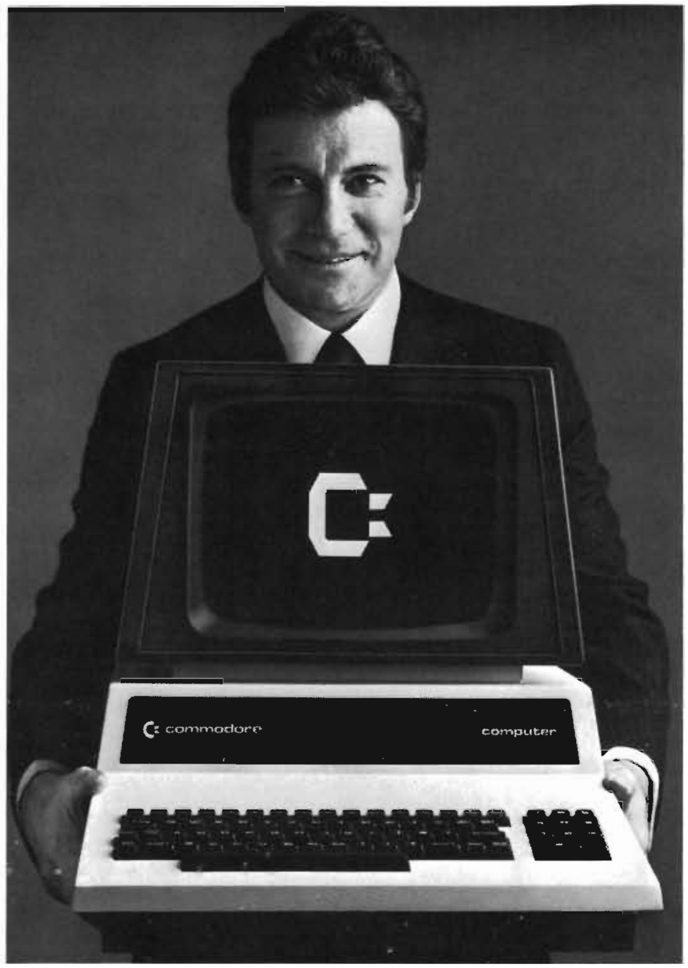
THE MICROCOMPUTER MAGAZINE

commodore

OCTOBER 1981

commodore COMPUTER

THE COMMODORE COMPUTERS
"FROM $300 TO $1995, THEY COST LESS AND GIVE YOU MORE FOR YOUR MONEY. READ OUR CHART."
—William Shatner



The idea of a computer in every office and home used to be science fiction. Now it's becoming a reality. The question is, with so many to choose from, which computer should you buy? When you consider the facts, the clear choice is Commodore.

COMPARE OUR $995 COMPUTER

FEATURES	COMMODORE 4016	APPLE II	IBM
Base Price	$995	$1,330	$1,565
12" Green Screen	Standard	299	345
IEEE Interface	Standard	300	NO
TOTAL	$995	$1,929	$1,910
Upper & Lower Case Letters	Standard	NO	Standard
Separate Numeric Key Pad	Standard	NO	Standard
Intelligent Peripherals	Standard	NO	NO
Real Time Clock	Standard	NO	NO
Maximum 5½" Disk Capacity per Drive	500K	143K	160K

Prices are as of the most recent published price lists, September, 1981 and approximate the capabilities of the (16K) PET® 4016. Disk Drives and Printers are not included in prices. Models shown vary in their degree of expandability.

Many experts rate Commodore Computers as the best desk-top computers in their class. They provide more storage power—up to 1,000,000 characters on 5¼" dual disks—than any systems in their price range. Most come with a built-in green display screen. With comparable systems, the screen is an added expense. Our systems are more affordable. One reason: we make our own microprocessors. Many competitors use ours. And the compatibility of peripherals and basic programs lets you easily expand your system as your requirements grow. Which helps explain why Commodore is already the No. 1 desk-top computer in Europe with more than a quarter of a million computers sold worldwide.

WE WROTE THE BOOK ON SOFTWARE.
The Commodore Software Encyclopedia is a comprehensive directory of over 500 programs for business, education, recreation and personal use. Pick up a copy at your local Commodore dealer.

FULL SERVICE, FULL SUPPORT.
Commodore dealers throughout the country offer you prompt local service. In addition, our new national service contract with TRW provides nationwide support. Visit your Commodore dealer today for a hands-on demonstration.

Commodore Computer Systems
681 Moore Road
King of Prussia, PA 19406

Canadian Residents:
Commodore Computer Systems
3370 Pharmacy Avenue
Agincourt, Ontario, Canada, M1W 2K4

Please send me more information.

Name __________

Company __________ Title __________

Address __________

City __________ State ______ Zip ______

Telephone __________

Interest Area __________

☐ Business ☐ Education ☐ Personal

commodore
COMPUTER

commodore

THE MICROCOMPUTER MAGAZINE

FEATURES

Cover Art by Don Dyen

DEPARTMENTS

commodore

Commodore Magazine is issued bi-monthly. Published by the Computer Systems Division, Commodore Business Machines, Inc., 681 Moore Rd., King of Prussia, PA 19406 (215) 337-7100. The U.S. Subscriber rate: $15.00 per year. In Canada and Mexico the Subscriber rate is $25.00. No material may be reprinted without permission. Volume 2, Number 4.

Q&A HOTLINE

Q. My computer is a Commodore 8032. Is there a machine code program that scrolls part of the screen, where the screen heading stays, but a file on the screen scrolls up or down by pressing a key? The reason is to be able to see all the items of a file, longer than the screen can hold the information. Secondly, is it possible to scroll a file at the right side of the screen, while the left part remains? Finally, is it possible to interface a standard electric typewriter (for example, an IBM) to the computer?

N. Schlebaum

A. *The 8032 has a special screen controller chip that can be programmed by simply PRINTing certain PET ASCII codes. You can use these character codes to set up a window, which is a rectangular area of the screen that will scroll while the rest of the screen stays put. To keep the top of the screen fixed while the bottom scrolls, position the cursor to the top of the area you want to scroll and PRINT CHR$(15). Now the lower section will scroll, but the top will remain fixed. Here is a table of the special CHR$ codes for the 8032's screen:*

CHR$(14) Display text instead of graphic characters
CHR$(15) Set top left corner of window
CHR$(21) Delete line (scrolls screen below cursor up 1 line)
CHR$(22) Erase from cursor to end of line
CHR$(25) Scroll screen window up 1 line
CHR$(142) Display graphic characters instead of text
CHR$(143) Set bottom right corner of window
CHR$(149) Insert line (scroll screen below cursor down 1 line)
CHR$(150) Erase from beginning of line to cursor
CHR$(153) Scroll screen window down 1 line

Electronic typewriters that have been adapted for communications, such as I/O Selectrics, can be used with a PET or CBM as long as a special interface is used. This interface provides the PET/CBM with RS232 communications and also converts the PET ASCII codes to IBM's EBCDIC (tilt-and-rotate) codes. A normal electric typewriter cannot be used with a computer, except with an adaptor that sits on top of the keyboard and physically presses the keys, and such adaptors aren't considered reliable.

Q. I recently purchased a PET 2001 8K unit, which I am very happy with. It is fun to use—so much fun that I probably spend too much time on it. Although the unit was sold to me as a 3.0 ROM machine, I have since determined that it has a 2.0 ROM. The manual I reveived with the computer was a CBM User Manual, Model 2001-16, 16N, 32, 32N, dated June, 1979. Is there a more appropriate manual for my 3.0 model. Also, I have tried to code the TIM monitor that is listed in the PET User Manual. I tried to use a BASIC loader, but I found that the monitor listing has no code for the following locations:

$0408 to $040B inclusive
$051C to $052C inclusive
$074D to $0754 inclusive
$0757 to $077F inclusive

I am at a loss as to what should be placed in these locations, and the code won't run without the numbers. Could you tell me where I can find the proper values for these locations.

W. D. English
Orange, CA

A. *If your machine doesn't have the machine language monitor in ROM, it can be obtained on tape from a Commodore dealer. This is certainly more painless than using a BASIC loader. If your monitor is in ROM, typing SYS 4 should set you into it. The "PET/CBM Personal Computer Guide" from Osborne Books is an excellent supplement to any owner's library.*

Continued on Page 4.

Q&A HOTLINE

Q. I am currently teaching high school mathematics and computer programming. We have a Commodore PET (1979 vintage, small, with non-standard keyboard). I am planning on teaching a course in assembly language programming and want to know if the PET has a "mini-assembler." If not, can an assembler be purchased for the PET?

D. Lufkin, San Leandro, CA

A. *There is no mini-assembler in the PET's machine language monitor. There are several commercially available assemblers, including the disk-based "Assembler Development System" from Commodore, the tape-based "4K Assembler" in BASIC from AB Computers, and the disk-based "MAE: Macro Assembler Editor" from Eastern House Software.*

Q. Could you please address these questions: Does the PET have a built-in parallel interface? If so, what is it? Is there a Commodore dealer in Hawaii? I have no idea what peripherals are possible with the PET and what connections I need to hook them up. Could you tell me where I could find this information? Are CBM peripherals compatible with the PET?

E. Wong, Honolulu, Hawaii

A. *The interfaces on the PET and CBM machines are identical: an IEEE-488 bus and an 8-bit parallel "user" port. The IEEE-488 bus is used with standard Commodore peripherals. The parallel port isn't a parallel printer interface, if you're thinking about a Centronics-type parallel interface, but can be adapted to one with software and a cable. There are adapters for PET to RS232, available from Connectivut Micro Computers (CMC) and The Net Works (TNW).*

Currently, there are two dealers in Hawaii:
Computerland/Hawaii
567 S. King St.
Honolulu, Hawaii 96813
(808) 521-8002

PHS Business Systems
1336 Dillingham Blvd.
Honolulu, Hawaii 96817
(808) 841-3443

Q. I need some information on PET tapes. I have a Heathkit System and would like to make an interface to load PET tapes. Can you tell me the PET tape layout, i.e., headerbits, mark & space, baud rate, etc.?

R. Whitten

A. *The following is an excellent description of the PET/CBM tape format from COMPUTE magazine, issue #1, pages 80-81.*

CASSETTE FORMAT REVISITED

Pulse position modulation is used in the PET (cries of 'what's that?'). At regular intervals a byte marker pulse is written on the tape, followed by bit pulses, the elspsed time defining the '0' or '1.' This method has several advantages. Because the bit pulses are referenced to the byte marker, data is fairly immune to variations in tape speed. If 8 bits do not follow the marker, there has been an error.

Three times periods are defined: Long (L) = 336 ±5uS (1.49kHz), Medium (M) = 256 ±5uS (1.49kHz), Short (S) = 176 ±5uS (2.84kHz), and these are used to define a Word Marker = LLMM, '1' = MMSS and '0' = SSMM, where LLMM Means: long '1,' long '0,' medium '1,' Med '0.' Now to words, ASCII'A' = 01000001 which, when preceded by the word marker and terminated with odd parity gives

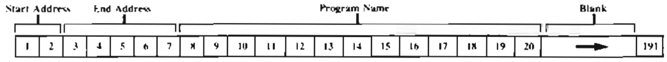
LLMMMMSSSSMMSSMMSSMMSSMM
SSMMMMSSSSMMMMSS
· mkr 100000101 · gives a character length of 8.96mS.

Note 8 bits plus parity to accommodate graphics. Now since it is inefficient to start and stop the tape for each character, they are stored in memory (635-825 or 827-1017 for = 1 or = 2) until sufficient to make up a block of data. Now since data blocks are 191 bytes each a further check is possible (long or short block error). Each block is written twice and if an error is found in the first block, the second is used. Only if the corresponding byte in both blocks is in error can we not recover data.

Now the crunch. The cassette motor takes time to run up to speed and the interblock gap is there to allow for this. If we try to read the tape while the motor is still accelerating, errors are likely to occur. If the first byte or two are not recognized, the block is discarded. Should this contain the mark of end-of-file (EOF) or end-of-tape (EOT) the PET would crash. Owing to a bug in the operating system the inter-block gap is too short . . . It is relatively simple to turn the cassette motor on before the buffer is full and patches for this have been printed in IPUG circulars but most comprehensively in the TIS Workbooks.

Two more definitions: Block end market = LL plus leader, Leader = over 50 cycles of shorts.

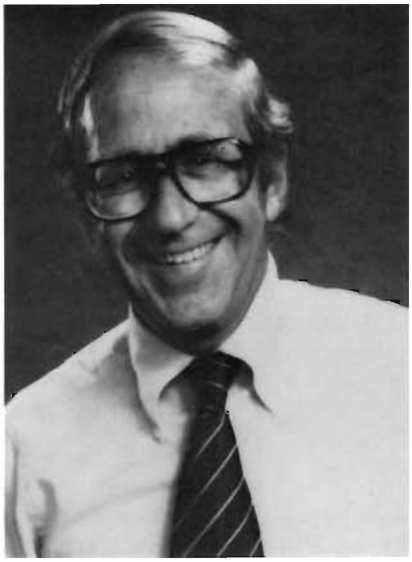
PROGRAM FILE	DATA FILE
Header 192 bytes	Header 192 bytes
Repeated header	Repeated header
Program (one long block)	Data block 192 bytes
	Repeated data block
Repeated prog.	Data block as reqd.
	Repeated as reqd.
End block	
	Data + spaces to end
Repeated end.	Repeat above
	End block
	Repeat end.



ALL BLOCK TYPES

Leader	First Leader—two seconds approximately
Count Down (9bytes)	First pass $89,88,...81 Second pass $09,08,...01
Type	Program, Data, Header
Data Area	If data or header—191 bytes; If program, length of program
Check-Sum	Exclusive OR checksum of above (header type)
Block End Marker	Block end marker, 1 cycle long
Leader	Approximately 450 cycles of leader (0.16 seconds) ■

Message from the President



In life, as in business, prosperity and growth are achieved when the fullest use of skills and energies can be combined to produce an end product called *realized potential.* But we have over 30 million persons in the U.S.A who don't have a full chance to realize their dreams, join the mainstream of society, or fulfill their potential. These are America's disabled.

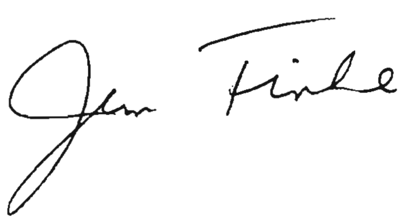
The United Nations has declared 1981 to be the "International Year of Disabled Persons," and various actions are being taken to awaken Americans to this important minority. From a corporate viewpoint, they represent not only a significant segment of the population, but a virtually untapped resource as well.

At Commodore, we feel especially close to the disabled because of our work in pioneering and marketing the microcomputer worldwide. The microcomputer is truly one of the most signigficant aids to enrich the life of the disabled. It expands their capabilities, extends their reach, and, because of the limited size but great power of the micro, it gives the user control—and in the case of the handicapped—more control over more things than previously thought possible.

In December 1981, Commodore will be co-sponsoring (along with Careers for the Disabled, Inc.) the first National Careers for the Disabled Symposium in Baltimore, Md.

We hope that all of the readers of Commodore Magazine, especially those who despite their disabilities have found new hope and new capabilities through the microcomputer, will support this symposium.

If our society is to prosper and grow, and to realize its potential, we are handicapping ourselves by not doing all in our power to harness the talents of previously considered disabled persons. ■



James Finke
President

COMMODORE NEWS

Editor's Notes

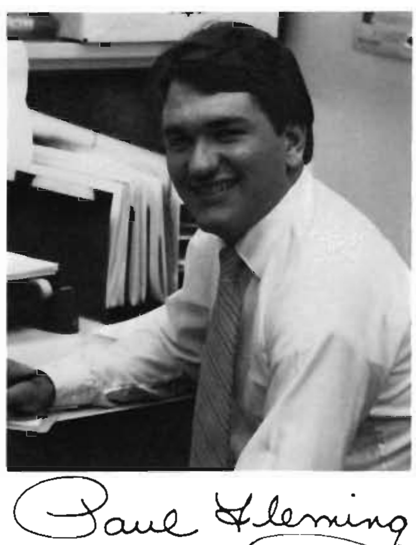
If you have been subscribing to this publication for more than one year, you may be confused or at least amused at its seemingly endless "new looks." In December, 1980, you were reading The PET User Club Newsletter. Then, with the arrival of a new year, a copy of Commodore INTERFACE arrived at your door. And now you are reading COMMODORE Magazine. Although this may be disconcerting to some, we feel it reflects the positive direction of the publication and our company.

Now, we can truly label our publication "the microcomputer magazine." Throughout its pages, you'll discover new and different ideas about Commodore and the microcomputer industry. The magazine contains the most current news on Commodore products and innovations. Articles by Jim Butterfield, and other highly regarded experts, provide up-to-date and useful information on hardware, software, and programming. Enlightening interviews and application articles give you, the reader, added insight into the direction Commodore is heading in the microcomputer industry.

This magazine, like the company from which it takes its new name, is strong and viable, with a message to convey not only to Commodore users, but to the whole microcomputer industry; that's why we call it "THE microcomputer magazine." And, while there's always room for improvement, we have made substantial advances in a relatively short period of time. Since the first edition of INTERFACE was introduced in February, 1981, this publication has more than tripled in size—and quality. One reason is that we have heard—and heeded—our readers' advice.

While you may not care to read every article in this magazine, please remember there are other readers who have requested to see specific information they feel is important and relative. This is a magazine for ALL Commodore users, from the person who has just walked out of a dealer's store with their very first computer, to the seasoned "compuvet" who has followed Commodore since we pioneered the microcomputer field.

Help us to make this your magazine. Take advantage of the reader questionnaire card that is enclosed with each issue. Because, good or bad, we want to hear from you.■



Paul Fleming
Editor

To the Editor

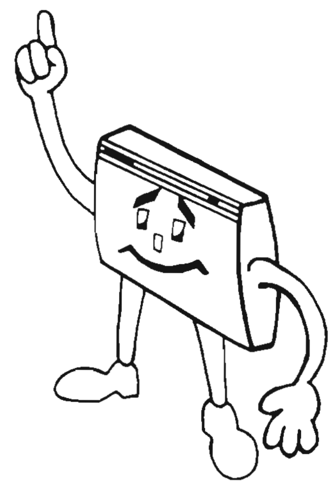
Gentlemen:

On June 30, 1981, we purchased a VIC 20 at Computerland in Dresher, PA. While typing some programs included in the manual, we discovered an error in one program. On page 68, line 130 reads "130 FOR J = 1 TO 2000: NEXTN." It should read "130 FOR J = 1 TO : NEXTJ." Please correct this because it may confuse many of your customers. Thank you.

I really enjoy your VIC and hope to see more of your work soon.

Sincerely yours,
Kara Jean Kortwright

NOTE: *Thanks Kara for your very helpful comments. You'll be happy to know the change has been made in a more recent version of the VIC 20 manual.*■



Commodore Launching Its Biggest Ad Campaign Ever with Star Trek's William Shatner as Spokesperson

Commodore is ready to launch the biggest national advertising campaign in its 23-year history, and has signed internationally known actor William Shatner to help project its message to the consumer and business marketplaces via print and broadcast media.

Even as you read this magazine, the campaign will be breaking, and will include Shatner, known throughout the world as Captain James Kirk, commander of the Starship Enterprise on Star Trek. Shatner will be promoting the full line of Commodore microcomputer products.

The hard hitting message carried by the new Commodore promotional and advertising program will be the many price/performance ratio benefits offered by the company's wide range of high-quality, competitively-priced micros.

Commodore products will be advertised and promoted in consumer, trade, educational, and hobbyist magazines, as well as in newspaper markets throughout the country. The initial phase of the Commodore campaign will run until Christmas, and will also include national radio broadcast messages by Shatner.■

Kit Spencer Named VP of Marketing of Commodore's Computer Systems Division

Christopher ''Kit'' Spencer has been named vice president of marketing of the Computer Systems Division of Commodore Business Machines, Inc.

The announcement was made by Jack Tramiel, founder, chief executive officer, and vice chairman of the board of Commodore International Ltd., the parent company of Commodore.

''Kit Spencer is one of the world's recognized leaders in marketing microcomputer systems,'' said Tramiel. ''He has built an enviable record of achievement in Europe, and we feel confident that he will affect a major impact on Commodore sales here in the U.S.''

In his new position, Spencer will have responsibility for marketing Commodore's line of high quality, competitively priced microcomputer systems. His efforts will be concentrated on sales to dealers and industry, as well as in building a strong base of Commodore user groups throughout the country.

Spencer has been with Commodore for seven years, most recently as European marketing manager. Previously he served as Commodore's general manager for the Computer Systems Division in the United Kingdom, and as marketing manager for calculators in the U.K. Prior to joining Commodore, Spencer held marketing management positions with Philips and Bowmar.■

Attention: All PET Users Clubs

We need you! Commodore is compiling a comprehensive list of all user clubs throughout the country. Once this list is obtained, we will begin forwarding valuable information to clubs on a regular basis, including hardware and software updates, technical bulletins, new product announcements, and troubleshooting tips.

Remember, the Commodore Magazine evolved from an earlier publication known as the PET Users Club Newsletter. While the name of the magazine has changed, its purpose and audience remain the same. We want to share as much information with as many users clubs as possible.

So please, make sure we know who you are. Send your club's name, address, and other pertinent information to:

Commodore Users Clubs
c/o Editor
Commodore Magazine
681 Moore Road
King of Prussia, PA 19406 ■

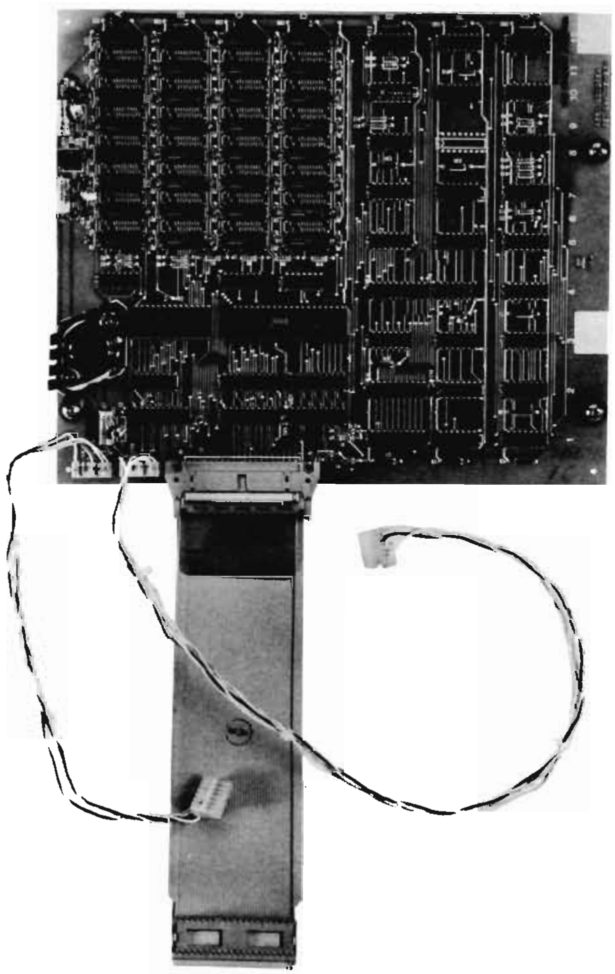
96K-bytes of RAM is now a reality!

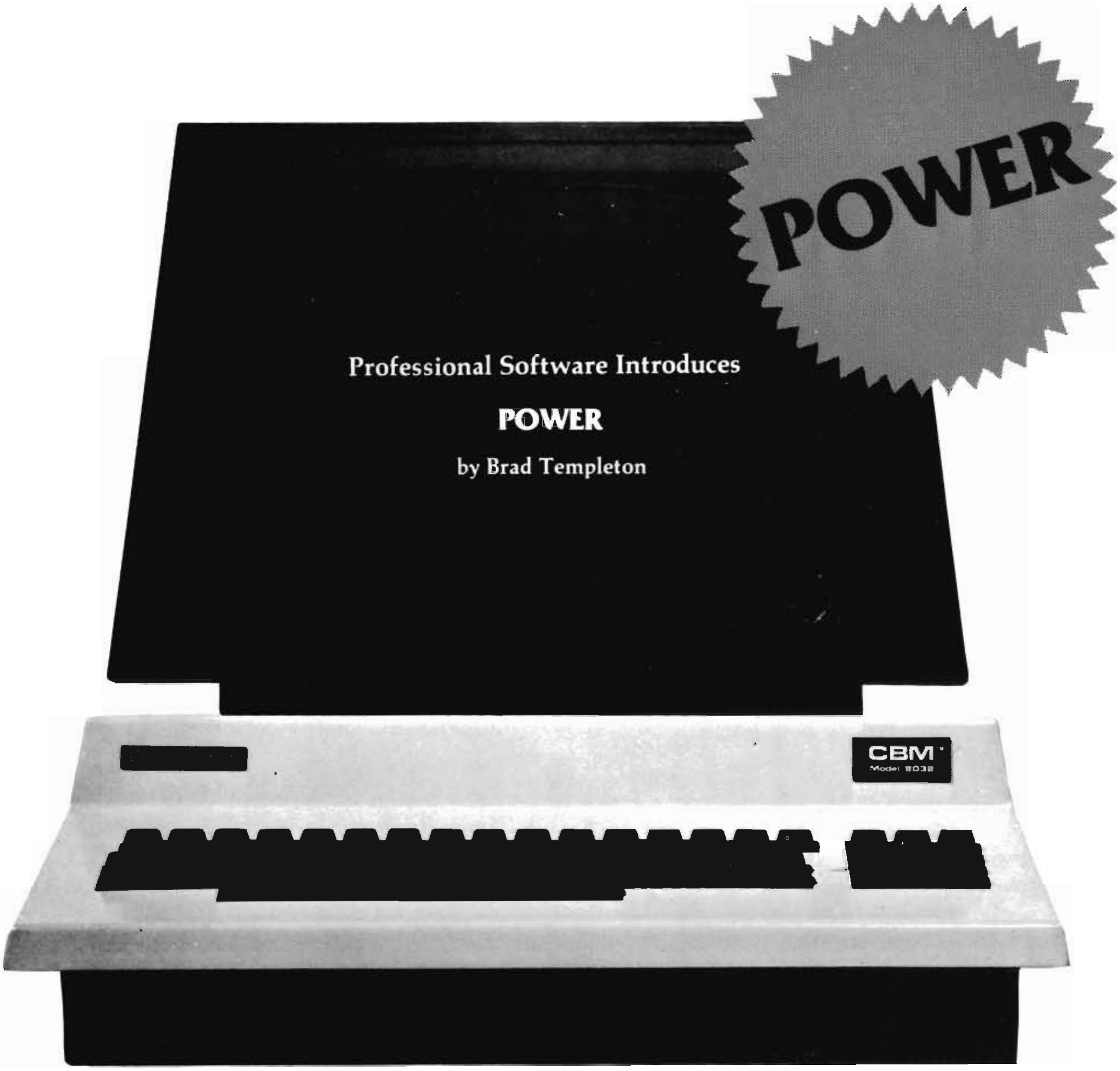
First shipments of the Expansion Memory Board, which adds 64K-bytes of RAM to the CBM 8032, have just been delivered to Commodore dealers. The expansion memory is mapped into main memory in 32K-byte blocks, and one 32K-byte expansion block can reside in memory at one time.

The Expansion Memory Board is mounted in the CBM 8032 console by four mounting brackets, and is connected to the computer by two power cables and one logic cable.

Control of the expansion memory is through a register on the Expansion Memory Board that provides selection of two 16K-byte blocks, write protection, and enabling of the expansion memory. Additional bits in the register allow the programmer to PEEK at I/O and screen memory.

The add-on board retails for $500.00 and is available through authorized Commodore dealers. For more details on the Expansion Memory Board, see the "Excerpts from a Technical Notebook" in this magazine. ■




POWER
Professional Software Introduces
POWER
by Brad Templeton
CBM

New Low-Priced Single Disk Drive Now Being Delivered

Commodore's CBM 2031, a new single floppy disk unit, is now being delivered to Commodore dealers.

Selling for $695.00, the new drive provides a lower price entry point for Commodore computer owners who want to put their data on floppy diskettes, or work with the wide range of computer programs available on that medium. Owners of the new single disk drive may expand their disk system by adding additional 2031's or other Commodore disk drives and running them in tandem.

The CBM 2031 stores up to 170K bytes on a single 5¼-inch floppy diskette, and incorporates an IEEE-488 interface for use with Commodore's PET and CBM computer equipment.

The new drive is based on the same technology used in Commodore's existing 4040 dual disk drive unit and incorporates the latest disk operating system. The 2031 diskettes are read/write compatible with the CBM 4040 disk unit.■

Welcome Aboard!

Congratulations to the following groups, who have joined the lengthening ranks of dealers selling Commodore computer products. . .

The Computer Center, Inc.
433 Valley Ave.
Birmingham, AL 35209
205-942-8567

Microage Computer Center
101 East 9th Ave. Suite 9B
Anchorage, AK 99501
907-277-8515

City Business Machines
6800 Interstate 30
Little Rock, AR 72209
501-568-4044

Computerworld
1375 Mission St.
San Francisco, CA 94103
415-431-0844

Exectec
1725 S. Bascom Ave. #209
Campbell, CA 95008
408-377-2600

Med Data Systems
348 Hilgard Ave. Box 24885
Los Angeles, CA 90024
213-279-2450

Net Profit
2908 Oregon Ct. Bldg. G1
Torrance, CA 90503
213-320-0756

Palos Verdes Reading Ctr.
716 Yarmouth Rd. #203
Palos Verdes Est, CA 90274
213-377-1404

Platt Music
3600 "C" Thomas Rd.
Santa Clara, CA 95050
408-988-0301

Repo Depo
1669 Old Bayshore Hwy
Burlingame, CA 94010
415-692-5000

Scientific Resources
22634 Oak Canyon Rd.
Salinas, CA 93908
408-242-4484

Micro Age Computer Store
27605 S. Harvana St.
Aurora, CO 80014
303-696-6950

Business Machine Sales, Inc.
425 Tamiami Trail, North
Naples, FL 33940
813-262-5960

Computerland/Tallahasse
1815-6 Thomasville Rd.
Tallahasse, FL 32303
904-224-9341

Taylor Business Machines
3394 NE Jacksonville Rd.
Ocala, FL 32670

Computerland/Topeka
911-A West 37th
Topeka, KS 66611
913-267-6530

Tech Hi Fi
48 Teed Dr.
Randolph, MA 02368
617-961-5700

Central School Supply Co.
Box 54366, 310 Airport Rd.
Jackson, MS 39208
601-932-1901

Computer Technology Assoc.
1704 Moon N.E. #7
Albuquerque, NM 87112
505-298-2140

Computer Technology Assoc.
1675 Lohman #1
Las Cruces, NM 88001
505-524-4600

Computerland/Massena
40-50 Main St.
Massena, NY 13662
315-769-9971

Dumac Cash Register Co., Inc.
1628 Erie Blvd. East
Syracuse, NY 13210
315-472-4531

Grand Central Camera
420 Lexington Ave.
New York, NY 10017
212-986-2270

J. Cameron Assoc., Inc.
29 Goodway Dr.
Rochester, NY 14623
716-473-4590

Upstate Computer Shop/Capit
1823 Western Ave./Guilerland
Albany, NY 12203
518-456-3019

Computer Services, Inc.
401 West Main, #460
Norman, OK 73069
405-364-5305

Computerland/Portland
327 SW Morrison St.
Portland, OR 97208
503-295-1928

Computer Specialties, Inc.
831 Lancaster Dr., NE #330
Salem, OR 97301
503-399-0534

Microage Computer Store
1517 NE 122nd Ave.
Portland, OR 97220
503-256-4713

Sirco Business Computer
401 E. 10th Ave. #1B
Eugene, OR 97401
206-343-9296

Robec, Inc.
Technology Center, Route 309
Montgomeryville, PA 18936
215-628-4960

The Stereo House
Route 15
Lewisburg, PA 17837
717-524-9182

Alamo Data Systems
7070 San Pedro
San Antonio, TX 78216
512-821-5892

Computer Technology Asso.
San Miquel Sq. Space 115
Midland, TX 79703
915-699-5046

Computer Technology Assoc. E.
9530 Viscount
El Paso, TX 79925
915-593-6655

Computer Technology Assoc. W.
118 Castellano
El Paso, TX 79912
915-581-3500

The Computer Experience, Inc.
315 Pinewood
San Antonio, TX 78216
512-826-7138

CalPro The World of Computers
3119 Waterick Rd.
Lynchburg, VA 24502
804-237-3825

Computer Corner
27908 85th NW
Stanwood, WA 98292
206-629-3318

Computerland/Casper
138 A. South Kimball
Casper, WY 82601
307-234-2879

The Information Solution To Your Office Productivity Bottleneck!

Now . . . turn your Commodore computer into a powerful business system with Info-2001™ Software Applications.

Info-2001™ Software Applications are designed by expert consultants to meet the needs of the professional practitioner and businessman. All systems have been thoroughly field-tested at many business sites to insure complete functionality.

Professional Solutions!

For the attorney, consultant, engineer, architect, and other professionals . . . a Time Management/Client Billing System with these advanced features:

- Timekeeping and Disbursement Accounting - to keep track of time and expenses by client and matter.
- Client Billing - to print computerized bills on professional billheads.
- Accounts Receivable - to provide Aged Receivables reports for improved "cash flow" control.

The "parameter-driven" design allows you to customize the system to conform to the specific practices of your office. You get the best of both worlds—a "field-proven" packaged system (at off-the-shelf prices) with the "flexibility" to meet your specific needs.

Business Solutions!

For the small to medium size business... a comprehensive Management Accounting System with these modules:

- Sales Order Entry
- Inventory Management
- Accounts Receivable
- Accounts Payable
- Payroll
- General Ledger

All software includes an instructionally designed Operator's Manual, functional input forms, and training by your authorized dealer using a computer-simulation model.

