064,122
49890 :141,227,173,076,174,162,155
49896 :201,127,144,009,240,044,229
49902 :201,160,144,060,076,149,004
49908 :162,056,173,227,173,233,244
49914 :032,141,227,173,076,174,049
49920 :162,201,192,176,012,056,031
49926 :173,227,173,233,064,141,249
49932 :227,173,076,174,162,024,080
49938 :173,227,173,105,128,141,197
49944 :227,173,173,243,173,240,229
49950 :004,206,243,173,096,173,157
49956 :241,173,208,005,169,000,064
49962 :141,242,173,096,173,243,086
49968 :173,005,212,240,035,173,118
49974 :227,173,201,032,176,041,136
49980 :201,013,240,110,201,020,077
49986 :240,004,165,212,208,013,140
49992 :173,243,173,208,008,169,022
49998 :001,141,235,173,076,078,014
```

```
50004 :163,076,028,163,173,241,160
50010 :173,208,005,169,000,141,018
50016 :242,173,076,066,163,173,221
50022 :227,173,201,141,240,066,126
50028 :201,148,208,012,165,212,030
50034 :208,008,169,001,141,240,113
50040 :173,076,066,163,056,173,059
50046 :227,173,233,064,141,227,167
50052 :173,076,028,163,173,243,220
50058 :173,208,017,169,000,141,078
50064 :243,173,165,212,208,011,132
50070 :169,000,141,242,173,076,183
50076 :058,163,206,243,173,169,144
50082 :001,141,242,173,169,000,120
50088 :141,235,173,076,186,163,118
50094 :169,001,141,235,173,169,038
50100 :000,141,240,173,133,212,055
50106 :173,227,173,201,032,176,144
50112 :102,201,008,208,005,160,108
50118 :128,140,145,002,201,009,055
50124 :208,005,160,000,140,145,094
50130 :002,201,013,208,005,072,199
50136 :032,053,165,104,201,014,017
50142 :208,005,160,001,140,236,204
50148 :173,201,017,208,008,238,049
50154 :222,173,072,032,206,164,079
50160 :104,201,018,208,008,160,171
50166 :001,140,242,173,140,241,159
50172 :173,201,019,208,017,160,006
50178 :000,132,009,140,222,173,166
50184 :072,032,206,164,169,240,123
50190 :141,223,173,104,201,029,117
50196 :208,007,230,009,072,032,066
50202 :210,164,104,201,020,208,165
50208 :005,072,032,092,165,104,246
50214 :096,201,141,208,005,072,249
50220 :032,053,165,104,201,142,229
50226 :208,005,160,000,140,236,031
50232 :173,201,145,208,008,206,229
50238 :222,173,072,032,206,164,163
50244 :104,201,146,208,008,160,127
50250 :000,140,242,173,140,241,242
50256 :173,201,147,208,005,072,118
50262 :032,003,164,104,201,148,226
50268 :208,005,072,032,080,166,143
50274 :104,201,157,208,007,198,205
50280 :009,072,032,210,164,104,183
50286 :096,032,058,169,169,000,122
50292 :133,251,169,224,133,252,254
50298 :169,000,141,225,173,141,203
50304 :226,173,141,036,164,169,013
50310 :224,141,037,164,169,000,101
50316 :170,168,138,153,255,255,255
50322 :136,208,249,238,037,164,154
50328 :173,037,164,201,255,208,166
50334 :239,160,064,169,000,153,175
50340 :000,255,136,016,250,169,222
50346 :000,133,009,141,222,173,080
50352 :169,240,141,223,173,173,015
50358 :244,173,240,006,160,007,244
50364 :169,240,145,251,162,024,155
50370 :024,189,196,169,105,212,065
50376 :141,107,164,189,170,169,116
50382 :141,106,164,169,032,160,210
50388 :079,153,255,255,136,192,002
50394 :255,208,248,202,224,255,074
50400 :208,224,032,210,164,032,070
50406 :074,169,096,169,000,141,111
50412 :226,173,165,009,074,010,125
50418 :046,226,173,010,046,226,201
50424 :173,010,046,226,173,141,249
50430 :225,173,172,222,173,185,124
50436 :118,169,133,251,024,185,116
50442 :144,169,109,226,173,133,196
50448 :252,024,173,225,173,101,196
50454 :251,133,251,169,000,101,159
50460 :252,133,252,024,165,252,082
50466 :105,224,133,252,165,009,154
50472 :041,001,240,008,169,015,002
50478 :141,223,173,076,205,164,004
50484 :169,240,141,223,173,096,070
50490 :169,255,133,202,165,009,223
50496 :133,211,048,014,201,080,239
50502 :144,021,169,000,133,009,034
50508 :238,222,173,076,241,164,166
50514 :230,009,206,222,173,048,202
50520 :024,169,079,133,009,173,163
50526 :222,173,133,214,048,013,129
50532 :201,025,144,012,206,222,142
50538 :173,032,135,167,076,008,185
50544 :165,238,222,173,169,001,056
50550 :141,234,173,173,244,173,232
50556 :240,015,160,007,032,058,124
50562 :169,177,251,077,223,173,176
50568 :145,251,032,074,169,174,213
50574 :222,173,189,170,169,133,174
50580 :209,024,189,196,169,105,016
50586 :212,133,210,032,168,168,053
50592 :096,238,222,173,169,000,034
50598 :133,009,141,243,173,141,238
50604 :242,173,141,241,173,032,150
50610 :210,164,173,033,208,041,239
50616 :015,205,246,173,240,003,042
50622 :032,244,160,173,033,208,016
50628 :141,246,173,096,032,058,174
50634 :169,169,001,141,244,173,075
50640 :165,009,208,003,076,066,223
50646 :166,160,007,177,251,077,028
50652 :223,173,145,251,056,165,209
50658 :251,233,008,133,253,165,245
50664 :252,233,000,133,254,165,245
50670 :009,041,001,208,025,160,170
50676 :007,177,251,041,240,074,010
50682 :074,074,074,141,228,173,246
50688 :177,253,041,240,013,228,184
50694 :173,145,253,136,016,233,194
50700 :172,222,173,200,024,185,220
50706 :144,169,105,224,141,238,015
50712 :173,056,185,118,169,233,190
50718 :001,141,230,173,173,238,218
50724 :173,233,000,141,231,173,219
50730 :169,008,141,229,173,160,154
50736 :004,173,230,173,141,222,223
50742 :165,173,231,173,141,223,136
50748 :165,056,169,080,229,009,000
50754 :074,105,000,170,024,008,191
50760 :040,046,255,255,008,056,220
50766 :173,222,165,233,008,141,252
50772 :222,165,173,223,165,233,241
50778 :000,141,223,165,202,208,005
50784 :231,136,240,004,040,076,055
50790 :197,165,040,206,230,173,089
50796 :208,003,206,231,173,206,111
50802 :229,173,208,185,160,007,052
50808 :177,251,077,223,173,145,142
50814 :251,174,222,173,189,170,025
50820 :169,133,253,024,189,196,072
50826 :169,105,212,133,254,056,043
50832 :169,079,229,009,170,164,196
50838 :009,177,253,136,145,253,099
```

```
50844 :200,200,202,224,255,208,165
50850 :244,169,032,160,079,145,223
50856 :253,198,009,032,210,164,010
50862 :169,000,141,234,173,032,155
50868 :074,169,169,000,141,244,209
50874 :173,096,032,058,169,172,118
50880 :222,173,200,056,185,118,122
50886 :169,233,008,133,253,185,155
50892 :144,169,233,000,133,254,113
50898 :024,165,254,105,224,133,091
50904 :254,160,007,177,253,041,084
50910 :015,240,003,076,116,167,071
50916 :136,016,244,160,007,177,200
50922 :251,077,223,173,141,245,064
50928 :173,173,244,173,240,008,227
50934 :173,245,173,145,251,076,029
50940 :146,166,165,009,041,001,012
50946 :240,018,024,165,251,105,037
50952 :008,141,230,173,165,252,209
50958 :105,000,141,231,173,076,228
50964 :180,166,165,251,141,230,129
50970 :173,165,252,141,231,173,137
50976 :169,008,141,229,173,160,144
50982 :004,173,230,173,141,210,201
50988 :166,173,231,173,141,211,115
50994 :166,056,169,080,229,009,247
51000 :074,170,024,008,040,110,226
51006 :255,255,008,024,173,210,219
51012 :166,105,008,141,210,166,096
51018 :173,211,166,105,000,141,102
51024 :211,166,202,208,231,136,210
51030 :240,004,040,076,187,166,031
51036 :040,238,230,173,208,003,216
51042 :238,231,173,206,229,173,068
51048 :208,187,024,165,251,105,020
51054 :008,133,253,165,252,105,002
51060 :000,133,254,165,009,041,206
51066 :001,240,031,160,007,177,226
51072 :251,041,015,010,010,010,209
51078 :010,141,228,173,177,253,092
51084 :041,015,013,228,173,145,243
51090 :253,177,251,041,240,145,229
51096 :251,136,016,227,160,007,181
51102 :177,251,077,223,173,141,176
51108 :245,173,173,244,173,240,132
51114 :005,173,245,173,145,251,138
51120 :032,210,164,032,058,169,073
51126 :174,222,173,189,170,169,255
51132 :133,253,024,189,196,169,128
51138 :105,212,133,254,056,169,099
51144 :079,229,009,170,160,078,157
51150 :177,253,200,145,253,136,090
51156 :136,202,224,255,208,244,201
51162 :169,032,164,009,145,253,222
51168 :169,000,141,234,173,032,205
51174 :074,169,173,243,173,201,239
51180 :080,240,003,238,243,173,189
51186 :096,032,058,169,169,224,222
51192 :141,158,167,024,169,224,107
51198 :105,001,141,155,167,160,215
51204 :000,185,064,255,153,000,149
51210 :255,200,208,247,238,155,033
51216 :167,238,158,167,173,155,050
51222 :167,201,255,208,234,169,232
51228 :000,160,000,153,000,254,083
51234 :200,208,250,160,192,153,173
51240 :000,255,136,192,255,208,062
51246 :248,056,165,251,233,064,039
51252 :133,251,165,252,233,001,063
51258 :133,252,162,001,189,170,197
51264 :169,141,246,167,024,189,232
51270 :196,169,105,212,141,247,116
51276 :167,202,189,170,169,141,090
51282 :249,167,024,189,196,169,052
51288 :105,212,141,250,167,162,101
51294 :008,160,000,185,255,255,189
51300 :153,255,255,200,208,247,138
51306 :238,247,167,238,250,167,133
51312 :202,208,238,162,024,189,111
51318 :170,169,141,029,168,024,051
51324 :189,196,169,105,212,141,112
51330 :030,168,169,032,160,079,000
51336 :153,255,255,136,192,255,102
51342 :208,248,032,074,169,169,018
51348 :127,141,000,220,173,001,042
51354 :220,201,251,008,169,127,106
51360 :141,000,220,040,208,009,010
51366 :160,000,234,202,208,252,198
51372 :136,208,249,096,169,000,006
51378 :133,254,032,058,169,173,229
51384 :227,173,041,001,240,008,106
51390 :169,015,141,224,173,076,220
51396 :095,168,169,240,141,224,209
51402 :173,173,227,173,074,010,008
51408 :038,254,010,038,254,010,044
51414 :038,254,133,253,173,236,021
51420 :173,208,014,024,169,222,006
51426 :101,253,133,253,169,169,024
51432 :101,254,076,140,168,024,227
51438 :169,222,101,253,133,253,089
51444 :169,171,101,254,133,254,046
51450 :032,074,169,096,173,235,005
51456 :173,208,016,169,000,141,195
51462 :234,173,032,168,168,032,045
51468 :044,169,230,009,032,210,194
51474 :164,096,032,125,164,032,119
51480 :058,169,160,007,174,234,058
51486 :173,240,005,169,000,076,181
51492 :191,168,177,253,045,224,070
51498 :173,174,224,173,224,240,226
51504 :208,004,074,074,074,074,044
51510 :141,228,173,173,223,173,141
51516 :201,015,240,010,173,228,159
51522 :173,010,010,010,010,141,164
51528 :228,173,169,255,174,234,025
51534 :173,208,005,173,223,173,009
51540 :073,255,049,251,013,228,185
51546 :173,192,007,208,008,174,084
51552 :244,173,240,003,077,223,032
51558 :173,174,234,173,208,023,063
51564 :174,241,173,208,005,174,059
51570 :242,173,240,013,077,223,058
51576 :173,072,173,227,173,009,179
51582 :128,141,227,173,104,145,020
51588 :251,136,016,148,173,243,075
51594 :173,005,212,240,005,169,174
51600 :000,141,242,173,032,074,038
51606 :169,096,164,009,173,227,220
51612 :173,032,058,169,145,209,174
51618 :032,074,169,096,072,120,213
51624 :173,014,220,041,254,141,243
51630 :014,220,169,052,133,001,251
51636 :104,096,072,169,054,133,040
51642 :001,173,014,220,009,001,092
51648 :141,014,220,088,104,096,087
51654 :072,152,072,138,072,169,105
51660 :169,072,169,109,072,008,035
51666 :032,074,169,120,076,071,240
51672 :254,032,058,169,104,170,235
51678 :104,168,104,064,000,064,214
```