For a demonstration of these "Information Solutions" see your local authorized Commodore dealer today!

InfoDesigns

INFORMATION SOLUTIONS USING MICROCOMPUTER TECHNOLOGY

Info-Designs, Inc. • P.O. Box 5340 • W. Bloomfield, Michigan • (313) 582-9090

PET PREDICTS

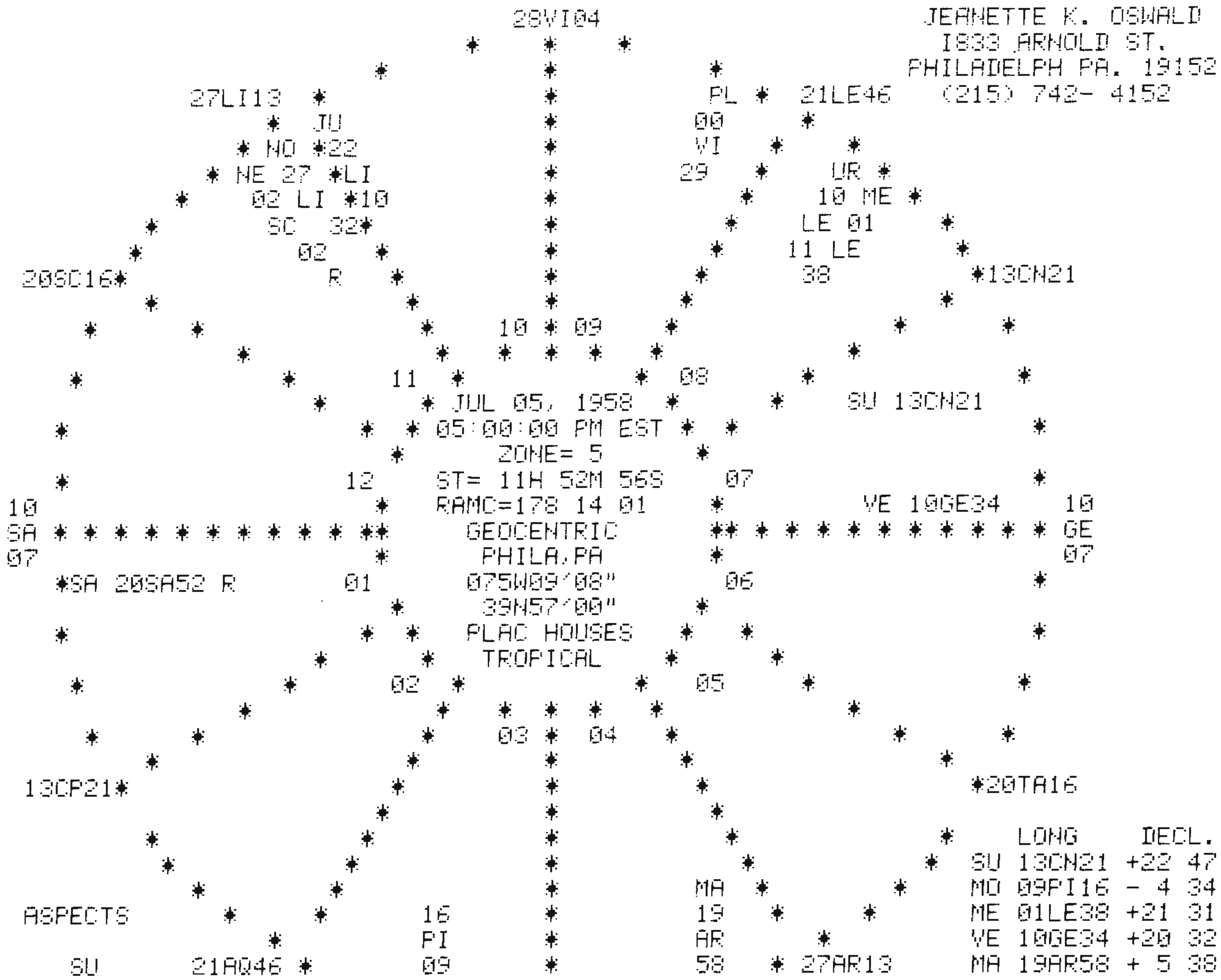
One may wonder what an article about astrology is doing in a computer magazine. After all, astrology has been around for thousands of years—since Biblical times, at least. And computers are, well...relatively speaking, kind of brand new. Yet, Jeanette Oswald, highly-acclaimed Philadelphia astrologer, uses a PET computer, a 4040 disk drive, 2022 printer, and an M-64 package from Matrix Software (Big Rapids, Michigan) to produce astrological charts for her clients.

Mrs. Oswald estimates that her computer saves her approximately two hours of time for each chart she prepares. After inputting three pieces of information about the client, (date of birth, exact time of birth, and location) a chart is produced in 2.5 minutes. This natal chart gives Mrs. Oswald the information she requires to provide the client with feedback. Her expertise in interpreting these charts, and her other areas of background in para-psychology, contribute to an accuracy rate of 85%.

But Mrs. Oswald is not merely an astrologer with individual clients; her involvement with the study of stars has broadened to include the potential of astrology as a diagnostic tool in medicine. Physicians have consulted with her on numerous occasions for this purpose. She has also been involved with astrology as it relates to corporate environments and believes it has much potential in the business world.

Mrs. Oswald does, on occasion, encounter individuals who doubt the validity of her capabilities with astrology; some have even told her they believe astrology is another form of fortune-telling. But with an 85 percent accuracy rate, she does not have far to go to prove her point. ➔

— *Jody Miller*




```
                                                              LONG    DECL.
                                                           SU 13CN21 +22 47
                                                           MO 09PI16 - 4 34
 ASPECTS                                                   ME 01LE38 +21 31
                                                           VE 10GE34 +20 32
   SU                                                      MA 19AR58 + 5 38
MO TRI4A  MO                                               JU 22LI10 - 7 28
ME ...    ...    ME                       CARD FIX MUT     SA 20SA52R-21 45
VE ...    SQR1A  ...    VE                FIRE   1   2   1 UR 10LE11 +18 18
MA SQR7A  ...    ...    ...    MA         EARTH  0   0   1 NE 02SC02R-10 31
JU ...    SES2S  ...    SES3S  OPP2A  JU  AIR    2   0   1 PL 00VI29 +21 58
SA ...    ...    ...    ...    TRI1A  SXT1A  SA  WATER 1 1 1 NO 27LI32 -10 36
UR ...    INC1A  ...    SXT0S  ...    QTL0S  ...    UR     MC 28VI04 +    46
NE ...    ...    SQR0A  ...    ...    ...    SSQ4S  ...    NE        AS 10SA07 -21 58
PL SSQ2A  ...    ...    ...    ...    ...    ...    ...    SXT2A  PL  VT 24CN05 +21 18
NO ...    SES3A  SQR4S  SES2A  ...    CJN5S  ...    ...    CJN4A  SXT3A  NO
MC QTL3S  ...    SXT4A  ...    ...    ...    ...    SSQ3S  ...    ...    ...    MC
AS ...    SQR1S  ...    OPP0A  ...    SSQ3S  ...    TRI0A  ...    ...    SSQ2A  QTL0A  AS
----PF=06LE02-------------360 DEGREE MIDPOINT SORT---------------EP=28SA23----
19AR58= *MA* 06CN06=ME/VE 16LE22=VE/JU 16VI50=ME/NE 20LI18=PL/AS 18SA40=MO/MC
24AR55=MO/VE 09CN01=MA/MC 20LE20=UR/PL 21VI06=UR/NE 22LI10= *JU* 20SA52= *SA*
11TA19=SU/MO 10CN22=VE/UR 20LE43=SU/MC 26VI20=JU/PL 25LI41=SA/PL 00CP43=MO/JU
15TA16=VE/MA 13CN21= *SU* 21LE18=VE/NE 26VI44=SU/AS 27LI06=JU/NE 05CP39=MO/NE
20TA27=MO/ME 20CN32=VE/PL 29LE51=ME/MC 28VI04= *MC* 02SC02= *NE* 21CP04=MA/JU
24TA43=MO/UR 22CN30=SU/ME 00VI29= *PL* 01LI15=NE/PL 04SC06=MC/AS 24CP42=MO/AS
01GE40=SU/MA 26CN46=SU/UR 02VI46=SU/JU 02LI06=SU/SA 09SC28=SA/MC 26CP00=MA/NE
04GE53=MO/PL 01LE38= *ME* 04VI08=UR/MC 05LI53=ME/AS 16SC09=JU/AS 00AQ04=MO/SA
10GE34= *VE* 04LE19=VE/MC 07VI41=SU/NE 10LI07=JU/MC 21SC04=NE/AS 15AQ03=MA/AS
10GE48=ME/MA 05LE54=ME/UR 10VI20=VE/AS 10LI09=UR/AS 21SC31=JU/SA 29AQ25=MA/SA
15GE04=MA/UR 06LE55=SU/PL 11VI54=ME/JU 11LI15=ME/SA 25SC27=SA/NE 09PI16= *MO*
25GE14=MA/PL 10LE11= *UR* 14VI17=PL/MC 15LI03=NE/MC 10SA07= *AS* 15PI43=VE/SA
26GE58=SU/VE 16LE04=ME/PL 15VI10=JU/UR 15LI31=SA/UR 15SA29=SA/AS 23PI37=MO/MA
```

...An example of a natal chart done by Mrs. Jeanette Oswald. This particular chart shows that the person was born on July 5, 1958 at 6:00 PM E.D.T. in Philadelphia, Pennsylvania.

Columbus, Ga.-Based Commodore Dealer

Using, Programming, Selling Home Computers Changes Life of Quadriplegic Gene Spalding

In July of 1979, Gene Spalding had the world on a string. Just past his 31st birthday and 10 years out of Cornell University where he had earned a B.S. in Electrical Engineering, he had a partnership in a successful telephone consulting business in Columbus, Georgia, owned a home with a swimming pool, had three cars, and had just been married. And his company had just made its biggest sale ever, a $150K deal!

But then, on July 31st, just five days after returning from London where he had honeymooned with his wife Starr, disaster struck. Gene dove into the pool at his home and broke his neck.

''All I remember is that I woke up under water and couldn't move,'' said Gene, ''Luckily, Starr and my business partner were there, jumped in, and saved me. I spent the next year or so bouncing in and out of hospitals, and, quite obviously, it made a major change in our lives.''

The ''major change'' Gene speaks of was certainly that. A quarter-inch break in his spinal cord at the neck resulted in his being paralyzed from the neck down with only minimal movement of his right arm. Now a C-4 level quadriplegic rather than an on-the-go business person, he faced a crisis and was at a crossroad few of us ever see.

''If the year in the hospital was bad, the next year was even worse,'' Gene lamented. ''I sat around the house in a wheelchair or in bed, watching soap operas on TV, generally feeling sorry for myself, and dreaming about the life that I could no longer—or so I thought—lead.

But then, in the fall of 1980, another ''major change'' took place in the life of Gene Spalding.

''My vocational rehabilitation counselor and I discussed the possibility of a home computer—a microcomputer—and I said that if I could get my hands on one, I might be able to utilize my knowledge of the telephone industry and possibly create a job for myself,'' Gene said.

''I started working with the computer, and, at the same time, I became aware of a course being given by the Georgia Institute of Technology in Atlanta called the 'Georgia Computer Programming Course for the Severely Handicapped.' They take people such as myself and teach them COBOL programming for mainframe computers. They allowed me to take that course via 'remote control' down here by linking my home computer with the big computer at Georgia Tech. Outside of that course, I've been completely self-taught on the micro.

''Through Dennis Pellerin, my friend and now my business partner, I started using a Commodore CBM microcomputer and was able to write some BASIC programs to take information from software-controlled telephone systems, process the data, and give reports back to management. I think it's a very good service, and one I was able to use to literally get back in business. I've sold the service to Tom's Foods, which is a division of General Mills, and I've sold it also to the Columbus Health Department. Finally, after a horrendous two years, I had things going my way, at least a little bit.''

Another plus came from Gene's use of the microcomputer as well. ''Because of his almost never-ending typing on the computer with a 'typing pen',' said Starr, ''he started regaining some strength and movement in his right arm. Certainly he had made physical progress as a direct result of using the micro.

Having taken an important first step on the road to vocational as well as physical and emotional rehabilitation, Gene, his wife Starr, and the Pellerins, established a business to utilize Gene's new found skill and devotion to the Commodore micro. The Pellerins, Dennis and his wife Maria, who already had an on-going urban planning business in Columbus, and the Spaldings, established Computer Systems, Inc., a Commodore microcomputer dealership, in this city some 90 miles southwest of Atlanta. Oddly enough, the signing of the business partnership agreement papers was around the 4th of July, giving Gene Spalding a truer meaning of ''Independence Day'' then he ever dreamed possible. ➔

COMPUTER SYSTEMS, INC., the Commodore dealership in Columbus, GA, is run by two unique and hard-working couples. Standing, from left, are Maria and Dennis Pellerin, and Starr Spalding, while Gene Spalding is in his motorized wheelchair.

''Now I'm in charge of the computer sales part of the CSI operation,'' said Gene, ''and we're doing well selling Commodore CBM 8000 series micros. We're offering Legal Time Accounting (LTA) packages, business accounting programs, word processing, VisiCalc financial forecasting, and educational systems. We're also very excited about offering Commodore's new low-priced VIC 20, which is a great entry-level computer. It's a great way to get into computing for under $300, and it truly excites me to think how much the computer can mean to others because of what it has meant to me.''

The Spaldings have just purchased a new ''old'' home, a 150-year-old Victorian which is being modified to meet Gene's needs. ''We're using micros around the house as well,'' said Gene, to do things like turn on lights, dial the phone, and so on. And I'm using Scott's voice entry system which is a great help to handicapped persons.

''From a handicapped person's point of view, being able to once again control things is simply phenomenal. I use the home computer to work the phone, the lights, and to activate a digital plotter which allows me to draw,'' said Gene, why I can't tell you how much it has helped.

''And,'' says Gene, ''both as a Commodore dealer and as a handicapped person, I'm very proud that Commodore is co-sponsoring the first National Careers for the Disabled Symposium in Baltimore in December. You can bet that Starr and I will be there to meet with attendees from throughout the United States, to tell them how we worked hard together to establish, or should I say re-establish ourselves and our lives. In truth I must say that without the microcomputer things just wouldn't be the same—they'd be an awful lot worse!''

Truely, Gene Spalding's story is an inspirational one. But he doesn't see it that way. He's a rare combination of dreamer and realist, and his use of those two qualities, which have helped him to utilize 21st century computer technology to help himself in the 1980s, make him a very special person indeed. ■

—Dave Kaminer

DIAGNOSIS: Ineffective accounts receivable and billing system
PROGNOSIS: Continued decline under present system
Rx: Commodore's Medical Accounting System (MAS)

Many doctor's offices today are faced with the same complexities encountered by major corporations worldwide—mounting paperwork. With the burden of insurance forms and financial procedures established by the medical community over the last decade, even the smallest physician's office needs a fast, flexible, and inexpensive accounts receivable and billing system. In the past, doctors have had a choice of numerous medical billing systems, all of which boasted their own particular features and advantages. Yet, an obvious reluctance still exists on the part of physicians to make the commitment to computerizing office operations. One of the obstacles in reaching a decision is cost. Doctors do not want to buy a mini-computer system at a cost of $40,000 when they are terribly unsure of the results it will achieve. Office staffs also become intimidated at the thought of a computer handling information that has always been their responsibility.

Commodore's Medical Accounting System (MAS), developed by Cimarron Corporation of California, can make the same claims as many of the automated systems on the market today—cost effective, performing receivables, billing, aging, and revenue analysis. However, it can also make a much more exclusive claim that many of the competitive systems fall far short of—low cost.

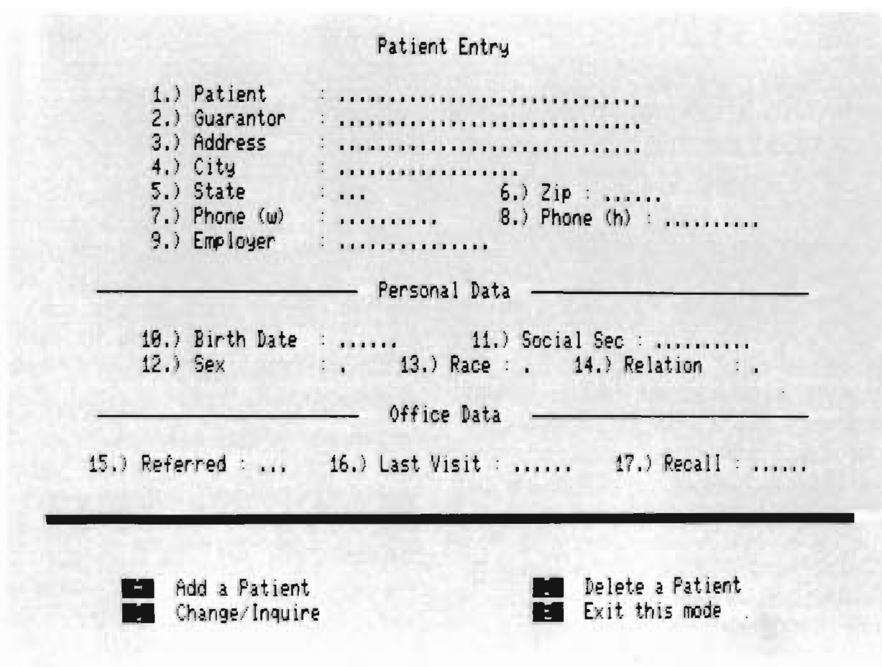
Incorporating the MAS software, a total medical accounting system, which includes a Commodore 8032 computer, 8050 disk drive, and a properly interfaced 4022 or ASCII printer, is available at a fraction of the cost of most automated billing systems.

MAS is actually a series of programs that are linked together under one Mas- →

ter Menu, or main selection program. This menu allows the operator to select specific programs to accomplish daily, weekly, and monthly office procedures. Imagine having an "instructor" stay with your operator throughout the business day, giving easy-to-follow instructions, pointing out errors, and assuring the operator that a procedure has been performed successfully. In essence, MAS provides the same service, by giving clear step-by-step instructions to the operator, and making it easy to select each desired function and perform it correctly. In fact, any task, whether creating cash and billing journals, generating aged trial balances, or printing statements, becomes much simpler than performing it manually. Little or no bookkeeping education is even needed to use MAS.

The master menu of MAS allows you to choose from the following functions:

1. Patient File Maintenance
2. Guarantor File Maintenance
3. Work Ticket Entry
4. Quick Entry of New Patients
5. Cash Receipts/Adjustments
6. End of Day Post
7. End of Month Post
8. Insurance Forms
9. Statements
10. Reports
11. Utilities
12. Diskette Backup
13. Interface Menu



Basically, MAS will lead the operator through each step of the chosen function until it is successfully completed. It will then return the operator to the Master Menu screen to choose another procedure.

For example, choosing number 4 on the Master Menu will allow the operator to perform quick entry of patients. This is a convenient program that allows the operator to quickly enter a new patient/guarantor during hectic business hours, without the necessity of moving back and forth through the main menu. All information is entered in a "linked" fashion, resulting in substantial time saving. Assume a patient visits your office for the first time. The operator enters the patient's relevant personal data into MAS, which accepts the information and automatically, without operator intervention, provides a screen to complete the necessary guarantor data. Upon acceptance of guarantor data, the system provides a Work Ticket Entry screen, which is used to enter the treatment, diagnosis, and insurance information for the new patient.

The whole MAS package is based on this simple but extremely effective "Ticket Entry" program, which allows you to proceduralize daily operations by processing all charges and payments through the use of a "traveler." The traveler is a work sheet containing procedure and diagnostic codes that can be custom-designed for each practice. A physician, after performing a service, completes the traveler form by indicating the procedure and the diagnosis. The doctor then submits the completed form to the operator, who enters the information into MAS. As patients complete treatment, their files are updated through this Ticket Entry program. Simultaneously, receivables and all ancillary reports are also made current.

The Total Picture

These features just begin to scratch the surface of the benefits derived from using MAS. The system does more than just maintain patient/guarantor information. It displays guarantor's accounts receivable information, from current balance to 90 days overdue. Up to 500 active patients and 400 guarantors can be maintained at one time. The work ticket entry procedure allows for complete insurance billing, multiple place of service, diagnostic and procedure codes and two fee structures with override. Separate programs are available for cash and adjustments entry which, together with ticket entry, result in the automatic posting of a daily Charges and Payments Journal. MAS, which never stops making your job easier, even creates the practice's day sheet as well as the bank's daily deposit slip by payment category.

Insurance Forms

MAS prints the "Superbill" and generates an industry standard insurance form—Medicare, Blue Cross, etc. Also, aged trial balances are run prior to statement generation to ensure careful management of cash flow. These insurance forms can be printed on a one-by-one basis at the time of a patient's visit.

Reports

To assist in office management, nine reports are available to reflect the health of your practice. These reports include:

- Procedure Code Report, with month-to-date and year-to-date totals.
- Diagnostic Code Report, featuring month-to-date and year-to-date activity totals.
- Patient Referral Report, with date of last visit.
- Guarantor Aging Analysis.
- Patient Recall Report—by period.
- Activity Report by Treating Doctor, featuring month-to-date and year-to-date revenue.
- Activity Report by Referring Doctor.
- Activity Report for Outside Services.
- Activity Report by Payment Types.

These reports, along with the versatile procedures available through MAS, combine to bring the doctor what any true businessman desires—complete and accurate control of receivables, billing, aging, and revenue analysis. Of course, there are alternatives—even successful ones. Yet, these solutions may include overhead increases, expensive outside services, and minimal control over an important investment—a physician's livelihood. MAS is the true exception—providing a legitimate business solution at a minimal cost. →

The MAS package can be purchased from any authorized Commodore dealer. Retail price of MAS is $595.00.

JUST WHAT THE DOCTOR ORDERED:
Selections from the MAS Menu

Patient File Maintenance allows you to add, change, inquire about, or delete patient information. This patient information includes pertinent data such as: address, city, state, ZIP code, phone number, employer, personal information (birth date, sex, etc.), and office information (patient's last visit date, recall date, etc.)

Guarantor File Maintenance provides you with an accurate record of guarantor information for each patient. Data is entered in much the same way as the Patient File Maintenance procedure. In addition, this menu selection lets you enter receivables data, including insurance, and credit codes for the guarantor. The credit code will identify the guarantor's credit status as good, questionable, or poor risk.

Work Ticket Entry is the nucleus of MAS and is used frequently throughout the business day. This menu selection allows you to proceduralize daily operations. Patient and guarantor data can be added, inquired about, changed, and deleted easily. All the necessary information for standard services are provided with the Work Ticket Entry selection, including: physician's name, procedure performed, diagnosis, location code, and fees.

Quick Entry of New Patient is a convenient selection that allows the operator to easily enter a new patient/guarantor during normally hectic business hours, without the necessity of moving back and forth through the main menu. This results in time and key stroke savings.

Cash Receipts Adjustment provides you with a convenient method of recording payments and accounts receivable adjustments. The information gathered with this menu selection is used to generate an audit trail, and may also be used as a convenient deposit slip for use with your bank.

End of Day Post is an automatic feature of MAS. When posting receivables or general ledger in a manual system, the bookeeper must be accurate, while handling some figures three or four times. Now, MAS completely eliminates the posting job by letting the system do it for you. At the same time, MAS generates a "daily journal," which provides you with substantial daily post material for your files.

End of Month Post is similar to End of Day Post except that it automatically updates all the year-to-date files.

Insurance Forms offers an alternative to printing insurance forms at point of service. So if a patient is carried on assignment, or it is just more efficient to print all insurance forms at one time, this menu selection may be used.

Statements allows you to print statements for each guarantor, with active account balances.

Reports permits you to retrieve numerous information that is stored by MAS. Each report addresses a different segment of the practice, including: patient list, guarantor list, guarantor accounts receivable aging report, procedure code report, diagnostic code report, physician report, referral report, insurance report, location report, outside services report, payment type report, activity reports, and recall reports. These reports provide timely information about the practice, while helping to identify trends, both from a medical perspective and from a revenue standpoint.

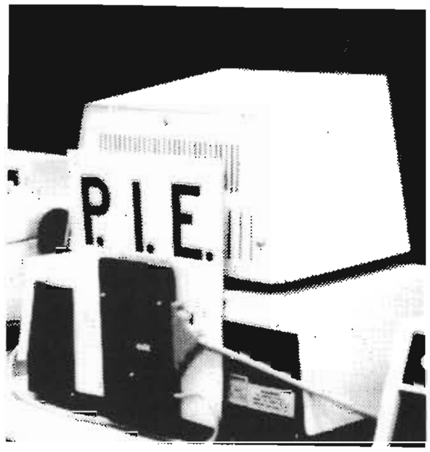
Utilities lets you create, and later change, payment, doctor, place of service, diagnostic, and procedure codes.

Diskette Backup lets the user copy diskettes to guard against damage and loss of data.

Interface Menu allows you to link MAS with other systems. Presently, this special function provides for an easy interface to Wordcraft 80, Commodore's word processing package. In the future, MAS will provide for transferring patient file data to Wordcraft 80. ■

—Paul Fleming

PIE-C



PET/CBM* IEEE-488 TO PARALLEL PRINTERS By LemData Products

P.I.E.-C MEANS—Professional design, Indispensible features, Excellent quality and Cost effectiveness. You can't buy a better parallel interface for your PET/CBM.

Our P.I.E.-C will interface your PET/CBM through the IEEE-488 bus to . . . . . .

the NEC Spinwriter, the C. Itoh Starwriter, printers by Centronics, Epson, Anadex, Escon Products, the Paper Tigers by IDS, the MIPLOT by Watanabe, the DIP printers, the AJ-841, the OKIDATA printers, plus ALL OTHER parallel ASCII printers.

Assembled with custom case, CBM-TO-ASCII code converter and appropriate cable, the **P.I.E.-C** is only $119.95 (+ $5 S&H). Md. Res. +5% tax. Specify printer and CBM models.

LemData Products, P.O. Box 1080, Columbia, Md. 21044 Phone (301) 730-3257
*PET/CBM are trademarks of Commodore Business Machines

A Wooden Computer?

Not from Commodore!

So why should the desk look like wood? A pleasant cream and charcoal trimmed desk looks so much better with Commodore systems. One look and you'll see. Interlink desks are right. By design.

PRICE: $299

In short, as Commodore dealers, we won't settle for anything that looks good only in the catalog! Our customers won't let us. They don't buy pictures. And neither should you. This is why **we will let you use one of our desks for a week** and then decide. If for any reason you don't like it, just return it in good condition for a cheerful refund.

The specifications only confirm the obvious:

- Cream and charcoal color beautifully matches the Commodore hardware and blends with your decor.
- An ideal 710 mm (28") keyboard height yet no bumping knees because a clever cutout recesses the computer into the desk-top.
- High pressure laminate on both sides of a solid core for lasting beauty and strength.
- Electrostatically applied baked enamel finish on welded steel legs—no cheap lacquer job here.
- T-molding and rounded corners make a handsome finish on a durable edge that won't chip.
- Knocked down for safe, inexpensive shipment.
- Patented slip joints for quick easy assembly.
- Leveling glides for uneven floors.
- Room enough for a Commodore printer on the desk, yet fits into nearly any den or office niche—H: 660 mm (26") W: 1170 mm (46") D: 660 mm (26").
- Matching printer stand available with slot for bottom feeding.

If your Commodore dealer doesn't carry our desks yet, send a check for $299 and we will ship your desk freight paid!

Name____________________

Address____________________

City__________ **St**_____ **Zip**_____

Interlink, Inc., Box 134, Berrien Springs, MI 49103
Master Charge and Visa welcome. Call our order line:
616-473-3103

The PET as an Aid to Handicapped Students

by
William L. Brown and Mary Schillinger
Miller Elementary School, Brighton, MI

Handicapped children encounter a variety of obstacles when attempting to meet the requirements of a regular classroom. Many of these problems can be overcome with the use of a PET Computer. The PET is being used at Miller School in Brighton, Michigan with students who have a variety of handicaps. Children who are Physically or Otherwise Health Impaired, (POHI), Emotionally Impaired and Learning Disabled, (LD), are among the population of students receiving individualized instruction on the PET. This is a joint project involving the Principal, Learning Disabilities Teacher and Classroom Teachers.

Lightpens are used by students to further reduce the need for a fine motor response.

Problems Encountered by Handicapped Students

Students with poor fine motor and/or visual motor skills often find the large amounts of written work required overwhelming. They become *frustrated* by the physical demand placed upon them and often lose sight of the skill being presented. This results in minimal work output, student and teacher frustration, and slow skill attainment. Motivation for the task is lost and time-on-task becomes drastically reduced.

The average turnaround time for written work in the classroom is approximately three or four days. This feedback time is too slow for the LD students to benefit. The student has often repeated the same mistake several times by then and imprinted the incorrect response rather than the correct one.

Dittos and seatwork assignments are often crowded and visually confusing to the student with poor visual skills. The LD child is not able to discriminate the individual items and becomes confused. When writing an assignment, this child cannot present work in an organized, readable manner.

Classroom management becomes difficult when some students require a great deal of extra time to complete work. The teacher often lacks patience because of conflicts over allocation of time and attention to competing needs.

How the PET is Used

The PET has become a valuable aid in overcoming many of these problems presented above. Where a written response to questions is required, a multiple choice question/answer program is used. This eliminates the need for a written response.

The dexterity required to manipulate the PET keys is minimal. In some instances, fine motor dexterity has been increased through the use of PET games such as ALIENS. In cases where manual dexterity is extremely limited, a "light pen" is used to further reduce the need for a fine motor response.

Items on the PET are presented one at a time. The screen is free from other distracting stimuli, therefore the student with poor visual organization skill does not have difficulty keeping track of where he should be at any given time. The movement of printing and indicators on the screen helps maintain the student's attention and increases time-on-task. When print appears in a left-to-right, top-to-bottom motion, it reinforces that progression in the student.

The novelty of the PET is a primary factor in enhancing motivation. It is a popular teaching tool with students and the handicapped child is envied by his peers because he is able to use it. This goes a long way toward removing the negative attitude which can develop as the result of a handicapped child's perception of being treated "specially."

Immediate feedback is essential for all students and particularly handicapped learners. This is not often possible with regular seatwork. The PET provides item-by-item feedback and allows the students to learn from their mistakes before the lesson is over, thus increasing accuracy and successful learning. The student who takes longer to respond is not penalized. "Patience" is a key aspect of the computer; it can give clues, prompt for a response, and wait almost forever for the students to perform. This removes a great deal of anxiety for some students.

With the PET, a student does not have to produce great volumes of written work to demonstrate mastery of a skill. A representative sample of items covering the skill to be tested can be presented. The PET will detail those skills not mastered and present them again.

Programs and Formats

A multiple choice program is being used for material from reading series workbooks and comprehension questions about a story. The Specific Skills Series by Richard A. Boning (Barnell Loft, LTD 1976) is easily adapted to this type of program. The questions are presented one at a time and errors are reviewed at the end of the program. This provides immediate re-teaching in case of errors.

Vocabulary development exercises are programmed in a similar manner. The content can be derived from the school's reading series, spelling text or some other source such as the *Reader's Digest* "Word Power" series. Isolated words and meanings are used as well as incomplete sentences. A tape recorder or language master may be used in conjunction with these exercises to ensure proper pronunciation.

The well known game of "Hangman" is used for spelling drill on the PET. This requires a knowledge of phonetic patterns and logic in a form of "guess with feedback." Highly interesting stories with common spelling and punctuation errors are also used. One student practiced spelling when items were read to him to be entered into the computer for a multiple choice program. His spelling was proofread and corrected on the PET before the items were recorded on cassettes.

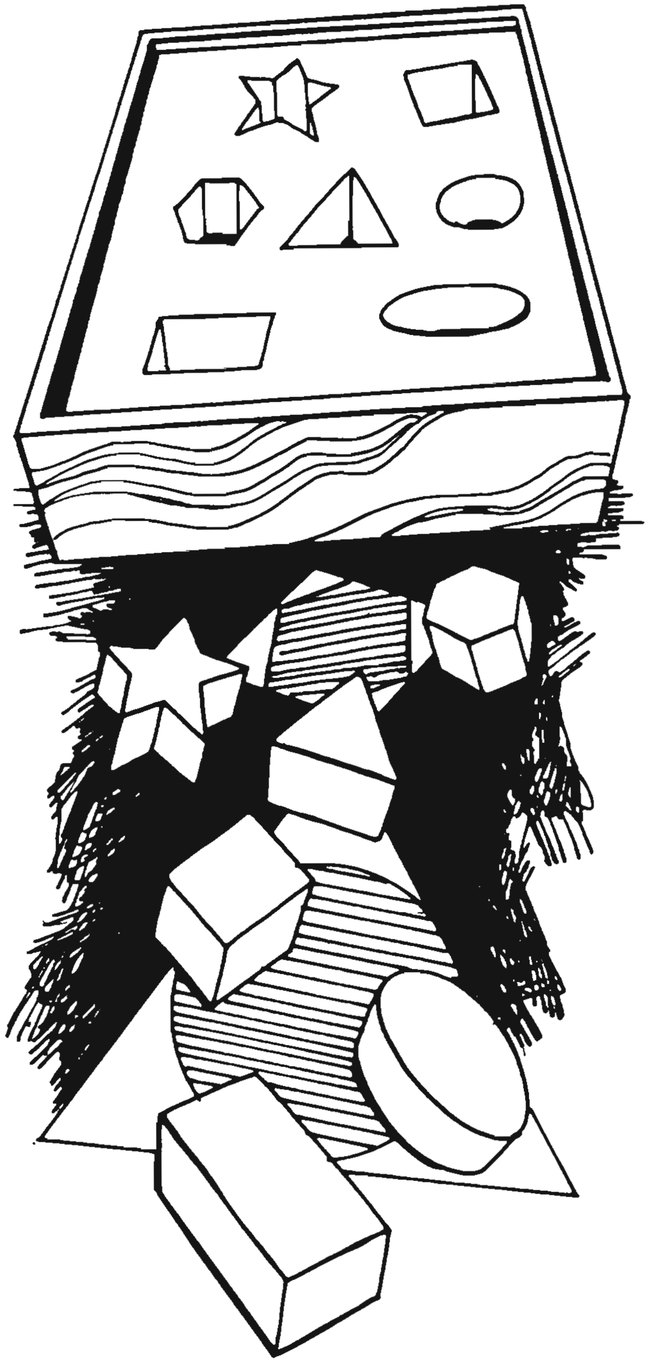
The PET has made the wearisome task of math fact drill challenging and fun. A selection of appropriate levels of difficulty, coupled with a timed exposure of appropriate length, help the student concentrate and independently practice math facts.

Students often become confused on problems requiring borrowing and carrying. The steps must be done in the correct sequence. PET programs for these drills guide the student through each step of the process with the flashing "cursor" as the visual cue. There is also a "Long Division" program which teaches this skill at appropriate levels of difficulty, with cues presented whenever an error is made.

A future article is planned which will present data regarding our research experience with handicapped children. The variables which are being studied include time on task, task completion, accuracy, and short and long term retention of skills.

References

1) Susan Semancik, Manual Alphabet Tutorial on a PET," *COMPUTE.*, Volume 1, Number 3 (March/April 1980)
2) Susan Semancik, "Micros With The Handicapped," *Compute* Magazine, Vol. 3, No. 1 (January 1981)
3) Susan Semancik, "Micros With The Handicapped," *Compute* Magazine, Vol. 3, No. 4 (April 1981) ■



The Delmarva Computer Club

One of the earliest intelligence tests children can be given is the shapes recognition test—the old "fitting the right peg into the right shaped hole in the box" game. This can be used early because it provides non-verbal and non-written responses that give information about the child's shape recognition ability, eye-hand coordination, attention span, reaction time, ability to follow directions and ability to complete a task. For a motor-impaired handicapped child, many of these abilities cannot be tested for by the same methods simply because of the motor coordination prerequisite for the tests. If a child is non-verbal, and can not physically hold or place a peg, the "peg-box" game cannot be used to acquire the same information. This is an area where computer simulation can be advantageously used.

The pegs, as well as the box, can be drawn on the computer's screen, with reverse-fielding for the cutouts in the box corresponding to the shapes of the pegs. A cursor or flashing region can be delayed at a user-determined rate over each peg's representation on the screen. This cursor would continuously cycle over the pegs until a response is received by the computer. If the handicapped child has enough motor coordination to hit any key on the computer's keyboard, this would be enough to select the peg over which the cursor is currently delayed. Other devices such as sip-and-puff switches, cheek switches, eyelid switches, or others suitable for the user's handicap, can also be used to signal a response to the computer through the computers's parallel user's port.

Besides being able to test a handicapped person at an early age, computer simulation can also be used to accurately time each event of the test, and to allow the test to be given without constant supervision. Even non-handicapped users can benefit from computer simulation. For instance, a test can be easily modified or enhanced without having to physically acquire new materials. New board arrangements or different shapes for the "peg-box" game can be implemented by changing only the drawings on the screen. The shapes recognition test could easily be expanded to test for color discrimination, pattern recognition, counting ability, or number concepts. This first part, of a two-part article, presents a "peg-box" program that can be run on any ROM, 40 column screen, CBM/PET computer with at least 8K of memory. The second part of the article will endeavor to show how some of the above enhancements can be implemented on the computer.

We are hopeful that this application for education in general, and for the handicapped in particular, will be useful to many of the readers of this magazine, and that it will encourage others to share ideas about computer applications for the handicapped. The club is currently marketing the Manual Alphabet Tutorial package of seven programs for the PET/CBM computers, to enable anyone to learn how to communicate with the deaf. The package price of $49.95 helps to fund current and future club projects in this area.

—Susan Semancik
Delmarva Computer Club

A student is learning the letter W through one of Delmarva's Manual Alphabet Tutorial programs.

The Delmarva Computer Club has been active for 2½ years in the area of computer aids for the handicapped by participating in national seminars and local demonstrations, by writing a continuing column for COMPUTE! magazine, and by developing computer programs and devices to help the blind, the deaf, and the motor-impaired. These are extensively used in such projects as a human communications workshop for the Marine Science Consortium, which conducts a marine science program for physically handicapped honor high school students from all over the country. The workshop aids the students in learning some basic communications skills necessitated by the variety of handicaps involved in the program.

Besides its work with the handicapped, the club is also actively engaged in computer demonstrations to local schools, groups, and businesses in order to promote computer awareness in the rural peninsula area where it is located. All correspondence concerning the Club should be addressed to the Delmarva Computer Club, c/o Jean Trafford, secretary, P.O. Box 36, Wallops Island, Virginia 23337.

```
100 REM>> PEG-BOX PROGRAM            <<<
110 REM>> COPYRIGHT (C) 1981         <<<
120 REM>> BY SUSAN SEMANCIK          <<<
130 REM>> DELMARVA COMPUTER CLUB     <<<
140 REM>> POB 36, WALLOPS ISLAND,VA <<<
150 :
160 REM *** REVERSE-FIELD BOX AREA ***
170 DATA169,7,133,0,169,128,133,1,162,9,169,160,160,25,145,0,136
180 DATA16,251,24,165,0,105,40,133,0,144,2,230,1,202,16,233,96,-1
190 I=900
200 READOP:IFOP<>-1THENPOKEI,OP:I=I+1:GOTO200
210 REM *** SHAPE TABLE ***
220 REM -1 FLAGS END OF TOP, -2 END OF BOTTOM, -3 END OF SHAPES
230 DATA32,32,233,223,32,32,-1,32,233,160,160,223,32,-2:REM PYRAMID
240 DATA32,160,160,160,160,32,-1,32,160,160,160,160,32,-2:REM RECTANGLE
250 DATA32,233,160,160,223,32,-1,32,95,160,160,105,32,-2:REM HEXAGON
260 DATA32,233,160,160,223,32,-1,233,160,160,160,160,223,-2:REM TRAPEZOID
270 DATA32,32,160,160,32,32,-1,32,32,160,160,32,32,-2:REM SQUARE
280 DATA32,32,32,32,32,32,-1,32,32,32,32,32,32,-2,-3:REM ERASER
290 I=634
300 READOP:IFOP<>-1THENPOKEI,OP:I=I+1:GOTO300
310 I=I+34
320 READOP:IFOP<>-2THENPOKEI,OP:I=I+1:GOTO320
330 READOP:I=I-40:IFOP<>-3THENPOKEI,OP:I=I+1:GOTO300
340 REM *** DRAW NORMAL-FIELD SHAPE ***
350 DATA160,0,177,15,145,0,200,192,6,208,2,160,40,192,46,208,241,96
360 REM *** DRAW REVERSE-FIELD SHAPE ***
370 DATA160,0,177,15,73,128,145,0,200,192,6,208,2,160,40,192,46,208,239,96,-1
380 I=826
390 READOP:IFOP<>-1THENPOKEI,OP:I=I+1:GOTO390
400 M=6+9*SGN(PEEK(50003)):DIMV(5),P(5),B(5),H(5),L(5),J(5):X=RND(-TI)
410 IFM=15ANDPEEK(50003)>1THENM=233
420 POKE829,M:POKE847,M
```

```
430 FORI=1TO5:V(I)=I:P(I)=I:READB(I):H(I)=INT(B(I)/256):L(I)=B(I)-256*H(I)
440 NEXTI:N=5:DATA32817,33057,32945,33073,32833:REM SHAPE POSITIONS IN BOX
450 REM *** DE CONTROLS DELAY TIME ***
460 DE=100:DC%=DE/5:NT=0:NR=0
470 FORI=1TO5:READJ(I):NEXTI:DATA5,3,3,3,5:REM MOVEMENT STEPS
480 FORI=1TO5:P(I)=I:NEXTI
490 REM *** DRAW BOX ***
500 PRINT"□";:SYS900
510 REM *** DRAW SHAPES IN BOX ***
520 FORI=1TO5:POKE0,L(I):POKE1,H(I):POKEM,122+(I-1)*6:POKEM+1,2:SYS844:NEXTI
530 REM *** MIX SHAPES IN ANS. AREA ***
540 FORI=5TO1STEP-1:A=INT(I*RND(1)+1):X=P(I):P(I)=P(A):P(A)=X:NEXTI
550 REM *** DRAW SHAPES IN ANS. AREA ***
560 FORI=1TO5:POKE0,153+8*(V(I)-1):POKE1,131:POKEM,122+(V(P(I))-1)*6
570 SYS826:NEXTI
580 FORS=1TO5:T=1
590 REM *** PICK BOX SHAPE TO MATCH ***
600 POKE0,L(S):POKE1,H(S):POKEM,122+(S-1)*6:GOTO640
610 REM *** SHOW BOX SHAPE TO MATCH ***
620 POKEB(S)+43,127:RETURN
630 POKEB(S)+43,255:RETURN
640 ONTGOSUB620,630:T=3-T
650 REM *** CURSOR DELAY & POSITION ***
660 I=1
670 IFP(I)=0THEN790
680 P=33569+8*(I-1):FORJ=0TO5:POKEP+J,104:NEXTJ
690 D=1:J=1:POKEP,98
700 REM ***TO INPUT THROUGH USER'S PORTREPLACE GET WITH PEEK(59471) IN 370,395
710 GETA$:IFA$<>""THEN820
720 J=J+1:IFINT(J/DC%)=J/DC%THENPOKEP+D,98:D=D+1:ONTGOSUB620,630:T=3-T
730 IFJ<=DETHEN710
740 J=1
750 GETA$:IFA$<>""THEN820
760 J=J+1:IFJ<DC%THEN750
770 REM *** ERASE CURSOR ***
780 FORJ=0TO5:POKEP+J,32:NEXTJ
790 I=I+1:IFI<6THEN670
800 GOTO660
810 REM *** USER SELECTS SHAPE ***
820 POKEB(S)+43,32:FORJ=0TO5:POKEP+J,32:NEXTJ:NT=NT+1
830 REM *** ERASE SELECTED SHAPE ***
840 POKE0,153+8*(I-1):POKE1,131
850 REM *** ALIGN SELECTED & BOX SHAPES ***
860 SP=122+(V(P(I))-1)*6
870 REM *** MOVE SHAPE UP TO BOTTOM OF BOX - ERASE, DRAW, DELAY ***
880 A=160:QL=209+8*(I-1):QH=130:FORK=1TO3:POKEM,122+5*6:SYS826
890 POKE0,QL:POKE1,QH:POKEM,SP:SYS826
900 FORJ=1TO5*DE:NEXTJ
910 E=QL:F=QH
920 QL=QL-A:IFQL<0THENQL=QL+256:QH=QH-1
930 NEXTK
940 IFI=STHEN1020
950 R=SGN(S+.5-I)
960 REM *** MOVE SHAPE LEFT OR RIGHT - ERASE, DRAW, DELAY ***
```

```
970 FORK=I+RTOSSTEPR
980 POKEM,122+5*6:SYS826
990 POKE0,145+8*(K-1):POKE1,129:POKEM,SP:SYS826
1000 FORJ=1TO5*DE:NEXTJ:NEXTK
1010 REM *** MOVE UP TO TOP OF BOX ***
1020 E=145+8*(S-1):F=129:K=4:IFJ(S)=3THEN1130
1030 A=160:IFK=5THEN1050
1040 A=120:IFJ(S)=5THENA=200
1050 E=E-A:IFE<0THENE=E+256:F=F-1
1060 TT=0:GOSUB1080:GOTO1120
1070 REM *** ERASE, (DELAY), DRAW, DELAY ***
1080 POKEM,122+5*6:SYS826
1090 IFTTTHENFORJ=1TO3*DE:NEXTJ
1100 POKE0,E:POKE1,F:POKEM,SP:SYS826
1110 FORJ=1TO5*DE:NEXTJ:RETURN
1120 K=K+1:IFK<6THEN1030
1130 IFV(P(I))=STHEN1330
1140 REM *** IF NO MATCH                              - FLASH WRONG SHAPE.
1150 TT=1:FORK=1TO3:GOSUB1080:NEXTK
1160 REM - MOVE SHAPE DOWN (ERASE,DRAW, DELAY).
1170 FORK=J(S)TO2STEP-1:POKEM,122+5*6:SYS826
1180 A=160:IFK=4THENA=200
1190 E=E+A:IFE>255THENE=E-256:F=F+1
1200 POKE0,E:POKE1,F:POKEM,SP:SYS826
1210 FORJ=1TO5*DE:NEXTJ:NEXTK
1220 REM - MOVE SHAPE LEFT OR RIGHT     (ERASE,DRAW,DELAY).
1230 IFI=STHEN1280
1240 FORK=S+SGN(-R)TOISTEP-R:POKEM,122+5*6:SYS826
1250 POKE0,209+8*(K-1):POKE1,130:POKEM,SP:SYS826
1260 FORJ=1TO5*DE:NEXTJ:NEXTK
1270 REM - MOVE BACK INTO POSITION.
1280 POKEM,122+5*6:SYS826
1290 POKE0,153+8*(I-1):POKE1,131:POKEM,SP:SYS826
1300 REM - KEEP LOOKING FOR A MATCH. ***
1310 GOTO790
1320 REM *** IF MATCH                                 - FILL BOX SHAPE.
1330 POKEM,122+5*6:SYS826
1340 POKE0,L(S):POKE1,H(S):POKEM,122+5*6:SYS826:SYS844
1350 FORJ=1TO5*DE:NEXTJ
1360 REM - ELIMINATE SHAPE FROM CHOICE.
1370 P(I)=0:NR=NR+1
1380 REM - CONTINUE WITH NEXT BOX SHAPE. ***
1390 NEXTS
1400 PRINT"█████████████NUMBER TRIED ="NT,"NUMBER RIGHT = "NR,"SCORE =";
1410 PRINTINT((NR/NT)*100+.5)
1420 PRINT"█          █HIT ANY KEY TO CONTINUE█"
1430 PRINT"████████████████████/⌒\\█████████/@@@@@\█████████@ O O @";
1440 PRINT"█████████@\ - /@██████████@@\—/@@███████████@@@\_/@@@TT███████";
1450 T=1
1460 GETA$:IFA$<>""THEN480
1470 T=3-T:ONTGOTO1480,1490
1480 PRINT"● ●████";:GOTO1460
1490 PRINT"O O████";:GOTO1460
READY.
```

The Fortune Society—Revisited

Victor B., a 19 year-old who came to the Fortune Society last year, is tutoring other teenagers who have been in trouble. He is here with his student Luis P. (right).

The Fortune Society, the New York city-based tutoring and counseling center for ex-offenders which was featured in a recent issue of INTERFACE (May 1981), has made a few more innovative advances in the brand-new technology of utilizing computers within the criminal justice system.

Fortune offers several types of programs aimed at motivating young people who, at one time or another, experienced little more than failure in school. One of these programs, "Playing to Win," developed in 1979 by Antonia Stone, is receiving national acclaim for its success. Playing to Win utilizes PET Computers, combined with one-on-one tutoring, to help reinforce basic math, verbal, or problem-solving skills. The participants are ex-offenders and residents of New York's juvenile detention facilities.

Playing to Win's most recent innovation involves initiating a similar program at Spofford Juvenile Center, a secure detention facility located in New York City which currently houses 192 delinquents and juvenile offenders awaiting sentencing. The Playing to Win program at Spofford will be a joint project of the Fortune Society and the Juvenile Justice Department of the City of New York.

Under the direction of Ms. Stone, several participants of the Fortune Society's Playing to Win program were chosen to staff the program at Spofford. One of the staff members, Victor B., was featured in the May issue of INTERFACE. At that time, he was unsure whether or not he would be selected for the program. But hard work and determination at Fortune as a Counselor-in-Training paid off, and he was among the 10 chosen—from a group of 82—to go to Spofford to assist in the program. An ex-offender himself, his job at Spofford includes tutoring the inmates in math, verbal, and problem-solving areas.

The program, which began on July 23, has never before been attempted in a city prison, and certainly not using a staff consisting of ex-inmates. According to David Rothenberg, executive director and founder of Fortune, both Playing to Win and the program at Spofford exemplify the philosophy of Fortune; that "something can work" in our criminal justice system. "We recognize that no one blueprint can be used for all people, since everyone is different," said Mr. Rothenberg. "Many of the staff at the Fortune Society have been in prison and recognize that the institutional approach assumes that all persons convicted of a crime will respond in exactly the same manner," he said.

"We have attempted to create an atmosphere which permits people to feel comfortable—and to be able to reassess what they have been doing with their own lives."

"Fortune provides educational testing so it can be determined what the present realistic job possibilities are. If the individual has severe literacy limitations, he or she is given the opportunity to enter the one-to-one tutoring program. Each student has his/her own teacher and can learn at his/her own pace. The teachers are volunteers who are trained by our Education staff. Adult illiterates, we have learned, respond best in the private one-to-one tutoring mode so they can make mistakes without fear of ridicule and embarrassment."

"The counseling and education programs can make vocational training a reality for people who otherwise would have no occupational skills. A job readiness component prepares individuals to enter the job market," Rothenberg added. This is where the PET computer becomes an active part of Fortune's philosophy. Because when Victor's programming skills become sharp enough, he just may land a job as a programmer. ■

—Jody Miller

An In-Service Computer Course for Teachers

by Dr. Daniel Friedman

It all started with a telephone call from Mr. Meizner. He demanded to talk with me, and was persistent in his efforts. No, he would not discuss his concerns with anyone else and, as the president of Meizner Business Machines, Inc., of Pelham, N.Y., he commanded my attention.

It was not long after our first telephone encounter that Mr. Meizner was in my office extolling the virtues of the PET and telling me, a veteran educator, of an idea that he had to train teachers to use microcomputers. Imagine, I was being sweet-talked and slowly convinced to take on still another teaching task. The goal of helping teachers to develop computer skills was not new to me, but the prospect of doing it in a shortened time-frame, off-campus in a high school, and only with micros, was a challenge.

I looked into this man's twinkling eyes to try and fathom his true motivation since, after all, he was a business man and I was an educator. What did he really want? I guessed that he must have wanted me to sell machines for him. My fears were dispelled over the months to come by our relationship in what I came to understand was a deep and sincere commitment on his part to both education and learning.

The proposal sounded far-fetched, fanciful, and unrealistic. This businessman wanted me to teach about computers and learning to a group of unselected teachers at some still unnamed school, at some unspecified time in the future, using machines that were strange to me. He only wanted my consent, and assured me that he would do the rest. His eloquent presentations were frequently accompanied by crystal ball gazing. We speculated on the power of the marvelous machine to give immediate feedback, to diagnose student responses, to offer branched learning opportunities, to sort, to randomize, and to retrieve data.

We talked about the "media" of computers and wondered if the book industry would survive. Of course we also explored such questions as "Will the computer replace or supplement the role of the teacher?" "What criteria should be used in evaluating educational software?" "How will personal computers contribute to the education of young people in the future?" "How will the present adult population learn to cope with the new technology?" and so on.

Although his proposal was lacking in specificity, its goals were obviously so noble and important that I accepted the assignment. The following string of connections and cooperative commitments quickly occured:

- The White Plains, N.Y. school district agreed to offer the course for in-service credit. They also provided the free use of a fine computer laboratory equipped with 20 PET microcomputers.
- Commodore agreed to provide me with a PET 2001 micro with both a cassette player and floppy disk drive for storage so that I could learn the special operating features of the PET, its graphics, and the 3.0 version of PET BASIC.
- Mr. Meizner agreed to do all the advertising and handle the registration for the course.
- Iona College in New Rochelle, N.Y., agreed to permit me to offer the course as a service to the educational community.

The first day of the course arrived and almost 30 people showed up for this FREE, VOLUNTARY, AFTER-SCHOOL, HANDS-ON COURSE. Mr. Meizner dropped in to see the fruits of his labor, offered the students some words of encouragement, and then disappeared never to be seen again by the students. This first session seemed like a sideshow at the circus. Although the session was carefully structured, students seemed to come and go. In retrospect, the fact that the course was free probably encouraged a flexibility and fluidity that were new to me. It took a full five sessions for some students to decide if they would or would not stay with the course. A total of 16 students ultimately completed the 20-session course. We met twice a week from 3:30 to 5:00 p.m.

Other planners and teachers of courses about computers for teachers might be interested in several observations:

- Students often came early and stayed late indicating a high level of motivation and a desire for more hands-on experience (remember, the ratio of students per machine was excellent, i.e. 1.5 students per machine). After the third session, the school district and Mr. Daigle, the teacher in charge of this computer laboratory, agreed to permit teachers to use the lab during the regular school day for practice and skill sessions on their own time.
- One of the students, Kevin Dougherty, who was the audio-visual coordinator for the school, volunteered and constructed an interface connecting a 21-inch television set to the computer creating an instant large-screen teaching device. This helped students through the peer process of learning. Fellow students demonstrated on the screen for the class. It was a marvelous aid.
- A guest lecturer was invited to explain some of the subtleties involved in the manipulation of programs and files between memory and storage.
- A couple of high school students infiltrated the course and asked permission to come and help their teachers learn about the computers. These young students were encouraged and provided individual help and assistance in a relatively open learning environment. Teachers were learning from their students!
- The materials distributed to each student, at a cost of $10 each, were: one floppy disk, one ten-minute blank cassette tape, and a copy of the *PET CBM Personal Computer Guide*. In addition, an extensive set of teacher-

➔

Inset I
Syllabus for
Computers, Programming, and Classroom Instruction

Session	Topic	Keywords
1	Orientation & introduction to operating the computer	LOAD, SAVE
2	The BASIC language: the commands	CLR, CONT, LIST, LOAD, NEW, RUN, SAVE, VERITY
3	Concepts of programming; correcting keyboard entries	PRINT, AND VARIATIONS
4	Programming for learning; the why and how of interactive computing and program components	
5	Variables	LET, FOR-NEXT, IF-THEN, ON-GOTO
6	Formats for Computer-Based learning programs	
7 & 8	Graphics	
9	The use of Sub-Routines	GOSUB, ON-GOSUB
10	The use of Data Statements	INPUT, READ-DATA, RESTORE
11	Numerics, arithmetic functions, and randomization	RND
12	Character strings and string functions	ASC, CHR$, LEFT$, LEN, MID$, RIGHT$, STR$, BAL
13	Simulation programs in education	
14	Array and matrix operations	DIM
15	Designing educational programs	POKE, PEEK, SYS, USR
16	Programming concepts: structured programming and creative programming	
17 & 18	Files	OPEN, CLOSE, CMD, GET#, INPUT#, PRINT#
19	Criteria for evaluating educational programs	
20	Designing systems for learning	

prepared materials were distributed at the session.

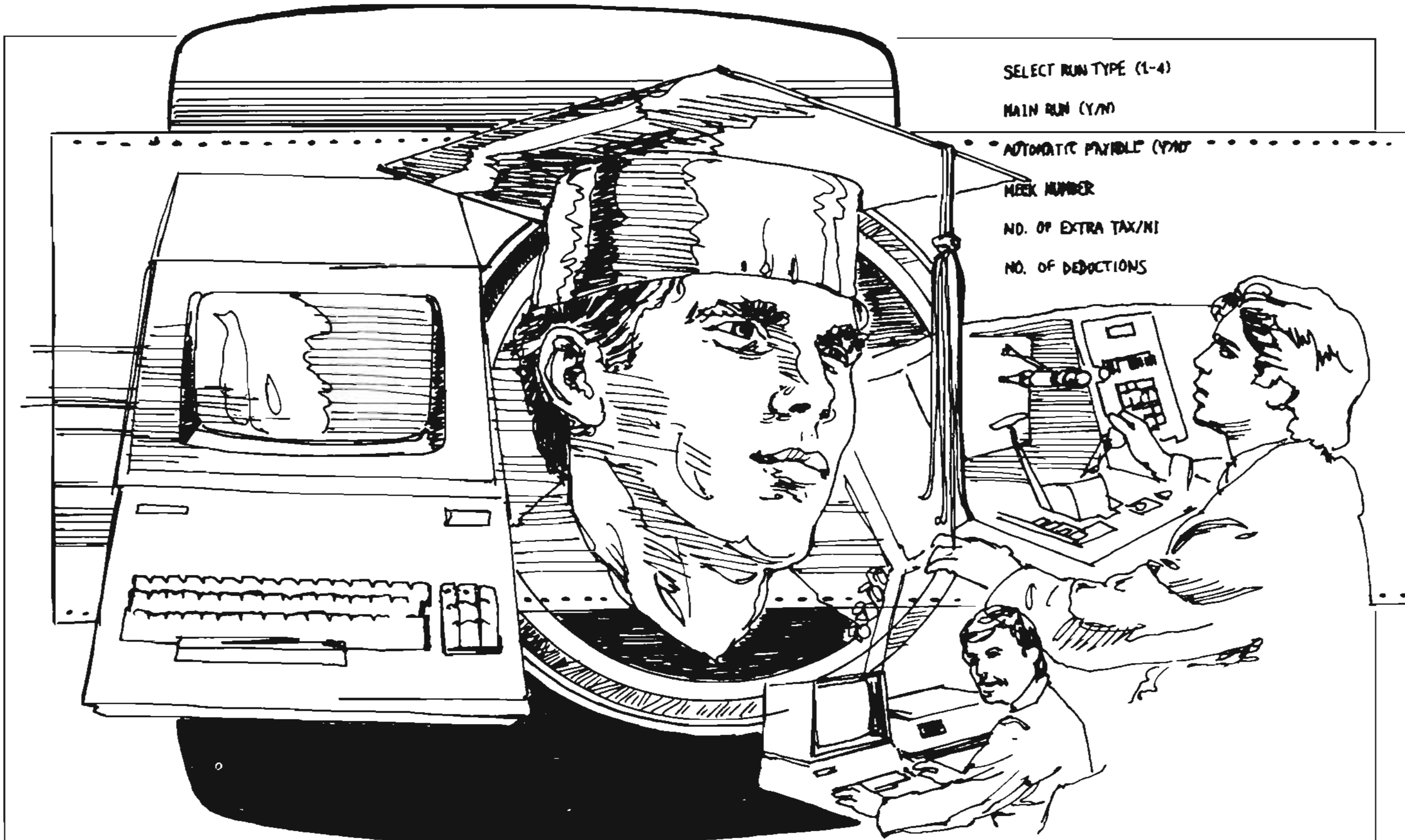
- The high school students referred to the laboratory as "The Zoo where the PETS lived!"
- Participants were encouraged to develop programs to be used in their classrooms. The specific direction for formatting the screen, establishing conditions for input data, receiving non-standard input data, and the other problems inherent in "user friendly programs" were discussed.

The course ran smoothly with a minimum of frustration for participants. The major aggravations were caused not by the hardware (which was reliable and efficient), but by the peculiarities of our configuration of peripherals. We were sharing one floppy disk storage device and one printer among 3 PET micros. There were no automatic controls present and consequently, we sometimes experienced program failures and print-outs of random materials from other machines when someone else was using the system. It took some time for everyone to adjust to this eccentricity.

There was general agreement that the topics were appropriate and that the course was worthwhile. The course was a success.

Although the course is completed, the real test is yet to come. Will teachers be using micros in their classrooms to help students learn? Will teachers be able to develop motivating and challenging programs for learning? We are at a primitive stage in our understanding of the uses of educational computing. ■

Dr. Friedman is currently the director of graduate programs in education at Iona College in New Rochelle, New York.



Computer Education: A handy reference

While the Autumn season often evokes images of Halloween, pumpkin pie, blustery days, and football games, it is also the time of year when many high school seniors begin to ponder their future. For some, that means finding a college to begin their career path. And there are others who may be at a dead-end in their present job or vocation, and want to start over again. More and more, these people are turning towards colleges that offer curriculum and degrees in data processing, computer engineering, and the information sciences. The computer trade, particularly in the microcomputer area, is growing at a phenomenal rate, becoming one of the nations's largest industries.

For those considering a computer career, we have compiled a list of trade schools, colleges, and universities that offer relevant programs. We apologize if we have left any schools off our list. Regardless, it will get those future programmers and analysts off to a good start.

Trade and Technical Schools

Source:
"Handbook of Trade and Technical Careers and Training" National Association of Trade and Technical Schools, 2021 K Street, N.W., Washington, D.C. 20006

Colleges and Universities

Source:
Chronicle Four Year College Data Book, Chronicle Guidance Publications, Inc., Moravia, NY 13118

Note:
The degree offered by the schools listed are abbreviated as follows:

B—Bachelor's Degree
M—Master's Degree
D—Doctorate

COMPUTER ENGINEERING TECHNOLOGY

AL	AUBURN U, AUBURN	B
CA	U OF CA, SAN DIEGO	B
	U OF LA VERNE	B
FL	EMBRY-RIDDLE AERO U	B
	FL INST. OF TECH	B
IN	IN INST OF TECH	B
	IN U-PURDUE U, INDIANAPOLIS	B
	PURDUE U, WEST LAFAYETTE	B
	U OF EVANSVILLE	B
IA	GRACELAND C	B
MA	BOSTON U	BM
	SOUTHEASTERN MA U	B
MI	WESTERN MI U	B
NY	RENSSELAER POLYTECH INST	BMD
	ROCHESTER INST OF TECH	B
OH	CASE WESTERN RESERVE U	BMD
OR	OR INST OF TECH	B
PA	BUCKNELL U	B
PR	U OF PR, MAYAGUEZ	B
SC	CLEMSON U	BM
TN	MEMPHIS ST U	B
WA	EASTERN WA U	B
	SEATTLE U	B

COMPUTER & INFORMATION SCIENCES (General)

AL	AL AG & MECH U	B
	AL ST U	B
	AUBURN U, AUBURN	BMD
	JACKSONVILLE ST U	B
	JUDSON C	B
	LIVINGSTON U	B
	TROY ST. U, DOTHAN, Fort Rucker	B
	TROY ST. U, TROY	BM
	U OF AL, BIRMINGHAM	BMD
	U OF AL, HUNTSVILLE	M
	U OF AL, UNIVERSITY	BM
	U OF SOUTH AL	B
AZ	AZ ST U	BM
	NORTHERN AZ U	B
	U OF AZ	MD
AR	AR ST U, STATE UNIVERSITY	BM
	AR TECH U	B
	HARDING U	B
	JOHN BROWN U	B
	U OF AR, FAYETTEVILLE	BM
	U OF AR, LITTLE ROCK	B
CA	CA INST OF TECH	BMD
	CA POLYTECH ST U	BM
	CA ST C, STANISLAUS	B
	CA ST POLYTECH U, POMONA	B
	CA ST U, CHICO	BM
	CA ST U, DOMINGUEZ HILLS	B
	CA ST U, FULLERTON	BM
	CA ST U, HAYWARD	B
	CA ST U, LOS ANGELES	B
	CA ST U, NORTHRIDGE	BM
	CA ST U, SACRAMENTO	BM
	INTERNATIONAL C	BMD
	LOMA LINDA U	B
	LOYOLA MARYMOUNT U	M
	MILLS C	B
	NAVAL POSTGRADUATE S	BM
	NORTHROP U	B
	PACIFIC UNION C	B
	POINT LOMA C	B
	SAN DIEGO ST U	BM
	SAN FRANCISCO ST U	B
	SAN JOSE ST U	BM
	STANFORD U MD	M
	U OF CA, BERKELEY	BMD
	U OF CA, DAVIS	MD
	U OF CA, IRVINE	BMD
	U OF CA, LOS ANGELES	MD
	U OF CA, RIVERSIDE	B
	U OF CA, SAN DIEGO	BMD