```
51684 :128,192,000,064,128,192,164
51690 :000,064,128,192,000,064,170
51696 :128,192,000,064,128,192,176
51702 :000,064,128,192,000,064,182
51708 :000,001,002,003,005,006,013
51714 :007,008,010,011,012,013,063
51720 :015,016,017,018,020,021,115
51726 :022,023,025,026,027,028,165
51732 :030,031,000,080,160,240,049
51738 :064,144,224,048,128,208,074
51744 :032,112,192,016,096,176,144
51750 :000,080,160,240,064,144,214
51756 :224,048,128,208,000,000,140
51762 :000,000,001,001,001,002,055
51768 :002,002,003,003,003,004,073
51774 :004,004,005,005,005,005,090
51780 :006,006,006,007,007,007,107
51786 :068,170,170,174,138,138,164
51792 :106,000,196,170,168,200,152
51798 :168,170,196,000,206,168,226
51804 :168,174,168,168,206,000,208
51810 :228,138,136,234,138,138,086
51816 :132,000,174,164,164,228,198
51822 :164,164,174,000,234,042,120
51828 :042,044,042,170,074,000,232
51834 :138,142,142,138,138,138,190
51840 :234,000,206,170,170,170,054
51846 :170,170,174,000,196,170,246
51852 :170,202,138,138,132,002,154
51858 :198,168,168,196,162,162,176
51864 :172,000,234,074,074,074,012
51870 :074,074,078,000,170,170,212
51876 :170,170,174,174,074,000,158
51882 :170,170,074,068,068,164,116
51888 :164,000,230,036,068,068,230
51894 :068,132,230,000,070,162,076
51900 :130,194,130,130,230,000,234
51906 :032,114,036,047,036,034,237
51912 :032,032,004,004,004,004,024
51918 :004,000,004,000,160,170,032
51924 :014,010,014,010,000,000,004
51930 :074,226,132,228,036,232,122
51936 :074,000,066,162,164,064,242
51942 :160,160,096,016,040,068,002
51948 :130,130,130,068,040,000,222
51954 :000,160,068,238,068,160,168
51960 :000,000,000,000,000,014,006
51966 :000,096,032,064,001,001,192
51972 :002,006,004,008,072,000,096
51978 :068,172,164,164,164,164,138
51984 :078,000,078,162,036,066,180
51990 :130,138,228,000,174,168,092
51996 :238,034,034,042,038,000,158
52002 :078,162,130,196,164,168,164
52008 :072,000,068,170,170,070,078
52014 :162,164,072,000,000,000,188
52020 :068,000,000,068,004,008,200
52026 :016,032,078,128,078,032,166
52032 :016,000,132,074,034,020,084
52038 :036,064,132,000,004,004,054
52044 :014,254,010,004,014,000,116
52050 :032,032,032,047,032,032,033
52056 :032,032,000,015,240,000,151
52062 :000,000,000,000,004,004,102
52068 :004,004,004,004,244,004,108
52074 :032,032,032,044,038,034,062
52080 :034,034,034,034,054,028,074
52086 :000,000,000,000,136,136,134
52092 :132,132,130,130,129,241,250
52098 :031,024,040,040,072,072,153
52104 :136,136,240,016,022,031,205
52110 :031,022,016,016,000,009,236
52116 :015,015,015,006,240,000,183
52122 :064,064,064,065,067,066,032
52128 :066,066,144,144,102,105,019
52134 :105,102,144,144,098,098,089
52140 :146,146,098,098,242,002,136
52146 :002,066,066,239,226,066,075
52152 :066,002,066,130,066,130,132
52158 :066,130,066,130,015,007,092
52164 :023,099,163,163,161,001,038
52170 :012,012,012,012,012,012,018
52176 :012,012,015,000,000,000,247
52182 :240,240,240,240,008,008,166
52188 :008,008,008,008,008,248,252
52194 :161,081,161,081,161,081,184
52200 :161,081,015,014,012,012,015
52206 :172,088,168,088,050,050,086
52212 :050,051,050,050,050,050,033
52218 :002,002,002,003,048,048,099
52224 :048,048,000,000,000,224,064
52230 :032,032,047,047,002,002,168
52236 :002,063,032,032,032,032,205
52242 :002,002,002,254,034,034,090
52248 :034,034,140,140,140,140,140
52254 :140,140,140,140,063,063,204
52260 :048,048,048,048,048,048,068
52266 :240,240,240,000,000,015,009
52272 :015,015,016,016,016,016,142
52278 :028,028,028,252,050,050,234
52284 :050,062,000,000,000,000,172
52290 :204,204,204,204,003,003,120
52296 :003,003,064,160,172,162,124
52302 :142,138,110,000,128,128,212
52308 :198,168,168,168,198,000,216
52314 :032,032,100,170,174,168,254
52320 :102,000,032,064,068,234,084
52326 :074,070,066,004,128,132,064
52332 :192,164,164,164,164,000,188
52338 :008,040,010,042,044,042,044
52344 :170,064,192,064,074,078,250
52350 :078,074,234,000,000,000,000
52356 :196,170,170,170,164,000,234
52362 :000,000,198,170,170,198,106
52368 :130,130,000,000,206,168,010
52374 :142,130,142,000,000,064,116
52380 :234,074,074,074,078,000,178
52386 :000,000,170,170,174,174,082
52392 :074,000,000,000,170,170,070
52398 :070,162,162,012,006,004,078
52404 :228,036,068,132,230,000,106
52410 :070,162,130,194,130,130,234
52416 :230,000,032,114,036,047,139
52422 :036,034,032,032,004,004,084
52428 :004,004,004,000,004,000,220
52434 :160,170,014,010,014,010,076
52440 :000,000,074,226,132,228,108
52446 :036,232,074,000,066,162,024
52452 :164,064,160,160,096,016,120
52458 :040,068,130,130,130,068,032
52464 :040,000,000,160,068,238,234
52470 :068,160,000,000,000,000,218
52476 :000,014,000,096,032,064,202
52482 :001,001,002,006,004,008,024
52488 :072,000,068,172,164,164,136
52494 :164,164,078,000,078,162,148
52500 :036,066,130,138,228,000,106
52506 :174,168,238,034,034,042,204
52512 :038,000,078,162,130,196,124
52518 :164,168,072,000,068,170,168
```

```
52524 :170,070,162,164,072,000,170
52530 :000,000,068,000,000,068,186
52536 :004,008,016,032,078,128,066
52542 :078,032,016,000,132,074,138
52548 :034,020,036,064,132,000,098
52554 :004,010,010,254,010,010,116
52560 :010,000,196,170,168,200,056
52566 :168,170,196,000,206,168,226
52572 :168,174,168,168,206,000,208
52578 :228,138,136,234,138,138,086
52584 :132,000,174,164,164,228,198
52590 :164,164,174,000,234,042,120
52596 :042,044,042,170,074,000,232
52602 :138,142,142,138,138,138,190
52608 :234,000,206,170,170,170,054
52614 :170,170,174,000,196,170,246
52620 :170,202,138,138,132,002,154
52626 :198,168,168,196,162,162,176
52632 :172,000,234,074,074,074,012
52638 :074,074,078,000,170,170,212
52644 :170,170,174,174,074,000,158
52650 :170,170,074,068,068,164,116
52656 :164,000,226,034,066,079,233
52662 :066,130,226,002,066,130,034
52668 :066,130,066,130,066,130,008
52674 :082,169,084,162,089,164,176
52680 :082,169,012,012,012,012,243
52686 :012,012,012,012,015,000,013
52692 :000,000,240,240,240,240,148
52698 :008,008,008,008,008,008,010
52704 :008,248,161,081,161,081,196
52710 :161,081,161,081,004,009,215
52716 :002,004,169,082,164,089,234
52722 :050,050,050,051,050,050,031
52728 :050,050,002,002,002,003,101
52734 :048,048,048,048,000,000,190
52740 :000,224,032,032,047,047,130
52746 :002,002,002,063,032,032,143
52752 :032,032,002,002,002,254,084
52758 :034,034,034,034,140,140,182
52764 :140,140,140,140,140,140,100
52770 :063,063,048,048,048,048,096
52776 :048,048,240,240,240,000,088
52782 :000,015,015,015,000,032,123
52788 :032,032,172,108,044,012,196
52794 :050,050,050,062,000,000,014
52800 :000,000,204,204,204,204,112
52806 :003,003,003,003,000,013,095
```

Program 2: Custom-80

```
49152 :169,000,032,144,255,169,001
49158 :132,133,178,169,003,133,242
49164 :179,169,075,133,251,169,220
49170 :018,133,252,169,000,133,211
49176 :253,169,048,133,254,160,017
49182 :000,177,251,145,253,200,032
49188 :208,249,230,252,230,254,179
49194 :165,252,201,023,208,239,106
49200 :169,011,141,033,208,169,011
49206 :000,141,134,002,141,032,248
49212 :208,169,147,032,210,255,057
49218 :169,000,141,062,003,141,070
49224 :170,195,141,160,195,141,050
49230 :172,195,141,173,195,169,099
49236 :008,032,210,255,169,005,251
49242 :141,165,195,169,013,141,146
49248 :248,007,169,007,141,039,195
49254 :208,169,001,141,021,208,082
49260 :169,000,168,153,064,003,153
49266 :200,192,064,208,248,169,171
49272 :252,141,064,003,141,091,044
49278 :003,160,003,169,132,153,234
49284 :064,003,200,200,200,192,223
49290 :026,144,246,032,073,199,090
49296 :032,159,192,032,198,194,183
49302 :032,248,194,032,049,194,131
49308 :076,144,192,162,000,160,122
49314 :000,024,032,240,255,173,118
49320 :160,195,041,001,201,001,255
49326 :240,005,169,240,076,183,063
49332 :192,169,015,141,163,195,031
49338 :173,160,195,074,010,133,163
49344 :251,169,000,133,252,006,235
49350 :251,038,252,006,251,038,010
49356 :252,169,048,024,101,252,026
49362 :133,252,173,163,195,073,175
49368 :255,141,166,195,160,000,109
49374 :169,018,032,210,255,177,059
49380 :251,045,163,195,141,162,161
49386 :195,162,000,173,163,195,098
49392 :201,015,240,012,078,162,180
49398 :195,078,162,195,078,162,092
49404 :195,078,162,195,173,162,193
49410 :195,041,008,240,005,169,148
49416 :001,076,014,193,169,000,205
49422 :032,146,193,141,134,002,150
49428 :169,207,032,210,255,173,042
49434 :162,195,041,004,240,005,161
49440 :169,001,076,039,193,169,167
49446 :000,232,032,146,193,141,014
49452 :134,002,169,207,032,210,030
49458 :255,173,162,195,041,002,110
49464 :240,005,169,001,076,065,100
49470 :193,169,000,232,032,146,066
49476 :193,141,134,002,169,207,146
49482 :032,210,255,173,162,195,077
49488 :041,001,240,005,169,001,025
49494 :076,091,193,169,000,232,079
49500 :032,146,193,141,134,002,228
49506 :169,207,032,210,255,169,116
49512 :013,032,210,255,173,163,182
49518 :195,201,015,240,012,014,019
49524 :162,195,014,162,195,014,090
49530 :162,195,014,162,195,177,003
49536 :251,045,166,195,013,162,192
49542 :195,145,251,200,192,008,101
49548 :240,003,076,222,192,096,201
49554 :141,164,195,140,169,195,126
49560 :173,170,195,240,008,169,083
49566 :000,141,164,195,141,162,193
49572 :195,173,172,195,240,006,121
49578 :173,162,195,153,178,002,009
49584 :173,173,195,240,006,185,124
49590 :178,002,141,162,195,204,040
49596 :061,003,208,106,236,060,094
49602 :003,208,101,238,062,003,041
49608 :173,000,220,041,016,208,090
49614 :067,205,063,003,240,065,081
49620 :141,063,003,169,004,056,136
49626 :237,060,003,168,169,001,088
49632 :136,240,004,010,076,224,146
49638 :193,141,168,195,073,255,231
49644 :141,167,195,173,162,195,245
49650 :045,168,195,208,015,173,022
49656 :162,195,045,167,195,013,001
49662 :168,195,141,162,195,076,167
49668 :021,194,173,162,195,045,026
49674 :167,195,141,162,195,076,178
49680 :021,194,141,063,003,173,099
49686 :062,003,201,050,144,014,240
```