→

EDUCATION

	U OF CA, SANTA BARBARA	B
	U OF CA, SANTA CRUZ	BMD
	U OF SAN FRANCISCO	B
	U OF SOUTHERN CA	BMD
	WEST COAST U, LOS ANGELES	BM
	WEST COAST U, ORANGE	BM
CO	CO ST U	BMD
	MESA C	B
	METROPOLITAN ST C	B
	U S AIR FORCE ACADEMY	B
	U OF CO, BOULDER	MD
	U OF CO, COLORADO SPRINGS	B
	U OF SOUTHERN CO	B
CT	CENTRAL CT ST C	B
	HARTFORD GRADUATE CTR	M
	SACRED HEART U	B
	U OF BRIDGEPORT	BM
	U OF CT, STORRS	BMD
	U OF NEW HAVEN	BM
	YALE U	BMD
DE	GOLDEY BEACOM C	B
	U OF DE	BMD
DC	AMERICAN U	BM
	CATHOLIC U OF AMERICA	B
	GEORGE WASHINGTON U	BMD
	HOWARD U	M
	U OF DC	B
FL	EMBRY-RIDDLE AERO U	B
	FL ATLANTIC U	BM
	FL INST OF TECH	BM
	FL INTERNATIONAL U	BM
	NOVA U	BM
	U OF CENTRAL FL	BMD
	U OF FL	BM
	U OF MIAMI	BM
	U OF NORTH FL	B
GA	ARMSTRONG ST C	B
	AUGUSTA C	B
	CLARK C	B
	FORT VALLEY ST C	B
	MOREHOUSE C	B
	MORRIS BROWN C	B
	SPELMAN C	B
	U OF GA	B
HI	CHAMINADE U OF HONOLULU	B
	U OF HI, MANOA	BM
ID	ID ST U	B
	U OF ID	BM
IL	BRADLEY U	BM
	CHICAGO ST U	B
	CONCORDIA C	B
	DEPAUL U	BM
	DEVRY INST OF TECH	B
	ELMHURST C	B
	IL BENEDICTINE C	B
	IL INST OF TECH	BMD
	IL ST U	B
	KNOX C	B
	LAKE FOREST C	B
	LEWIS U	B
	NORTH CENTRAL C	B
	NORTHERN IL U	BM
	NORTHWESTERN U	BMD
	OLIVET NAZARENE C	B
	QUINCY C	B
	ROCKFORD C	B
	ROOSEVELT U	B
	SANGAMON ST U	BM
	SOUTHERN IL U, CARBONDALE	BM
	SOUTHERN IL U, EDWARDSVILLE	B
	U OF IL, URBANA-CHAMPAIGN	BMD
	WESTERN IL U	BM
IN	ANDERSON C	B
	BALL ST U	BM
	BUTLER U	B
	IN INST OF TECH	B
	IN U, BLOOMINGTON	BMD
	IN U—PURDUE U, FORT WAYNE	B
	PURDUE U, CALUMET	B
	PURDUE U, WEST LAFAYETTE	BMD
	ROSE-HULMAN INST OF TECH	BM
	ST JOSEPHS C	B
	U OF EVANSVILLE	B
IA	CENTRAL C	B
	DRAKE U	B
	GRACELAND C	B
	GRINNELL C	B
	IA ST U	BMD
	LUTHER C	B
	MARYCREST C	B
	MOUNT MERCY C	B
	ST AMBROSE C	B
	U OF IA	BMD
	WESTMAR C	B
KS	KS ST U	BMD
	KS WESLEYAN	B
	MANHATTAN CHRISTIAN C	B
	PITTSBURG ST U	B
	SOUTHWESTERN C	B
	TABOR C	B
	U OF KS	BMD
	WICHITA ST U	BM
KY	ASBURY C	B
	EASTERN KY U	B
	KY ST U	B
	MOREHEAD ST U	B
	MURRAY ST U	B
	TRANSYLVANIA U	B
	U OF KY	BM
	U OF LOUISVILLE	BM
	WESTERN KY U	BM
LA	GRAMBLING ST UB	
LA	STU & AG & MECH C, Baton Rouge	B
	LA ST U, SHREVEPORT	B
	LA TECH U	BM
	LOYOLA U	B
	MCNEESE ST U	B
	NICHOLLS ST U	B
	SOUTHERN U, BATON ROUGE	B
	TULANE U OF LA	B
	U OF NEW ORLEANS	B
	U OF SOUTHWESTERN LA	BMD
ME	U OF ME, ORONO	B
	U OF SOUTHERN ME	B
MD	JOHNS HOPKINS U	M
	LOYOLA C	B
	U OF BALTIMORE	M
	U OF MD, COLLEGE PARK	BMD
	U OF MD, EASTERN SHORE	B
	U OF MD, UNIV C	B
MA	BENTLEY C	B
	BOSTON C	B
	BOSTON U	BM
	BRANDEIS U	B
	CLARK U	B
	FITCHBURG ST C	B
	HARVARD U	MD
	MA INST OF TECH	BMD
	MERRIMACK C	B
	NORTH ADAMS ST C	B
	SOUTHEASTERN MA U	B
	U OF LOWELL	BM
	U OF MA, AMHERST	BMD
	U OF MA, BOSTON	B
	WESTFIELD ST C	B
	WORCESTER POLYTECH INST	BM
MI	ANDREWS U	B
	CALVIN C	B
	CENTRAL MI U	BM
	EASTERN MI U	B
	GRAND VALLEY ST COLLEGES	B
	HOPE C	B
	MADONNA C	B
	MARYGROVE C	B
	MI ST U	BMD
	MI TECH U	BM
	OAKLAND U	B
	U OF DETROIT	M
	U OF MI, ANN ARBOR	BMD
	U OF MI, DEARBORN	B
	U OF MI, FLINT	B
	WAYNE ST U	BMD
	WESTERN MI U	BM
MN	C OF ST CATHERINE	B
	C OF ST THOMAS	B
	CONCORDIA C, MOORHEAD	B
	MANKATO ST U	BM
	MOOREHEAD ST U	B
	ST CLOUD ST U	B
	ST MARYS C	B
	U OF MN, DULUTH	B
	U OF MN, TWIN CITIES	BMD
	WINONA ST U	B
MS	JACKSON ST U	BM
	MS ST U	BM
	MS VALLEY ST U	B
	U OF MS, UNIVERSITY	B
	U OF SOUTHERN MS	BM
MO	CENTRAL MO ST U	B
	MARYVILLE C, ST LOUIS	B
	MO INST OF TECH	B
	MO VALLEY C	B
	MO WESTERN ST C	B
	NORTHEAST MO ST U	B
	NORTHWEST MO ST U	B
	ROCKHURST C	B
	ST LOUIS U	B
	SOUTHEAST MO ST U	B
	SOUTHWEST MO ST U	B
	STEPHENS C	B
	U OF MO, COLUMBIA	M
	U OF MO, ROLLA	BMD
	WASHINGTON U, ST LOUIS	BMD
	WESTMINSTER C	B
	WILLIAM WOODS C	B
MT	MT ST U	B
	U OF MT	B
NE	CHADRON ST C	B
	HASTINGS C	B
	KEARNEY ST C	B
	U OF NE, LINCOLN	BM
	U OF NE, OMAHA	B
NH	U OF NH	BM
NJ	FAIRLEIGH DICKINSON U TEANECK-HACKENSACK	BM
	KEAN C OF NJ	B
	MONMOUTH C	B
	MONTCLAIR ST C	BM
	NJ INST OF TECH	BM
	RUTGERS ST U OF NJ, NEWARK	B
	RUTGERS ST U OF NJ, NEW BRUNSWICK	BMD
	ST PETERS C	BM
	STEVENS INST OF TECH	BMD
	WILLIAM PATERSON C OF NJ	B
NM	EASTERN NM U, PORTALES	B
	NM INST OF MINING & TECH	BMD
	NM ST U, LAS CRUCES	BMD
	U OF NM, ALBUQUERQUE	BM
NY	ALFRED U	B
	BARUCH C	BM
	CITY C OF CUNY	BM
	COLGATE U	B
	C OF STATEN ISLAND	B
	COLUMBIA U, TEACHERS C	M
	CORNELL U, ENDOWED COLLEGES	MD
	ELMIRA C	B
	HOFSTRA U	B
	HUNTER C	B
	IONA C	B
	MANHATTAN C	B
	MERCY C	B
	NY INST/TECH, METRO CTR	B
	NY INST/TECH, OLD WESTBURY	BM
	NY U	BMD
	PACE U, NEW YORK	B
	PACE U, PLEASANTVILLE/BRIARCLIFF	B
	POLYTECH INST OF NY	BMD
	PRATT INST	BM
	QUEENS C	BM
	RENSSELAER POLYTECH INST	BMD
	ROCHESTER INST OF TECH	BM
	ST JOHNS U	B
	SIENA C	B
	SUNY AT ALBANY	M
	SUNY AT BINGHAMTON	B
	SUNY AT BUFFALO	BMD
	SUNY, C AT BROCKPORT	B
	SUNY, C AT CORTLAND	B
	SUNY, C AT OLD WESTBURY	B
	SUNY, C AT OSWEGO	B
	SUNNY, C AT PLATTSBURGH	B
	SUNNY, C AT POTSDAM	B
	SUNY, C/TECH AT UTICA/ROME	B
	SUNY, MARITIME C	B
	SUNY AT STONY BROOK	BMD
	SYRACUSE U	BMD
	UNION C	BM
	U OF ROCHESTER	MD
	U/ST/NY/REGENTS EXTER DEGREE	B
	UTICA C OF SYRACUSE U	B
	YESHIVA U	B
NC	APPALACHIAN ST U	B
	CATAWBA C	B
	DUKE U	BMD
	EAST CAROLINA U	B
	JOHNSON C SMITH U	B
	NC ST U, RALEIGH	B
	ST ANDREWS PRESBYTERIAN C	B
	SHAW U	B
	U OF NC, CHAPEL HILL	MD
	U OF NC, WILMINGTON	B
	WESTERN CAROLINA U	B
	WINSTON-SALEM ST U	B
ND	JAMESTOWN C	B
	ND ST U, FARGO	BM
	U OF MD, GRAND FORKS	B
OH	BOWLING GREEN ST U - BOWLING GREEN	BM
	CENTRAL ST U	B
	CLEVELAND ST U	BM
	DENISON U	B
	HEIDELBERG C	B
	KENT ST U, KENT	B
	MARIETTA C	B
	MOUNT UNION C	B
	MOUNT VERNON NAZARENE C	B
	MUSKINGUM C	B
	OH INST OF TECH	B
	OH NORTHERN U	B
	OH ST U, COLUMBUS	BMD
	OH U, ATHENS	B
	U OF DAYTON	BM
	WRIGHT ST U, DAYTON	BM
	XAVIER U	B
	YOUNGSTOWN ST U	B
OK	BETHANY NAZARENE C	B
	CENTRAL ST U	BM
	LANGSTON U	B

	OK BAPTIST U	B
	OK CHRISTIAN C	B
	OK ST U	BMD
	ORAL ROBERTS U	B
	SOUTHEASTERN OK ST U	B
	U OF OK, NORMAN	BMD
	U OF SCI & ARTS OF OK	B
	U OF TULSA	B
OR	OR INST OF TECH	B
	OR ST U	BMD
	PORTLAND ST U	B
	U OF OR, EUGENE	BM
PA	ALBRIGHT C	B
	BEAVER C	B
	BLOOMSBURG ST C	B
	CALIFORNIA ST C	B
	CHEYNEY ST C	B
	CLARION ST C, CLARION	B
	DREXEL U	B
	EAST STROUDSBURG ST C	B
	EDINBORO ST C	B
	GANNON U	B
	GENEVA C	B
	INDIANA U OF PA	B
	JUNIATA C	B
	KUTZTOWN ST C	B
	LA SALLE C	B
	LEBANON VALLEY C	B
	LEHIGH U	BMD
	LOCK HAVEN ST C	B
	LYCOMING C	B
	MANSFIELD ST C	B
	MARYWOOD C	BM
	MESSIAH C	B
	MILLERSVILLE ST C	B
	MORAVIAN C	B
	PA ST U, CAPITOL	B
	PA ST U, UNIVERSITY PARK	BMD
	POINT PARK C	B
	SHIPPENSBURG ST C	BM
	SLIPPERY ROCK ST C	B
	SUSQUEHANNA U	B
	TEMPLE U	B
	U OF PA	BMD
	U OF PITTSBURGH, PITTSBURGH	BMD
	U OF SCRANTON	B
	VILLANOVA U	BM
	WEST CHESTER ST C	BM
	WESTMINSTER C	B
	WILKES C	B
PR	U OF PR, RIO PIEDRAS	B
RI	BROWN U	BMD
	BRYANT C	B
	U OF RI	BM
SC	FRANCIS MARION C	B
	LANDER C	B
	NEWBERRY C	B
	U OF SC, COLUMBIA	B
	U OF SC, SPARTANBURG	B
SD	NATIONAL C	BB
	SD S OF MINES & TECH	B
	U OF SD, VERMILLION	B
TN	DAVID LIPSCOMB C	B
	EAST TN ST U	B
	FREED-HARDEMAN C	B
	MIDDLE TN ST U	B
	TN TECH U	B
	UNION U	B
	U OF TN, CHATTANOOGA	B
	U OF TN, KNOXVILLE	BM
TX	ABILENE CHRISTIAN U	B
	AMERICAN TECH U	M
	ANGELO ST U	BM
	AUSTIN C	B
	BAYLOR U	B
	CORPUS CHRISTI ST U	BM
	DEVRY INST OF TECH	B
	EAST TX ST U	BM
	LAMAR U, BEAUMONT	B
	MIDWESTERN ST U	B
	NORTH TX ST U	M
	PAN AMERICAN U	B
	ST EDWARDS U	B
	ST MARYS U OF SAN ANTONIO	B
	SAM HOUSTON ST U	B
	SOUTHERN METHODIST U	BMD
	SOUTHWEST TX ST U	BM
	STEPHEN F AUSTIN ST U	BM
	TX A&I U	M
	TX A&M U, COLLEGE STATION	BMD
	TX CHRISTIAN U	B
	TX SOUTHERN U	B
	TX TECH U	BM
	TRINITY U	BM
	U/HOUSTON, CENTRAL	BM
	U/TX, ARLINGTON	BM
	U/TX, AUSTIN	BMD
	U/TX, DALLAS	BMD
	U/TX, EL PASO	B
	U/TX OF THE PERMIAN BASIN	B
	U/TX, TYLER	B
	WEST TX ST U	B
UT	BRIGHAM YOUNG U	BM
	U OF UT	BMD
	UT ST U	B
VT	U OF VT	BM
VA	C OF WILLIAM & MARY	B
	GEORGE MASON U	B
	JAMES MADISON U	B
	MARY WASHINGTON C	B
	OLD DOMINION U	BM
	U OF VA	BMD
	VA POLYTECH INST & ST U	B
WA	EASTERN WA U	B
	EVERGREEN ST C	B
	GONZAGA U	B
	SEATTLE PACIFIC U	B
	U OF PUGET SOUND	B
	U OF WA	BMD
	WALLA WALLA C	B
	WA ST U	BMD
	WESTERN WA U	B
WV	DAVIS & ELKINS C	B
	MARSHALL U	B
	U OF CHARLESTON	B
	WV INST OF TECH	B
	WV ST C	B
	WV U	BM
	WV WESLEYAN C	B
WI	U OF WI, EAU CLAIRE	B
	U OF WI, LA CROSSE	B
	U OF WI, MADISON	BMD
	U OF WI, MILWAUKEE	BM
	U OF WI, PARKSIDE	B
	U OF WI, RIVER FALLS	B
	U OF WI, WHITEWATER	B
WY	U OF WY	BM

COMPUTER PROGRAMMING

AL	AL AG & MECH U	B
CA	COLEMAN C	B
CT	EASTERN CT ST C	B
FL	EMBRY-RIDDLE AERO U	B
	FL INST OF TECH	M
HI	HI LOA C	B
IL	ROCKFORD C	B
IN	TAYLOR U	B
IA	CLARKE C	B
LA	NORTHEAST LA U	B
MA	ATLANTIC UNION C	B
	FRAMINGHAM ST C	B
MI	MADONNA C	B
	NORTHERN MI U	B
	WAYNE ST U	B
MN	MANKATO ST U	B
MS	U OF SOUTHERN MS	B
MO	FONTBONNE C	B
MT	MT ST U	B
NH	DANIEL WEBSTER C	B
NY	ELMIRA C	B
	SUNY, C AT OLD WESTBURY	B
OH	MALONE C	B
	TIFFIN U	B
	U OF CINCINNATI	B
	U OF TOLEDO	B
PA	SPRING GARDEN C	B
PR	INTERNATIONAL INST OF THE AMERICAS OF WORLD U	B
	U OF PR, REG COLLEGES ADMIN	B
SD	NATIONAL C	B
TN	LANE C	B
TX	AMERICAN TECH U	B
	TX CHRISTIAN U	M
WA	GRIFFIN C	B
WI	RIPON C	B

COMPUTER SERVICE TECHNICIAN

Ariz.:	ABC Trade School, Tuscon
Ariz.:	Arizona Tech, Phoenix
Calif.:	Condie College of Business and Technology, San Jose
Calif.:	Control Data Institute, Anaheim
Calif.:	Control Data Institute, Los Angeles
Calif.:	Control Data Institute, San Francisco
Fla.:	Tampa Technical Institute, Tampa
Fla.:	United Electronics Institute, Tampa
Ga.:	Control Data Institute, Atlanta
Ill.:	Control Data Institute, Bensenville
Ill.:	Control Data Institute, Chicago
Ill.:	DeVry Institute of Technology, Chicago
Ill.:	Electronics Technical Institute, Chicago
Kan.:	Wichita Automotive and Electronics Institute, Wichita
Md.:	Arundel Institute of Technology, Baltimore
Mass.:	Associated Technical Institute, Woburn
Mass.:	Control Data Institute, Burlington
Mass.:	Sylvania Technical School, Waltham
Mich.:	Control Data Institute, Southfield
Minn.:	Brown Institute, Minneapolis
Minn.:	Control Data Institute, Minneapolis
Minn.:	Control Data Institute, St. Paul
Minn.:	Dunwoody Industrial Institute, Minneapolis
Minn.:	Northwestern Electronics Institute, Minneapolis
Mo.:	Basic Institute of Technology, St. Louis
Mo.:	Control Data Institute, St. Louis
Mo.:	Ranken Technical Institute, St. Louis
N.J.:	Brick Computer Science Institute, Brick
N.J.:	Lincoln Technical Institute, Pennsauken
N.J.:	Lyons Institute, Cherry Hill
N.J.:	The Plaza Technical Institute, Paramus
N.Y.:	Advanced Training Center, Tonawanda
N.Y.:	Albert Merill School, New York
N.Y.:	Control Data Institute, New York
N.Y.:	PSI Institute, New York
N.Y.:	SCS Business and Technical Institute, New York
N.Y.:	Suburban Technical School, New York
N.Y.:	Technical Career Institutes, New York
Ohio:	Control Data Institute, Independence
Pa.:	Electronic Institute, Harrisburg
Pa.:	Electronic Institute, Pittsburgh
Pa.:	Lincoln Technical Institute, Allentown
Pa.:	Lyons Technical Institute, Philadelphia
Pa.:	Lyons Technical Institute, Upper Darby
Pa.:	R.E.T.S. Electronic School, Upper Darby
Texas:	Control Data Institute, Dallas
Va.:	Control Data Institute, Arlington
Va.:	Electronic Computer Programming Institute, Norfolk
W. Va.:	United Electronics Institute, Charleston

DATA PROCESSING

Ala.:	Alabama College of Technology, Birmingham
Ala.:	Herzing Institute, Birmingham
Ariz.:	Arizona Tech, Phoenix
Ark.:	Arkansas College of Technology, Little Rock
Calif.:	American Business College—Technical Division, San Diego
Calif.:	Computer Learning Center of Los Angeles, Los Angeles
Calif.:	Condie College of Business and Technology, San Jose
Calif.:	Control Data Institute, Aneheim
Calif.:	Control Data Institute, Los Angeles
Calif.:	Control Data Institute, San Francisco
Fla.:	Garces Commercial College, Hialeah
Fla.:	Garces Commercial College, Miama
Ga.:	Control Data Institute, Atlanta
Ga.:	DeVry Institute of Technology, Atlanta
Ill.:	Control Data Institute, Bensonville
Ill.:	Control Data Institute, Chicago
Iowa:	United Electronics Institute, West Des Moines
Kan.:	Bryan Institute, Wichita
Kan.:	Electronic Computer Programming Institute, Topeka
Kan.:	Electronic Computer Programming Institute, Wichita
Ky.:	United Electronics Institue, Louisville
Mass.:	Control Data Institute, Burlington
Mass.:	Control Data Institute, Quincy
Mich.:	Control Data Institute, Southfield
Minn.:	Brown Institute, Minneapolis
Minn.:	Control Data Institute, Minneapolis
Minn.:	Minnestoa School of Business—Technical Division, Minneapolis
Mo.:	Bryan Institute, Bridgeton
Mo.:	Bryan Institute, Webster Groves
Mo.:	Control Data Institute, St. Louis
Mo.:	Electronic Computer Programming Institute, Kansas City
Mo.:	Missouri Institute of Technology, Kansas City
Neb.:	Electronic Computer Programming Institute, Omaha
N.J.:	Brick Computer Science Institute, Brick
N.J.:	Chubb Institute for Computer Technology, Short Hills
N.J.:	Electronic Computer Programming Institute, Paterson
N.J.:	Empire Technical Schools of New Jersey, East Orange
N.Y.:	Advanced Training Center, Tonawanda
N.Y.:	Albert Merrill School, New York
N.Y.:	Control Data Institute, New York
N.Y.:	Empire Technical School, New York
N.Y.:	PSI Institute, New York
N.Y.:	SCS Business and Technical Institute, New York
N.Y.:	Technical Career Institutes, New York
N.C.:	Raleigh School of Data Processing, Raleigh
Ohio:	Columbus Paraprofessional Institute, (Indianoia Ave.) Columbus
Ohio:	Control Data Institute, Independence
Ohio:	Ohio School of Career Technology, Columbus
Okla.:	Bryan Institute, Tulsa
Ore.:	Oregon Career Institute, Portland
Pa.:	Computer Communications Institute, Upper Darby
Pa.:	FPM Data Processing School, New Kensington
Pa.:	Maxwell Institute, Norristown
Pa.:	Philadelphia Training Center, Philadelphia
Pa.:	Wilma Boyd Career School, Pittsburgh
Texas:	CBM Education Center, San Antonio
Texas:	Control Data Institute, Dallas
Texas:	DeVry Institute of Technology, Dallas
Texas:	Texas Institute, Dallas
Va.:	Computer Learning Center, Springfield
Va.:	Control Data Institute, Arlington
Va.:	Electronic Computer Programming Institute, Nortolk
Wash.:	ITT Peterson School of Business—Technical Division, Seattle
Wash.:	Seattle Opportunities Industrialization Center, Seattle ■

Astronomy and the Microcomputer in Rural U.S.

by Everett Q. Carr
Herkimer BOCES Planetarium

Microcomputers with 8 kilobytes (8K) or more of random access memory (RAM) have numerous applications in astronomy education. These uses include star catalogs, computer generated ephemerides, weather forecasting, stellar object horizon coordinates, telescope control, constellation recognition, randomly generated spectra for red and blue measurements and calculations and the study of Kepler's Laws with animation. There are computer simulations for multistage rocket design, rocket launches and landings, interplanetary flight, interstellar flight and planetary exploration.

This year a group of 16 determined youngsters in our planetarium's bi-weekly Elementary Honors Seminar (ages 10, 11 and 12) used Commodore PET computers in a simulation of the manned Mars Expedition of 2014 AD. They had two 32-page manuals, one for spaceship and rocket launch design, staffing and stores selection; the second for planning the Mars colony design and the planet exploration for water, gas and oil. At 9:34 AM, they launched from Earth and executed a Homan transfer orbit to Mars. They were to land on Mars, do a survey of Solus Lacus region by Mars Flyer and crawler and drill for water and gas to a depth of 400 meters. They returned to earth at 2 p.m., in time for buses to take them to their home schools.

In our rural school system of 13 comprehensive K-12th grades (ages 5-18) schools, the student population totals 15,000 children. However, the smallest school has 365 students and the largest 2200. Right now there are about 135 PET computers, one TRS-80, and two Apple II computers in the system. I forecast there will be 750 computers in these schools by 1984. How that goal will be achieved is by no means clear. But the same has been true since 1977 when the PET and the original TRS-80 first appeared. Our schools found money in remediation funds by using computers in the "basics" of reading, writing and arithmetic. Our current PET program catalog for elementary school level lists over 500 commercial programs in those basics. These programs, while not yet forming a complete curriculum for the elementary school, are converging on that objective. A complete single subject curriculum in arithmetic and algebra will undoubtedly be available for a few thousand dollars within 2-3 years.

Constellation recognition is a favorite first (or second) program for our students learning BASIC on PETs. It exploits their ability at graphics and the elementary techniques of constructing computer-aided instruction (CAI) programs.

A recent development of our planetarium staff is the application of the video disc to elementary and middle school science education. We have developed an adapter interfacing a 32K PET computer through its user port to the Pioneer VP1000 Laser Disc player and a color TV monitor. The effect is to create an "intelligent video disc" system that allows student-interactive computer/video disc programs. Some authorities state the "intelligent video disc" is the most important tool developed in education since the invention of the book. Our experience in testing 3rd graders with a program entitled "Weather" has demonstrated that 9-year-olds are capable of handling questions and answers suitably reinforced with aural and visual clues containing words like precipitation, rendezvous and climatic. The Pioneer Laser Disc became available locally in July 1980 and our system was first tested with students in October.

Briefly, an "intellitent video disc" system allows access to any of the 54,000 individual frames which constitute a half-hour TV program. The TV frames can be displayed individually for indefinite periods at the command of the computer program. Further, the computer program can employ the video disc information in various modes of instruction such as computer-aided instruction (CAI) drill and practice, simulations, the delivery of straightforward cognitive level information and higher order cognitive tasks.

We hope to begin development of video discs designed as a well-organized curriculum in astronomy. ■

Reprinted from "Sky & Telescope"

Teaching Astronomy with Computers

by Claire J. Carr and Everett Q. Carr, *Herkimer BOCES Planetarium*

All across the country, school planetariums are running into time and money problems. Operating hours are reduced, planetarium teachers diverted to classroom work, and limits put on buses that carry classes to the planetarium. Administrators who are forced to trim programs often scrap astronomy if it is not a formal part of the core curriculum.

In New York State, where we teach, the 9th grade Earth science units on astronomy are not mandatory. Fewer than four percent of the teachers we surveyed in our area use astronomy in their courses, usually for "fast" classes. For high school Project Physics, only about two percent of the experiment handbook is concerned with astronomy. We asked 17 nearby schools about telescopes, and found 14 had none; two scopes were in disrepair, and one in regular use by an experienced teacher and his class.

In 1974, when the Board of Cooperative Educational Services (BOCES) put up a new building for Herkimer, Fulton and Montgomery counties in upstate New York, it included a planetarium in the complex. BOCES also built conditioning, welding shops and a conditioning, and welding shops and a modern computer instruction center. The planetarium offers a wide variety of programs for grades K through 12, and special services for gifted and talented children starting in grade 4.

We try to teach how science and communications work with selected projects. One popular session takes just five hours for a student to build an 8x refracting telescope from a kit, and construct a mount for it. The five hours include a juice break and lunch in addition to telescope building. We encourage students to use their telescopes at night to track planets or the moons of Jupiter.

Another day is devoted to the hemisphere project described above, with the students additionally using an atlas to find stars' coordinates, and punching holes to represent major northern constellations. They compare their star lists to one stored on a cassette in our Commodore PET 2001 computer to verify their data.

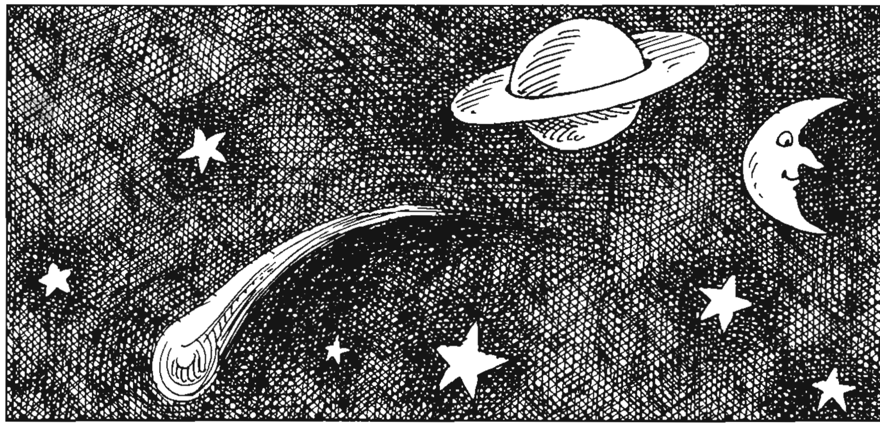
It takes 20 junior high school students with 20 Commodore PET computers and five tutors about 40 hours to become effective programmers.

We find computers extremely versatile for our teaching. In about 15 class hours the students complete 20 lessons in the BASIC language. They apply this to problems in algebra, trigonometry, and geometry, including solving simple triangles and generating planet positions from assumed circular orbits. Another program checks student solutions for accuracy. The computer also assists with a full-day simulation of a voyage to Mars and back. This year they left at 9:30 one morning, and returned from Mars at 2 p.m. with "Martian" soil samples, just in time to catch buses back to their schools.

Students have also "designed" space cities and studied Mayan astronomy and archaeoastronomy with Colgate University experts, and with a professor from Utica College they probed the possibilities of communicating with planets elsewhere.

Last July we taught an intensive programming course to 20 students from grades 7 to 9, each using a microcomputer. The astronomical applications included finding great-circle distances (from the latitudes and longitudes of two points on Earth), as well as deriving sunrise and sunset times, moon positions, and eclipse seasons. The students' intensity was remarkable—there was not a single absence during the entire two weeks.

Our PET computer is proving itself useful in many other ways, too. In a planetarium show for the deaf, it displays the script for students to see. And it also guides a special-effects image on the dome by pulsing stepping motors attached to a steerable mirror. →



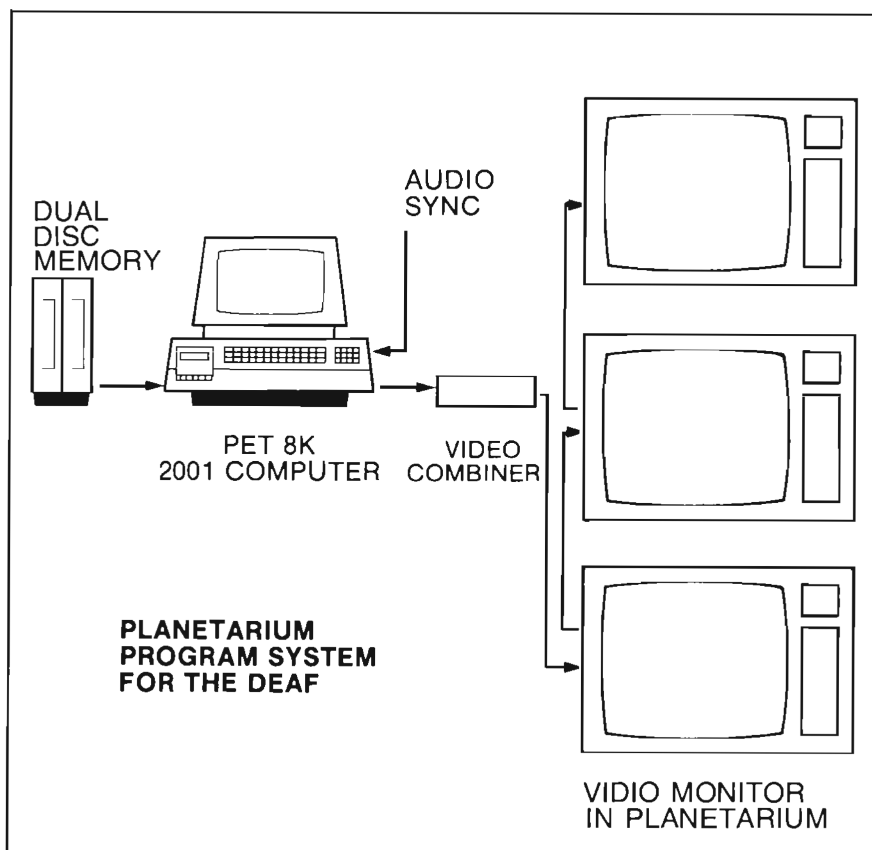
The planetarium offers other services as well. A senior honors seminar presents lectures by experts in astronomy, many other sciences, the arts, and humanities. With NASA's lunar sample education project, we are able to take actual samples brought back by Apollo to students either at BOCES or at their own schools.

In an era where administrations must trim budgets, school buses cost more than $1 per mile, and low rural tax bases offer little hope of prosperity, these advances in curriculum and program give planetarium astronomy strength to survive. ■

Inside the Mars-bound Project Pegasus space capsule, student astronauts control their progress with a computer terminal and video monitor.

The Commodore PET Computer is proving a useful tool in the planetarium. Tests have already been run in a unique approach to planetarium programs for the deaf. The script of a planetarium program is stored in a dual minifloppy disc memory. Up to eight pages of a planetarium script are "dumped" from the disc memory into the PET's 8K of RAM or random access working memory for transfer to the computer video display. A video combiner changes the computer video into a standard composite video to drive three 25-inch RCA monitor TVs around the planetarium dome. Pulses on the audio tape synchronize the spoken script on tape with the video script displayed. The system could be altered to display large characters in a Time Square sign approach for the visually handicapped as well.

The PET computer has also been tested as program generator in positioning images on the planetarium dome. The computer program, through its IEEE 488 output port and an adapter interface, supplies reversible drive signals to stepper motors. The motors move a first surface mirror horizontally and vertically in front of a slide projector. The computer simply keeps track of the number of pulses sent to each motor to position the projected slide anywhere on the dome within range. ■



VIC=20

The friendly computer

How to Tell a Game Machine from a True Computer

There are many game machines on the market—devices that let you play arcade-style games on your television set. But why buy a just game machine when you can buy a personal computer for the same price?

Commodore's VIC 20 lets you plug in the same exciting game cartridges available on most game machines. But the VIC is a full-featured *computer*, and that makes a BIG difference.

Here's a simple test to help you find out whether you're buying a game machine or a computer. Simply ask the salesperson if the device you're evaluating has these features:

1. Does it have a full-size typewriter keyboard?
   (Some game machines try to pass themselves off as computers by including flat plastic keypads or calculator keys that look like the real thing but are difficult to use. So be careful!)
2. Can you use all three types of storage media: plug-in program cartridges, cassette tapes, AND floppy disks?
   (Many game machines offer a cassette recorder and tapes but do NOT offer a disk drive)
3. Is full-power BASIC built-in, or do you have to pay extra for an "extended" computing language?
   (The VIC 20 has a fully extended BASIC built into its operating system.)
4. Is a self-teaching instruction book included or do you have to pay extra for it?
   (If there's no instruction book, chances are you're not buying a real computer. The VIC includes a free "friendly computer guide" tailored for first-time computerists and veterans as well.)
5. How many accessory ports are there?
   (Several ports mean you can hook the computer to special accessories like a printer, disk drive, cassette recorder, modem and game controls. The VIC has four accessory ports and a growing line of special accessories.

The advantages of owning a personal computer instead of a game machine are obvious. Make sure *you* make the right decision before you buy. ■

—Michael Tomczyk

Joystick Control on the VIC

Like all other input and output, the joysticks are controlled using the VIC's 6522 peripheral interface adapters (PIAs). The 6522 is a versatile and complex device. Fortunately, it isn't necessary to delve deeply into the mysteries of the 6522 PIA to read the joysticks.

Each 6522 has two Input/Output ports, called port A and port B. Each of the ports has a control register attached, called the DATA DIRECTION REGISTER (DDR). This highly important register controls the direction of the port. By using this register, you can use the port for input, output, or both at the same time. To set one bit of the port to output, set the corresponding bit of the Data Direction Register to 1. To set a bit of the port for input, set the corresponding bit of the DDR to 0. For example, to set bit 7 of port A to input, and the rest of the bits to output, poke 127 in the DDR for port A.

To read the joystick, one port (and one DDR) of each of the 6522 PIAs on the VIC must be used.

The joystick switches are arranged as follows:

```
                    TOP
FIRE BUTTON
Switch 4
(FR)

                 Switch 0
                   (S1)
                     |
  Switch 2  ---------+---------  Switch 3
    (S2)             |             (S3)
                     |
                   (S1)
                 Switch 1
```

Switch 0, switch 1, switch 2 and the Fire button can be read from PIA #1, which is located beginning at location $9110. Switch 3 must be read from the other 6522 (PIA #2), which is located beginning at location $9120.

Now, the key locations for the joystick are as follows:

HEX	DECIMAL	PURPOSE
9113	37139	Data direction register for I/O port A on PIA #1
9111	37137	Output register A Bit 2 Joy switch 0 Bit 3 Joy switch 1 Bit 4 Joy switch 2 Bit 5 Fire button
9122	37154	Data direction register for I/O port B on PIA #2
9120	37152	Output register B Bit 7 Joy switch 3

To read the joystick inputs, you first set the ports to input mode by setting the DDR to 0. This can be done by a POKE. Then the value of the switches can be read by two peeks. Sounds easy, right? There is only one problem . . . PIA#2 is also used for reading the keyboard. Setting the DDR can disrupt the keyscan rather badly. So you have to make sure you restore the DDR to the original condition if you want to use the keyboard afterwards.

To make things really easy, you can use the following program. Lines 10 to 25 are initialization. The rest of the program, beginning at line 9000, can be called as a subroutine whenever you want to read the joystick.

```
10 DIM JS(2,2):
   POKE37139,0:DD=
   37154:PA=37137:PB=37152
20 FORI=oTO2:FORJ=oTO2:
   READJS(J,I):NEXTJ,I
25 DATA-23,-22,-21,-1,0,1,21,22,23
30GOSUB9000:PRINT JS(X+1,
   Y+1):GOT030
9000 POKEDD,127:S3=
   -((PEEK(PB)AND128)=0):
   POKEDD,255
9010 P=PEEK(PA):S1=
   -((PAND8)=0):S2=((PAND16)=
   0):S0=((PAND4)=0)
9020 FR=-((PAND32)=0):X=
   S2+S3:Y=S0S1:RETURN
```

The variables S0, S1, S2, and S3 will be 0 normally, and will be set to 1 (or −1) when the joystick is pointed in that direction. Two of the variables will be set to 1 on diagonal moves. FR will be 1 when the firing button is pressed, and 0 otherwise.

The AND function is used to pick out one bit of the joystick port. The bits are numbered from 7 (most significant bit) to 0 (least significant bit). By ANDing the 6522 port with a number whose value is a power of two, a single bit is selected. (For example, to pick bit 3, AND using 2, 3 or 8).

The JS array in the program is set up to make it easy to move around the screen using the joystick. The numbers in the DATA statement of line 25 can easily be changed for other purposes. For example . . .

To 'decode' the joystick in this pattern:

```
           TOP
FIRE
            0
          7 | 1
          6-8-2
          5 | 3
            4
```

This data statement should be changed to:

25 DATA 7,0,1,6,8,2,5,4,3 ■

—Andy Finkel

VIC=20

The friendly computer

Commodore Announces Availability of Newest VIC 20 Accessories

If you're planning to expand your VIC 20 system during the next few months, it might be helpful to know which accessories will be available next. The following list of peripherals will give you an indication of which items can be purchased between now and Christmas. Those products which do not have availability dates shown are being sold now, although some dealers may have trouble keeping accessories in stock due to the heavy demand for VIC 20 products. Often, a visit to your Commodore dealer will get you on the list for first delivery of these accessories.

VIC 20—The "Friendly Computer"
Commodore's revolutionary personal computer features color, sound, graphics, programmable function keys, built-in BASIC, expandable memory, low-priced peripherals and more! Connects to any TV or monitor. Includes RF modulator, switchbox, cables and self-teaching instruction book. **$299.95.**

Commodore Datassette
Provides handy economical storage of user-written or prerecorded programs, using ordinary audio tape cassettes. Works like standard tape recorder, includes tape counter. **$75.00.**

VIC Single Disk Drive
Fast, high capacity storage and retrieval of data on standard 5¼ inch floppy diskettes. Stores up to 170K on each diskette, with read/write compatibility with PET/CBM computer systems. First availability: DECEMBER. **$599.00.**

VIC 20 Graphic Printer
Economical dot matrix printer makes paper copies of BASIC programs, letters, business data, and graphic displays. Connects directly to the VIC, prints all characters including letters, numbers and graphics, as well as double-size characters. Prints 80-columns wide, 30 characters per second. Reliable tractor feed mechanism. Device 4/5 and test switch. Accepts sprocketed 8-inch wide roll or sheet paper. **$395.00**

VIC Graphic Printer Paper
High-quality 15# bond paper for your tractor feed printer is available in 1000-sheet packages directly from most Commodore dealers. Each sheet is 8½ x 11 inches. **$15.00 per pack.**

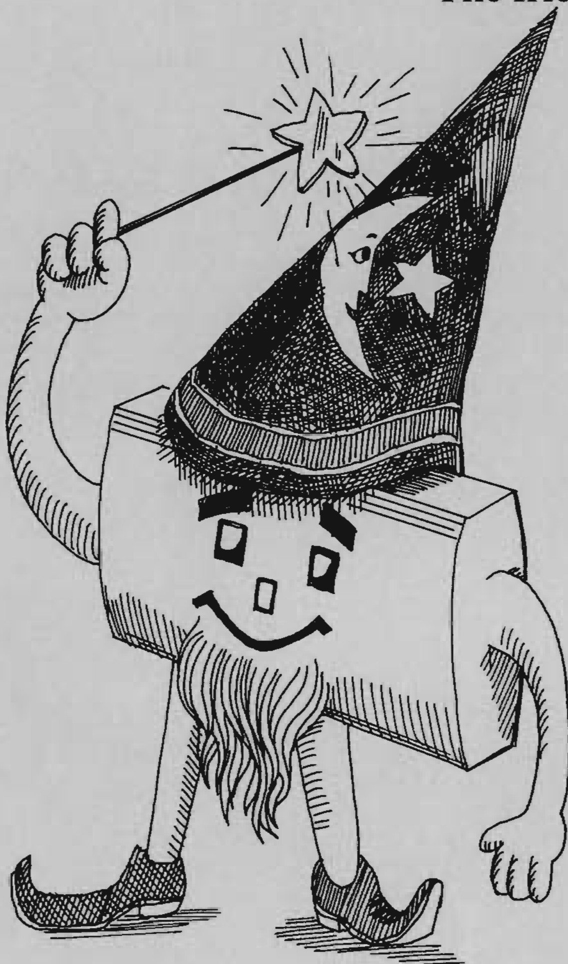
VIC 3K Memory Expander Cartridge
Plugs directly into the VIC's expansion port, expands memory to 8K RAM total. **$39.95.**

RS232C Terminal Interface Cartridge
Provides interface between the VIC 20 and RS232 telecommunications modems. Connects to the VIC's user port. Available: November. **$49.95.**

In addition to the products mentioned above, Commodore is also developing several other peripherals which will follow the same "VIC 20" philosophy—to produce "friendly" accessories at affordable prices. Some of these products include: an IEEE-488 interface cartridge, a low-priced modem, a 6-slot expansion module (accepts 3 different types of cartridges), and a 16K RAM expansion cartridge (note the 16K cartridge only operates with the expansion module). These items are scheduled for introducton after Christmas. ■

VIC 20

The friendly computer



The VIC Magician

Your First Computer "Magic"

by Michael Tomczyk
VIC Product Manager

Writing programs for Commodore's VIC 20 is a lot like performing magic. The results are certainly astounding, and your friends are sure to be amazed.

Actually, computing isn't much different from magic if you're talking about illusions. For example, the VIC automatically tells you if you make a programming mistake by displaying an "error message" on the screen. That doesn't mean there's an "intellectual rabbit" hiding under the keyboard. It simply means the VIC 20 is a "logical" machine.

We're going to be exploring the VIC's peculiar logic—and some magic, too—in this series of articles which focus on elementary BASIC programming. The purpose is to show first-time computer owners how to COMPUTE, with secondary emphasis on hard-line programming. The philosophy is: "you don't have to know how to repair a car in order to drive one; likewise, you don't have to be a computer scientist to "drive" the VIC 20."

That's the beauty of Commodore's "friendly computer." It's easy to learn, fun to "drive," and you don't need a license (or PhD.) to use it.

Everyone likes to perform magic with their new computer, but doing these neat tricks the salesperson showed you in the store doesn't seem so easy when you get the thing home.

Here are some of the favorite programs of Commodore's "VIC Group." These programs are not only fun, but they incorporate some helpful computing techniques which you might want to mix, match, and experiment with. Most of these programs are explained in the VIC owner's guide.

Before we begin, here's a quick refresher on how to enter a program into the VIC 20:

1. Type the program line-by-line as shown, including the line number.
2. Hit the RETURN key at the END of each numbered line of instructions.
3. Type the word RUN and hit RETURN to make the program execute.
4. To stop a program which is "running," hit the RUN/STOP key.
5. You can RUN a program over and over by STOPping it and typing RUN (because the program stays in the VIC's memory when you type it in).
6. Before typing a new program, type the word NEW and hit RETURN to erase the old program.
7. If a program "hangs up," hold down the RUN/STOP key and hit the RESTORE key. This resets the VIC without losing the program.

VIC-TRICK #1 (NAMES, NAMES, NAMES)

```
10 PRINT "YOUR FRIEND'S NAME . . .";
20 GOTO 10
```

This is a fun program which most of us have used over the years to impress our friends who never saw a personal computer close up before. Type in a person's name, with some dots to make it more readable on the screen. The PRINT command displays the name, or any message between the quotation marks, on the screen. The GOTO command tells the VIC to go back to line 10 and print it again. The semicolon at the end of line 10 makes each message appear right next to the last one. Try changing the semicolon to a comma, or leaving it off entirely, and see what happens.

VIC-TRICK #2 (255 COLORS)

```
10 FOR X = 1 TO 255 STEP 1
20 POKE 36879, X
30 PRINT CHR$ (147)
40 FOR T=1 TO 700: NEXT T: NEXT X
```

This little program displays the VIC's 255 screen and border color combinations. It's very helpful because you can go

VIC=20
The friendly computer

through all 255 combinations step-by-step, and find the color combination you like best for a particular program. The POKE command in line 20 is the key. RUN the program until you see a color combination you like. Hit the RUN/STOP key to freeze the colors. Write down the POKE command shown on the screen for reference. Now type CONT and hit RETURN to continue the program from where it left off. This is the "lazy" approach to choosing colors. The best way is to check page 37 or 134 in the VIC owner's guide.

VIC-TRICK #3 (THE ROLLING SCREEN WINDOW)

```
10 POKE 36867,4: PRINT CHR$ (147)
20 PRINT "YOUR MESSAGE HERE . . ."
30 FORX=0TO120:POKE 36865,X:NEXT
40 GOTO20
```

You can change the screen window of the VIC 20 by using some of the special "POKE" commands, which change the size and position of the VIC's screen window. This little program makes use of these commands to make your message scroll downward across the screen. The message in line 20 should be 22 letters long. Try typing 22 hearts (Hold down the SHIFT key and type S which has the heart on the front) instead of a message.

VIC-TRICK #4 (THE SEASICK PROGRAM)

```
10 PRINT CHR$ (147) "SEASICK"
20 FOR L = 0 TO 6.28 STEP .1
30 POKE 36864, 5 + 4 * SIN(L)
40 POKE 36865, 27 + 4 * COS(L)
50 NEXT: GOTO 20
```

Michael Tomczyk, VIC Product Manager

This program makes the screen move around . . . and around . . . and around. We call it the Seasick program because that's how you might feel if you stare at it too long. The programming magic here is the VIC's ability to move the screen around using POKE statements.

VIC-TRICK #5 (DRAWING A HIGH RESOLUTION CIRCLE)

```
10  FOR S = 7168 TO 7679: POKE S,0: NEXT
20  POKE36879,8:PRINT CHR$ (147);
30  FOR S = 7680 TO 8185: POKES, 160: NEXT
40  POKE 36869,255
50  FOR L = 0 TO 7: FOR M = 0 TO 7
60  POKE 7680+M*2+L, L*8+M        2
70  NEXT M, L
80  FOR X = 0 TO 63
90  Y1 = 32 + SQR (64*X-X*X)
100 Y2 = 32 - SQR (64*X-X*X)
110  FOR Y = Y1 TO Y2 STEP Y2-Y1
120  CH = INT (X/8) *8 + INT (Y/8)
130 RO = (Y/8 - INT (Y/8 - INT(Y/8))*8
140 BY = 7168 + 8*CH + RO
150 BS = 7-(X-INT(X/8)*8)
160 POKE BY, PEEK (BY) OR (2 BS)
170 NEXT
180 GOTO 130
```

This 18-line program looks like a lot . . . but it does a lot. This is our first program which actually draws something on the VIC 20 screen in high resolution, dot programmable graphics. Dot programmable graphics are different from VIC graphics, in that VIC graphics are made up of 8x8 dot blocks (64 dots per block). Dot programming lets you access each dot individually, and "draw" in high resolution programmable graphics. The mechanics of how to do this are discussed in the upcoming VIC 20 PROGRAMMERS REFERENCE GUIDE. But, if you really want to get into programmable graphics and plotting, we suggest you watch for the VIC 20 SUPER EXPANDER CARTRIDGE, which will be introduced by Christmas. This special cartridge gives you 3K RAM extra memory and adds several new commands to VIC BASIC that let you plot individual points, lines, arcs and circles . . . and even let you "paint" closed figures on the screen in color! The Super Expander also has built-in music writing commands and a special "music mode."

We hope this brief "magical" introduction to the VIC gives you some interesting programs to experiment with. In future columns, we'll continue our exploration of the VIC's capabilities and give you a magician's hat full of programming tips and tricks to help you become a "VIC Wizard."

If you have a particular topic you'd like us to discuss in this column, please drop a line to VIC MAGICIAN, in care of this magazine. ■

Some Thoughts on the PET/CBM Systems

by
Elizabeth Deal
Malvern, Pa

This is a review of several attributes of the PET/CBM computers I found to be important. The word "PET" will be used throughout to include both the PETs and CBMs.

There is a term in the personal computer business that has caught my attention lately—"user friendly." It is a strange term but one that fits the PET so well, one grows to accept it. For the PET has "friendly" features undreamed of in some other personal computers, with the exception of the VIC. Generally, computers share a lot of features. What sets them apart may or may not be of importance to most applications. But whether the computer is easy to use is quite important, whether the user be a child, an old pro, or an elderly or handicapped person.

Do not underestimate the weight of this aspect, for this is what will make using a computer a smooth and pleasant activity or turn it into drudgery. And it makes little difference whether you plan to program your own computer, use existing programs on the market, or switch between those activities. Working with the computer can be very rewarding when it does what you planned and in a manner that harmonizes with your way of doing things.

From the moment the PET is turned on, it works smoothly. The key reasons for the ease of use of the PET are, in my opinion, its outstanding cassette tape system, screen editor, the keyboard and meaningful way of communicating its messages to you. Needless to say, most of the other features of the PET work nicely together.

Let's look at these features that make the PET so "user friendly."

The Tape System

The PET tape system works very well. It is slow but reliable. And it is inexpensive. Part of the reason for reliability is the use of a very special and technically accurate recording method. Another part is that all information, programs and data are recorded twice. A VERIFY command is available and should become part of anyone's habit in SAVEing things for subsequent use. It checks whether the information on the tape matches the information in the computer.

The tape recorder, of course, needs a bit of care to function properly. The recording head must be demagnetized every once in a while. The recording and the erasing heads and the rubber wheel need to be gently scrubbed with alcohol (*not* rubbing alcohol—that leaves deposits) or a Freon type head cleaner and a Q-tip.

How do you tell when this procedure is needed? Timing of recording sessions and going by the book is a bore. Don't do it. A somewhat unreliable—though enlightening—way is to listen to the CB2 sound as the tape goes by. When the sound becomes sloppy it's time to scrub. The third way seems to work well and is the easiest of all. Forget the whole machine until the first occurrence of ?VERIFY ERROR. (I really mean first, no point wasting time in risking a second error.) Then keep your PET turned on, clean the recorder very carefully, making sure that not one little bit of the Q-tip stays behind. And finally, keep it running for several minutes to dry, SAVE the program again on another tape and the next message, invariably, will be VERIFYING . . . OK. You should never experience LOAD ERROR if you do this.

The brand of cassettes that are used is also important. It must be computer-grade tape. But this does not necessarily mean expensive. Such tapes often cost much less per foot than the heavily advertised tapes used for sound recordings. There are many places where these tapes can be obtained. Ten minute tapes can be purchased for under 60 cents and a thirty minute tape for around 75 cents.

Management of tapes you are working with is something you will pick up as you go along. You'll find your own scheme of labelling them and ways of knowing which is the last good version of the program you were writing just before it got mangled in the never-ending improvement process. You have to be neat in this department but the PET will help, because no matter what you put on the label, the PET will keep it straight. In case of total confusion several VERIFY commands will display the complete names of files on the screen. The visible part of the file name can be up to sixteen characters long. A simple scheme of 1NAME, 2NAME, 3NAME etc., permits you to LOAD by asking simply for a number, as in LOAD"3, while at the same time displaying the full name. There is no need for guessing what is coming in, and no guessing as to the names of the programs that are passed by.

Most people do not have much trouble devising methods of tape management. Some big system programmers do. They are used to putting their "jobs" in little bins at night and getting something back in the bin in the morning. Having been spoiled by tidy computer operators who managed all this, they are often the ones least able to adapt to a home computer where they have to do all the dirty work themselves, including turning on the computer.

Overall, PET files are compatible with one another whether it is a screen, tape or a disk file. This means that switching from a tape system to a disk system is a matter of minor programming changes. Changes do need to be made in the file set up, but once you learn how to do it, it is not difficult. Numerous, good commercially available programs support both types of devices and permit a choice to be made by the user at any time. This is an important aspect since it permits flexibility as your system changes. ➡

The reliability of PET's recording system is not limited to tapes. Disk drives equipped with DOS2.0 and higher are equally reliable. However, that subject is beyond the scope of this article.

A bit of perspective concerning cassettes: It is a pleasure to watch a group of small boys look through some tapes, pick a game, and load it into the PET without any difficulty by use of any variety of "put it in" commands they can think of. It is frustrating to watch a dealer or another adult with a different computer model, first fumble for the program's one letter name or its loading address in hex, have trouble loading it and giving up with a meager explanation that the cassettes just don't seem to load "today."

I suspect the reason they don't load, or save for that matter, has nothing to do with "today." I think it's bad design of the equipment. I have seen very little use of cassettes other than with PETs at computer stores I have visited. I sometimes use a CBM disk drive and love it. But a first time computer buyer who feels forced to acquire a disk drive because his cassettes won't work for him has been ripped off.

The Screen Editor
It is a programmed device within the PETs operating system that together with the keyboard is the messenger between people and the rest of the PET. A good screen editor simplifies the work you do. A poorly thought out editor could make work difficult because instead of concentrating on WHAT it is you want to write you would have to concentrate on HOW to put it in.

PET has a superb screen editor. The more I use it the more I like it and the better I have learned to use it. One of its characteristics has a seemingly magical quality to it (of course, it is not at all magical, it's just been programmed nicely) in that you practically never have to retype anything that already exists on the screen or within a Basic program. If you typed RUN, the program ran and you want to execute it again, you do not need to retype the word RUN as long as it is on the screen. Placing the cursor over RUN and hitting RETURN acts as if you had typed it again. It is very handy in error-prone commands.

You may in this fashion duplicate lines for a BASIC program, again avoiding a lot of retyping. For example, in one place in the program you may have a line:
150 PRINT"THIS IS ORIGINAL LINE",
and you would like to say in another place:
485 PRINT"THIS IS DUPLICATE LINE".
No need to retype. Place the cursor over150, overtype 485, move the cursor to "O" in ORIGINAL, insert a space, overtype DUPLICATE, hit return key and line 485 has been entered into the BASIC program. The original line 150 is still there. Nothing could be more "user friendly."

The importance of a good screen editor may not become apparent until editing a mistake on one line in a program for the twentieth time begins to hurt your fingers and your nerves. But at least your computer can do the work it's are so good at—accuracy.

The ease with which text can be modified on the screen makes the PET particularly well suited for schools and homes with small semi-literate children. Since typing is a skill children are not born with it is important to provide them with a system that minimizes the typing effort. They still will need to learn the keyboard and will in due time. But when a computer can do so much work for them, the learning is easier. PET carries that burden very nicely.

The Keyboard
The ancient QWERTY design of the keyboard is still with us and PET is no exception. Everybody uses it because everybody has always used it. People and children who cannot type find it frustrating to use as they try to both spell the words correctly and look up the letters in an out of order arrangement. PET computers, as opposed to CBM computers, have an invention in a "graphic" keyboard which, for the nonprofessional typist, is quite easy and pleasant to use. Not only does it contain visible graphic characters that exist within the system, it also permits typing without shifting. (On all Commodore computers the upper and lower case letters as well as the graphic symbols are provided.) The only characters that need to be shifted are certain combinations of screen editing keys. The physical layout of the keyboard makes it possible to do so with one hand. Very handy. Being able to actually see a graphic character on the keyboard and screen as in:
PRINT "•"
makes use of the keyboard less abstract. In contrast, having to write:
PRINT CHR$(145)
to represent the big fat dot on the screen is less user-oriented, though equally accurate, and sometimes even necessary for compatibility reasons. The nice thing about the PET is that both ways are OK.

The keyboard and the screen editor work well together. Editing functions are available at the fingertips. These are: four cursor movement keys—up, down, left and right, one key to wipe out the screen and/or to place the cursor in the upper left hand corner, and the last key is used to delete letters you do not want or to insert extra room within a line so that something can be typed into that room. No programming tricks are needed to use screen editing functions. But these functions can be coded into a program for some elegant screen display manipulation. You have the best of two worlds.

The business keyboard also contains these editing functions. And the computers equipped with the 80-column screens have several useful additional editing keys. For example, you can scroll the screen down or split the contents of the screen at the cursor position in order to insert some text in that place. You can live without this feature, but in programming, it is visually pleasing to have it.

As you can see, the keyboard has also been designed in a "user friendly" manner. It works well because it permits one to type or make corrections with the least amount of hassle.

Two Way Communication
This is the fourth aspect of the excellent human interface of the PET. You talk to

your PET via the keyboard. The screen editor helps you talk correctly so that the PET can understand your instructions. The cassette system helps save your instructions for the next day so that you do not have to retype them over and over. Your job being done, it is PET's turn to interpret instructions you have entered. Your PET will not understand an incorrectly typed instruction. Your PET will not accept numeric values that are too large to handle. It will not accept letters when numbers are expected. You'll know about those errors soon enough. PET will say nasty

?SYNTAX ERROR
?ILLEGAL QUANTITY ERROR
?FILE NOT FOUND ERROR
?DEVICE NOT PRESENT ERROR

These messages may look cryptic. But their meaning comes across as you look over the code that might have caused it. You do not have to open a book to understand an error message. Contrast this with systems that say MD ERROR or ERROR 17. The two-way key communication breaks down. You do not know what the computer is saying. You have to waste your time looking for the book.

WRAP UP

In case you haven't guessed already, I am a big fan of Commodore computers. Even though I accept the existence of many good features of other systems on the market, the PET, to me, is still the best value in a microcomputer.

Both hardware and software support for the PET has been incredible over the past few years. Should you need something extra, it can often be purchased separately from numerous cottage industries that have appeared on the scene.

Several examples come to mind. The list is not meant to exclude anybody's product—but to keep it short I'll list a few things I've seen. High resolution (pixel level) graphics from MTU, Steve Punter's Wordpro from Professional Software, Visicalc from PersonAl Software, Jinsam Data Base system from Jinni Micro Systems Inc., Ozz information management from Commodore. There are music synthesis systems, one put out by AB Computers, another by MTU. MTU also has an advanced music system that synthesizes instruments in real time. There is color in the VIC computer. There are chips (Palo Alto IC-s Toolkit) you can plug in that add some fancy commands to BASIC resident in the PET. There are large and small assemblers for the machine language people. There are instruments that can be used with the PET via its IEEE-488 interface. It's all there.

The support of the user has been made possible by various user groups, the most notable being the Toronto group with Jim Butterfield. These people have shared their discoveries with the general public. You find that most questions, that occur to you, have already been raised and answered.

Finally, relevant information regarding Commodore computers is available through sources such as: PET Revealed by Nick Hampshire, COMPUTE! magazine, Commodore Canada—TRANSACTOR, PET/CBM Computer Guide by Osborne & Donahue, and others. They are worth looking into if you want to learn more and to expand this editorial material.

Elizabeth Deal is a Malvern, Pa.-based free-lance writer whose work frequently appears in COMPUTE! ■

Some Commodore Disk Utilities

Reprinted from the "Transactor"

The following routines are for use with 4040 and 8050 disk units.

The first two are for reading the disk ID from the specified drive DR, device DV. Line 150 sets the drive. Line 160 sets the track. Line 170 tells the disk to read any header on track 18. Initializing is not necessary! 180, 190 and 200 wait for the DOS to finish the read. Line 210 does any error processing (i.e. read error or no disk in drive). The first character of the ID is read from DOS memory (line 220) and the DOS puts it in the command channel. Line 250 reads the second character and both characters are put in 'ID$.'

This routine is particularly useful (and fast) to see if the user has placed a disk in the drive. It can also be used to detect insertion of incorrect diskettes. The software would have to anticipate ID numbers, perhaps ID's that were selected by the program in an earlier formatting operation.

The second two will return the BLOCKS FREE count BF, from the specified drive DR, device DV. BF is reset before entering. Initializing is necessary here! The block-free count is not stored on the disk but rather calculated from the Block Availability Map. Line 170 sends the DOS off to that routine in disk ROM! The result is placed in disk RAM where it is read, once again, into the command channel by lines 180 and 210. A few calculations and presto! Block Free!

Knowing blocks-free from within a program can be especially useful for anticipating the nasty DISK FULL error.

Versions for both 4040 (DOS 2.0) and 8050 (DOS 2.5) have been provided. The fifth program was written by John Collins of the U.K. It's the definitive subroutine for determining host equipment.

```
100 REM ID READER FOR 4040
110 ID$="" : REM RESET ID$
120 DR=0   : REM DRIVE #
130 DV=8   : REM DEVICE #
140 OPEN15,DV,15 : REM UNLESS ALREADY OPEN
150 PRINT#15,"M-W"CHR$(18) CHR$(0)
    CHR$(1)CHR$(DR)
160 PRINT#15,"M-W"CHR$(43)
    CHR$(16)CHR$(1)CHR$(18)
170 PRINT#15, "M-W"CHR$(4) CHR$(16)
    CHR$(1)CHR$(176+DR)
180 PRINT#15,"M-R"CHR$(4)  CHR$(16)
190 GET#15,X$
200 IF ASC(X$)  127 THEN 180
210 IF ASC(X$)     1 THEN PRINT#15, "M-E"
    CHR$(37)CHR$(217):PRINTDS,DS$:END
220 PRINT#15,"M-R"CHR$(41)CHR$(16)
230 GET#15,A$
240 ID$=A$
250 PRINT#15,"M-R"CHR$(42)CHR$(16)
260 GET#15,A$
270 ID$=ID$+A$
280 PRINTID$
```

```
100 REM ID READER FOR 8050
110 ID$=""  : REM RESET ID$
120 DR=0 : REM DRIVE #
130 DV=8 : REM DEVICE#
140 OPEN15,DV,15 : REM UNLESS ALREADY OPEN
150 PRINT#15,"M-W"CHR$(18) CHR$(0)
    CHR$(1)CHR$(DR)
160 PRINT#15,"M-W"CHR$(43) CHR$(16)
    CHR$(1)CHR$(18)
170 PRINT#15,"M-W"CHR$(4)  CHR$(16)
    CHR$(1)CHR$(176+DR)
180 PRINT#15,"M-R"CHR$(4)  CHR$(16)
190 GET#15,X$
200 IF ASC(X$)  127 THEN 180
210 IF ASC(X$)    1 THEN PRINT#15, "M-E"
    CHR$(179)CHR$(238):PRINTDS,DS$:END
220 PRINT#15,"M-R"CHR$(41)CHR$(16)
230 GET#15,A$
240 ID$=A$
250 PRINT#15,"M-R"CHR$(42)CHR$(16)
260 GET#15,A$
270 ID$=ID$+A$
280 PRINTID$
```

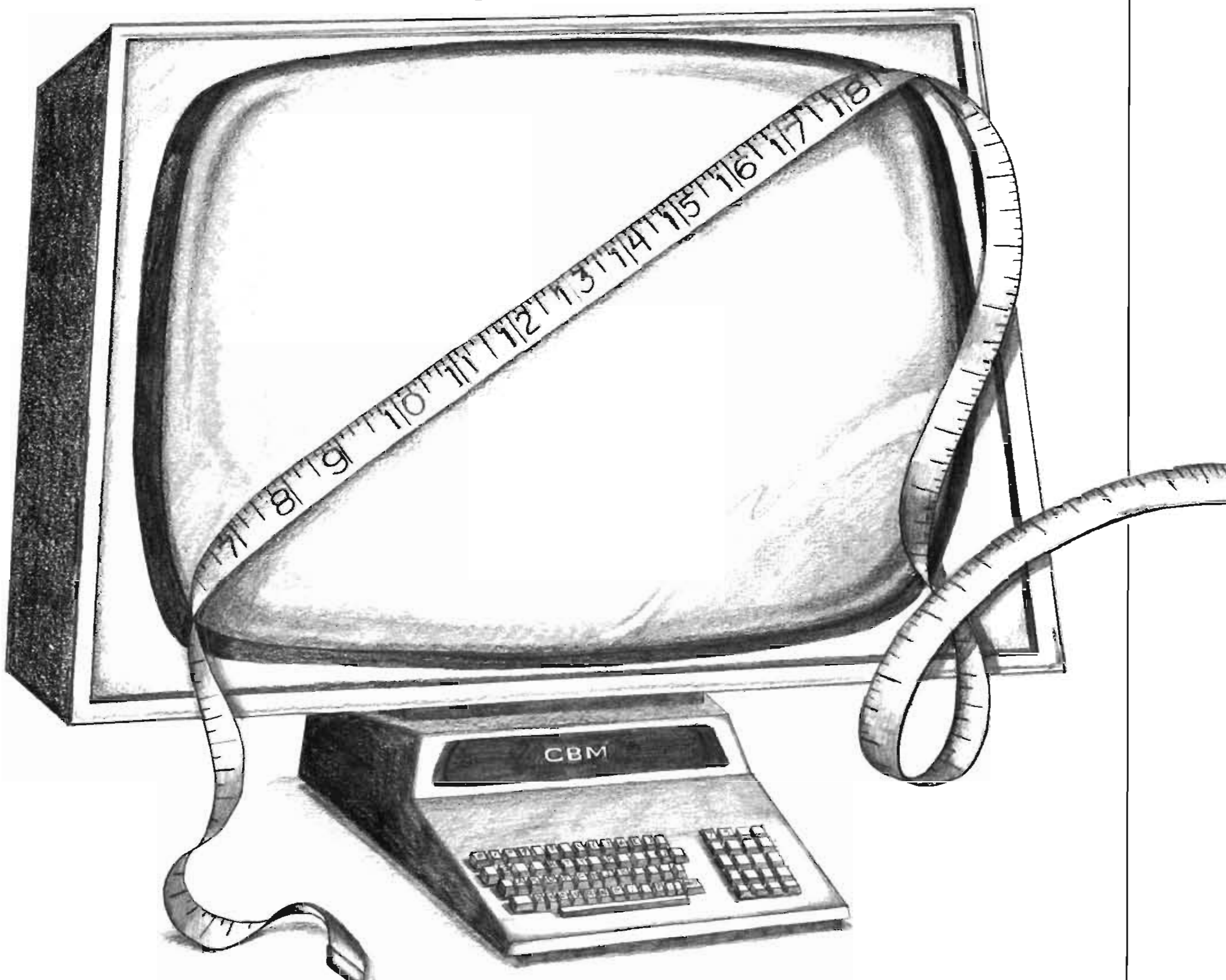
```
100 REM SUBROUT. RETURNS BLOCKS-FREE FOR
    DOS 2.0
```

```
110 DR=0 : REM DR = DRIVE #
120 DV=8 : REM DV = DEVICE#
130 BF=0 : REM RESET BLOCK FREE COUNT
140 OPEN15,DV,15 : REM UNLESS ALREADY OPEN
150 PRINT#15,"I"+STR$(DR) : REM UNLESS
    ALREADY INIT'D
160 PRINT#15,"M-W"CHR$(18)CHR$(0)CHR$(1)
    CHR$(DR)
170 PRINT#15,"M-E"CHR$(52)CHR$(219)
180 PRINT#15,"M-R"CHR$(119)CHR$(67)
190 GET#15,A$
200 BF=ASC(A$+CHR$(0)) : REM IN CASE A$=""
210 PRINT#15,"M-R"CHR$(120)CHR$(67)
220 GET#15,A$
230 BF=BF+ASC(A$+CHR$(0))*256
240 PRINT BF
```

```
100 REM SUBROUT. RETURNS BLOCKS-FREE FOR
    DOS 2.5 (8050)
110 DR=0 : REM DR = DRIVE #
120 DV=9 : REM DV = DEVICE#
130 BF=0 : REM RESET BLOCK FREE COUNT
140 OPEN15,DV,15    :REM UNLESS ALREADY OPEN
150 REMPRINT#15,"IO"
160 PRINT#15,"M-W"CHR$(18)CHR$(0)CHR$(1)
    CHR$(DR)
170 PRINT#15,"M-E"CHR$(231)CHR$(211)
180 PRINT#15,"M-R"CHR$(158)CHR$(67)
190 GET#15,A$
200 BF=ASC(A$+CHR$(0) : REM IN CASE A$=""
210 PRINT#15,"M-R"CHR$(160)CHR$(67)
220 GET#15,A$
230 BF=BF+ASC(A$+CHR$(0))*256
240 PRINT BF
```

```
100 REM VERSION TEST FOR PET/CBM AND DISK
110 REM BY: JOHN COLLINS
120 REM PET/CBM TEST TP;
130 REM 2001 BASIC 1.0 = 0
140 REM 2001 BASIC 2.0 = 1
150 REM 4032 BASIC 4.0 = 2
160 REM 8032 BASIC 4.0 = 3
170 :
180 A=PEEK(57345):TP=0:IF A THEN TP=1:
    IF A AND 1 THEN TP=3:IF A AND 4 THEN
190 :
200 REM DISK TEST    TD;
210 REM 2040 (DOS 1.0) = 1
220 REM 4040 (DOS 2.0) = 2
230 REM 8050 (DOS 2.5) = 3
240 :
250 OPEN 15, 8, 15
260 PRINT#15,"M-R"CHR$(255)CHR$(255)
270 GET#15,A$
280 CLOSE 15
290 A=ASC(A$):TD=1:IF A AND 16 THEN TD=3:IF A
    AND 1 THEN TD=2
300 :
310 REM RESULTS:
320 PRINT TP, TD ■
```

PROGRAMMER'S TIPS



Two Points of View on the 'FAT FORTY'

Converting Programs to 4032 12-inch

The addition of some new features to the 40 column PET has brought about some problems with program compatability between the 4032 10-inch and 4032 12-inch CRT display machines. In some cases, the changes required to program for proper operation on the 'FAT FORTY' may be trivial, and in other cases the conversion may be next to impossible. In the following text, I will attempt to cover the areas where possible failure can occur and what changes need implementing.