```
49692 :201,100,144,005,169,000,135
49698 :141,062,003,169,014,141,052
49704 :164,195,173,164,195,172,079
49710 :169,195,096,206,165,195,048
49716 :208,065,173,000,220,041,247
49722 :015,141,162,195,041,001,101
49728 :208,003,206,061,003,173,206
49734 :162,195,041,002,208,003,169
49740 :238,061,003,173,162,195,140
49746 :041,004,208,003,206,060,092
49752 :003,173,162,195,041,008,158
49758 :208,003,238,060,003,173,011
49764 :162,195,201,015,240,008,153
49770 :169,051,141,062,003,032,052
49776 :120,194,169,005,141,165,138
49782 :195,096,173,060,003,201,078
49788 :255,208,008,169,003,141,140
49794 :060,003,206,160,195,173,159
49800 :060,003,201,004,208,008,108
49806 :169,000,141,060,003,238,241
49812 :160,195,173,061,003,201,173
49818 :255,208,014,169,007,141,180
49824 :061,003,173,160,195,056,040
49830 :233,064,141,160,195,173,108
49836 :061,003,201,008,208,014,155
49842 :169,000,141,061,003,173,213
49848 :160,195,024,105,064,141,105
49854 :160,195,169,016,141,063,166
49860 :003,096,173,160,195,074,129
49866 :074,074,074,074,074,141,201
49872 :053,003,173,160,195,041,065
49878 :063,141,052,003,173,053,187
49884 :003,010,010,010,024,105,126
49890 :153,141,001,208,173,052,186
49896 :003,010,010,024,105,055,183
49902 :141,000,208,169,000,042,030
49908 :141,016,208,096,169,000,106
49914 :141,170,195,141,171,195,239
49920 :141,172,195,141,173,195,249
49926 :032,228,255,208,001,096,058
49932 :201,147,208,006,169,001,232
49938 :141,170,195,096,201,019,072
49944 :208,006,169,000,141,160,196
49950 :195,096,201,157,208,008,127
49956 :169,255,141,060,003,076,228
49962 :120,194,201,029,208,008,034
49968 :169,004,141,060,003,076,245
49974 :120,194,201,145,208,008,162
49980 :169,255,141,061,003,076,253
49986 :120,194,201,017,208,008,046
49992 :169,008,141,061,003,076,018
49998 :120,194,201,088,208,003,124
50004 :076,124,195,201,133,208,253
50010 :006,169,001,141,172,195,006
50016 :096,201,136,208,006,169,144
50022 :001,141,173,195,096,201,141
50028 :083,208,004,032,125,197,245
50034 :096,201,076,208,004,032,219
50040 :046,197,096,096,169,075,031
50046 :133,251,169,018,133,252,058
50052 :169,000,133,253,169,048,136
50058 :133,254,160,000,177,253,091
50064 :145,251,200,208,249,230,147
50070 :252,230,254,165,252,201,224
50076 :023,208,239,000,000,000,114
50082 :000,000,000,000,000,000,162
50088 :000,000,000,000,000,000,168
50094 :158,029,029,029,029,029,221
50100 :029,029,029,029,029,029,098
50106 :029,029,029,029,029,067,142
50112 :085,083,084,079,077,045,133
50118 :056,048,013,144,029,029,005
50124 :029,029,029,029,067,076,207
50130 :082,032,045,032,067,076,032
50136 :069,065,082,032,067,085,104
50142 :082,082,069,078,084,032,137
50148 :067,072,065,082,065,067,134
50154 :084,069,082,013,029,029,028
50160 :029,029,029,029,072,079,251
50166 :077,069,032,045,032,071,060
50172 :079,032,084,079,032,070,116
50178 :073,082,083,084,032,067,167
50184 :072,065,082,065,067,084,187
50190 :069,082,013,029,029,029,009
50196 :029,029,029,067,085,082,085
50202 :083,079,082,032,075,069,190
50208 :089,083,032,077,079,086,222
50214 :069,032,065,082,079,085,194
50220 :078,068,032,067,072,065,170
50226 :082,032,083,069,084,013,157
50232 :029,029,029,029,029,029,230
50238 :070,049,032,045,032,083,117
50244 :084,079,082,069,032,067,225
50250 :072,065,082,065,067,084,253
50256 :069,082,032,073,078,032,190
50262 :066,085,070,070,069,082,016
50268 :013,029,029,029,029,029,250
50274 :029,070,055,032,045,032,105
50280 :071,069,084,032,067,072,243
50286 :065,082,065,067,084,069,030
50292 :082,032,070,082,079,077,026
50298 :032,066,085,070,070,069,002
50304 :082,013,029,029,029,029,083
50310 :029,029,088,032,045,032,133
50316 :080,085,084,032,082,069,060
50322 :068,069,070,073,078,069,061
50328 :068,032,067,072,065,082,026
50334 :065,067,084,069,082,083,096
50340 :032,073,078,013,029,029,162
50346 :029,029,029,029,032,032,094
50352 :083,067,082,069,069,078,112
50358 :032,056,048,013,029,029,133
50364 :029,029,029,029,074,079,201
50370 :089,083,084,073,067,075,153
50376 :032,067,079,078,084,082,110
50382 :079,076,083,032,067,085,116
50388 :082,083,079,082,032,077,135
50394 :079,086,069,077,069,078,164
50400 :084,013,029,029,029,029,181
50406 :029,029,032,032,065,082,243
50412 :079,085,078,068,032,069,135
50418 :088,080,065,078,068,069,178
50424 :068,032,067,072,065,082,122
50430 :065,067,084,069,082,032,141
50436 :065,078,068,013,029,029,030
50442 :029,029,029,029,032,032,190
50448 :066,085,084,084,079,078,236
50454 :032,083,069,084,083,032,149
50460 :065,078,068,032,082,069,166
50466 :083,069,084,083,032,080,209
50472 :073,088,069,076,083,000,173
50478 :032,224,197,008,173,215,127
50484 :198,208,002,040,096,040,124
50490 :176,031,169,008,170,160,004
50496 :000,032,186,255,173,215,157
50502 :198,162,199,160,198,032,251
50508 :189,255,169,000,162,000,083
50514 :160,048,032,213,255,032,054
50520 :234,198,096,032,203,199,026
50526 :169,008,162,001,160,000,082
```

```
50532 :032,186,255,173,215,198,135
50538 :162,199,160,198,032,189,022
50544 :255,169,000,170,160,048,146
50550 :032,213,255,032,236,199,061
50556 :096,032,224,197,008,173,086
50562 :215,198,208,002,040,096,121
50568 :040,176,042,032,045,199,158
50574 :169,008,170,160,255,032,168
50580 :186,255,173,215,198,162,057
50586 :199,160,198,032,189,255,163
50592 :169,048,133,252,169,000,163
50598 :133,251,169,251,162,000,108
50604 :160,056,032,216,255,032,155
50610 :234,198,096,032,203,199,116
50616 :169,008,162,001,160,000,172
50622 :032,186,255,173,215,198,225
50628 :162,199,160,198,032,189,112
50634 :255,169,048,133,252,169,204
50640 :000,133,251,169,251,162,150
50646 :000,160,056,032,216,255,165
50652 :032,236,199,096,160,000,175
50658 :162,011,024,032,240,255,182
50664 :169,032,162,040,032,210,109
50670 :255,202,208,250,160,000,033
50676 :162,011,024,032,240,255,200
50682 :162,000,189,192,198,032,255
50688 :210,255,232,224,007,208,112
50694 :245,162,000,169,164,032,010
50700 :210,255,138,072,032,228,179
50706 :255,168,104,170,152,201,044
50712 :000,240,243,201,020,240,200
50718 :042,201,034,240,235,201,215
50724 :013,240,065,201,032,144,219
50730 :227,201,128,176,223,224,197
50736 :016,240,219,157,199,198,053
50742 :232,072,169,157,032,210,158
50748 :255,104,032,210,255,169,061
50754 :164,032,210,255,076,014,049
50760 :198,224,000,240,193,169,072
50766 :157,032,210,255,169,032,165
50772 :032,210,255,169,157,032,171
50778 :210,255,032,210,255,202,230
50784 :169,164,032,210,255,076,234
50790 :014,198,142,215,198,160,005
50796 :000,162,011,024,032,240,065
50802 :255,162,017,169,032,032,013
50808 :210,255,202,208,250,174,139
50814 :215,198,208,001,096,160,236
50820 :000,162,011,024,032,240,089
50826 :255,162,000,189,216,198,134
50832 :032,210,255,232,224,018,091
50838 :208,245,032,228,255,240,078
50844 :251,201,068,240,009,201,102
50850 :084,208,243,056,008,076,069
50856 :172,198,024,008,160,000,218
50862 :162,011,024,032,240,255,130
50868 :162,017,169,032,032,210,034
50874 :255,202,208,250,040,096,213
50880 :159,078,065,077,069,058,186
50886 :155,000,000,000,000,000,097
50892 :000,000,000,000,000,000,204
50898 :000,000,000,000,000,000,210
50904 :153,018,084,146,065,080,250
50910 :069,032,079,082,032,018,022
50916 :068,146,073,083,075,063,224
50922 :032,183,255,041,191,208,120
50928 :001,096,162,011,160,000,158
50934 :024,032,240,255,169,018,216
50940 :032,210,255,169,150,032,076
50946 :210,255,169,000,032,189,089
50952 :255,169,015,162,008,160,009
50958 :015,032,186,255,032,192,214
50964 :255,162,015,032,198,255,169
50970 :032,207,255,032,210,255,249
50976 :201,013,208,246,169,015,116
50982 :032,195,255,032,204,255,243
50988 :096,169,002,160,199,162,064
50994 :071,032,189,255,169,015,013
51000 :168,162,008,032,186,255,099
51006 :032,192,255,169,015,032,245
51012 :195,255,096,073,048,120,087
51018 :169,127,141,013,220,169,145
51024 :001,141,026,208,173,060,177
51030 :003,141,018,208,169,027,140
51036 :141,017,208,169,199,141,199
51042 :021,003,169,250,141,020,190
51048 :003,088,169,147,032,210,241
51054 :255,160,000,169,195,133,254
51060 :252,169,174,133,251,177,248
51066 :251,240,011,032,210,255,097
51072 :200,208,246,230,252,076,060
51078 :121,199,169,008,133,251,247
51084 :169,006,133,252,165,251,092
51090 :133,253,165,252,024,105,054
51096 :212,133,254,162,000,160,049
51102 :004,138,145,251,169,000,097
51108 :145,253,232,200,192,036,198
51114 :208,243,165,251,024,105,142
51120 :040,133,251,165,252,105,098
51126 :000,133,252,165,253,024,241
51132 :105,040,133,253,165,254,114
51138 :105,000,133,254,224,128,014
51144 :208,211,096,120,169,000,236
51150 :141,026,208,169,255,141,122
51156 :013,220,169,049,141,020,056
51162 :003,169,234,141,021,003,021
51168 :169,000,141,021,208,088,083
51174 :169,147,032,210,255,096,115
51180 :032,073,199,169,001,141,083
51186 :021,208,169,004,141,136,153
51192 :002,096,173,018,208,201,178
51198 :146,208,021,169,000,141,171
51204 :018,208,169,028,141,024,080
51210 :208,169,001,141,025,208,250
51216 :104,168,104,170,104,064,218
51222 :169,146,141,018,208,169,105
51228 :021,141,024,208,169,001,080
51234 :141,025,208,076,049,234,255
51240 :000,000,000,000,000,000,040
```

Program 3: Custom Character Loader

```
10 INPUT"FILENAME:";N$,D                :rem 205
20 F$=N$:ZK=PEEK(53)+256*PEEK(54)-LEN(F$)
   :POKE 782,ZK/256                     :rem 180
25 POKE781,ZK-PEEK(782)*256:POKE780,LEN(F
   $):SYS65469                          :rem 39
30 POKE780,1:POKE781,D:POKE782,0:SYS65466
                                        :rem 177
40 POKE780,0:POKE781,222:POKE782,169:SYS6
   5493                                 :rem 115
50 CLOSE1:PRINT:PRINT"{CLR}"CHR$(142)
                                        :rem 90
```

BEFORE TYPING...

Before typing in programs, please refer to "How To Type COMPUTE!'s Gazette Programs," "A Beginner's Guide To Typing In Programs," and "The Automatic Proofreader" that appear before the Program Listings.

Treasure Hunt

(Article on page 110.)

Program 1: Treasure Hunt—

VIC Version

Memory expansion required.

```
20 IFPEEK(44)<32THENPOKE56,28:POKE52,28
                                  :rem 53
25 PRINT"{RVS}{CLR}{7 DOWN}{YEL}****TREAS
   URE{2 SPACES}HUNT****"         :rem 236
26 PRINT"{3 DOWN}{PUR}{RIGHT}DEFINING
   {2 SPACES}CHARACTERS"          :rem 73
30 FORI=7168TO7679:POKEI,PEEK(25600+I):NE
   XT                             :rem 100
35 FORX=832TO936:READA:POKEX,A:NEXT:rem 4
40 FORI=7168+35*8TO7168+47*8+7:READA:POKE
   I,A:NEXT                       :rem 158
41 FORI=7168+58*8TO7168+61*8+7:READA:POKE
   I,A:NEXT                       :rem 160
45 GOSUB800                       :rem 129
49 GOSUB200                       :rem 127
50 SH=36876:SL=36875:V=36878:POKE36869,PE
   EK(36869)AND240OR15:RN=RN+1:P=1:Q=22
                                  :rem 29
55 GOSUB510:GOSUB1000:GOSUB1300   :rem 121
60 CL=8118                         :rem 4
65 SYS832                          :rem 7
70 TL=CL:Z=CL:ONPEEK(830)GOSUB91,92,93,94
   ,95,96,97,98                   :rem 59
75 CL=Z:POKESL,220:POKEV,5:GOSUB300
                                  :rem 199
76 IFFLANDNOTDFTHENGOSUB650       :rem 38
77 POKESH,.:POKESL,.:POKEV,.:IFDFTHEN1600
                                  :rem 86
79 POKETL,32:POKETL+Q,32          :rem 230
80 POKECL,46:POKECL+Q,47:POKECM+CL,3:POKE
   CM+Q+CL,3:GOSUB540:IFGC<2ANDGB<2THEN50
                                  :rem 217
85 GOSUB600:IFDFTHEN1600          :rem 204
90 GOTO65                          :rem 13
91 Z=Z-Q:RETURN                   :rem 243
92 Z=Z-21:RETURN                    :rem 6
93 Z=Z+P:RETURN                   :rem 242
94 Z=Z+23:RETURN                    :rem 8
95 Z=Z+Q:RETURN                   :rem 245
96 Z=Z+21:RETURN                    :rem 8
97 Z=Z-P:RETURN                   :rem 248
98 Z=Z-23:RETURN                   :rem 14
200 PRINT"{DOWN}{RVS}{CYN}ENTER SKILL LEV
    EL 1-5"                       :rem 45
210 GETA$:IFA$=""THEN210          :rem 73
220 AA=VAL(A$):IFAA<1ORAA>5THENPRINT"
    {2 UP}";:GOTO200              :rem 156
230 RETURN                        :rem 117
300 REM COLLISION CHK             :rem 249
302 TC=0:IFPEEK(CL)=35ORPEEK(CL+Q)=35ORPE
    EK(CL)=36ORPEEK(CL+Q)=36THENTC=P
                                  :rem 58
303 IFPEEK(CL)=37ORPEEK(CL+Q)=37ORPEEK(CL
    )=38ORPEEK(CL+Q)=38THENTC=P    :rem 5
304 IFPEEK(CL)=39ORPEEK(CL+Q)=39ORPEEK(CL
    )=40ORPEEK(CL+Q)=40THENTC=P   :rem 252
305 IFPEEK(CL)=41ORPEEK(CL+Q)=41THENTC=P
                                  :rem 229
306 IFTCTHENCL=TL:RETURN          :rem 116
308 IF(PEEK(CL)=47ANDPEEK(CL+Q)=32)OR(PEE
    K(CL)=32ANDPEEK(CL+Q)=46)THENRETURN
                                  :rem 193
309 IFPEEK(CL)=32ANDPEEK(CL+Q)=32THENRETU
    RN                            :rem 215
310 IFPEEK(CL)=46ANDPEEK(CL+Q)=47THENRETU
    RN                            :rem 218
312 IFPEEK(CL)=42ORPEEK(CL+Q)=42THEN400
                                   :rem 85
315 IFPEEK(CL)=61ORPEEK(CL+Q)=61THENGB=GB
    -P:SC=SC+10*AA:POKESL,.:POKESH,180:PO
    KEV,15                        :rem 233
320 IFPEEK(CL)=60ORPEEK(CL+Q)=60THENGC=GC
    -P:SC=SC+AA:POKESL,.:POKESH,240:POKEV
    ,15                            :rem 87
323 IFPEEK(CL)=43ORPEEK(CL+Q)=43ORPEEK(CL
    )=44ORPEEK(CL+Q)=44THEN450    :rem 112
325 IFPEEK(CL)=45ORPEEK(CL+Q)=45THEN650
                                  :rem 102
330 RETURN                        :rem 118
400 REMDEAD FROM SKUL             :rem 249
405 POKETL,32:POKETL+Q,32:POKECL,46:POKEC
    L+Q,47:POKECL+CM,0:POKECL+CM+Q,0:GOTO
    700                            :rem 76
450 REM AT BOG                    :rem 234
452 D=INT(RND(1)*10)+1            :rem 165
455 IFPEEK(831)<>0ANDPEEK(CJ)=59ANDD>2THE
    NCL=CH:POKECI,32:SC=SC+100*AA:FG=P:RE
    TURN                          :rem 118
460 IFPEEK(831)<>0ANDPEEK(CJ)<>59ANDD>2TH
    ENCL=CL+47:GOTO300            :rem 80
465 CL=CH+23:POKETL,32:POKETL+Q,32:POKECL
    ,46:POKECM+CL,3:GOTO700       :rem 224
500 REM BORDER                     :rem 55
510 PRINT"{CLR}":CM=30720         :rem 253
520 FORI=7680TO7701:POKEI,35:POKECM+I,3:N
    EXT                           :rem 214
525 FORI=7723TO8185STEP22:POKEI,35:POKECM
    +I,3:NEXT                     :rem 128
530 FORI=8162TO8142STEP-1:POKEI,35:POKECM
    +I,3:NEXT                     :rem 109
535 FORI=8164TO7702STEP-22:POKEI,35:POKEC
    M+I,3:NEXT                    :rem 168
536 RETURN                        :rem 126
539 REM SCOR+TRES CHST             :rem 87
540 PRINT"{HOME}{22 DOWN}{RIGHT}{CYN}ROUN
    D"RN"SCORE"SC;:IFFGTHEN555    :rem 140
550 ON(INT(RND(1)*15))GOSUB555,555,555,55
    5,555,555,561,555,555,555,555,555,563
    ,555,555                      :rem 239
555 RETURN                        :rem 127
561 IFPEEK(CI)=32ANDPEEK(CJ)=32THENPOKECI
    ,58:POKECJ,59:POKECI+CM,6:POKECJ+CM,6
                                  :rem 127
562 RETURN                        :rem 125
563 IFPEEK(CI)=58THENPOKECI,32:POKECJ,32
                                  :rem 219
564 RETURN                        :rem 127
600 REMMOVESKULLS                 :rem 143
605 D=INT(RND(1)*AA)+1:ONDGOSUB641,642,64
    3,644,645                      :rem 34
608 TS=SK:Z=SK:POKESK,32           :rem 39
610 ONINT(RND(1)*8)+1GOSUB91,92,93,94,95,
    96,97,98                      :rem 180
615 SK=Z:IFPEEK(SK)=32THENONDGOSUB1381,13
    82,1383,1384,1385:GOTO625     :rem 171
620 IFPEEK(SK)=46ORPEEK(SK)=47THEN400
                                   :rem 2
622 SK=TS                          :rem 28
625 POKESK,42:POKESK+CM,7:RETURN   :rem 59
641 SK=S1:RETURN                   :rem 20
642 SK=S2:RETURN                   :rem 22
643 SK=S3:RETURN                   :rem 24
644 SK=S4:RETURN                   :rem 26
```