Level 1
Programs Loading Below Basic ($0400)

Standard BASIC programs should work without any modification, unless they employ PEEKs and POKEs or if the program loads into memory below BASIC. The latter problem can be a bit tricky to spot unless you know specifically what to look for. If the program does load below BASIC (say $033A), but does not use locations $03E9-03F9, one method that will correct the problem is:

1) get the program into memory (don't execute)
2) SYS to the monitor (SYS 4)
3) display hex $03E9-03F9
4) modify these registers with the following values

[$03E9: 10,10,09,10,00,00,00,00,00,00,00,00,00,00,00,00,00]

5) resave the program via the monitor

Tape Unit #2

Another area where the standard BASIC program can fail is in the utilization of the second cassette unit for sequential file access. If any program calls files from tape unit #2, unpredictable effects can result, depending on the data coming into the buffer. To date, nothing can be done to resolve this problem.

Peeks and Pokes

Decimal location 151, which is often used to check if a particular key has been pressed, is still the same on the 12-inch. However, the value of the keys have changed, so expected values for certain keys will return false information. The following table will assist in the conversion of a program with this problem.

KEY	OLDV	NEWV	KEY	OLDV	NEWV	KEY	OLDV	NEWV
@	15	64	A	48	65	B	30	66
C	31	67	D	47	68	E	63	69
F	39	70	G	46	71	H	38	72
I	53	73	J	45	74	K	37	75
L	44	76	M	29	77	N	22	78
O	60	79	P	52	80	Q	64	81
R	55	82	S	40	83	T	62	84
U	61	85	V	23	86	W	56	87
X	24	88	Y	54	89	Z	32	90
1	26	49	2	18	50	3	25	51
4	42	52	5	34	53	6	41	54
7	58	55	8	50	56	9	57	57
0	10	48						

When POKEs to this problem area are used for saving byte variables (whatever purpose), they must be moved to a free spot elsewhere in memory. If space is free just below $03E9, this could be a good area for relocating the byte variables.

Level 2
BASIC Programs with Machine Language Utilities
BASIC programs using machine language utilities that reside in the second cassette buffer can work properly, provided they don't use the taboo area of the buffer (namely decimal 1001-1017). Again, if the utility uses this area, the space must be relinquished to the PET operating system, in order to obtain successful execution of the program. Usually, in the case of small machine language utilities, it shouldn't be too difficult to understand and relocate it to an area of free memory.

Level 3
Machine Language Programs
This will be the most difficult area to troubleshoot. If you are going to attempt modifying this type of program, be prepared to spend a good deal of time. Making the necessary changes to get the program working will most likely require a considerable amount of effort, which I personally don't recommend. In most cases the author should be contacted and he should facilitate the changes. If you are really desperate, here are a few helpful hints which may assist you:

1) Use Supermon or Extramon to locate any absolute occurrences of memory addresses from $03E9-$03F9, and reassign new values.
2) Check hi-low tables for references to the same address locations and if any, reassign new values.
3) Seek all immediate operations involving hex $03 and $e9-f9 . . . if any, look at code where occurrence takes place and evaluate.
4) Check all JSR & JMP occurrences into the 'E block ROM.' All other ROMs can be ignored since they are identical.

Factory CRT Setup (Size)
One other problem that may or may not be encountered is screen setup. If the user decides on entering screen text mode via 'PRINT CHR$(14)' the top line of screen may run off the upper edge and not be visible. To restore screen to normal graphics mode, enter 'PRINT CHR$(142)'. One easy solution to this problem is to use 'POKE 59468,14.' This will turn the PET to text mode without opening up pixel lines between text.

Conclusion
The changes required to existing software may be a problem now but, at the same time, these changes bring the 4032 to a closer compatability with the 8032 model. Features such as repeat keys, scroll—up and down, bell and more have been implemented. These changes make the 4032 a much more desirable product. ■

Joe Ferrari
Commodore Canada

The Fat 40
A New Generation 40 Column PET

There's a new kid on the block. It arrived without any fanfare or advanced publicity: the Fat 40—a large-screen 40 column PET.

The most noticeable feature of the fat 40 is its big 12-inch screen. You'd almost think you had an 8032 80-column computer until you look at the keyboard and notice that the keyboard layout is graphics rather than ASCII business. When you turn it on, the screen characters come up fat and crisp.

I'd heard that Commodore would be fitting 12-inch screens to their 40-column computers, and had thought that this would be only a screen change. Most of us, I think, were unprepared for a noticeably new machine with new logic. The logic board has changed so that it is similar in layout to that of the 8032. More importantly, the program logic has been changed so that we have more than a new size of machine: we have a new style.

The changes are improvements, but we'll have to recover from the shock of having a new system introduced with no advanced fanfare. Most programs which work on the skinny 40's will move to the fat 40 without trouble; I'll try to itemize possible conflict areas later in this article.

A Junior 80?
The fat 40 picks up much of the style of the 80 column machine. In fact, many of the features of the 8032 are now available in 40 columns, including ringing the bell, erasing part or all of a line, and screen tabulation. Windows—the ability to pick out a piece of the screen—and up/down

→

scrolling, are not available.

The user will notice one of the features right away—as he turns the machine on, a bell chimes. Yes, there's an electronic bell in there, and it chimes any time you get near the end of a line. You can make it ring from your program with PRINT CHR$(7). The bell may be turned off or changed by poking the contents of location 1004 decimal—note that this is a different location from that used in the 80-column machine (231 decimal).

Another feature that's hard to miss is the repeating keys. The cursor movement keys, the Space key, and the Insert/Delete key repeats automatically if the key is held down for a moment. Very handy.

The user may switch to Text mode (uper/lower case) with PRINT CHR$(14). This is easier to remember than the POKE address, and gives you another bonus: the screen lines are readjusted to give you somewhat more pleasant text appearance. You may return to graphics with PRINT CHR$(142).

There are commands for clearing all or part of a line; it takes a little dexterity to get them to work right since you must carefully place the cursor before using them. To clear a line up to, but not including, the cursor position, type PRINT CHR$(15). To clear from the cursor to the end of line, type PRINT CHR$(16). This appears to be an error on Commodore's part, since PRINT CHR$(22) does the job on the 8032 and is more consistent. It seems as if someone has confused decimal 16 with hexadecimal 16 (22) in the coding. I would anticipate this being corrected in future ROMs; in the meantime, you can print both characters and create programs which will move gracefully through any such future corrections: PRINT CHR$(16);CHR$(22). This Erase-to-End sequence is often very handy: it allows you to clear a line before printing on it.

Screen tabulation is included. It will probably be less useful on a 40-column machine as compared to its 80-column big brother, but it's not hard to use. Set or reset tab positions by positioning the cursor and then printing CHR$(137). You can move to the next tab location when desired by printing CHR$(9).

System organization changes

Users who fiddle with the innards of the machine—or who have programs that do so—should be aware of certain internal changes in the fat 40.

The keyboard is decoded in a different manner from previous 40-column machines; you should not depend that PEEK(151) will give you the same values for specific keys being pressed. You may still expect that this value will be 055 if no key is pressed—but the other values will have moved around.

The Screen Wrap table, which tells us which rows of the screen are joined together into double lines, is still in the same place. But this means that some of the 80-column variables needed to be moved clear of this area; values previously stored in locations E3 to E8 have been relocated to the area 03E9 to 03EF. You don't need to know all of their functions, but you must leave these new address locations alone, or you'll interfere with screen/keyboard operation.

80-column tyros might wish that all the wide screen features had been implemented on the fat 40. It probably wasn't possible: the two machines need to be organized differently. Users who go looking for the 80-column screen windows or for the input and output vectors will be disappointed: they aren't there.

Summary

It's an exciting new machine. It has good new features, and should be very convenient. I wish that Commodore had told us it was coming . . . but in any case, I'm glad it's here.■

Jim Butterfield
Toronto

SUPERPET

The SuperPET—A First Look

Reprinted from the "Transactor"

Introduction
The dawning of a new age in the microcomputer industry is fast approaching. The announcement of the SuperPET Computer by Commodore has demonstrated that a new breed of micros is on its way. With 5 languages currently available, and a whole lot of RAM space, it makes the SuperPET one of the most versatile computers on the market today.

I was able to get my hands on one of them for testing and evaluation. Since it is a pre-release version, all information contained in this article is subject to change. There is a lot to be said about the SuperPET, but I will discuss mostly Waterloo microBASIC and its features.

Powering up the SuperPET brings up the usual Commodore BASIC, but flicking a switch located on the bottom right side of the compter will bring up Waterloo microSystems menu. All the languages available are boot-loaded from disk, with the exception of the monitor, which is resident in ROM. Selecting b [return] will load Waterloo microBASIC with 30150 bytes free.

Editor Features
For those of you who frequently use a developer's toolkit, you will be pleased to know that Waterloo microBASIC includes the following features in their editor for the development programs.

A) DELETE
B) RENUMBER
C) AUTOLINE
D) MERGE
E) STEP

Debugging Programs
Here is one feature that I really like. When a program has been interrupted either by the operator or due to an error in the program, changes or modifications can be made and execution of the program can continue at the point of interruption, or at any location desired, without the loss of the current variables.

Saving and Retrieving Program Files
Unlike Commodore BASIC, Waterloo microBASIC provides several options on how a program can be saved and retrieved.

1) STORE "FILENAME"

This option will output the program to the disk in a compressed form. Two advantages of saving programs in this manner are:
a) occupies less space on disk
b) faster in storage and retrieval

2) SAVE "FILENAME"

This second option will save the program as it is typed on the keyboard , and will appear on the directory as a sequential file. This method should be used only while the program is under development. The advantage of this mode is that portions of the program can be saved and eventually merged with other programs by specifying a line range.

3) LOAD "FILENAME"

This command retrieves program files that were saved with the 'STORE' command.

4) OLD "FILENAME"

The "old" option will load a program file that was created with the 'SAVE' command. Any existing program in memory is cleared and the new program is then loaded.

5) MERGE "FILENAME"

This option adds the program specified to the existing program in memory. Be very careful when using this command, for any new line number coming in, that already exists in memory, will be deleted and the new line inserted in its place.

String and Variable Representation
If you have ever encountered a point in program development where you can't think of an appropriate name for a variable, you won't have such a problem in microBASIC, because all string and variable names can have up to 31 characters with upper and lower case representation.

Repetitive Structures
Waterloo microBASIC supports a variety of statements that make coding easier to implement and also much easier to understand. Since we are all familiar with the FOR-NEXT LOOP structure, I will skip it entirely and go on to some you may or may not know.

WHILE-ENDLOOPS
Delimits a group of statements to be executed repetitively, if the value of the expression is true (non-zero).

Example:

```
X=15
WHILE X
    PRINT X
    X=X-1
ENDLOOP
```

The example above will execute statements starting with WHILE and ending with ENDLOOP, until the value of variable X is equal to zero. As you can see, ENDLOOP is similar to saying GOTO only without specifying a line number. As a point of interest, Waterloo microBASIC executes much more efficiently with structured loops, in comparison with GOTO statements.

LOOP-UNTIL
Delimits a group of statements to be executed repetitively, if the value of the expression in the UNTIL statement is true (non-zero). Control is then passed to the statement immediately following the UNTIL statement. ➔

Example:

```
X=15
LOOP
    PRINT X
    X=X-1
UNTIL X=4
```

LOOP-ENDLOOP
Defines a group of statements to be repeated infinitely. To exit loop, the STOP key is pressed or an IF-THEN statement is introduced.

Example:

```
X=0
LOOP
    PRINT X
    X=X+1
    IF X=10 THEN QUIT
ENDLOOP
```

Execution of the statements falling within LOOP and ENDLOOP will be repeated until the value of X is equal to 10. Control is then passed to the statement following ENDLOOP.

IF-THEN-ELSE
Those accustomed to Commodore BASIC are familiar with the IF-THEN statement. MicroBASIC offers an extension to this statement with the ELSE option. The set of statements following ELSE are executed if the condition is false. ENDIF delimits the ELSE statement group.

Example:

```
COUNTER%=1
LOOP
  TESTER% = INT (COUNTER%
    / 2)* 2
  IF TESTER% = COUNTER%
    PRINT COUNTER%,
      "NUMBER IS EVEN"
  ELSE
    PRINT COUNTER%,
      "NUMBER IS ODD"
  ENDIF
COUNTER% = COUNTER%+1
IF COUNTER% = 10 THEN QUIT
ENDLOOP
```

The above example demonstrates the power of the ELSE option with the IF-THEN statement, especially if more than one statement is to be executed in each condition. I must say that utilization of structured coding makes for far more readable code.

Although there are many more features that I have not covered in this article, I hope it has given you a small taste of what to expect with Waterloo microBASIC on the SuperPET. ■

—Joseph P. Ferrari

Four PET/CBM Keyboards

by
Elizabeth Deal
Malvern, Pa

Programs containing PEEK(151)* are common in game applications and other uses where speed is of the essence. Location 151 contains a key number of a key that is being currently pressed. Unless made general, such programs cannot run on every release of the PET because of different ways in which the keyboard is decoded.

The lookup tables contained within this article should make it easier to convert such programs from one machine to another.

The key that is being pressed is decoded by the PET into a key value between 1 and 254. A value of 255 means no key was pressed. PET stores the key number in location 151 on the upgrade and Basic 4.0 computers, and in location 515 on the original PETs with small keyboards. Locations 152 and 516, respectively, indicate whether a shift key has also been pressed. If it is, the value is one, otherwise it is zero.

As you use the table, note the following nuances in the decoding methods:

1. Original, small keyboard PETs decode keys in a way almost identical to large graphic keyboard PETs (Basic2). That is the reason for one column in the table for both systems. Three exceptions: in the upgrade systems the RETURN key gives only 27 and the SPACE key gives only 6. SHIFT keys (L-S and R-S in the table) have no effect—they return 255.

2. On the 40-column CBM with Basic 4.0, the two ZERO keys are decoded differently from one another, while on the 80-column CBM, both ZERO keys return the same value.

3. The elegant decoding scheme used on the 80-column CBMs is such that the key number is equivalent to its ASCII value, with the exception of all keys on the numeric keypad and several special characters. Those decode into an ASCII value with the highest bit set, i.e. 128 added.

So there. No need to avoid programs that a friend gives you if they contain the speedy keyboard check of PEEK(151) and PEEK(152). You can try to write programs on your computer that could also run successfully on your friend's computer.

If your computer is not listed in the table you can make up your own list by running a routine similar to this:

```
10 PRINTPEEK(151)
20 write tape, disk or printer
30 GOTO10
```

This will permit you to check the table and to get all the key numbers except the STOP key.

*PEEK 515 on older PETs →

Elizabeth Deal is a Malvern, Pa.-based free-lance writer whose work frequently appears in COMPUTE! ■

KEY#	PET-GR-ORIG/UPGR	KEY#	CBM-40COL B4	KEY#	CBM-80COL B4
1	=	1		1	
2	.	2		2	
3		3	COLON AND *	3	STOP AND RUN
4	STP AND RUN	4	STOP AND RUN	4	
5	<	5	9 AND >	5	
6	SPC	6	6 AND &	6	
7	[	7	3 AND #	7	
8	RVS AND OFF-RVS	8	←	8	
9	-	9	1	9	TAB
10	0	10	/ AND ?	10	
11	R-S	11		11	
12	>	12	CLS AND HOME	12	
13	SPC	13	M	13	RET
14	]	14	SPACE	14	
15	@	15	X	15	
16	L-S	16	RVS AND OFF-RVS	16	
17	+	17	2	17	CD AND CU
18	2	18		18	RVS AND OFF-RVS
19		19		19	CLS AND HOME
20	?	20	0	20	DEL AND INST
21	COMMA	21	COMMA AND <	21	
22	N	22	N	22	
23	V	23	V	23	
24	X	24	Z	24	
25	3	25	3	25	
26	1	26		26	
27	RET	27		27	
28	SEMIC	28	.	28	
29	M	29	. AND >	29	CR AND CL
30	B	30	B	30	
31	C	31	C	31	
32	Z	32		32	SPACE
33	*	33	4	33	
34	5	34	[	34	
35		35	O	35	
36	COLON	36	CD AND CU	36	
37	K	37	U	37	
38	H	38	T	38	
39	F	39	E	39	
40	S	40	Q	40	
41	6	41	DEL AND INST	41	
42	4	42	P	42	
43	RET	43	I	43	
44	L	44	\	44	COMMA AND <
45	J	45	Y	45	- AND =
46	G	46	R	46	. AND >
47	D	47	W	47	/ AND ?
48	A	48	TAB	48	
49	/	49	6	49	1 AND !
50	8	50	@	50	2 AND QUOTE
51		51	L	51	3 AND #
52	P	52	RET	52	4 AND $
53	I	53	J	53	5 AND %
54	Y	54	G	54	6 AND &

KEY#	PET-GR-ORIG/UPGR	KEY#	CBM-40COL B4	KEY#	CBM-80COL B4
55	R	55	D	55	7 AND '
56	W	56	A	56	8 AND (
57	9	57	5	57	9 AND)
58	7	58	SEMIC AND +	58	COLON AND *
59	↑ AND π	59	K	59	SEMIC AND +
60	O	60	J	60	
61	U	61	H	61	
62	T	62	F	62	
63	E	63	S	63	
64	Q	64	ESC	64	
65	DEL AND INS	65	9	65	A
66	CD AND CU	66		66	B
67		67	↑	67	C
68	)	68	7	68	D
69	\	69	0 IN TOP ROW	69	E
70	'	70	7 AND '	70	F
71	$	71	4 AND $	71	G
72	QUOTE	72	1 AND !	72	H
73	CR AND CL	73		73	I
74	CLS AND HOME	74		74	J
75	←	75	CR AND CL	75	K
76	(	76	8	76	L
77	&	77	- AND =	77	M
78	%	78	8 AND (	78	N
79	#	79	5 AND %	79	O
80	!	80	2 AND QUOTE	80	P
81		81		81	Q
82		82		82	R
83		83		83	S
84		84		84	T
85		85		85	U
86		86		86	V
87		87		87	W
88		88		88	X
89		89		89	Y
90		90		90	Z
155		155		155	ESC
174		174		174	.
176		176		176	0
177		177		177	1
178		178		178	2
179		179		179	3
180		180		180	4
181		181		181	5
182		182		182	6
183		183		183	7
184		184		184	8
185		185		185	9
192		192		192	@
219		219		219	[
220		220		220	\
221		221		221	]
222		222		222	↑
223		223		223	←

Using USR
by David Hsieh

One of the least used and most versatile commands in PET Basic is the USR(command. The big problem is that the User's manual makes it very hard to understand exactly what it does and how to use it. USR can be thought of as user-defined command with one parameter. Similiar to a user defined function, it is much more flexible because it can access a complete machine language subroutine and is not limited to math functions.

So what do you use it for? This article presents two useful applications: TABLER and ADRENALIN. TABLER allows the user to translate simple tables quickly (machine language and cleanly. Other than initializing the table and subroutine, there are no confusing pokes or peeks. The same is true of ADRENALIN, which helps to speed up program execution of self-contained subroutines.

The USR command must be initialized by pointing to the location of the machine language subroutine. This is done by POKEing 1, low address (58 for the 2nd cassette buffer) and POKE 2, high address (3 for the 2nd cassette buffer). When the USR command is executed, the PET does a machine language jump to the location pointed to and continues executing in MACHINE LANGUAGE (not BASIC), until it hits a return (RTS instruction). The argument in the parentheses is passed to the FLOATING POINT ACCUMULATOR (FAC), where it can be manipulated in floating point form or converted to a binary integer. Both TABLER and ADRENALIN use the FLPINT ROM subroutine to convert the FAC (at $5E-$63) to an integer (at $61-$62), and then back to floating point again using INTFLP subroutine. FLPINT Is located at $D09A and INTFLP is at $D26D. While many math applications for USR would use a floating point parameter (like a base 10 logarithm routine), for non mathematical subroutines it is easier to use integer parameters. FLPINT performs this function. Since the USR command returns a floating point value back to BASIC, subroutines that use integers must convert using INTFLP as well.

About TABLER:
TABLER is a very handy subroutine. Many of the programs I write use a GET statement, instead of INPUT, to enter choices (in a menu, for example). The problem is often how to decode the choice into a number code without using 15 IF/THEN statements. The solution is to use machine language to translate a table of characters into a sequence number, and use an ON/GOTO statement. Listing 1 is the BASIC loader program and listingH2 is the machine language source program.

Using the subroutine is very easy. Once the routine has been pointed to, the table itself is set up and POKEd into memory. An ASCII value is used as the argument for USR, and it is translated into a number representing where it appeared in the translation table. The first entry will return a value of one, the second a two, etc. If you try to translate an argument not in the table, USR returns a value of 32767.

The primary limitation of TABLER is that it can only translate numbers less than 256 and only 255 table entries are allowed. Other than that, it is a very speedy (compared with multiple IF/THENs) utility.

About ADRENALIN:
Speed on microcomputers has always been a problem. There are many hints that can be used to help speed up program execution, especially in Microsoft BASIC. Some of these, such as using variables as often as possible, and ordering them according to usage, are easy to implement and not much of an inconvenience.

One such trick I find particularly bothersome is putting subroutines at the beginning of long programs. This speeds up execution because the interpreter has to search for the correct line number every time a subroutine is called, and by putting out used subroutines at the start of programs (instead of the end or middle) when there are less line numbers to search through. This sounds fine, but when you want to list the main program, you have to pass through all the subroutines first. Since you usually debug the main routine more often than each subroutine (I think), this can be quite inconvenient and confuses the program listings as well.

The ADRENALIN program (listing 3 and 4) significantly speeds up programs and allows you to write your subroutines anywhere you want. Its use is very simple. The program is called by the USR function. A zero between the parentheses signals the start of a subroutine, and any non zero value signals the end. Don't forget to POKE 1,58:POKE 2,3 at the beginning of the program.

How Does It Work?
The ADRENALIN routine intercepts subroutines just as they are being called and fools the PET into thinking the start of a subroutine is actually the start of program memory. This means that whenever a subroutine has a jump (GOTO or GOSUB), the line number search starts at the beginning of the subroutine, instead of at the beginning of the program. Comparative benchmark results are better for longer programs, but as an example, I benchmarked a simple bubble sort with a four K "filler" program tacked on in front. Adrenalin averaged 49.5 seconds faster than the unmodified program!

Limitations
ADRENALIN must be used carefully. It has several severe limitations, the foremost of which is that jumps must not go before the subroutine start when ADRENALIN is in effect (the interpreter will not be able to find the line number). This includes the RETURN statement. ADRENALIN must be turned off BEFORE exiting ADRENALIN. The other limitation is that the USR command used to turn ADRENALIN on must be at the end of the line. If it is not, the program will crash after the first jump. The off command can be anywhere within a line.

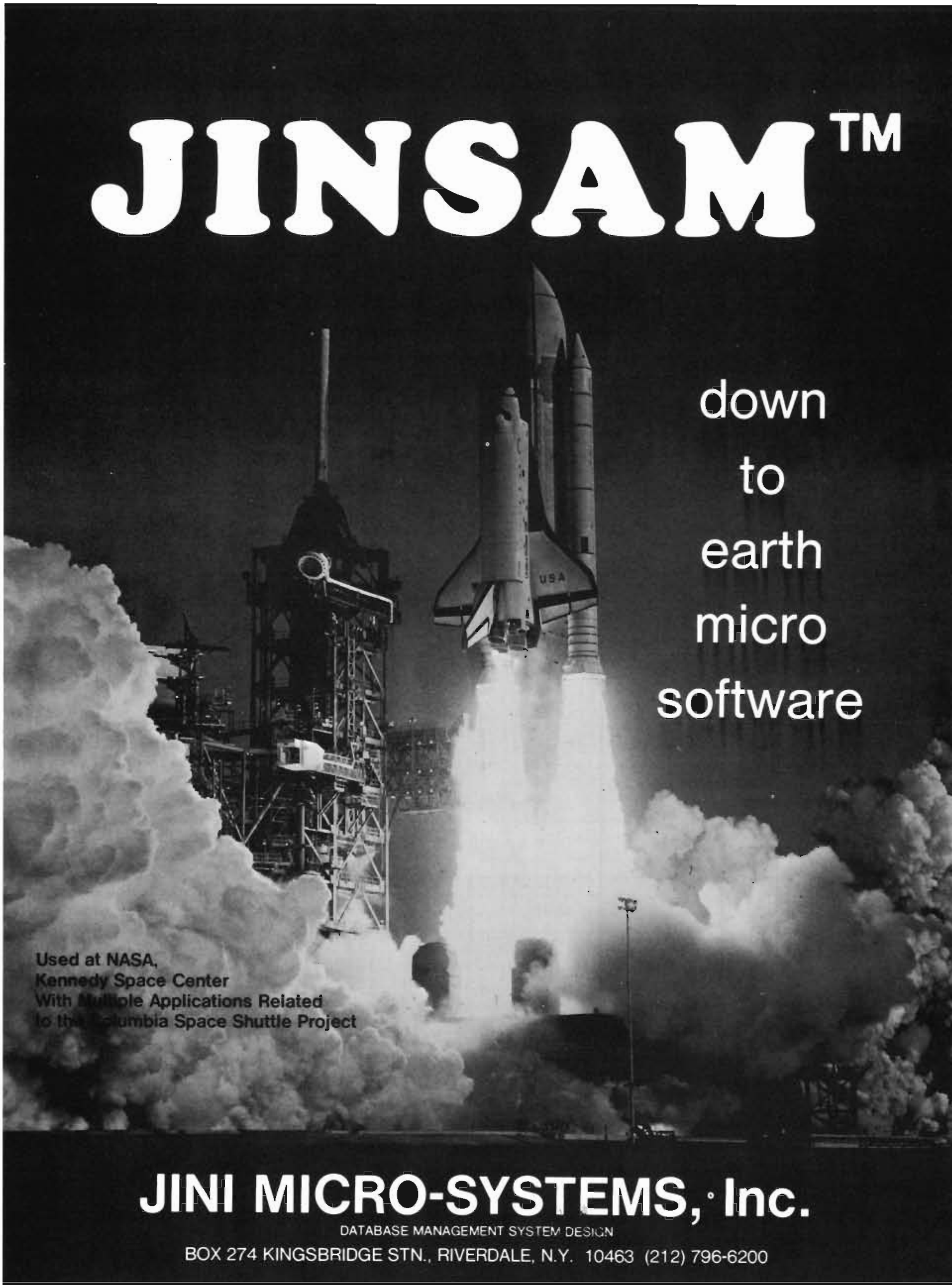
Conclusion:
USR, while being one of the most versatile commands in PETBasic, is also one of the least used. Hopefully this article helps to clear up some of the mysteries regarding USR, and also provides some useful application subroutines.

```
;***************
;
;ADRENALIN
;
;    VERSION 1.0
;
;
;***************
;
;
*=$33A
;
;
;ROM ROUTINES
;***************
;
INTFLP=$D26D
FLPINT=$D09A
BASTRT=$28
TXTPTR=$77
BUFFER=$027A
;
;TEXT   STARTS HERE
 JSR FLPINT ;CONVERT TO INTEGER
 CLC
 LDA $61
 ADC $62
 BNE RESET
SET LDA BASTRT
 STA BUFFER
 LDA BASTRT+1
 STA BUFFER+1
 LDA TXTPTR
 STA BASTRT
 LDA TXTPTR+1
 STA BASTRT+1
 RTS
RESET LDA BUFFER
 STA BASTRT
 LDA BUFFER+1
 STA BASTRT+1
 RTS
 .END
```

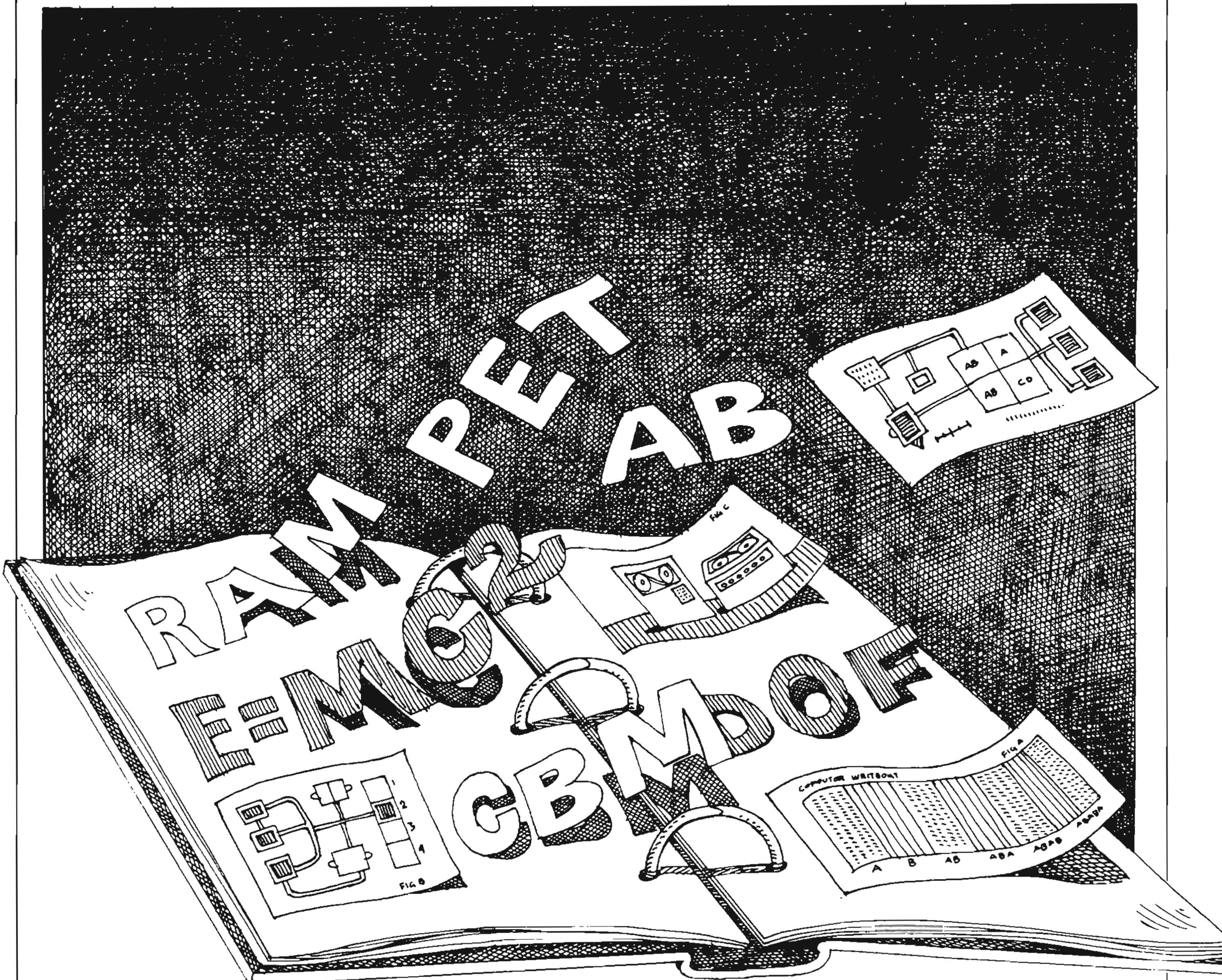
```
;***************
;
;TABLE CONVERTER
;
;    VERSION 1.0
;
;
;***************
;
;
*=$33A
;
;
;ROM ROUTINES
;***************
;
INTFLP=$D26D
FLPINT=$D09A
;
;TEXT   STARTS HERE
 JSR FLPINT ;CONVERT TO INTEGER
 LDX $027A ;TABLE LENGTH
 LDA $62 ;LOBYTE ONLY
LOOP1 CMP $027B,X ;COMPARE LOOP
 BEQ END1 ;FOUND
 DEX
 BNE LOOP1 ;LOOP OVER?
 LDA #$FF ;NOT FOUND
 STA $61
 LDA #$7F
END1 STA $62
END JMP INTFLP
 .END
```

```
 1000 FORI=826TO858:READA:POKEI,A:NEXT:
      POKE1,58:POKE2,3:RETURN
 1010 DATA 32,154,208,24,165,97,101
 1020 DATA 98,208,11,165,40,141,122
 1030 DATA 2,165,41,141,123,2,96
 1040 DATA 173,122,2,133,40,173,123
 1050 DATA 2,133,41,96,245
READY.
```

```
 1000 POKE1,58:POKE2,3:FORI=826TO870:
      READII:POKEI,II:NEXTI:RETURN
 5010 DATA 32,154,208,24,165,97,101
 5020 DATA 98,208,23,165,40,141,122
 5030 DATA 2,165,41,141,123,2,165
 5040 DATA 119,105,1,133,40,165,120
 5050 DATA 105,0,133,41,96,173,122
 5060 DATA 2,133,40,173,123,2,133
 5070 DATA 41,96,167
READY. ■
```



JINSAM™
down
to
earth
micro
software
Used at NASA,
Kennedy Space Center
With Multiple Applications Related
to the Columbia Space Shuttle Project
JINI MICRO-SYSTEMS, Inc.
DATABASE MANAGEMENT SYSTEM DESIGN
BOX 274 KINGSBRIDGE STN., RIVERDALE, N.Y. 10463 (212) 796-6200



Excerpts from a Technical Notebook

BASIC Modem Driver for the CBM 8010

Based on research and articles for 'The Transactor'

The manual supplied with the 8010 modem is fairly good, but I found that I still had to read between the lines to get it right. My objective was to be able to have two PETs connected by modem, talking freely with each other. This would require a program in each PET which would send characters keyed in to the modem and echo them to the screen, while displaying any received characters in reverse video to distinguish them from the operator's text.

The first program that I wrote was based on what I had read in the manual, that if 'ST' (the system STatus variable) is zero, a character is ready to be read from the modem; so my first program looked like this:

```
100 open5,5 :rem open a file to the modem
110 get#5,a$:if st<>0 then print "(rvs)"a$"(offrvs)";
    :rem read modem
120 geta$ :if a$<>"" then print#5,a$; :printa$; :ram send
    key & echo
130 goto 110
```

This program was totally unacceptable, because it dropped 50% of all characters transmitted to it. However, I had read about the SRQ method of using the modem. The SRQ (Service ReQuest) line is one of the main reasons who the PET IEEE is not a full IEEE-488 bus. The SRQ line is fully implemented in the PET hardware, but is not used in software! What SRQ does is to allow an IEEE peripheral to tell a controller (the PET) that it needs servicing. The SRQ is attached to bit 7 of address 59427. So, my next program:

```
100 open5,5
110 if (peek(59427) and 128)=0 goto140 :rem if no SRQ,
    read keyboard
120 get#5,a$ :print''(rvs)''a$''(offrvs)''; :rem read modem
    & print char
140 geta$ :ifa$<>'''' then print#5,a$; :printa$; :rem send
    char & echo
150 goto 110
```

In theory then, this program should GET incoming characters and print them to the screen as soon as they are received. Well don't bother trying this program, because it won't work. The snag here is that SRQ does not get cleared when we GET the character, so the program will continue to GET characters, even if one isn't there. Curiously, SRQ is reset by reading the IEEE Output Register, which is at address 59426. So if we add this line to the program above:

```
130 x=peek(59426) :rem dummy read of IEEE Output
    Register resets SRQ
```

it will work now. It WILL, honestly. Once I had discovered how easy it was to communicate via the 8010, I decided to add a little style. So, the next program evolved a short while later. The next program uses the 'window' facility of the CBM 8032, so it will not work on 40-column PETs.

A Dual Window Modem Communicator for the CBM 8032/8010

by Paul Higginbottom, CBM Canada

The upper window displays characters typed locally and the bottom window shows remote activity. I couldn't get the modem to drop a single character, no matter how fast I typed, when using this program. Some characters appear within parentheses () inside quotes in the program. They are: (hm) = 'home cursor' (clr) = 'clear screen' (dn) —'cursor down'.

```
1000 open 5,5:print'' (hm hm clr)'' :rem open file to modem
     :clear window
1010 print''(clr)'' :rem clear the screen
1020 for i=0 to 79:poke 33728+i,64:next :rem draws
     window separator line
1030 cd$='' (dn dn dn dn dn dn dn dn dn dn dn dn)'' :rem 12
     cursor downs
1040 xl=0:yl=0:x2=0:y2=0 : rem setup screen
     coordinates
1050 si=59427:ms=128:io=59426:ze=0 :rem SRQ in
     addr & IEEE out reg addr
1060 tl=224:ml=13:cp=198:cl=216bl=225:hf=232
1070 ll=80:sl=32768:sz=33808:us=127:bp=11
1080 poke sl,160:poke s2,160:es$=chr$(27)
2000 if not peek(si) and ms then 3000 :rem if no SRQ go
     read keyboard
2010 get#5,a$:poke io,ze:print''(hm hm)'';:poke
     hf,ze:poke tl,ml
2020 p=s2+x2+y2*11:poke p,peek(p) and us
2030 print''(hm)''left$(cd$,y2)spc(x2):poke hf,ze:print a$
     es$;
2040 x2=peek(cp):y2=peek(cl)-ml
2050 p=s2+x2+y2*11:poke p,peek(p) or ms
3000 geta$:if a$='''' then 2000 :rem if no key pressed go
     check modem
3010 print''(hm hm)'';:poke bl,bp:poke hf,ze
3020 p=sl+xl+yl*11:poke p,peek(p) and us
3030 print''[hm]''left$(cd$,yl)spc(xl):poke hf,ze:print a$
     es$;
3040 xl=peek(cp):ylApeek(cl)
3050 p=sl+xl+yl*11:poke p,peek(p) or ms
3060 print#5,a$; :goto 2000
```

NOTES ON THE VARIOUS DOS VERSIONS

I. Overview of DOS Versions

DOS 1.0 is an 8k DOS which works exclusively with 2040/3040 dual disk units. The directory provides for 152 file entries and 670 blocks (171.5k bytes/disk) are available for user data. DOS 1.0 supports Sequential files and Block Read/Write commands allowing direct disk access.

A BASIC 4.0 program will work with DOS 1.0 provided that the extended features of BASIC 4.0 are not used for disk access; however, it is not recommended that DOS 1.0 and BASIC 4.0 be used together.

DOS 2.1 is a 12k DOS which works with the 4040 dual disk unit. Model 2040/3040 disk units can be upgraded to DOS 2.1 by replacement of ROM chips. Reliability of the recording format of DOS 2.1 was improved by removing one block from tracks 18 thru 24. As a result the directory holds 144 (8 fewer than DOS 1.0) file entries and 664 (6 fewer than DOS 1.0) blocks, or 170k bytes/disk, are available for user data.

The Relative Record file structure was added to DOS 2.1 to provide for random access to files. The Block Read/Write commands of DOS 1.0 are not supported, instead the corresponding 'U1' and 'U2' utility commands must be used for upward compatibility with future CBM disk products.

In general, software which does not depend upon physical device attributes should be upward compatible for all versions of DOS. Programs using the Block Read/Write commands are very vulnerable to DOS changes.

DOS 2.5 is a 16k DOS used in all 8050 dual disk units. All of the features of DOS 2.1 are included in DOS 2.5, adapted for additional capacity. DOS 2.5 also includes enhancements such as disk insertion detect, expanded error recovery techniques and a utility loader for self-testing and extended utility functions. The directory provides 224 file entries and 2052 blocks (525.3k bytes/disk) are available for user data.

II. 4040/8050 Power-up Diagnostics

DOS 2.1 and DOS 2.5 perform self-tests on all RAM and ROM within the disk unit. The RAM test consists of an address-sensitive read/write test. The ROMs are check-summed to a single-byte check sum which is equal to the high-byte of the address of each ROM.

If a failure is detected, the DOS will blink all of the LEDs on the disk unit a specific number of times to indicate the problem. If all the LEDs do not blink together than the pattern may not indicate the actual problem. The flashing code is repeated continously with a one second pause between repititions.

Diagnostic Messages—4040/8050 LED's

# Flashes	Resource	Component	Location
1	Zero Page	6532	E1, C1
2	ROM $e000-$ffff	6332	L1
3	ROM $c000-$dfff	6332	H1
4	Not Used		
5	Controller ROM, Zero Page	6530, 6502	K3, H3
6	Common RAM $1000-$13ff	2114	C4, C5
7	Common RAM $2000-$23ff	2114	D4, D5
8	Common RAM $3000-$33ff	2114	E4, E5
9	Common RAM $4000-$43ff	2114	F4, F5
10	Controller ROM	6530, 6502	K3, H3

III. BAM (Block Allocation Map) Formats

A. 2040/3040 BAM Format—Track 18 Sector 00

Byte	Data	Definition
0-1	18-00	Tract-Sector of first directory block
2	01	Identifies DOS 1.0 format
3	00	Reserved for future DOS use
4-143		Bit map of available blocks, tracks 1-35

B. 4040 BAM Format—Track 18 Sector 00

Byte	Data	Definition
0-1	18-00	Tract-Sector of first directory block
2	65	ASCII 'a' identifies DOS 2.1 format
3	00	Reserved for future DOS use
4-143		Bit map of available blocks, tracks 1-35

C. 8050 BAM Format—Track 18 Sector 00

Byte	Data	Definition
0.1	38-03	Track-Sector of second BAM block
2	67	ASCII 'c' identifies DOS 2.5 format
3	00	Reserved for future DOS use
4	01	Lowest track # mapped in this BAM block
5	51	Highest track #(+1) mapped in this BAM block
6		Nr. of unused blocks on track #1
7-10		Bit map of available blocks on track # 1
11-255		Bit map of available blocks, tracks 2-50

D. 8050—Second BAM Block Format—Track 38 Sector 03

Byte	Data	Definition
0-1	39-01	Track-Sector of first directory block
2	67	ASCII 'c' identifies DOS 2.5 format
3	00	Reserved for future DOS use
4	51	Lowest track # mapped in 2nd BAM block
5	78	Highest track # (+1) mapped in 2nd BAM block
6		Nr. of blocks unused on track # 51
7-10		Bit map of available blocks on track # 51
11-140		Bit map of available blocks, tracks 52-77

E. Structure of BAM Entries for one Track—All DOS Versions
Each track has five bytes allocated to map it. A map bit=1 means the block is available; bit=0 means the block has been used. Blocks are mapped by bytes, the high order bit of each mapping the lowest block.

Byte	Definition
1	Number of available blocks for this track
2	Bit Map blocks 0-7. Bit 7= block 0, bit 0 = block 7
3	Bit map blocks 8-15. Bit 7=block 8, bit 0=block 15
4	Bit map blocks 16-23. Bit 7=block 16, bit 0=block 23
5	Bit map blocks 24-31. Bit 7=block 24, bit 0=block 31

IV. Directory Header and Directory Block Formats

A. 2040/3040 Directory Header—Track 18 Sector 00

Byte	Data	Definition
144-161		Diskette name, padded with shifted spaces
162-163		Diskette ID Nr.
164-165	32	Spaces
166-170	160	Shifted spaces
171-255	00	Not used

Note: ASCII data may appear in bytes 180-191 on some diskettes.

B. 2040/3030 Directory Blocks—Track 18 Sectors 01 thru 19

Byte	Data	Definition
0-1		Track-Sector pointer to next directory block
2		File Type
3-4		Track-sector pointer to first file block
5-20		File name, padded with shifted spaces
21-27		Reserved for future file info
28-29		Track-sector link for replacement
30-31		Number of blocks used by the file
32-255		Seven more 32-byte entries (same as 2-31 above, plus two additional unused bytes)

C. 4040 Directory Header—Track 18 Sector 00

Byte	Data	Definition
144-161		Diskette name, padded with shifted spaces
162-163		Diskette ID Nr.
164	160	Shifted space
165-166	50, 65	ASCII '2a' identifies DOS version & format
167-170	160	Shifted spaces
171-255	00	Not used

Note: ASCII data may appear in bytes 180-191 on some diskettes.

D. 4040 Directory Blocks—Track 18 Sectors 01 thru 18

Byte	Data	Definition
0-1		Track-Sector pointer to next directory block
2		File type
3-4		Track-Sector pointer to first file block
5-20		File name, padded with shifted spaces
21-22		Track-Sector of 1st side sector if Relative file
23		Record length if Relative file
24-27		Reserved for future file info
28-29		Track-Sector pointer for replacement
30-31		Number of blocks used by the file
32-255		Seven more 32-byte file entries (same as 2-31 above, plus two additional unused bytes)

E. 8050 Directory Header—Track 39 Sector 00

Byte	Data	Definition
0-1	38-00	Track-Sector pointer to first BAM block

2	67	ASCII 'c' identifies DOS 2.5 format
3	00	Reserved for future DOS use
4-5		Not used
6-21		Diskette name, padded with shifted spaces
22-23	160	Shifted spaces
24-25		Diskette ID Nr.
26	160	Shifted space
27-28	50,67	ASCII '2c' identifies DOS version & format
29-32	160	Shifted spaces
33-255	00	Not used

F. 8050 Directory Blocks—Track 39 Sectors 01 thru 29

Byte	Data	Definition
0-1		Track-Sector pointer to next directory block
2		File type
3-4		Track-Sector pointer to first file block
5-20		File name, padded with shifted spaces
21-22		Track-Sector of 1st side sector if Relative file
23		Record length if Relative file
24-27		Reserved for future file info
28-29		Track-Sector pointer for replacement
30-31		Number of blocks used by the file
32-255		Seven more 32-byte file entries (same as 2-31 above, plus two additional unused bytes)

G. Notes to Directory Block formats—all DOS versions:

1. 32 bytes per file entry, except the first entry is 30 bytes
2. Total of eight (8) file entries per directory block
3. File Types are:

Scratched files	$00	
Sequential data	$01	
Program files	$02	
User-defined	$03	
Relative Record	$04	(DOS 2.1 & up only)

4. File Type codes are OR'ed with $80 when file is properly closed
5. Track value of 00 in byte zero indicates the last used block in the directory. Sector value then shows next byte to use.

V. Disk File Formats—All DOS Versions

A. Program Files

Byte	Definition
0-1	Track-Sector pointer to next program block
2-255	Up to 254 bytes of BASIC program text. End-of-file is marked by three consecutives bytes of $00.

B. Sequential and Relative Record Data

Byte	Definition
0-1	Track-Sector pointer to next sequential data block
2-255	Up to 254 bytes of data with carriage returns as terminators between data items.

Notes: Track link of $00 in byte zero indicates last data block. Sector link is then next byte position to receive data. End of Relative Record data indicated by reading $ff.

C. Relative File Side Sector Format

Byte	Definition
0-1	Track-Sector pointer to next side sector
2	Side sector number
3	Relative Record length
4-5	Track-Sector pointer—1st side sector
6-7	Track-Sector pointer—2nd side sector
8-9	Track-Sector pointer—3rd side sector
10-11	Track-Sector pointer—4th side sector
12-13	Track-Sector pointer—5th side sector
14-15	Track-Sector pointer—6th side sector
16-255	Track-Sector pointer to 120 data blocks Total of 720 blocks (182.8k bytes of Relative data per file.)

Overview of CBM 64k Memory Expansion Board

The expansion board adds 64k-bytes of RAM to the CBM 8032, giving a total of 96k-bytes of RAM. The add-on memory is mapped into the 6502's main memory space in 16k-byte blocks, two at a time. A write-only control register at hex address $fff0 provides for expansion board enable/disable, bank-selection of 16k-byte blocks, write-protect of expansion RAM and a 'peek-thru' feature for I/O and screen memory.

A diskette containing programs for controlling the expansion memory is supplied with the board. These include: an extended memory manager, a loader which prompts the user and then loads one of three versions of CBM BASIC into add-on memory, a functional equivalent of TIM which provides access to add-on memory and a program which extends 4.0 BASIC to provide commands for use of add-on memory from BASIC programs. Also included are a set of short BASIC programs which demonstrate use of the BASIC command extensions.

The expansion board is disabled at power-up and the system is in all respects an 8032. When the memory manager routines are loaded and started, access to the add-on RAM is enabled. The add-on RAM is configured as two 16k-byte blocks at hex $8000-$bfff and $c000-$ffff plus two alternate blocks. Either or both of the currently selected blocks can be write-protected via the control register.

Since the add-on memory is addressed contiguously from $8000-$ffff a means of accessing screen RAM ($8000-$8fff) and I/O memory ($e800-$efff) is required. By storing an appropriate bit pattern into the control register, memory accesses in these address ranges can be directed to either add-on memory or to the screen or I/O memory. Either of the active 16k-byte blocks may be write-protected separately and 'peek-thru' for screen or I/O set separately.

The BASIC command extensions reside at hex $7800, leaving 29k-bytes of contiguous BASIC program space in low memory. Commands are provided to allow loading additional program/data files into add-on memory and for 'downloading' programs into low memory for execution. Thus a BASIC menu program and several frequently-used modules of an application could be loaded to add-on RAM for fast program chaining without re-loading from disk each time. Also, assembler language applications may access both low memory and add-on RAM as needed, the only problem being that the programmer must know when bank selection is required. ■

Compiled by Dave Middleton

So You Want To Program in Machine Language...

As we have discussed in previous articles, it is sometimes more efficient and in some cases, almost imperative, to write a program in Machine Language. How can we do this? Well, one way, and probably the best way to get started, is to buy an Assembler Development System. But why do it the easy way when everything we need is right in front of us. Machine Language is eventually what the Assembler generates so let's first take a look at that. In this article I will provide you with the tools needed to get started, some theory behind how the PET/CBM memory operates, and finally a sample program.

The tools needed are:

1. Instructions for the Machine Language Monitor
2. A Memory Map for the system you are working on
3. A Storage Device (either disk or tape)
4. A copy of the 6502 Microprocessor Instruction Set
5. MOS Hardware Manual (optional)
6. MOS Programming manual (optional)
7. A system with a Machine Language Monitor

I have provided, at the end of this article, a copy of the Machine Language Monitor Instructions, a BASIC 4.0 Memory Map, and the 6502 Instruction Set. I will assume that you have the necessary storage medium and a PET/CBM with a resident Monitor. For those of you with a system without a Monitor, you should get a hold of one (available from your dealer).

Memory

Memory, quite simply, is a series of consecutive storage areas, numbered, in our case, from 0 (low memory) to 65536 (high memory). One storage area can usually hold one alphanumeric character. This memory is of two types: RAM or Random Access Memory, and ROM or Read Only Memory. RAM is memory which can be updated (you can store or change information in this type of storage area), whereas ROM is memory which can only be looked at (you cannot change or store information here; it is usually already precoded with information, as in the PET's case). On the PET/CBM, RAM is the amount of storage space that you have to store a program, variables, etc., although 1024 storage areas are used by the system to store information. This can be changed by you, but BEWARE, if you do, you might be in for trouble. So RULE 1: Don't mess with this area.

8K, 16K, 32K, what does it all mean? When you bought your machine, you were told it had so much "K" of memory. "K" is equal to 1024 storage locations or "bytes" (which is computer terminology for one storage location). So to figure out how much memory you have to store a program in, subtract 1K (1024 bytes from above), and that's what is left for you. So for example, on an 8032, which has 32K of memory, you would have only 31K of memory in which to program (31744 bytes). This is validated when you power on your mach ine and the system displays "31743 bytes free." I know, there is one less than what you calculated, right? True, because 1 byte is used to display the message.

There is however, some extra RAM that is included in each system. This is Video RAM, or Screen RAM, and is 2K (2048) bytes. On 40 column machines, 1000 bytes (40 columns x 25 rows) are used for display purposes, while on 80 column machines 2000 bytes (80 columns x 25 rows) are used. So when you turn on your machine, you are looking at Screen RAM. If you store something at a Screen RAM location, you will see it appear on the screen (we will see an example of this later).

What is stored in RAM and ROM? All the necessary programming relating to running BASIC, Input/Output routines (printing to the screen, writing to cassette and disk, etc.), and any other functions necessary to run a program on your PET/CBM, are stored in ROM. This is hardwired (or burned in) in each PET/CBM, and is the same every time you power up your machine. RAM is empty basically on power up and is used for storing and writing programs. The exceptions are Screen RAM, which contains whatever you type on the screen, and the first 1024 bytes (memory locations 0-1024), which contain information used by the Operating System (ROM). Why not put this information in ROM? The main reason is because the programs stored in ROM need a place to write out variables and store information, and since this can only be done in RAM (we are able to update and change it), this area was set aside for this purpose when the Operating System was developed.

POKE and PEEK

There are a number of different ways to access and update memory, two of which you may already be familiar with, the PEEK and POKE BASIC statements. The PEEK statement (Format: PEEK(address)) allows you to access the particular memory location specified by "address," and when executed, will return the decimal value that is stored at that location. The POKE statement (Format: POKE address, value) allows you to update the RAM memory location specified by "address," and when executed, will place the "value" specified at that memory location. As you can see, the PEEK function can be used to look at any memory location, whereas the POKE function can only be executed on a RAM memory location. As I discussed before, locations 0 through 1024 contain information used by the Operating System. It is possible to change the values of certain of these locations (refer to the Memory Map) to do certain things. We will do this to illustrate the function of the POKE and PEEK statements. Let's try some examples.

Example 1

Location 231 (hex 00E7) stores the timer for the bell on 80-column machines. Try typing:

?peek(231)

If you havn't already changed it, the value should be "16." To make the bell ring type:

?chr$(7)

The bell should ring. Let's change the timer now. To do this we must update that particular storage location. We do this with the POKE statement as follows:

poke 231,100

Now try to ring the bell using the second statement above. The bell should sound a little different. Continue to experiment using any value between 0 and 255 (the highest decimal number that can be stored in a byte). The lower the number, the faster the bell.

Example 2

Location 216 (hex 00D8) contains the row number where the cursor is presently located. Let's clear the screen, change the number, and see where the cursor ends up.

?"Sabc":poke216,15

The "S" refers to the shifted "CLR/HOME" key. Upon execution of this BASIC statement, the screen should clear, the letters "abc" should be printed at the top, the "ready" should end up somewhere near the bottom of the screen.

Example 3

Locations 32768 (hex 8000) to 34815 (hex 87FF) are the storage locations for screen images on the 80-column machine. Location 32768 being the first character position in the top, lefthand side of the screen. Let's put something there.

First, clear you screen, then cursor down to about the middle of the screen. Now type: poke 32768,65. Decimal 65 is really the representation for the letter "a." Another way would be to type: poke 32768, asc("a"). The ASC command here converts "a" to its decimal equivalent.

Now let's print something in column 79 of row one. The address then, is 32768 (row 1, column 1) plus 78 (row 1, column 79). Type: Poke 32768+78,asc("a"). This should make the letter "a" print at row 1, column 79.

As you can see, you can achieve many things in BASIC through these two statements, which may have been harder to do in BASIC itself. Please refer to the Memory Map when trying anything, but just remember that some locations were not meant to be changed (only experimentation will tell you which ones).

The SYS Command

This command (Format: SYS address), when executed, will actually transfer control to the specified address and will execute the machine language instructions that start at this location and will not return to BASIC unless the appropriate machine language "RETURN" instruction is executed. This command is useful if you have some machine language code, for instance, a cursor positioning routine, that you may frequently use throughout your BASIC program. We will see a little more about the operation of the SYS command in the following example explaining the Monitor.

The Machine Language Monitor

The Monitor is basically a program which allows you to look at any PET/CMB memory location or to modify any RAM location. The catch is that everything you look at is in Hexadecimal (base 16). The way the Monitor is entered is via the SYS command. The resident Monitor (in newer machines) stares running whenever you execute a SYS command from immediate mode and the address that is SYSed to contain a Hex "0." Since memory location 4 contains a "0" on power up, we can type:

sys 4

to jump into the Monitor. What should be displayed now are the REGISTERS. We will talk more about them in a future article. It should look something like this:

```
b*
pc   irq  sr ac xr yr sp
.; 0005 e455 30 00 5e 04 f8
```

You are now in the Monitor. To illustrate some of the Monitor commands, we will key in and store a small machine language program that prints the alphanumeric character set. This time we will key in this program to get a feel for how the Monitor works. In the next issue we will explain the program and talk more about machine language.

Before starting through, we first need a place to store our program. By looking at the Memory Map, you will see that BASIC storage (RAM) starts at memory location 1024 (hex 0400). This seems like as good a place as any to store our program, but to be on the safe side let's start at location 1032 (hex 0408).

First let's display a portion of memory, starting at 1032, that is big enough to store our program.

```
m 0408-0418
.: 0408 aa aa aa aa aa aa aa aa
.: 0410 aa aa aa aa aa aa aa aa
.: 0418 aa aa aa aa aa aa aa aa
```

You should now have this portion of memory displayed. You will notice that again, all numbers are in hex, and that the first four digits indicate the hex memory address of the first byte of the line—there are 8 bytes of memory addresses per line.

To enter information into the memory addresses, simply cursor up to the top line and start to type over the current information as shown below. Pressing <RETURN> on any line updates memory as displayed on that line. Be sure to follow the syntax as is displayed, for example, 2 character hex number, "space," next 2 character hex numbers, etc.

```
.: 0408 a2 0d 20 d2 ff a2 20 8a
.: 0410 20 d2 ff e8 e0 60 d0 f7
.: 0418 00 4c 0f 04 <RETURN>
```

You have now entered a machine language program. Of course you want to save it, so type:

```
.s "filename",08,0408,0420
```

"08" specifies the disk drive, whereas "01" would mean tape #1. "0408" specifies the beginning of the program and "0420" specifies the end. We use this number in order to include everything that appears on the line that starts with address "0418"

To load this program back again, you can either load it directly from BASIC, ie, "DLoad", or you can enter the Monitor and type:

```
.l "filename",08
```

Now, just to prove that you have really tuped in a machine language program, let's run it. Type:

```
.g 0408
```

This tells the PET/CBM to go to address 0408 and begin executing the instruction that is stored at that location. You should get the character set printed.

Now, let's exit the Monitor by typing "X". This will return us to "ready." Even though we have exited, our program still resides in memory. We can execute it again by typing SYS 1032 (hex 0408). This will transfer control to memory location 1032 and begin executing the machine code starting there. You will notice however, that when the program is finished, it returns to the Monitor. This is because the final instruction told the system to do so. We will discuss how to change this next time.

In the next issue, we will dissect this program and find out what it's all about. We will also try some more examples, and learn a little more about machine code. Any comments, ideas, or questions can be sent to me in care of the Editor.

—Dave Scott

PET 4.0 ROM Routines

Compiled by Jim Butterfield

There are some differences between usage between the 40- and 80-column machines.

Hex	Decimal	Description
0000-0002	0-2	USR jump
0003	3	Search character
0004	4	Scan-between-quotes flag
0005	5	Input buffer pointer; # of subscripts
0006	6	Default DIM flag
0007	7	Type: FF=string, 00=numeric
0008	8	Type: 80=integer, 00=floating point
0009	9	Flag: DATA scan; LIST quote; memory
000A	10	Subscript flag; FNX flag
000B	11	0=INPUT; $40=GET; $98=READ
000C	12	ATN sign/Comparison Evaluation flag
000D-000F	13-15	*Disk status DS$ descriptor
0010	16	*Current I/O device for prompt-suppress
0011-0012	17-18	Integer value (for SYS, GOTO etc)

0013-0015	19-21	Pointers for descriptor stack
0016-001E	22-30	Descriptor stack(temp strings)
001F-0022	31-34	Utility pointer area
0023-0027	35-39	Product area for multiplication
0028-0029	40-41	Pointer: Start-of-Basic
002A-002B	42-43	Pointer: Start-of-Variables
002C-002D	44-45	Pointer: Start-of-Arrays
002E-002F	46-47	Pointer: End-of-Arrays
0030-0031	48-49	Pointer: String-storage(moving down)
0032-0033	50-51	Utility string pointer
0034-0035	52-53	Pointer: Limit-of-memory
0036-0037	54-55	Current Basic line number
0038-0039	56-57	Previous Basic line number
003A-003B	58-59	Pointer: Basic statement for CONT
003C-003D	60-61	Current DATA line number
003E-003F	62-63	Current DATA address
0040-0041	64-65	Input vector
0042-0043	66-67	Current variable name
0044-0045	68-69	Current variable address
0046-0047	70-71	Variable pointer for FOR/NEXT
0048-0049	72-73	Y-save; op-save; Basic pointer save
004A	74	Comparison symbol accumulator
004B-0050	75-80	Misc work area, pointers, etc
0051-0053	81-83	Jump vector for functions
0054-005D	84-93	Misc numeric work area
005E	94	Accum#1: Exponent
005F-0062	95-98	Accum#1: Mantissa
0063	99	Accum#1: Sign
0064	100	Series evaluation constant pointer
0065	101	Accum#1 hi-order (overflow)
0066-006B	102-107	Accum#2: Exponent, etc.
006C	108	Sign comparison, Acc#1 vs #2
006D	106	Accum#1 lo-order (rounding)
006E-006F	110-111	Cassette buff len/Series pointer
0070-0087	112-135	CHRGET subroutine; get Basic char
0077-0078	119-120	Basic pointer (within subrtn)
0088-008C	136-140	Random number seed.
008D-008F	141-143	Jiffy clock for TI and TI$
0090-0091	144-145	Hardware interrupt vector
0092-0093	146-147	BRK interrupt vector
0094-0095	148-149	NMI interrupt vector
0096	150	Status word ST
0097	151	Which key down; 255=no key
0098	152	Shift key: 1 if depressed
0099-009A	153-154	Correction clock
009B	155	Keyswitch PIA: STOP and RVS flags
009C	156	Timing constant for tape

009D	157	Load=0, Verify=1
009E	158	Number of characters in keybd buffer
009F	159	Screen reverse flag
00A0	160	IEEE output; 255=character pending
00A1	161	End-of-line-for-input pointer
00A3-00A4	163-164	Cursor log (row, column)
00A5	165	IEEE output buffer
00A6	166	Key image
00A7	167	0=flash cursor
00A8	168	Cursor timing countdown
00A9	169	Character under cursor
00AA	170	Cursor in blink phase
00AB	171	EOT received from tape
00AC	172	Input from screen/from keyboard
00AD	173	X save
00AE	174	How many open files
00AF	175	Input device, normally 0
00B0	176	Output CMD device, normally 3
00B1	177	Tape character parity
00B2	178	Byte received flag
00B3	179	Logical Address temporary save
00B4	180	Tape buffer character; MLM command
00B5	181	File name pointer; MLM flag, counter
00B7	183	Serial bit count
00B9	185	Cycle counter
00BA	186	Tape writer countdown
00BB-00BC	187-188	Tape buffer pointers, #1 and #2
00BD	189	Write leader count; read pass1/2
00BE	190	Write new byte; read error flag
00BF	191	Write start bit; read bit seq error
00C0-00C1	192-193	Error log pointers, pass1/2
00C2	194	0=Scan/1-15=Count/$40=Load/$80=End
00C3	195	Write leader length; read checksum
00C4-00C5	196-197	Pointer to screen line
00C6	198	Position of cursor on above line
00C7-00C8	199-200	Utility pointer: tape, scroll
00C9-00CA	201-202	Tape end addrs/End of current program
00CB-00CC	203-204	Tape timing constants
00CD	205	0=direct cursor, else programmed
00CE	206	Tape read timer 1 enabled
00CF	207	EOT received from tape
00D0	208	Read character error
00D1	209	# characters in file name
00D2	210	Current file logical address
00D3	211	Current file secondary addrs
00D4	212	Current file device number
00D5	213	Right-hand window or line margin
00D6-00D7	214-215	Pointer: Start of tape buffer
00D8	216	Line where cursor lives
00D9	217	Last key/checksum/misc.
00DA-00DB	218-219	File name pointer
00DC	220	Number of INSERTs outstanding
00DD	221	Write shift word/read character in
00DE	222	Tape blocks remaining to write/read
00DF	223	Serial word buffer

00E0-00F8	224-248	(40-column) Screen line wrap table
00E0-00E1	224-225	*(80-column) Top, bottom of window
00E2	226	*(80-column) Left window margin
00E3	227	*(80-column) Limit of keybd buffer
00E4	228	*(80-column) Key repeat flag
00E5	229	*(80-column) Repeat countdown
00E6	230	*(80-column) New key marker
00E7	231	*(80-column) Chime time
00E8	232	*(80-column) HOME count
00E9-00EA	233-234	*(80-column) Input vector
00EB-00EC	235-236	*(80-column) Output vector
00F9-00FA	249-250	Cassette status, #1 and #2
00FB-00FC	251-252	MLM pointer/Tape start address
00FD-00FE	253-254	MLM, DOS pointer, misc.
0100-010A	256-266	STR$ work area, MLM work
0100-013E	256-318	Tape read error log
0100-01FF	256-511	Processor stack
0200-0250	512-592	MLM work area; Input buffer
0251-025A	593-602	File logical address table
025B-0264	603-612	File device number table
0265-026E	613-622	File secondary adds table
026F-0278	623-632	Keyboard input buffer
027A-0339	634-825	Tape#1 input buffer
033A-03F9	826-1017	Tape#2 input buffer
033A	826	*DOS character pointer
033B	827	*DOS drive 1 flag
033C	828	*DOS drive 2 flag
033D	829	*DOS length/write flag
033E	830	*DOS syntax flags
033F-0340	831-832	*DOS disk ID
0341	833	*DOS command string count
0342-0352	834-850	*DOS file name buffer
0353-0380	851-896	*DOS command string buffer
03E9	1001	(Fat 40) New key marker
03EA	1002	(Fat 40) Key repeat countdown
03EB	1003	(Fat 40) Keyboard buffer limit
03EC	1004	(Fat 40) Chime time
03ED	1005	(Fat 40) Decisecond timer
03EE	1006	(Fat 40) Key repeat flag
03EE-03F7	1006-1015	(80-column) Tab stop table
03EF	1007	(Fat 40) Tab work value
03F0-9	1008-1017	(Fat 40) Tab stops
03FA-03FB	1018-1019	Monitor extension vector
03FC	1020	*IEEE timeout defeat
0400-7FFF	1024-32767	Available RAM including expansion
8000-83FF	32768-33791	(40-column) Video RAM
8000-87FF	32768-34815	*(80-column) Video RAM
9000-AFFF	36864-45055	Available ROM expansion area
B000-DFFF	45056-57343	Basic, DOS, Machine Lang Monitor
E000-E7FF	57344-59391	Screen, Keyboard, Interrupt programs
E810-E813	59408-59411	PIA 1 - Keyboard I/O
E820-E823	59424-59427	PIA 2 - IEEE-488 I/O
E840-E84F	59456-59471	VIA - I/O and timers
E880-E881	59520-59521	(80-column) CRT Controller
F000-FFFF	61440-65535	Reset, I/O handlers, Tape routines

The Machine Language Monitor

TIM is the Terminal Interface Monitor program for MOS Technology's 65xx microprocessors. It has been expanded and adapted to function on the PET/CBM and is included in ROM.