```
645 SK=S5:RETURN                          :rem 28
650 REMSTAGGER                            :rem 140
655 POKETL,32:POKETL+Q,32:POKECL,46:POKEC
    L+Q,47:POKECL+CM,4:POKECL+CM+Q,4
                                          :rem 81
660 POKEV,5:POKESH,0:FORX=1TO3:POKESL,200
    :FORL=1TO10:NEXT:POKESL,244:FORL=1TO2
    5:NEXT                                :rem 233
662 POKESL,0:FORL=1TO200:NEXT:NEXT
                                          :rem 202
665 TL=CL:Z=CL:ONINT(RND(1)*8)+1GOSUB91,9
    2,93,94,95,96,97,98                   :rem 196
670 CL=Z:IFFL=.THENFL=P:GOTO300           :rem 221
680 IFFL=PTHENFL=.:GOTO300                :rem 126
700 REMDEADSOUND                          :rem 18
710 POKEV,5:POKESH,0:FORN=1TO5:POKESL,255
    :FORX=1TO200:NEXT:POKESL,180:FORX=1TO
    100:NEXT                              :rem 89
720 POKESL,0:FORX=1TO200:NEXT:NEXT:DF=1:R
    ETURN                                 :rem 29
800 PRINT"{CLR}{RVS}{CYN}YOU ARE HUNTING
    {SPACE}LOST{2 SPACES}PIRATE TREASURE
    {SPACE}ON{4 SPACES}A SECLUDED ISLAND.
    {4 SPACES}";                          :rem 92
820 PRINT"{RVS}{BLU}GUIDE THE TREASURE
    {4 SPACES}HUNTER WITH A JOYSTICKAND C
    OLLECT THE GOLD{2 SPACES}";           :rem 52
830 PRINT"{RVS}{BLU}COINS, GOLD BARS AND
    {2 SPACES}A TREASURE CHEST. THE TREAS
    URE CHEST IS{5 SPACES}";              :rem 81
832 PRINT"{RVS}SURROUNDED BY A BOG.
    {2 SPACES}";                          :rem 168
835 PRINT"YOU HAVE A 70% CHANCE OF CROSSI
    NG BY PRESS- ING THE FIRE BUTTON AS";
                                          :rem 227
838 PRINT"YOU APPROACH.{9 SPACES}";
                                          :rem 40
839 PRINT"{RVS}{PUR}THE TREASURE HAS A
    {4 SPACES}DEATH CURSE ON IT AND IS GU
    ARDED BY THE EVILSPIRITS";            :rem 46
840 PRINT" OF ANCIENT{4 SPACES}PIRATES WH
    O MAKE THE{2 SPACES}TREASURE CHEST
    {8 SPACES}DISAPPEAR AND";             :rem 146
841 PRINT" REAPPEAR FROM TIME TO TIME.
    {3 SPACES}";                          :rem 92
843 PRINT"{RVS}{GRN}PRESS ANY KEY";
                                          :rem 57
844 GETA$:IFA$=""THEN844                  :rem 99
855 PRINT"{CLR}{BLU}THE EVIL SPIRITS ALSO
     HAVE PLACED KEGS OF{3 SPACES}RUM ARO
    UND THE ISLAND ";                     :rem 179
856 PRINT"TO DISTRACT TREASURE{2 SPACES}H
    UNTERS.{14 SPACES}";                  :rem 114
857 PRINT"{PUR}IF THE TREASURE HUNTERDRIN
    KS RUM, HE WILL{3 SPACES}STAGGER AND
    {SPACE}YOU CAN'T GUIDE ";             :rem 69
858 PRINT"HIM.{12 SPACES}";               :rem 189
860 PRINT"{RVS}{RED}YOU EARN POINTS AS
    {4 SPACES}SHOWN:{16 SPACES}";         :rem 53
865 PRINT"{DOWN}COIN=1 X SKILL LEVEL
    {2 SPACES}{DOWN}BAR=10 X SKILL LEVEL
    {2 SPACES}{DOWN}TREASURE CHEST=
    {7 SPACES}";                          :rem 169
870 PRINT"{5 SPACES}100 X SKILL LEVEL";
                                          :rem 139
880 RETURN                                :rem 128
999 REMBUILD PALMGROVE                    :rem 172
1000 TB=INT(RND(1)*8)+2:PRINT"{HOME}":I=I
     NT(RND(1)*3)+1:FORX=0TOI:PRINT"
     {DOWN}":NEXT                         :rem 149
1005 PRINTTAB(TB)"{GRN}&%'&%'"            :rem 87
1006 PRINTTAB(TB)"($&%')&%'"              :rem 175
1010 PRINTTAB(TB)"{RIGHT}$($&%'$)"
                                          :rem 157
1015 PRINTTAB(TB)"{3 RIGHT}$($)$":rem 106
1020 PRINTTAB(TB)"{3 RIGHT}$ $ $" :rem 21
1023 PRINTTAB(TB)"{5 RIGHT}${BLK}
     {2 SPACES},+"                        :rem 241
1025 PRINTTAB(TB)"{GRN}{5 RIGHT}${BLU}:;
     {SPACE}{BLK},"                       :rem 122
1027 PRINTTAB(TB)"{8 RIGHT},+"            :rem 152
1028 PRINTTAB(TB)"{5 RIGHT}+,+," :rem 153
1030 CH=PEEK(209)+256*PEEK(210)-(88-(TB+7
     )):CJ=CH+Q:CI=CJ-P                   :rem 20
1035 RETURN                               :rem 169
1300 REM BUILD SCREEN                     :rem 216
1301 FORI=PTO7*AA                         :rem 8
1302 X=INT(RND(1)*396)+7702               :rem 196
1305 IFPEEK(X)<>32ORPEEK(X+P)<>32ORPEEK(X
     -P)<>32ORPEEK(X+Q)<>32ORPEEK(X-Q)<>3
     2THEN1302                            :rem 36
1310 POKEX,45:POKECM+X,4:NEXT             :rem 54
1320 FORI=PTO6*AA                         :rem 8
1325 X=INT(RND(1)*396)+7702               :rem 201
1330 IFPEEK(X)<>32ORPEEK(X+P)<>32ORPEEK(X
     -P)<>32ORPEEK(X+Q)<>32ORPEEK(X-Q)<>3
     2THEN1325                            :rem 39
1335 POKEX,42:POKECM+X,0:NEXT             :rem 54
1360 FORI=PTOAA                           :rem 172
1365 SK=INT(RND(1)*396)+7702:IFPEEK(SK)<>
     42THEN1365                           :rem 206
1370 POKESK+CM,7                          :rem 182
1375 ONIGOSUB1381,1382,1383,1384,1385
                                          :rem 241
1380 NEXT:GOTO1400                        :rem 67
1381 S1=SK:RETURN                         :rem 70
1382 S2=SK:RETURN                         :rem 72
1383 S3=SK:RETURN                         :rem 74
1384 S4=SK:RETURN                         :rem 76
1385 S5=SK:RETURN                         :rem 78
1400 GC=25:B=60:FORI=1TO25                :rem 244
1410 GOSUB1450                            :rem 16
1415 IFPEEK(X)<>32ORPEEK(X+Q)><32ORPEEK(X
     -Q)><32THEN1410                      :rem 146
1420 POKEX,B:POKECM+X,7:NEXT              :rem 20
1430 GB=5:B=61:FORI=1TO5                  :rem 147
1435 GOSUB1450                            :rem 23
1437 IFPEEK(X)<>32ORPEEK(X+Q)><32ORPEEK(X
     -Q)><32THEN1435                      :rem 157
1440 POKEX,B:POKECM+X,7:NEXT              :rem 22
1445 RETURN                               :rem 174
1450 X=INT(RND(1)*415)+7702:RETURN
                                          :rem 218
1600 REMGAMEOVER                          :rem 1
1605 POKESH,0:POKESL,0:POKE36869,PEEK(368
     69)AND240OR0                         :rem 187
1610 PRINT"{CLR}{3 DOWN}{RVS}{YEL}ANOTHER
      VICTIM OF THE PIRATE'S CURSE!!!"
                                          :rem 134
1611 PRINT"{DOWN}{RVS}{RED}SKILL LEVEL:"A
     A                                    :rem 140
1612 PRINT"{DOWN}{RVS}{RED}ROUNDS:"RN
                                          :rem 143
1613 PRINT:PRINT"{RVS}{RED}SCORE:"SC
                                          :rem 221
1615 PRINT:PRINT"{RVS}{RED}PLAY AGAIN?"
                                          :rem 104
1620 GETA$:IFA$=""THEN1620                :rem 181
1625 IFA$="Y"THENRN=0:FL=0:FG=0:DF=0:SC=0
     :PRINT"{CLR}":GOTO49                 :rem 49
1630 END                                  :rem 161
6035 DATA120,8,72,152,72,138,72,173,19,14
```

```
     5,72,173,34,145,72,169,0,141,62,3,14
     1,63,3,169                    :rem 246
6040 DATA127,141,34,145,173,32,145,73,255
     ,41,128,42,8,169,195,141,19,145,173,
     17,145,73                     :rem 207
6045 DATA255,41,60,74,74,40,42,168,41,16,
     201,16,208,3,141,63,3,152,41,15,162,
     0,232,224                     :rem 163
6050 DATA9,240,8,221,160,3,208,246,142,62
     ,3,104,141,34,145,104,141,19,145,104
     ,170,104                      :rem 108
6055 DATA168,104,40,88,96,2,3,1,5,4,12,8,
     10                            :rem 108
6060 DATA255,255,255,255,255,255,255,255,
     28,28,28,28,28,28,28,28       :rem 170
6065 DATA129,227,247,255,255,255,255,156,
     15,127,127,227,143,63,113,243
                                   :rem 195
6070 DATA248,254,254,227,249,252,207,227,
     199,159,62,56,120,96,96,64,121,120
                                   :rem 197
6075 DATA28,14,6,6,7,60,126,90,126,126,36
     ,60,24,28,63,31,255,255,127,124,56
                                   :rem 168
6080 DATA60,254,255,255,243,120,56,60,28,
     62,62,62,62,62,62,28          :rem 246
6085 DATA60,126,219,255,102,60,24,255,189
     ,189,189,189,36,36,36,102     :rem 9
6087 DATA15,31,63,63,63,63,63,63,248,252,
     254,254,254,254,254,254       :rem 156
6090 DATA,,,24,24,,,,,,,127,127,,,0
                                   :rem 173
```

Program 2:

Treasure Hunt—64 Version

```
10 POKE53280,6:POKE53281,1         :rem 189
21 POKE56,48:POKE52,48:CLR         :rem 24
25 PRINT"{CLR}{7 DOWN}{13}*************
   {RVS}TREASURE HUNT{OFF}**************"
                                   :rem 127
26 PRINT"{7 DOWN}{BLU}{9 SPACES}REDEFININ
   G CHARACTERS"                   :rem 138
28 POKE56334,PEEK(56334)AND254:POKE1,PEEK
   (1)AND251                       :rem 139
30 FORI=0TO511:POKEI+12288,PEEK(53248+I):
   NEXT                            :rem 177
32 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)
   OR1                             :rem 84
40 FORI=12288+35*8TO12288+47*8+7:READA:PO
   KEI,A:NEXT                      :rem 252
41 FORI=12288+58*8TO12288+61*8+7:READA:PO
   KEI,A:NEXT                      :rem 254
45 POKE53272,21:GOSUB800:SN=54272:POKESN+
   24,15:POKESN+5,17:POKESN+6,240:rem 240
49 POKESN,100:GOSUB200             :rem 70
50 POKE53272,(PEEK(53272)AND240)OR12:RN=R
   N+1:P=1:Q=40                    :rem 49
55 GOSUB510:GOSUB1000:GOSUB1300    :rem 121
60 CL=1902                         :rem 254
65 JS=PEEK(56320)AND15             :rem 244
70 TL=CL:Z=CL:ONJS-4GOSUB94,92,93,99,96,9
   8,97,99,95,91,99                :rem 2
75 CL=Z:POKESN+1,50:POKESN+4,33:GOSUB300
                                   :rem 209
76 IFFLANDNOTDFTHENGOSUB650        :rem 38
77 POKESN+4,32:IFDFTHEN1600        :rem 119
79 POKETL,32:POKETL+Q,32           :rem 230
80 POKECL,46:POKECL+Q,47:POKECM+CL,3:POKE
   CM+Q+CL,3:GOSUB540:IFGC<2ANDGB<2THEN50
                                   :rem 217
85 GOSUB600:IFDFTHEN1600           :rem 204
90 GOTO65                          :rem 13
91 Z=Z-Q:RETURN                    :rem 243
92 Z=Z-39:RETURN                   :rem 15
93 Z=Z+P:RETURN                    :rem 242
94 Z=Z+41:RETURN                   :rem 8
95 Z=Z+Q:RETURN                    :rem 245
96 Z=Z+39:RETURN                   :rem 17
97 Z=Z-P:RETURN                    :rem 248
98 Z=Z-41:RETURN                   :rem 14
99 RETURN                          :rem 82
200 PRINT"{DOWN}{RVS}{5}ENTER SKILL LEV
   EL 1-5"                         :rem 38
210 GETA$:IFA$=""THEN210           :rem 73
220 AA=VAL(A$):IFAA<1ORAA>5THEN210
                                   :rem 252
230 RETURN                         :rem 117
300 REM COLLISION CHK              :rem 249
302 TC=0:IFPEEK(CL)=35ORPEEK(CL+Q)=35ORPE
   EK(CL)=36ORPEEK(CL+Q)=36THENTC=P
                                   :rem 58
303 IFPEEK(CL)=37ORPEEK(CL+Q)=37ORPEEK(CL
   )=38ORPEEK(CL+Q)=38THENTC=P     :rem 5
304 IFPEEK(CL)=39ORPEEK(CL+Q)=39ORPEEK(CL
   )=40ORPEEK(CL+Q)=40THENTC=P     :rem 252
305 IFPEEK(CL)=41ORPEEK(CL+Q)=41THENTC=P
                                   :rem 229
306 IFTCTHENCL=TL:RETURN           :rem 116
308 IF(PEEK(CL)=47ANDPEEK(CL+Q)=32)OR(PEE
   K(CL)=32ANDPEEK(CL+Q)=46)THENRETURN
                                   :rem 193
309 IFPEEK(CL)=32ANDPEEK(CL+Q)=32THENRETU
   RN                              :rem 215
310 IFPEEK(CL)=46ANDPEEK(CL+Q)=47THENRETU
   RN                              :rem 218
312 IFPEEK(CL)=42ORPEEK(CL+Q)=42THEN400
                                   :rem 85
315 IFPEEK(CL)=61ORPEEK(CL+Q)=61THENGB=GB
   -P:SC=SC+10*AA:POKESN+1,30:POKESN+4,3
   3                               :rem 93
320 IFPEEK(CL)=60ORPEEK(CL+Q)=60THENGC=GC
   -P:SC=SC+AA:POKESN+1,80:POKESN+4,33
                                   :rem 211
323 IFPEEK(CL)=43ORPEEK(CL+Q)=43ORPEEK(CL
   )=44ORPEEK(CL+Q)=44THEN450      :rem 112
325 IFPEEK(CL)=45ORPEEK(CL+Q)=45THEN650
                                   :rem 102
330 RETURN                         :rem 118
400 REM DEAD FROM SKULL            :rem 69
405 POKETL,32:POKETL+Q,32:POKECL,46:POKEC
   L+Q,47:POKECL+CM,0:POKECL+CM+Q,0
                                   :rem 66
410 GOTO700                        :rem 101
450 REM AT BOG                     :rem 234
452 D=INT(RND(1)*10)+1             :rem 165
453 JB=NOT(-(PEEK(56320)AND16)/16)
                                   :rem 107
455 IFJBANDPEEK(CJ)=59ANDD>2THENCL=CH:POK
   ECI,32:SC=SC+100*AA:FG=P:RETURN
                                   :rem 70
460 IFJBANDPEEK(CJ)<>59ANDD>2THENCL=CL+47
   :GOTO300                        :rem 32
465 CL=CH+41:POKETL,32:POKETL+Q,32:POKECL
   ,46:POKECM+CL,3:GOTO700         :rem 224
500 REM BORDER                     :rem 55
510 PRINT"{CLR}":CM=54272          :rem 5
520 FORI=1024TO1063:POKEI,35:POKECM+I,14:
   NEXT                            :rem 245
525 FORI=1103TO2023STEP40:POKEI,35:POKECM
   +I,14:NEXT                      :rem 149
```

```
530 FORI=1982TO1943STEP-1:POKEI,35:POKECM
    +I,14:NEXT                          :rem 164
535 FORI=1984TO1064STEP-40:POKEI,35:POKEC
    M+I,14:NEXT                         :rem 216
536 RETURN                              :rem 126
539 REM SCOR+TRES CHST                  :rem 87
540 PRINT"{HOME}{24 DOWN}{RIGHT}{CYN}ROUN
    D"RN"SCORE"SC;:IFFGTHEN555          :rem 174
545 RD=INT(RND(1)*15)                   :rem 163
550 ONRDGOSUB555,555,555,555,555,555,561,
    555,555,555,555,555,563,555,555:rem 2
555 RETURN                              :rem 127
561 IFPEEK(CI)=32ANDPEEK(CJ)=32THENPOKECI
    ,58:POKECJ,59:POKECI+CM,6:POKECJ+CM,6
                                        :rem 127
562 RETURN                              :rem 125
563 IFPEEK(CI)=58THENPOKECI,32:POKECJ,32
                                        :rem 219
564 RETURN                              :rem 127
600 REMMOVESKULLS                       :rem 143
605 D=INT(RND(1)*AA)+1:ONDGOSUB641,642,64
    3,644,645                           :rem 34
608 TS=SK:Z=SK:POKESK,32                :rem 39
610 ONINT(RND(1)*8)+1GOSUB91,92,93,94,95,
    96,97,98                            :rem 180
615 SK=Z:IFPEEK(SK)=32THENONDGOSUB1381,13
    82,1383,1384,1385:GOTO625           :rem 171
620 IFPEEK(SK)=46ORPEEK(SK)=47THEN400
                                        :rem 2
622 SK=TS                               :rem 28
625 POKESK,42:POKESK+CM,7:RETURN        :rem 59
641 SK=S1:RETURN                        :rem 20
642 SK=S2:RETURN                        :rem 22
643 SK=S3:RETURN                        :rem 24
644 SK=S4:RETURN                        :rem 26
645 SK=S5:RETURN                        :rem 28
650 REMSTAGGER                          :rem 140
655 POKETL,32:POKETL+Q,32:POKECL,46:POKEC
    L+Q{SCR UP}{RVS}7:POKECL+CM,4:POKECL+
    CM+Q,4                              :rem 28
660 POKESN+1,40:POKESN+4,33:FORI=1TO10:NE
    XT:POKESN+1,45:FORI=1TO25:NEXT
                                        :rem 254
662 POKESN+4,32                         :rem 94
665 TL=CL:Z=CL:ONINT(RND(1)*8)+1GOSUB91,9
    2,93,94,95,96,97,98                 :rem 196
670 CL=Z:IFFL=.THENFL=P:GOTO300         :rem 221
680 IFFL=PTHENFL=.:GOTO300              :rem 126
700 REMDEADSOUND                        :rem 18
710 POKESN+1,40:POKESN+4,33:FORX=1TO5:POK
    ESN+1,40:FORL=1TO50:NEXT:POKESN+1,20
                                        :rem 84
715 FORL=1TO50:NEXT:NEXT:POKESN+4,32
                                        :rem 50
720 DF=1:RETURN                         :rem 171
800 PRINT"{CLR}{RVS}{4}YOU ARE HUNTING
    {SPACE}LOST PIRATE TREASURE ON ";
                                        :rem 225
810 PRINT"A SECLUDED ISLAND.{22 SPACES}";
                                        :rem 24
820 PRINT"{BLU}GUIDE THE HUNTER WITH JOYS
    TICK #2 TO THE";                    :rem 111
830 PRINT"COINS, GOLD BARS AND A TREASURE
     CHEST.{2 SPACES}";                 :rem 193
831 PRINT"THE TREASURE CHEST IS SURROUNDE
    D BY A{3 SPACES}";                  :rem 238
832 PRINT"BOG.{2 SPACES}YOU HAVE A 70% CH
    ANCE OF CROSSING ";                 :rem 60
835 PRINT"BY PRESSING THE FIRE BUTTON AS
    {SPACE}YOU{6 SPACES}";              :rem 38
838 PRINT"{PUR}THE TREASURE HAS A DEATH C
    URSE ON IT AND";                    :rem 169
839 PRINT"IS GUARDED BY THE EVIL SPIRITS
    {SPACE}OF{7 SPACES}";               :rem 183
840 PRINT"ANCIENT PIRATES WHO MAKE THE TR
    EASURE{3 SPACES}";                  :rem 26
841 PRINT"CHEST DISAPPEAR AND REAPPEAR FR
    OM TIME{2 SPACES}";                 :rem 63
842 PRINT"TO TIME.{32 SPACES}";         :rem 170
854 PRINT"{GRN}THE EVIL SPIRITS ALSO HAVE
     PLACED KEGS{2 SPACES}";            :rem 48
855 PRINT"OF RUM AROUND THE ISLAND TO DIS
    TRACT{4 SPACES}";                   :rem 157
856 PRINT"TREASURE HUNTERS.{23 SPACES}";
                                        :rem 113
857 PRINT"{7}IF THE TREASURE HUNTER DRI
    NKS RUM, HE{3 SPACES}";             :rem 115
858 PRINT"WILL STAGGER AND YOU CAN'T GUID
    E HIM.{3 SPACES}";                  :rem 141
860 PRINT"{RED}YOU EARN POINTS AS SHOWN:
    {15 SPACES}";                       :rem 35
865 PRINT"COIN=1 X SKILL LEVEL{20 SPACES}
    ";                                  :rem 149
866 PRINT"BAR=10 X SKILL LEVEL{20 SPACES}
    ";                                  :rem 114
867 PRINT"TREASURE CHEST=100 X SKILL LEVE
    L{8 SPACES}";                       :rem 176
880 RETURN                              :rem 128
999 REMBUILD PALMGROVE                  :rem 172
1000 TB=INT(RND(1)*8)+2:PRINT"{HOME}":I=I
     NT(RND(1)*3)+1:FORX=0TOI:PRINT"
     {DOWN}":NEXT                       :rem 149
1005 PRINTTAB(TB)"{GRN}&%'&%'"          :rem 87
1006 PRINTTAB(TB)"($&%')&%'"            :rem 175
1010 PRINTTAB(TB)"{RIGHT}$($&%'$)"
                                        :rem 157
1015 PRINTTAB(TB)"{3 RIGHT}$($)$":rem 106
1020 PRINTTAB(TB)"{3 RIGHT}$ $ $" :rem 21
1023 PRINTTAB(TB)"{5 RIGHT}${BLK}
     {2 SPACES},+"                      :rem 241
1025 PRINTTAB(TB)"{GRN}{5 RIGHT}${BLU}:;
     {SPACE}{BLK},"                     :rem 122
1027 PRINTTAB(TB)"{8 RIGHT},+"          :rem 152
1028 PRINTTAB(TB)"{5 RIGHT}+,+," :rem 153
1030 CH=PEEK(209)+256*PEEK(210)-(160-(TB+
     7)):CJ=CH+Q:CI=CJ-P                :rem 59
1035 RETURN                             :rem 169
1300 REM BUILD SCREEN                   :rem 216
1301 FORI=PTO7*AA                       :rem 8
1302 X=INT(RND(1)*720)+1064             :rem 182
1305 IFPEEK(X)<>32ORPEEK(X+P)<>32ORPEEK(X
     -P)<>32ORPEEK(X+Q)<>32THEN1302
                                        :rem 88
1306 IFPEEK(X-Q)<>32THEN1302            :rem 121
1310 POKEX,45:POKECM+X,4:NEXT           :rem 54
1320 FORI=PTO6*AA                       :rem 8
1325 X=INT(RND(1)*720)+1064             :rem 187
1330 IFPEEK(X)<>32ORPEEK(X+P)<>32ORPEEK(X
     -P)<>32ORPEEK(X+Q)<>32THEN1325
                                        :rem 91
1331 IFPEEK(X-Q)<>32THEN1325            :rem 124
1335 POKEX,42:POKECM+X,0:NEXT           :rem 54
1360 FORI=PTOAA                         :rem 172
1365 SK=INT(RND(1)*720)+1064:IFPEEK(SK)<>
     42THEN1365                         :rem 192
1370 POKESK+CM,7                        :rem 182
1375 ONIGOSUB1381,1382,1383,1384,1385
                                        :rem 241
1380 NEXT:GOTO1400                      :rem 67
1381 S1=SK:RETURN                       :rem 70
```

```
1382 S2=SK:RETURN                          :rem 72
1383 S3=SK:RETURN                          :rem 74
1384 S4=SK:RETURN                          :rem 76
1385 S5=SK:RETURN                          :rem 78
1400 GC=25:B=60:FORI=1TO25                 :rem 244
1410 GOSUB1450                             :rem 16
1415 IFPEEK(X)<>32ORPEEK(X+Q)><32ORPEEK(X
     -Q)><32THEN1410                       :rem 146
1420 POKEX,B:POKECM+X,7:NEXT               :rem 20
1430 GB=5:B=61:FORI=1TO5                   :rem 147
1435 GOSUB1450                             :rem 23
1437 IFPEEK(X)<>32ORPEEK(X+Q)><32ORPEEK(X
     -Q)><32THEN1435                       :rem 157
1440 POKEX,B:POKECM+X,7:NEXT               :rem 22
1445 RETURN                                :rem 174
1450 X=INT(RND(1)*755)+1064:RETURN
                                           :rem 220
1600 REMGAMEOVER                           :rem 1
1605 POKESN+4,32:POKE53272,21              :rem 135
1610 PRINT"{CLR}{3 DOWN}{RVS}{YEL}ANOTHER
      VICTIM OF THE PIRATE'S CURSE!!!"
                                           :rem 134
1611 PRINT"{DOWN}{RVS}{RED}SKILL LEVEL:"A
     A                                     :rem 140
1612 PRINT"{DOWN}{RVS}{RED}ROUNDS:"RN
                                           :rem 143
1613 PRINT:PRINT"{RVS}{RED}SCORE:"SC
                                           :rem 221
1615 PRINT:PRINT"{RVS}{RED}PLAY AGAIN?"
                                           :rem 104
1620 GETA$:IFA$=""THEN1620                 :rem 181
1625 IFA$="Y"THENRN=0:FL=0:FG=0:DF=0:SC=0
     :PRINT"{CLR}":GOTO49                  :rem 49
1630 END                                   :rem 161
6060 DATA255,255,255,255,255,255,255,255,
     28,28,28,28,28,28,28,28               :rem 170
6065 DATA129,227,247,255,255,255,255,156,
     15,127,127,227,143,63,113,243
                                           :rem 195
6070 DATA248,254,254,227,249,252,207,227,
     199,159,62,56,120,96,96,64,121,120,2
     8                                     :rem 91
6071 DATA14,6,6,7,                         :rem 160
6075 DATA60,126,90,126,126,36,60,24,28,63
     ,31,255,255,127,124,56                :rem 90
6080 DATA60,254,255,255,243,120,56,60,28,
     62,62,62,62,62,62,28                  :rem 246
6085 DATA60,126,219,255,102,60,24,255,189
     ,189,189,189,36,36,36,102             :rem 9
6087 DATA15,31,63,63,63,63,63,248,252,
     254,254,254,254,254,254               :rem 156
6090 DATA,,,24,24,,,,,,,127,127,,,0
                                           :rem 173
```

MLX

(Article on page 145.)

BEFORE TYPING...

Before typing in programs, please refer to "How To Type COMPUTE!'s Gazette Programs," "A Beginner's Guide To Typing In Programs," and "The Automatic Proofreader" that appear before the Program Listings.

```
10 REM LINES CHANGED FROM MLX VERSION 2.0
   0 ARE 750,765,770 AND 860               :rem 50
100 PRINT"{CLR}{6}";CHR$(142);CHR$(8);:
    POKE53281,1:POKE53280,1                :rem 67
101 POKE 788,52:REM DISABLE RUN/STOP
                                           :rem 119
110 PRINT"{RVS}{39 SPACES}";               :rem 176
120 PRINT"{RVS}{14 SPACES}{RIGHT}{OFF}
    {*}£{RVS}{RIGHT} {RIGHT}{2 SPACES}
    {*}{OFF}{*}£{RVS}£{RVS}
    {14 SPACES}";                          :rem 250
130 PRINT"{RVS}{14 SPACES}{RIGHT} {G}
    {RIGHT} {2 RIGHT} {OFF}£{RVS}£{*}
    {OFF}{*}{RVS}{14 SPACES}";             :rem 35
140 PRINT"{RVS}{41 SPACES}"                :rem 120
200 PRINT"{2 DOWN}{PUR}{BLK} MACHINE LANG
    UAGE EDITOR VERSION 2.01{5 DOWN}"
                                           :rem 237
210 PRINT"{5}{2 UP}STARTING ADDRESS?
    {8 SPACES}{9 LEFT}";                   :rem 143
215 INPUTS:F=1-F:C$=CHR$(31+119*F)
                                           :rem 166
220 IFS<256OR(S>40960ANDS<49152)ORS>53247
    THENGOSUB3000:GOTO210                  :rem 235
225 PRINT:PRINT:PRINT                      :rem 180
230 PRINT"{5}{2 UP}ENDING ADDRESS?
    {8 SPACES}{9 LEFT}";:INPUTE:F=1-F:C$=
    CHR$(31+119*F)                         :rem 20
240 IFE<256OR(E>40960ANDE<49152)ORE>53247
    THENGOSUB3000:GOTO230                  :rem 183
250 IFE<STHENPRINTC$;"{RVS}ENDING < START
    {2 SPACES}":GOSUB1000:GOTO 230
                                           :rem 176
260 PRINT:PRINT:PRINT                      :rem 179
300 PRINT"{CLR}";CHR$(14):AD=S:POKEV+21,0
                                           :rem 225
310 A=1:PRINTRIGHT$("0000"+MID$(STR$(AD),
    2),5);":";                             :rem 33
315 FORJ=ATO6                              :rem 33
320 GOSUB570:IFN=-1THENJ=J+N:GOTO320
                                           :rem 228
390 IFN=-211THEN 710                       :rem 62
400 IFN=-204THEN 790                       :rem 64
410 IFN=-206THENPRINT:INPUT"{DOWN}ENTER N
    EW ADDRESS";ZZ                         :rem 44
415 IFN=-206THENIFZZ<SORZZ>ETHENPRINT"
    {RVS}OUT OF RANGE":GOSUB1000:GOTO410
                                           :rem 225
417 IFN=-206THENAD=ZZ:PRINT:GOTO310
                                           :rem 238
420 IF N<>-196 THEN 480                    :rem 133
430 PRINT:INPUT"DISPLAY:FROM";F:PRINT,"TO
    ";:INPUTT                              :rem 234
440 IFF<SORF>EORT<SORT>ETHENPRINT"AT LEAS
    T";S;"{LEFT}, NOT MORE THAN";E:GOTO43
    0                                      :rem 159
450 FORI=FTOTSTEP6:PRINT:PRINTRIGHT$("000
    0"+MID$(STR$(I),2),5);":";             :rem 30
451 FORK=0TO5:N=PEEK(I+K):PRINTRIGHT$("00
    "+MID$(STR$(N),2),3);",";              :rem 66
460 GETA$:IFA$>""THENPRINT:PRINT:GOTO310
                                           :rem 25
470 NEXTK:PRINTCHR$(20);:NEXTI:PRINT:PRIN
    T:GOTO310                              :rem 50
480 IFN<0 THEN PRINT:GOTO310               :rem 168
490 A(J)=N:NEXTJ                           :rem 199
500 CKSUM=AD-INT(AD/256)*256:FORI=1TO6:CK
    SUM=(CKSUM+A(I))AND255:NEXT            :rem 200
510 PRINTCHR$(18);:GOSUB570:PRINTCHR$(146
    );                                     :rem 94
511 IFN=-1THENA=6:GOTO315                  :rem 254
515 PRINTCHR$(20):IFN=CKSUMTHEN530
                                           :rem 122
```

```
520 PRINT:PRINT"LINE ENTERED WRONG : RE-E
    NTER":PRINT:GOSUB1000:GOTO310:rem 176
530 GOSUB2000                      :rem 218
540 FORI=1TO6:POKEAD+I-1,A(I):NEXT:POKE54
    272,0:POKE54273,0              :rem 227
550 AD=AD+6:IF AD<E THEN 310       :rem 212
560 GOTO 710                       :rem 108
570 N=0:Z=0                         :rem 88
580 PRINT"{£}";                     :rem 81
581 GETA$:IFA$=""THEN581           :rem 95
582 AV=-(A$="M")-2*(A$=",")-3*(A$=".")-4*
    (A$="J")-5*(A$="K")-6*(A$="L"):rem 41
583 AV=AV-7*(A$="U")-8*(A$="I")-9*(A$="O"
    ):IFA$="H"THENA$="0"           :rem 134
584 IFAV>0THENA$=CHR$(48+AV)       :rem 134
585 PRINTCHR$(20);:A=ASC(A$):IFA=13ORA=44
    ORA=32THEN670                  :rem 229
590 IFA>128THENN=-A:RETURN         :rem 137
600 IFA<>20 THEN 630                :rem 10
610 GOSUB690:IFI=1ANDT=44THENN=-1:PRINT"
    {OFF}{LEFT} {LEFT}";:GOTO690   :rem 62
620 GOTO570                        :rem 109
630 IFA<48ORA>57THEN580            :rem 105
640 PRINTA$;:N=N*10+A-48           :rem 106
650 IFN>255 THEN A=20:GOSUB1000:GOTO600
                                   :rem 229
660 Z=Z+1:IFZ<3THEN580              :rem 71
670 IFZ=0THENGOSUB1000:GOTO570     :rem 114
680 PRINT",";:RETURN               :rem 240
690 S%=PEEK(209)+256*PEEK(210)+PEEK(211)
                                   :rem 149
691 FORI=1TO3:T=PEEK(S%-I)          :rem 67
695 IFT<>44ANDT<>58THENPOKES%-I,32:NEXT
                                   :rem 205
700 PRINTLEFT$("{3 LEFT}",I-1);:RETURN
                                     :rem 7
710 PRINT"{CLR}{RVS}*** SAVE ***{3 DOWN}"
                                   :rem 236
715 PRINT"{2 DOWN}(PRESS {RVS}RETURN{OFF}
     ALONE TO CANCEL SAVE){DOWN}":rem 106
720 F$="":INPUT"{DOWN} FILENAME";F$:IFF$=
    ""THENPRINT:PRINT:GOTO310      :rem 71
730 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR
     {RVS}D{OFF}ISK: (T/D)"        :rem 228
740 GETA$:IFA$<>"T"ANDA$<>"D"THEN740
                                   :rem 36
750 DV=1-7*(A$="D"):IFDV=8THENF$="0":"+F$:
    OPEN15,8,15,"S"+F$:CLOSE15     :rem 212
760 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
    ):POKE782,ZK/256                :rem 3
762 POKE781,ZK-PEEK(782)*256:POKE780,LEN(
    T$):SYS65469                   :rem 109
763 POKE780,1:POKE781,DV:POKE782,1:SYS654
    66                             :rem 69
765 K=S:POKE254,K/256:POKE253,K-PEEK(254)
    *256:POKE780,253               :rem 17
766 K=E+1:POKE782,K/256:POKE781,K-PEEK(78
    2)*256:SYS65496                :rem 235
770 IF(PEEK(783)AND1)OR(191ANDST)THEN780
                                   :rem 111
775 PRINT"{DOWN}DONE.{DOWN}":GOTO310
                                   :rem 113
780 PRINT"{DOWN}ERROR ON SAVE.{2 SPACES}T
    RY AGAIN.":IFDV=1THEN720       :rem 171
781 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
    ;E2$:CLOSE15:GOTO720           :rem 103
790 PRINT"{CLR}{RVS}*** LOAD ***{2 DOWN}"
                                   :rem 212
795 PRINT"{2 DOWN}(PRESS {RVS}RETURN{OFF}
     ALONE TO CANCEL LOAD)"        :rem 82
800 F$="":INPUT"{2 DOWN} FILENAME";F$:IFF
    $=""THENPRINT:GOTO310          :rem 144
810 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR
     {RVS}D{OFF}ISK: (T/D)"        :rem 227
820 GETA$:IFA$<>"T"ANDA$<>"D"THEN820
                                   :rem 34
830 DV=1-7*(A$="D"):IFDV=8THENF$="0":"+F$
                                   :rem 157
840 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
    ):POKE782,ZK/256                :rem 2
841 POKE781.ZK-PEEK(782)*256:POKE780,LEN(
    T$):SYS65469                   :rem 107
845 POKE780,1:POKE781,DV:POKE782,1:SYS654
    66                             :rem 70
850 POKE780,0:SYS65493             :rem 11
860 IF(PEEK(783)AND1)OR(191ANDST)THEN870
                                   :rem 111
865 PRINT"{DOWN}DONE.":GOTO310     :rem 96
870 PRINT"{DOWN}ERROR ON LOAD.{2 SPACES}T
    RY AGAIN.{DOWN}":IFDV=1THEN800
                                   :rem 172
880 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
    ;E2$:CLOSE15:GOTO800           :rem 102
1000 REM BUZZER                    :rem 135
1001 POKE54296,15:POKE54277,45:POKE54278,
     165                           :rem 207
1002 POKE54276,33:POKE 54273,6:POKE54272,
     5                              :rem 42
1003 FORT=1TO200:NEXT:POKE54276,32:POKE54
     273,0:POKE54272,0:RETURN      :rem 202
2000 REM BELL SOUND                 :rem 78
2001 POKE54296,15:POKE54277,0:POKE54278,2
     47                            :rem 152
2002 POKE 54276,17:POKE54273,40:POKE54272
     ,0                             :rem 86
2003 FORT=1TO100:NEXT:POKE54276,16:RETURN
                                    :rem 57
3000 PRINTC$;"{RVS}NOT ZERO PAGE OR ROM":
     GOTO1000                      :rem 89
```

Animating The VIC

(Article on page 135.)

Pop Up

```
1 PRINT"{CLR}":POKE36879,11:X=7912:R=0:Y=
  0:SC=0                           :rem 117
10 POKE37154,127:P=PEEK(37152)AND128:J0=-
   (P=0)                            :rem 51
20 P=PEEK(37151):J1=-((PAND8)=0):J2=-((PA
   ND16)=0)                        :rem 168
40 J3=-((PAND4)=0):R=(J0-J2)+22*(J1-J3):I
   FX+R<7680ORX+R>8185THEN57       :rem 241
50 IFPEEK(X+R)<81ANDPEEK(X+R)<>32THEN300
                                   :rem 163
55 POKEX,32:X=X+R:POKEX+30720,1:POKEX,94:
   IFR<>0THENSC=SC+1               :rem 126
57 POKE36878,15:POKE36876,INT(RND(1)*127+
   128)                             :rem 12
60 POKE36878,100-SQR(100):IFRND(1)*11>4TH
   EN10                             :rem 36
70 Y=INT(RND(1)*512+1):POKE38400+Y,INT(RN
   D(1)*9+8):POKE7680+Y,42+RND(1)*2:GOTO1
   0                                :rem 82
300 POKE36876,0:POKE36874,0:POKE36875,0
                                   :rem 207
302 POKE36878,15:FORI=200TO140STEP-1:POKE
    36874,I:POKE36875,I:FORE=1TO20:NEXT
                                   :rem 152
303 POKEX+30720,INT(I/14):NEXT:POKE36878,
    0:POKE36874,0:POKE36875,0      :rem 122
304 PRINT"{CLR}{WHT}{10 DOWN}{3 SPACES}FI
    NAL SCORE:"SC                   :rem 96
```

PRODUCT MART

Learn Computer Hardware Design With Your 64

EXPERIMENTS WITH COMPUTER CHIPS

This integrated hardware/software system interfaces with your 64. It includes a tutorial program to step you through experiments and display results.

This complete system requires no previous experience

Don't miss out on the hardware half of the computer field
Order LOGIC LAB64 Now
(Specify Disk or Tape)

Send Check or Money Order for $119 to:
ACCELERATED LEARNING TOOLS
P. O. Box 885, Hermosa Bch, CA 90254
Send for free brochure

OnTrack

Precision 1840/1841 Disk Drive Alignment System

★ **PRECISION DISK DRIVE ALIGNMENT SYSTEM**
- REQUIRES NO MODIFICATION TO YOUR DRIVE
- INCLUDES
- EASY TO INSTALL TRACK INDICATOR
  - SHOWS PRECISE TRACK BEING USED AT ALL TIMES
- ALIGNMENT/UTILITY DISKETTE
- REALIGNMENT PROBLEM-PREVENTION TIPS
- EASY TO USE PHOTO-ILLUSTRATED INSTRUCTIONS

★ **CURE/PREVENT HEAD ALIGNMENT PROBLEMS**
- FOR ALL 1540/1541 DISK DRIVES

★ **INVALUABLE BACK-UP COPY TOOL**
- WORKS WITH ALL COPY SOFTWARE
- REDUCES BACK-UP COPY TIME UP TO 90%
  - AVOIDS WRITING UNNECESSARY BAD BLOCKS

SPECIAL INTRODUCTORY OFFER THROUGH 31 OCTOBER 1984

FAST RESPONSE EACH POSTPAID **$12.95**
REGULAR PRICE $15.95 Check or Money Order

SchuLace ENTERPRISES
P.O.BOX 771 CASCADE, MARYLAND 21719
"Innovative yet Inexpensive!"

CONVERSE WITH YOUR COMPUTER

AT LAST! A FULL IMPLEMENTATION of the original ELIZA program is now available to run on your Commodore 64!

Created at MIT in 1966, ELIZA has become the world's most celebrated artificial intelligence demonstration program. ELIZA is a non-directive psychotherapist who analyzes each statement as you type it in and then responds with her own comment or question – and her remarks are often amazingly appropriate!

Designed to run on a large mainframe, ELIZA has never before been available to personal computer users except in greatly stripped down versions lacking the sophistication which made the original program so fascinating.

Now, our new Commodore 64 version possessing the **FULL** power and range of expression of the original is being offered at the introductory price of only $25. And if you want to find out how she does it (or teach her to do more) we will include the complete **SOURCE PROGRAM** for only $20 additional.

Order your copy of ELIZA today and you'll never again wonder how to respond when you hear someone say, "Okay, let's see what this computer of yours can actually do!"

ELIZA IS AVAILABLE IN THE FOLLOWING FORMATS:
(Please specify Disk or Cassette)

1. Protected Version $25
   (Protected Version can be run but not listed or modified)
2. Un-protected Commodore 64 BASIC Source Version $45
   (Source Version can be listed and modified as well as run)

Both versions include a six page user manual.

Please add $2.00 shipping and handling to all orders
(California residents please add 6% sales tax)

ARTIFICIAL INTELLIGENCE RESEARCH GROUP
921 North La Jolla Avenue, Dept. G
Los Angeles, CA 90046
(213) 656-7368 (213) 654-2214
MC, VISA and checks accepted

VISA MasterCard

CASSETTE INTERFACE ADAPTER

VIC 20* C-64*

- Signal level indicator enables perfect load every time
- Allows computer to control cassette motor
- Cassette on-line indicator
- Plug directly into cassette port
- Use standard cassette recorder
- Fully documented
- 90-day full warranty

TERMS:
Check or M.O. (allow 4-6 wks)
COD add $5.00

$25.95

Add $3.00 shipping. OK residents add 5% sales tax

DSM MARKETING
P.O. BOX 7647
OK CITY, OK 73153

*Trademarks of Commodore Electronics, Ltd.

SPEECH SYNTHESIS

NEW!

SAVE $5 ONLY **$49** KIT

UNLIMITED VOCABULARY
- Amplifier + Speaker
- Input for Ears Music Units

Built $59.95

SMART EARS!

SPEECH RECOGNITION $99
Vocal commands to a computer.

TALK TO YOUR MICRO

EARS opens the door to direct man-machine communication. The system comprises analogue frequency separation filters, preamps and signal conversion, together with a quality microphone and extensive software.

AUDIO VISION — CG
1279 N. NORMANDIE
LOS ANGELES, CA 90027
ADD $3. SHIPPING/CA. RES. ADD 6.5% Tx
(213) 660-5217

C-64 DISK CATALOGER

Catalog your entire disk library. No more fumbling to locate the program or file you want -

- Catalog, update or delete disk directory entry from Master Index.
- Review Master Index or any disk directory on screen or hard-copy.
- Printer dump of selected disk directories.
- Diskname print on standard labels.
- Search for any title in the entire library.
- Supports 1 or 2 drives.

MENU DRIVEN - USER FRIENDLY - FAST
- Special Introductory Price -
$14.95 (includes p & h)

Bon-Soft
13426 Laurel River Drive
Houston, TX 77083

CHIPMONK SOFTWARE

BIBLE MEMORIZER-Memory Verses
BIBLE SCRAMBLE-Book Quizzer
MUSICMAN-Fast Music Editor
QUICK FILE'EM-Easy Disk Filing
CAT N MOUSE-Word Recoginition
SPELLING SCRAMBLE-Best Speller
MEMORY TUTOR-Improves Memory
CHECKTRACKER 64
CHRISTMAS CAROLS-Words & Music
ELECTO GREETING CARD MAKER
RESET SWITCHES-$9.00

$15 Each 3/$35
631 N. Cherry
Battleground, Wn. 98604
Send SASE for catalog & Reviews

VIC-20 COMMODORE 64

THE RECIPE BOX

Now you can easily store and recall your favorite recipes on your Commodore computer. THE RECIPE BOX is a complete menu-driven disk system that comes with these additional features:
SEARCH BY INGREDIENT - Only have a pound of hamburger in the freezer? Let THE RECIPE BOX show you all the recipes that you have on file that use hamburger, or any other ingredient you choose.
SEARCH BY CATEGORY/INGREDIENT - Any combination of the above.
AUTOMATIC MEASUREMENT - THE RECIPE BOX will automatically scale up or down the amount of ingredients you need according to how many servings you want.
SCREEN OR PRINTED OUTPUT - Have printed copies to use in the kitchen or give to friends.
THE RECIPE BOX requires one disk drive and will run on a 5K VIC-20, Commodore 64. Please specify. Send check or money order for 19.95 to:

Aries Marketing Co.
P.O. Box 4196
Dept. G
4200 Shannon Drive
Baltimore, MD 21205
MD residents add 5% sales tax

FOR COMMODORE 64™

DISKMIMIC 5+™

- Backs up virtually all existing disks for Commodore 64™, including Copy Protected Versions. All Automatically.
- Supports 1541™ Drives.
- Don't be without back-up!!

Now Twice as Fast!
ONLY $49.95

A.I.D. CORP.
4020 HEMPSTEAD TURNPIKE
BETHPAGE, NEW YORK 11714
(516) 731-7100

Diskmimic 5™ is a trademark of A.I.D. Corporation
Commodore 64™ & 1541™ is a trademark of Commodore Electronics Ltd.

Shipping & Handling — $1.50 each
DEALER INQUIRIES INVITED

WIZARD'S DOMINION

ONLY THE BRAVEST DARE TO ENTER!! ARM YOUR CHARACTER WITH WEAPONS AND MAGIC. THEN FIND GOLD AND FIGHT OGRES AND GAIN NEW MAGICAL POWERS.

- *3-D PERSPECTIVE*
- *SUPERB GRAPHICS*
- *COMPLEX BATTLES*
- *LOTS OF MAGIC*
- *THOUSANDS OF CAVES*

COMMODORE 64
TI 99/4A (EXTENDED BASIC)
CASSETTE $19.95 DISK $21.95
DEALER INQUIRIES WELCOME

INSTRUCTIONS INCLUDED. To order send check or money order plus $1.50 shipping/handling to:

American Software
Design & Distribution Co.
P.O. Box 246 Dept. G-6
Cottage Grove, MN 55016

CARTRIDGE CRACKER™

WITH
ECA BACKUP™ AND SUPER SAVER™

PACKAGE INCLUDES:

1. EXPANSION BOARD
2. **CARTRIDGE CRACKER™** SOFTWARE TO BACKUP MOST CARTRIDGES.
3. **ECA BACKUP™** SOFTWARE TO BACKUP ELECTRONIC ARTS™ DISKS.
4. **SUPER SAVER™** SOFTWARE FOR FILE COPYING AND DISK TO TAPE COPYING.
5. USERS MANUAL.

FOR ARCHIVAL USE ONLY. IT IS ILLEGAL TO SELL OR DISTRIBUTE COPYRIGHTED SOFTWARE.

FOR **COMMODORE 64** **$44.95** + SHIPPING

FAMILY TREE (REVISED)
PET-I-GREE (NEW)

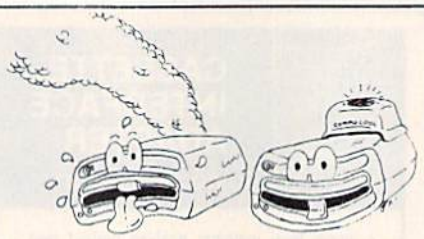
FAMILY TREE — Keep track of your ancenstry with pedigree charts and family record sheets from our popular program that has now been updated and revised. 664 names per disk, up to 6 generation charts, improved editing, more user friendly. $49.95 U.S.

PET-I-GREE — A new program for the dog breeder and kennel operator that keeps A.K.C. records and produces required pedigree charts and information files.

For information write or phone:
GENEALOGY SOFTWARE
Phone 519-344-3990

P.O. Box 1151
Port Huron, Michigan 48061

1046 Parkwood Ave.
Sarnia, Ontario N7V 3T9



ISN'T IT TIME you cooled off your 1540/41 disk drive?

Compu-Cool is a ***filtered*** cooling system that will eliminate those problems associated with heat in your disk drive.

Compu-Cool mounts directly on your disk drive without any hardware.

Send $39.95 (plus $2 shipping) to:

ADINC
P. O. Box 12094, Dept. G
San Antonio, TX 78212
(Texas residents add $2.40 tax)

NUMERIC KEY PAD
FOR **VIC-20 AND 64**

7	8	9	★	
4	5	6	/	←
1	2	3	+	↑
0	,	•	—	DEL
ENTER		SH	↑	

- 38 FUNCTIONS
- FULL CURSOR CONTROL
- NO SOFTWARE REQUIRED

$59.95 + SHIPPING

Bear Technologies INC.

VISA AND MASTERCARD ACCEPTED

3417 ROGER B. CHAFFEE BLVD.
GRAND RAPIDS, MI 49508
(616) 245-5061

ADD $3.00 SHIPPING + HANDLING
MI. RESIDENTS ADD 4% SALES TAX

PROTECT YOUR EXPENSIVE EQUIPMENT FROM DUST, LIQUIDS WITH A CROWN PROTECTIVE

COVER

- CUSTOM MADE TO FIT
- HEAVY 32 oz VINYL
- ANTI-STATIC
- SOFT LINED
- CHOICE of COLOR, TAN or BROWN

Covers for:

VIC20/C-64	7.00
C-1541 D/DRIVE	8.00
C-1525 PRINTER	10.00
DATASETTE (New)	5.00
DATASETTE (Old)	5.00
GEMINI 10/10X PRINTER	13.00
GEMINI 15/15X PRINTER	16.00
EPSON MX80 PRINTER	11.00
EPSON MX100 PRINTER	14.00
C-1702 MONITOR	19.00

Order by stating name and model of equipment for cover desired. Choice of color: TAN or BROWN. Enclose check or M.O.+1.50 shipping. Calif. Res. include 6.5% State Tax.

COVERS NOT NAMED ABOVE WILL BE FABRICATED TO YOUR SPECS. SEND YOUR REQUIREMENTS FOR LOW PRICE QUOTES.

CROWN CUSTOM COVERS
9606 SHELLYFIELD RD.,
DOWNEY, CA 90240

Successful Delivery!
Chromazone's New Arrival!

BRINK JINK

Maneuver through the mazes to make a million . . . if you dare!

Graphics for C-64, disk only, joystick required.

Send $29.95 in check or money order to:

Chromazone Software
P.O. Box 7325
San Jose, CA 95150-7325

CA residents add $1.95 tax

SOFTWARE PROTECTION HANDBOOK

This book "BLOWS THE LOCKS OFF" protected DISKS, CARTRIDGES, and TAPES! Protection "secrets" are clearly explained along with essential information and procedures to follow for breaking protected software. An arsenal of protection breaking software is included with all listings, providing you with the tools needed! Programs include high speed error check/logging disk duplicator... Disk picker... Disk editor... Cartridge to disk/tape saver and several others for error handling and advanced disk breaking. The cartridge methods allow you to save and run cartridges from disk or tape! The tape duplicator has never been beaten! This manual is an invaluable reference aid including computer and disk maps, as well as useful tables and charts.

C64 Book only.....$16.95 US
Book & Disk of all programs.....$29.95 US
Vic 20 book.....Cart. & Tapes only....$9.95 US

THIS MANUAL DOES NOT CONDONE PIRACY
DISTRIBUTING COPIED SOFTWARE IS ILLEGAL

PSIDAC, 7326 N. ATLANTIC, PORTLAND, OR 97217
-CHECK OR MONEY ORDER ONLY-

MENU PLANNER

Let **MENU PLANNER** assist you in arranging your dietary and shopping needs. The **MENU PLANNER** is a userfriendly kitchen helper for the C-64 with one 1541 drive. **MENU PLANNER** has the following features and more:

- **Store all of your favorite recipes**
- **Monitor your calorie intake**
- **Search by ingredient and calories**
- **Automatically change amount of *Ingredients to accomodate number* of servings**
- **Screen or printer capabilities for recipe and/or grocery list**

MEAL PLANNER-Only $15.95

LEATHER DUST COVERS FOR VIC & C-64

1st Class Protection for your Computer

Computer Cover	$14.95
1541 Cover	$13.95
Datasette Cover	$ 9.95

S & S Enterprises
P.O. Box 111
Hot Springs, SD 57747

VISA and Master Card accepted
Dealer inquiries invited

---

ditto

COPY DISKS AUTOMATICALLY

$39.95

- Copies 99% of currently available Commodore 64 disks.
- Supports 1 or 2 1541 drives.
- Takes approximately 25 minutes.
- Easy to use—menu driven.
- Currently available—future updates $17.
- ditto will even backup ditto.

ORDERS
800-762-5645
Hours 10-6 Mon.-Sat.

CARDINAL SOFTWARE
13646 Jefferson Davis Highway
Woodbridge, VA 22191

DOUBLE THE POWER OF YOUR COMMODORE 64!!!
ACQUIRE A POWERFUL TOOL OF PROFESSIONAL PROGRAMMERS!!!

P TECHNOLOGIES INTRODUCES THE

RAMDISK-64

The RAMDISK-64 doubles the memory capacity of the C-64 to 128K bytes of ram. Use the RAMDISK-64 as a second disk drive with the disk emulation software provided free ($49.95 value). Or use it with your own custom software as storage for printer spoolers, data bases, spreadsheets, hires graphic screens, whatever!!!

- Enclosed in sturdy cartridge case
- Easy to read User's Manual
- Can be used with other cartridges
- Disk emulation software provided
- Compatible with Basic disk commands
- Diagnostics provided
- Emulation source code provided on diskette
- 15 day free trial period — money-back guarantee!!!
- 90-day free replacement warranty
- Factory direct
- Many satisfied customers!!!
- Expandable to 256K with 256K Dram chips

INTRODUCTORY SALE!!!

Suggested retail price $199.95
Special Introductory Price **$149.95**
SAVE $50.00!!
Extender Board $10.00
(necessary if not using a motherboard)

Add $5.00 for shipping, handing, and insurance. Calif. residents add 6% sales tax. Personal checks take two weeks to clear. Order by phone or mail.

P Technologies
6905 Speckle Way, Sacramento, California 95842
(916) 920-3226

Learn to Play Lowball Draw

POKER!

at

Silicon Slick's Lowball
Poker Parlor!

A Game, Instructor & Analysis Tool!

YOU CAN

- Change House Rules
- Vary Number and Skill Level of Opponents
- Vary Size of Ante & Bet Limits
- Vary Number & Size of Blind Bets
- Set up & Analyze Special Situations

AND BEST OF ALL

Personalized Instruction from *Silicon Slick!*

$34.95

Requires Commodore-64® + Disk Drive

15 Day Money Back Guarantee
Call 208-524-5464 or Write
Snake River Software
2100 Belmont Ave.
Idaho Falls, ID 83401

DEALERS INQUIRES INVITED

HOME COMPUTER DESK PLANS

PROTECT YOUR INVESTMENT!

Designed by home computer user. All the room you need for computer monitor, printer, peripherals, etc. Shelves for software, everything at your fingertips. Fits COMMODORE, ATARI, APPLE I & II, IBM-PC, TRS 80. Bottom shelf slotted for printer paper plus storage. 28" deep x 51¾" high x 71¾" length. **Quality Plans, Instructions.**

PLANS - $10.00

CARPENTER'S CREATIVE DESIGNS
P.O.Box 122 / Desert Center, CA 92239

SIGN OF THE SPHINX

Something new from **Werewolf Software** — A world of enigmatic artifacts, disturbing visions, peculiar people. From an abandoned subway station to the Eternal Black Mass and beyond, you discover traces of forgotten knowledge which may lead you to the Crimson Altar and its final secret.

A morbid, suprising adventure. Sign of the Sphinx is disk-based to use memory more efficiently. If you appreciate detail, if you have a taste for the bizarre, then you should investigate the activity at the Sign of the Sphinx.

Text adventure on disk for the Commodore 64. $15 plus $1 shipping. Calif. residents add tax.

WEREWOLF SOFTWARE
109 Minna Street
Suite 353
San Francisco, CA 94105

C-64™ & VIC-20™

SUPER TYPEWRITER

The mini word processor you've wanted . . .

FEATURES:

- Changeable line width up to 80 characters
- Automatic margin setting
- Automatically centers each additional copy
- Upper and Lower Letters
- No more broken words with use of automatic carriage return
- Edit Text

All Poorhaus Programs user accessible for learning or adding personal touch. Simple to use. Load and follow instructions within programs.

Super Typewriter	**$24.95**
Home Inventory	**12.95**
Check Register	**19.95**
Black Jack	**9.95**
Loan Analyzer	**9.95**

Some VIC-20 Programs may need memory expansion.

POORHAUS SOFTWARE
P.O. Box 10782, Yakima, WA 98909
(509) 966-8461
SPECIFY TAPE OR DISK
MC, VISA, AND CHECKS ACCEPTED

NFL*

PROGNOSTICATOR

for FOOTBALL FANS with

$19.95 DISK/TAPE

☐ COMMODORE 64™
☐ VIC 20™ 8K+

- KEEP SCORES
- DISPLAY STANDINGS
- PREDICT WINNERS

PRINTOUTS '84 SCHEDULE

ORDER- Specify computer, tape or disk. Send check or money order for $19.95 (incl s&h) or COD($2 added). MN orders add 6%. Dealer inquiries welcome.
MORE- Ask for **catalog** of **MITE-y** home software.

* NFL is a trademark of the National Football League. COMMODORE 64, VIC 20 are tm of Commodore Electronics.

NELSON SOFTWARE
2232 OGDEN CT, ST. PAUL, MN 55119

VIC-20

NEW

VIC-20 INTERFACING BLUE BOOK

Did you know that your VIC can be used to control a 99¢ toy motor so effectively that it runs like a precision machine? Or that you can build an accurate digital thermometer using the VIC and four parts costing less than $5?

These and other 28 interfacing projects selected for usefulness, ease of construction and low cost are detailed in the VIC-20 Interfacing Blue Book, a veritable gold mine of practical information on how to build a variety of interfaces for your computer.

Projects include: Connecting VIC to your stereo; Pickproof digital lock; Capacitance meter; Liquid level sensor; Telephone dialer; Voice output; 8K/16K RAM/ROM expansion; 128K RAM expansion; 8-bit precision D/A; 8-bit A/D converter; MX-80 interface and more.

Written by a college professor in a friendly and informative style, the Blue Book gives you theory of operation, schematics, program listings, parts list, construction hints and sources of materials for each one of the 30 projects.

If you want to get the most out of your VIC this book is a must. Cost is $14.95 (less than 50¢ per project!). Price includes postage.

microsignal Dept. C
P.O. BOX 22
MILLWOOD NY 10546

VIC-20

ADVERTISERS INDEX

CARDCO "NOW" SOFTWARE

. . . available now for your Commodore-64™ and more!

A fine line of software developed by CARDCO for your Commodore-64 computer with all the features you should expect in much more costly software. CARDCO's "NOW" Series provides many unique and exclusive features and are packaged for easy reference, simple storage, instant recognition.

"WRITE NOW" . . . WORD PROCESSOR SOFTWARE . . . An excellent time saver, CARDCO offers the "Write Now" C/ Ø2 word processor program with built-in 80 column display. You see exactly what will print. All special codes can be transmitted to printers maintaining justification. Easy full-screen editing; works with any printer.

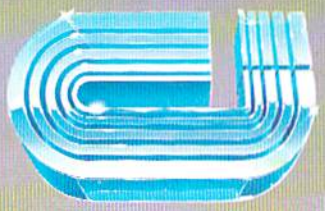
"MAIL NOW" . . . MAILING LIST SOFTWARE . . . CARDCO's D/ Ø1 "Mail Now" quickly (in memory) sorts by zip, category, name and state; fully compatible with "Write Now". Other fine features include: user-oriented; menu-driven operation; each disk supports 600 entries. Format can print single, double or triple labels across.

"SPELL NOW" . . . Cardware D/ Ø4 . . . a fine program designed as a spell checker for use with "Write Now" on the Commodore-64. A 34,000 word dictionary with two additional user constructed dictionaries. Menu-driven operation for ease of use. And "Spell Now" allows you to see each misspelled word in the context of your document for correction.

"FILE NOW" . . . D/ Ø5 . . . is a totally integrated, menu-driven database software package which interfaces with both the "Write Now!" for the 64 and the "Spell Now." 40K of working storage space is available with "File Now". "File Now" appears on the screen as index cards for easier manipulation of your data base; you see 5 index cards at a time. Cards are user defineable, i.e., user determines what goes where on the "index cards" and can sort by any given field. Every card has a general topic field which allows for quick sorting through cards.

"GRAPH NOW" INCLUDING . . . "PAINT NOW" . . . D/ Ø6 . . . This disk-based graphic/logo generator is totally menu-driven. Allows for the development of pies, charts, bar graphs and other vivid graphic illustrations. Also has the ability to design, and print logos and high resolution pictures. "Commodore-ready"; interfaces with CARDCO'S "Write Now" Word Processor, "Mail Now", "Spell Now" and "File Now".

Write for illustrated literature and prices or see CARDCO Computer Accessories and Software wherever Computers are sold.



cardco, inc.

300 S. Topeka Wichita, Kansas 67202 (316) 267-6525

"The world's largest manufacturer of Commodore accessories."

HOW TO BECOME COMPUTER LITERATE.

FEATURES	COMMODORE 64™	APPLE IIe™	IBM PC jr™	ATARI 800XL™
Price*	$219	$699	$669	$299
Built-in Memory	64K	64K	64K	64K
Typewriter Keyboard	YES (66 Keys)	YES (62 Keys)	"CHICKLET" (62 Keys)	YES (61 Keys)
Upper/Lower Case	YES	YES	YES	YES
Programmable Function Keys	YES	NO	YES	NO
AUDIO				
Polyphonic Tones	YES	NO	YES	YES
Music Synthesizer	YES	NO	NO	NO
Hi-Fi Output	YES	NO	YES	YES
VIDEO				
TV Output	YES	EXTRA COST	EXTRA COST	YES
Video Monitor Output	YES	YES	EXTRA COST	YES
INPUT/OUTPUT				
Intelligent I/O Bus	YES	NO	NO	YES
RS-232 Communications	YES**	EXTRA COST	YES**	EXTRA COST
"Smart" Peripherals	YES	NO	NO	YES

*Prices shown are common retail and may vary slightly in different markets.
**Requires an adapter to operate.

First you need the right input.

Like $219. That's what the Commodore 64™ costs. It's about one third the price of the Apple IIe™ or the IBM® PCjr.™

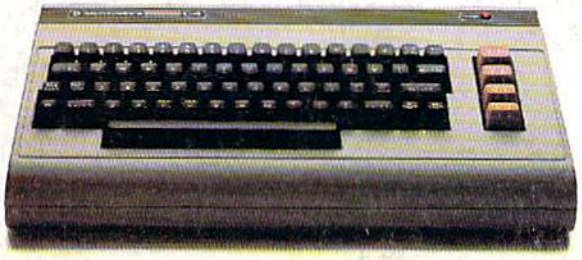
And 64K. That's how much memory the Commodore 64 has. It's also how much memory Apple IIe and the IBM PCjr have.

This computer lesson is brought to you as a public service by Commodore (certainly not by Apple or IBM), the only computer company that can afford to show you a chart like the one above.

But what you can't see above are the thousands of software programs that make the Commodore 64 fully capable of doing anything any "triple the price" computer can do; for fun or profit, for every member of the family; anything from soccer to spread sheets to space exploration.

Because the Commodore is so affordable, you can load up on Commodore peripherals. Like a disk drive, a printer or a telephone modem. All together they cost just a tad more than an IBM PCjr by itself. With no peripherals.

No wonder Commodore sells more computers than Apple and IBM combined.



commodore
COMPUTERS

IT'S NOT HOW LITTLE IT COSTS,
IT'S HOW MUCH YOU GET.