Commands typed on the PET/CBM keyboard can direct the Monitor to start executing a program, display registers and memory locations, set breakpoints, and load or save binary data. When modifying memory, the Monitor performs automatic read after write verification to insure that addressed memory exists, is R/W type, and is responding correctly.

Enter the Monitor via the SYS command. The location of the address used must contain a break instruction (00).

EXAMPLE: sys 4

TIM Connands

- M display memory
- R display register
- G begin execution
- X exit to basic
- L load
- S save

Examples

M Display Memory

```
.m c000,c010
.: c000   ld c7 48 c6 35 cc ef c7
.: c008   c5 ca df ca 70 cf 23 cb
.: c010   9c c8 9c c7 74 c7 lf c8
```

In a Display Memory command, the starting and ending addresses must be completely specified as 4 digit hex numbers. To modify a memory location, move the cursor up in the display, type to correction and press <RETURN> to enter the change. When doing a screen edit, the colon tells the Monitor that you are re-entering data.

R Display Registers

```
.r
   PC    IRQ SR AC XR YR SP
.; 0005  e455 30 00 5e 04 f8
```

Registers are saved and restored upon each entry or exit from the Monitor. They may be modified or preloaded as in the Display memory example above. The semicolon tells the Monitor that you are modifying registers. The register codes are represented as follows:

PC	Program counter
IRQ	IRQ vector location
SR	Stack Register
AC	Accumulator
XR	X Register
YR	Y Register
SP	Stack pointer

G Begin Execution

```
.g c38b
```

The GO command may have an optional target address. If none is specified, the value store in the PC is uded as the target.

X Exit to Basic

```
.x
ready
```

Causes a warm start of BASIC. In a warm start, memory is not altered in any way and BASIC resumes operation the way it was before the Monitor call was made.

L Load

```
.l''filename'',08
```

When LOADing, the device number (in this case 08, for disk) and the filename must be completely specified. Operating system prompts are the same as for BASIC. Memory addresses are loaded as specified in the file header that was set up by the SAVE command. If the DRIVE# is not specified when LOADing from disk, ie., ''l:filename'', the default value is 0 (zero).

S Save

```
.s''filename'',08,033a,0390
```

The SAVE command specifies the beginning (033a) and ending (0390) addresses where your data and/or machine language code is stored. These addresses must be entered as 4 hex digits, and specify the LOADing addresses that are stored in the file header. Any starting and ending addresses may be specified, but be sure to use an ending address at least 1 byte greater than your last byte of data. If DRIVE# is not specified when SAVEing to disk, ie , ''l:filename'', then the default value is 0 (zero).

To cancel a Display Memory, LOAD, or SAVE, press the <RUN/STOP> key. →

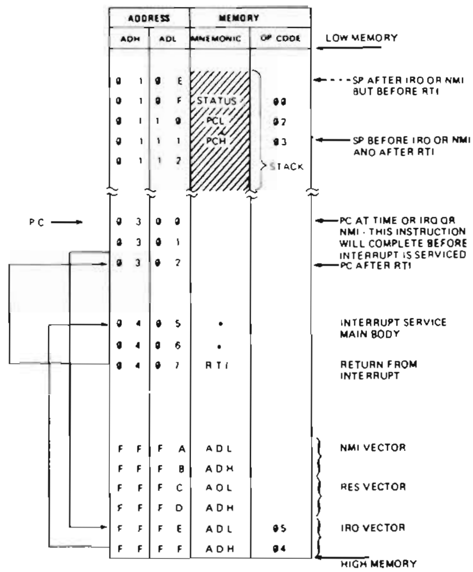


FIG. 1 IRQ, NMI, RTI, BRK OPERATION

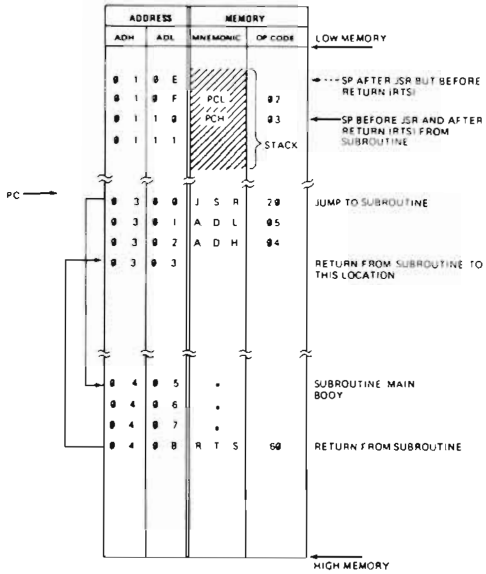


FIG. 2 JSR, RTS OPERATION

FOLD HERE —

ASSEMBLER DIRECTIVES

- .OPT - IF USED MUST BE THE FIRST EXECUTABLE STATEMENT IN THE PROGRAM.
- OPTIONS ARE - (OPTIONS LISTED ARE THE DEFAULT VALUE. TURNED OFF BY (NO) PREFIX.)
  - COUNT (COU OR CNT) - LIST ALL INSTRUCTIONS AND THEIR USAGE.
  - NOGENERATE (NOG) DO NOT GENERATE MORE THAN ONE LINE OF CODE FOR ASCII STRINGS.
  - XREF (XRE) - PRODUCE A CROSS-REFERENCE LIST IN THE SYMBOL TABLE.
  - ERRORS (ERR) - CREATE AN ERROR FILE.
  - MEMORY (MEM) - CREATE AN ASSEMBLER OBJECT OUTPUT FILE.
  - LIST (LIS) - PRODUCE A FULL ASSEMBLY LISTING.
- .BYTE - PRODUCES A SINGLE BYTE IN MEMORY EQUAL TO EACH OPERAND SPECIFIED.
- .WORD - PRODUCES TWO BYTES IN MEMORY EQUAL TO EACH OPERAND SPECIFIED.
- *= - DEFINES THE BEGINNING OF A NEW PROGRAM COUNTER SEQUENCE.
- .PAGE - ADVANCES THE LISTING TO THE TOP OF A NEW PAGE.
- .END - DEFINES THE END OF A SOURCE PROGRAM

LABELS.

LABELS BEGIN IN COLUMN 1 AND ARE SEPARATED FROM THE INSTRUCTION BY AT LEAST ONE SPACE
LABELS CAN BE UP TO 6 ALPHANUMERIC CHARACTERS LONG AND MUST BEGIN WITH AN ALPHA CHARACTER.
A, X, Y, S, AND P ARE RESERVED AND CANNOT BE USED AS LABELS.

LABEL = EXPRESSION CAN BE USED TO EQUATE LABELS TO INSTRUCTIONS.

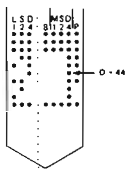
LABEL *=*+N CAN BE USED TO RESERVE AREAS IN MEMORY

CHARACTERS USED AS SPECIAL PREFIXES:

.	INDICATES AN ASSEMBLER DIRECTIVE
#	SPECIFIES THE IMMEDIATE MODE OF ADDRESSING.
$	SPECIFIES A HEXADECIMAL CHARACTER.
@	SPECIFIES AN OCTAL NUMBER
%	SPECIFIES A BINARY NUMBER.
'	SPECIFIES AN ASCII LITERAL CHARACTER.
( )	INDICATES INDIRECT ADDRESSING.
;	IN COLUMN 1 INDICATES A COMMENT.

ASCII CHARACTER SET (7-BIT CODE)

LSD \ MSD		0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NUL	DLE	SP	0	@	P		p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	
C	1100	FF	FS	,	<	L	\	l	¦
D	1101	CR	GS	-	=	M	]	m	
E	1110	SO	RS	.	>	N	↑	n	~
F	1111	SI	VS	/	?	☐	←	o	DEL



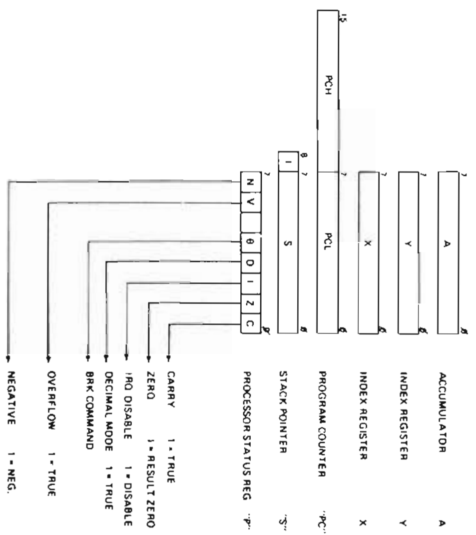
FOLD HERE —

MOS TECHNOLOGY, INC.

Valley Forge Corporate Center
950 Rittenhouse Road
Norristown, Pa. 19401

MCS6500 INSTRUCTION SET SUMMARY

PROCESSOR PROGRAMMING MODEL



INSTRUCTIONS		IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM.			IMPLIED		
MNEMONIC	OPERATION	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#
A D C	A+M+C → A (4)(1)	69	2	2	6D	4	3	65	3	2						
[illegible]																
A S L	C←[7 … 0]←0				0E	6	3	06	5	2	0A	2	1			
[illegible]																
B C S	BRANCH ON C=1 (2)															
B E Q	BRANCH ON Z=1 (2)															
[illegible]																
B M I	BRANCH ON N=1 (2)															
[illegible]																
B P L	BRANCH ON N=0 (2)															
B R K	(See Fig. 1)													00	7	1
[illegible]																
B V S	BRANCH ON V=1 (2)															
[illegible]																
C L D	0 → D													D8	2	1
C L I	0 → I													58	2	1
[illegible]																
C M P	A-M (1)	C9	2	2	CD	4	3	C5	3	2						
[illegible]																
C P Y	Y-M	C0	2	2	CC	4	3	C4	3	2						
D E C	M-1 → M				CE	6	3	C6	5	2						
[illegible]																
D E Y	Y-1 → Y													88	2	1
[illegible]																
I N C	M+1 → M				EE	6	3	E6	5	2						
I N X	X+1 → X													E8	2	1
[illegible]																
J M P	JUMP TO NEW LOC.				4C	3	3									
[illegible]																
L D A	M → A (1)	A9	2	2	AD	4	3	A5	3	2						

MNEMONIC	(IND, X)			(IND), Y			Z, PAGE, X			ABS, X			ABS, Y		
	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#
A D C	61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3
[illegible]															
A S L							16	6	2	1E	7	3			
[illegible]															
B C S															
B E Q															
[illegible]															
B M I															
[illegible]															
B P L															
B R K															
[illegible]															
B V S															
[illegible]															
C L D															
C L I															
[illegible]															
C M P	C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3
[illegible]															
C P Y															
D E C							D6	6	2	DE	7	3			
[illegible]															
D E Y															
[illegible]															
I N C							F6	6	2	FE	7	3			
I N X															
[illegible]															
J M P															
[illegible]															
L D A	A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3

MNEMONIC	RELATIVE			INDIRECT			Z, PAGE, Y			CONDITION CODES					
	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
A D C										✓	✓	✓	-	-	✓
[illegible]															
A S L										✓	✓	✓	-	-	-
[illegible]															
B C S	B0	2	2							-	-	-	-	-	-
B E Q	F0	2	2							-	-	-	-	-	-
[illegible]															
B M I	30	2	2							-	-	-	-	-	-
[illegible]															
B P L	10	2	2							-	-	-	-	-	-
B R K										-	-	-	1	-	-
[illegible]															
B V S	70	2	2							-	-	-	-	-	-
[illegible]															
C L D										-	-	-	-	0	-
C L I										-	-	-	0	-	-
[illegible]															
C M P										✓	✓	✓	-	-	-
[illegible]															
C P Y										✓	✓	✓	-	-	-
D E C										✓	✓	-	-	-	-
[illegible]															
D E Y										✓	✓	-	-	-	-
[illegible]															
I N C										✓	✓	-	-	-	-
I N X										✓	✓	-	-	-	-
[illegible]															
J M P				6C	5	3				-	-	-	-	-	-
[illegible]															
L D A										✓	✓	-	-	-	-

— FOLD HERE

MNEMONIC	OPERATION	IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM.			IMPLIED		
		OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#
L D X	M → X (1)	A2	2	2	AE	4	3	A6	3	2						
[illegible]																
L S R	0→[7 … 0]→C				4E	6	3	46	5	2	4A	2	1			
[illegible]																
O R A	A∨M → A	09	2	2	0D	4	3	05	3	2						
P H A	A → Ms S-1 → S													48	3	1
[illegible]																
P L A	S+1 → S Ms → A													68	4	1
[illegible]																
R O L	C←[7 … 0]←C				2E	6	3	26	5	2	2A	2	1			
R O R	C→[7 … 0]→C				6E	6	3	66	5	2	6A	2	1			
[illegible]																
R T S	(See Fig. 2) RTRN SUB													60	6	1
[illegible]																
S E C	1 → C													38	2	1
S E D	1 → D													F8	2	1
[illegible]																
S T A	A → M				8D	4	3	85	3	2						
[illegible]																
S T Y	Y → M				8C	4	3	84	3	2						
T A X	A → X													AA	2	1
[illegible]																
T S X	S → X													BA	2	1
[illegible]																
T X S	X → S													9A	2	1
T Y A	Y → A													98	2	1

MNEMONIC	(IND, X)			(IND), Y			Z, PAGE, X			ABS, X			ABS, Y		
	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#
L D X													BE	4	3
[illegible]															
L S R							56	6	2	5E	7	3			
[illegible]															
O R A	01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3
P H A															
[illegible]															
P L A															
[illegible]															
R O L							36	6	2	3E	7	3			
R O R							76	6	2	7E	7	3			
[illegible]															
R T S															
[illegible]															
S E C															
S E D															
[illegible]															
S T A	81	6	2	91	6	2	95	4	2	9D	5	3	99	5	3
[illegible]															
S T Y							94	4	2						
T A X															
[illegible]															
T S X															
[illegible]															
T X S															
T Y A															

MNEMONIC	RELATIVE			INDIRECT			Z, PAGE, Y			CONDITION CODES					
	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
L D X							B6	4	2	✓	✓	-	-	-	-
[illegible]															
L S R										0	✓	✓	-	-	-
[illegible]															
O R A										✓	✓	-	-	-	-
P H A										-	-	-	-	-	-
[illegible]															
P L A										✓	✓	-	-	-	-
[illegible]										(RESTORED)					
R O L										✓	✓	✓	-	-	-
R O R										✓	✓	✓	-	-	-
[illegible]										(RESTORED)					
R T S										-	-	-	-	-	-
[illegible]															
S E C										-	-	1	-	-	-
S E D										-	-	-	-	1	-
[illegible]															
S T A										-	-	-	-	-	-
[illegible]															
S T Y										-	-	-	-	-	-
T A X										✓	✓	-	-	-	-
[illegible]															
T S X										✓	✓	-	-	-	-
[illegible]															
T X S										-	-	-	-	-	-
T Y A										✓	✓	-	-	-	-

(1) ADD 1 TO "N" IF PAGE BOUNDRY IS CROSSED

(2) ADD 1 TO "N" IF BRANCH OCCURS TO SAME PAGE.
ADD 2 TO "N" IF BRANCH OCCURS TO DIFFERENT PAGE.

(3) CARRY NOT = BORROW.

(4) IF IN DECIMAL MODE Z FLAG IS INVALID.
ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT.

X INDEX X
Y INDEX Y
A ACCUMULATOR
M MEMORY PER EFFECTIVE ADDRESS
Ms MEMORY PER STACK POINTER

\+ ADD
— SUBTRACT
∧ AND
∨ OR

∀ EXCLUSIVE OR
✓ MODIFIED
\- NOT MODIFIED
M_7 MEMORY BIT 7
M_6 MEMORY BIT 6

N NO. CYCLES
\# NO. BYTES

— FOLD HERE

OP - CODE TABLE

MSD \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	LSD / MSD
0	BRK	ORA-IND, X				ORA-Z, Page	ASL-Z, PAGE		PHP	ORA-IMM	ASL A			ORA-ABS	ASL-ABS		0
[illegible]	[illegible]	[illegible]				[illegible]	[illegible]		[illegible]	[illegible]				[illegible]	[illegible]		[illegible]
2	JSR	AND-IND, X			BIT-Z, Page	AND-Z, Page	ROL-Z, Page		PLP	AND-IMM	ROL A		BIT-ABS	AND-ABS	ROL-ABS		2
[illegible]	[illegible]	[illegible]				[illegible]	[illegible]		[illegible]	[illegible]				[illegible]	[illegible]		[illegible]
4	RTI	EOR-IND, X				EOR-Z, Page	LSR-Z, Page		PHA	EOR-IMM	LSR A		JMP ABS	EOR-ABS	LSR-ABS		4
[illegible]	[illegible]	[illegible]				[illegible]	[illegible]		[illegible]	[illegible]				[illegible]	[illegible]		[illegible]
6	RTS	ADC-IND, X				ADC-Z, Page			PLA	ADC-IMM			JMP IND	ADC-ABS			6
[illegible]	[illegible]	[illegible]				[illegible]			[illegible]	[illegible]				[illegible]			[illegible]
8		STA-IND, X			STY-Z, Page	STA-Z, Page	STX-Z, Page		DEY		TXA		STY-ABS	STA-ABS	STX-ABS		8
[illegible]	[illegible]	[illegible]			[illegible]	[illegible]	[illegible]		[illegible]	[illegible]	[illegible]			[illegible]			[illegible]
A	LDY-IMM	LDA-IND, X	LDX-IMM		LDY-Z, Page	LDA-Z, Page	LDX-Z, Page		TAY	LDA-IMM	TAX		LDY-ABS	LDA-ABS	LDX-ABS		A
[illegible]	[illegible]	[illegible]			[illegible]	[illegible]	[illegible]		[illegible]	[illegible]	[illegible]		[illegible]	[illegible]	[illegible]		[illegible]
C	CPY-IMM	CMP-IND, X			CPY-Z, Page	CMP-Z, Page	DEC-Z, Page		INY	CMP-IMM	DEX		CPY-ABS	CMP-ABS	DEC-ABS		C
[illegible]	[illegible]	[illegible]				[illegible]	[illegible]		[illegible]	[illegible]				[illegible]	[illegible]		[illegible]
E	CPX-IMM	SBC-IND, X			CPX-Z, Page	SBC-Z, Page	INC-Z, Page		INX	SBC-IMM	NOP		CPX-ABS	SBC-ABS	INC-ABS		E
[illegible]	[illegible]	[illegible]				[illegible]	[illegible]		[illegible]	[illegible]				[illegible]	[illegible]		[illegible]

IMM — IMMEDIATE ADDRESSING – THE OPERAND IS CONTAINED IN THE SECOND BYTE OF THE INSTRUCTION.

ABS — ABSOLUTE ADDRESSING – THE SECOND BYTE OF THE INSTRUCTION CONTAINS THE 8 LOW ORDER BITS OF THE EFFECTIVE ADDRESS. THE THIRD BYTE CONTAINS THE 8 HIGH ORDER BITS OF THE EFFECTIVE ADDRESS.

Z. PAGE — ZERO PAGE ADDRESSING – SECOND BYTE CONTAINS THE 8 LOW ORDER BITS OF THE EFFECTIVE ADDRESS. THE 8 HIGH ORDER BITS ARE ZERO.

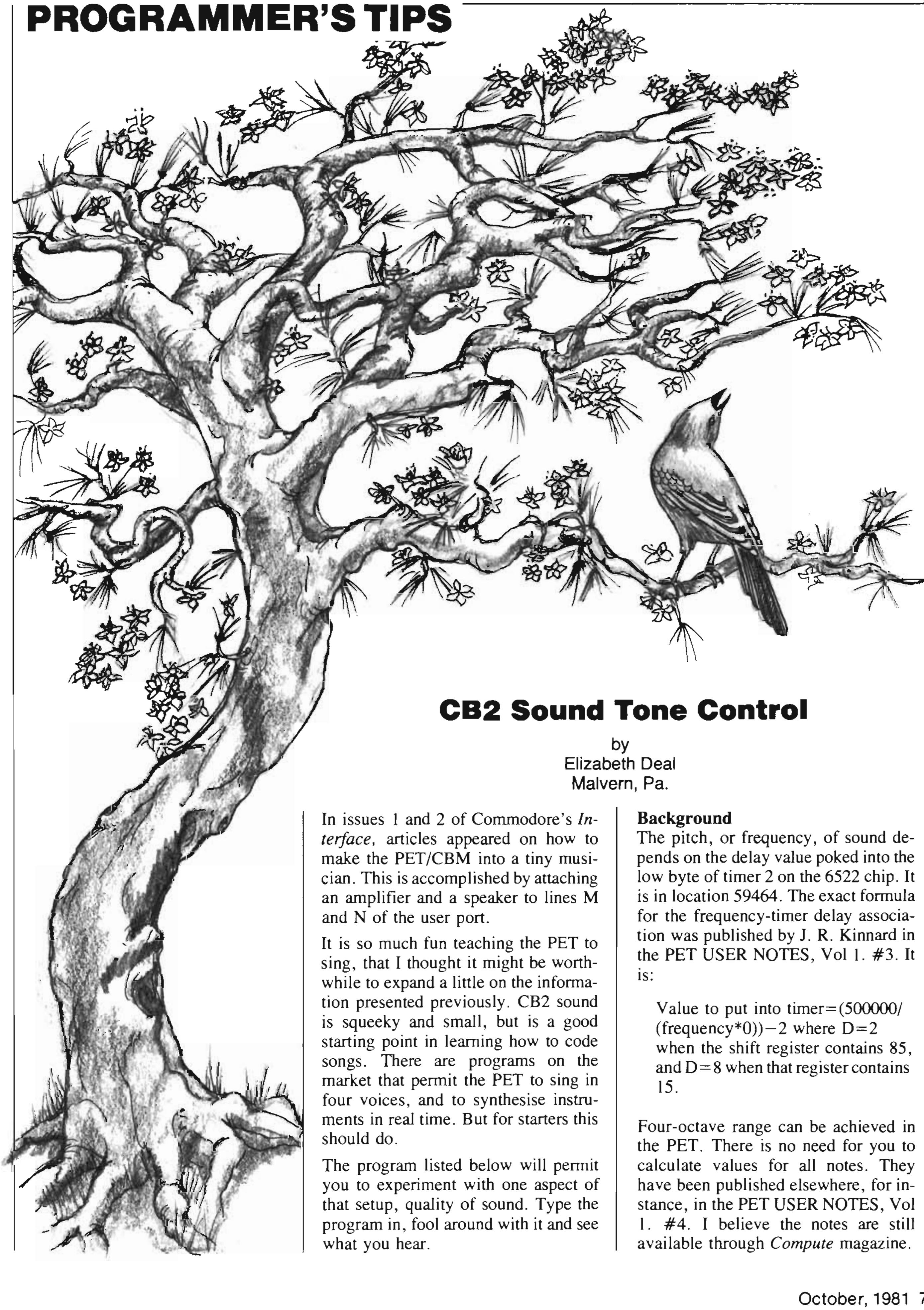
A. — ACCUMULATOR – ONE BYTE INSTRUCTION OPERATING ON THE ACCUMULATOR

Z. PAGE, X Z. PAGE, Y — ZERO PAGE INDEXED – THE SECOND BYTE OF THE INSTRUCTION IS ADDED TO THE INDEX (CARRY IS DROPPED) TO FORM THE LOW ORDER BYTE OF THE EA. THE HIGH ORDER BYTE OF THE EA IS ZEROS

ABS. X ABS. Y — ABSOLUTE INDEXED – THE EFFECTIVE ADDRESS IS FORMED BY ADDING THE INDEX TO THE SECOND AND THIRD BYTE OF THE INSTRUCTION

(IND, X) — INDEXED INDIRECT – THE SECOND BYTE OF THE INSTRUCTION IS ADDED TO THE X-INDEX, DISCARDING THE CARRY. THE RESULTS POINTS TO A LOCATION ON PAGE ZERO WHICH CONTAINS THE 8 LOW ORDER BITS OF THE EA. THE NEXT BYTE CONTAINS THE 8 HIGH ORDER BITS

(IND), Y — INDIRECT INDEXED – THE SECOND BYTE OF THE INSTRUCTION POINTS TO A LOCATION IN PAGE ZERO. THE CONTENTS OF THIS MEMORY LOCATION IS ADDED TO THE Y INDEX, THE RESULT BEING THE LOW ORDER EIGHT BITS OF THE EA. THE CARRY FROM THIS OPERATION IS ADDED TO THE CONTENTS OF THE NEXT PAGE ZERO LOCATION, THE RESULT BEING THE 8 HIGH ORDER BITS OF THE EA



CB2 Sound Tone Control

by
Elizabeth Deal
Malvern, Pa.

In issues 1 and 2 of Commodore's *Interface,* articles appeared on how to make the PET/CBM into a tiny musician. This is accomplished by attaching an amplifier and a speaker to lines M and N of the user port.

It is so much fun teaching the PET to sing, that I thought it might be worthwhile to expand a little on the information presented previously. CB2 sound is squeeky and small, but is a good starting point in learning how to code songs. There are programs on the market that permit the PET to sing in four voices, and to synthesise instruments in real time. But for starters this should do.

The program listed below will permit you to experiment with one aspect of that setup, quality of sound. Type the program in, fool around with it and see what you hear.

Background

The pitch, or frequency, of sound depends on the delay value poked into the low byte of timer 2 on the 6522 chip. It is in location 59464. The exact formula for the frequency-timer delay association was published by J. R. Kinnard in the PET USER NOTES, Vol 1. #3. It is:

> Value to put into timer=(500000/(frequency*0))−2 where D=2 when the shift register contains 85, and D=8 when that register contains 15.

Four-octave range can be achieved in the PET. There is no need for you to calculate values for all notes. They have been published elsewhere, for instance, in the PET USER NOTES, Vol 1. #4. I believe the notes are still available through *Compute* magazine.

PROGRAMMER'S TIPS

The character of tone depends on the value poked into the shift register on the 6522 chip. It is in location 59466. Consistent with its usual behavior, PET will see a number placed in location 59466 as an eight-bit byte. The shift register, as the name implies, will move, one by one, each bit of that value, onto the CB2 line of the user port (line M). It will do so at a regular rate, of x cycles in a second so that the result becomes audible when amplified. The rate at which bits are shifted is controlled by your selection of a value put into timer 2 (location 59464).

Time to do it

As you run this little routine you'll be able to see the bit-string patterns of the numbers you have put into the shift register, and you will be able to hear an association between the values, tone quality and pitch.

Note that the most regular square wave is produced by such patterns as:

```
00001111 = 15 decimal
00110011 = 51    "
01010101 = 85    "
```

Can you guess how the following will sound in comparison to the three patterns above?

```
11110000 = 240 decimal
11001100 = 204    "
10101010 = 170    "
```

Can you see why putting a value of zero or 255 (11111111 in the last instance) produces no sound?

When you change the value in 59466 to numbers other than those above you will notice a significant change in the quality of tone. I can't describe sound on paper. But your ears can.

When 15 is placed into the shift register and the timer contains a value of 140, you will hear a nice, perfect pitch, A-440 to which most instruments are tuned. There is nothing magical about using 140 in my program. You may change that value to whatever you desire. Certain low and high values will not give you any sound — don't be disturbed. It's normal.

Try to guess what will happen to that A-440 we have put in when you change the number in the shift register from 15 to 85. Then do it with your PET and listen.

The program is set up for a normal exit—when you have enough experimenting type in "Q" to quit. This will reset the auxiliary control register (location 59467) to zero, a state it must be in prior to reading or writing tape.

Note that if you exit this program on error or by use of a stop-key, you will have to reset the control register by yourself, whether the note is still sounding or not. The tapes cannot be read nor written when location 59467 contains 16. The tape will move, but nothing will be read or written correctly. The reason is that this register is essential for proper shifting of bits onto the cassette lines, but it is done under a different timing protocol. If you want to learn more details on that subject, read Nick Hampshire's "The PET Revealed."

Have fun with sound. It may even help you learn some bit-string patterns of the small decimal numbers in the process. They are handy in many situations.➔

Elizabeth Deal is a Malvern, Pa.-based free-lance writer whose work frequently appears in COMPUTE! ■

```
200 REM-----------------------------------------------------------------
210 REM EXPERIMENTS FOR CB2 SOUND TONE CONTROL
220 REM INCLUDES A TINY DECIMAL TO BIT-STRING CONVERSION SUBROUTINE
230 REM ELIZABETH DEAL
240 REM-----------------------------------------------------------------
250 PRINT"████";
260 PRINT"LOCATION 59464 WILL CONTAIN 140"
270 PRINT"WHICH WILL GIVE SOUND OF A-440.":PRINT
280 PRINT"YOU MY VARY THE OCTAVE AND TONE OF THAT"
290 PRINT"NOTE BY VARYING A NUMBER THAT GOES"
300 PRINT"INTO LOCATION 59466.":PRINT
310 PRINT"THE NOTE WILL SOUND UNTIL YOU PRESS":PRINT"ANY KEY.":PRINT:PRINT
320 :
330 NT=140:N0=0:N1=16
340 TL=59464:SR=59466:AC=59467
350 INPUT"'Q5' OR 0-255   15█████";X$
360 IFLEFT$(X$,1)="Q"THENPOKEAC,0:END
370 TC=VAL(X$)
380 IFTC<0ORTC>255GOTO350
390 GOSUB450:PRINT"]"TAB(20)"---> "BS$
400 POKEAC,N1:POKESR,TC:POKETL,NT
410 GETA$:IFA$=""GOTO410
420 POKEAC,N0
430 GOTO350:---
440 :
450 BS$="":B7=128:FORB1=0TO7:BS$=BS$+MID$(STR$(SGN(TCANDB7)),2):B7=B7/2:NEXTB1
460 RETURN
470 REM-----------------------------------------------------------------
READY.
```

Note	Freq	Poke
B	- 247	251
C	- 262	237
C#	- 277	224
D	- 294	211
D#	- 311	199
E	- 330	187
F	- 349	177
F#	- 367	168
G	- 392	157
G#	- 415	149
A	- 440	140
A#	- 466	132
B	- 494	125
C	- 523	118
C#	- 554	111
D	- 587	104
D#	- 622	98
E	- 659	93
F	- 698	88
F#	- 734	83
G	- 784	78
G#	- 831	73
A	- 880	69
A#	- 932	65

Using a Microcomputer as an Interactive Terminal

by Dr. Philip G. Barker

Reprinted from Commodore Club News, U.K.

Introduction

A microcomputer consists of an input unit (usually a keyboard), a display system (such as a CRT screen), memory elements (RAM and ROM) and processing logic (one or more CPUs). These components are inter-connected by means of an appropriate bus structure. Provided suitable interfaces are available, this bus also permits the attachments of a variety of external devices. The most commonly used add-on peripherals are document printers, disk units, tape cassettes and modems.

Microcomputers are most often used as stand-alone computing facilities. However, because they have all the necessary hardware elements available, they may also be employed to function as interactive terminal devices. If a microcomputer is to be used in this way, some form of interface unit will be needed. The nature of the interface will depend upon the distances that separate the micro and the host to which it is to be attached. When the distances involved are relatively short, as might be the case when a local mainframe is used, very simple interfaces can be employed. However, when significant distances separate the devices—as in the case of connection to a remote mainframe—some form of modem is needed.

The advantages that accrue from being able to perform this type of computer inter-connection are many-fold. Amongst the more important of these are:

(1) the ability to access a far wider range of software than is available on a single isolated microcomputer.
(2) the ability to use the attached host as a powerful backend data base processor enabling the archival, retrieval and sharing of both programs and data.
(3) the availability of more convenient program development tools such as editors, cross-assemblers, simulators and loaders. These permit software developed on the mainframe to be cross-loaded into an attached microcomputer for subsequent autonomous operation.

The ease with which these benefits may be derived will vary considerably with the nature of the computers involved and the kind of link that is to be implemented. This paper describes some of the problems that were encountered when a Commodore PET microcomputer was used as an interactive terminal device attached to a remote mainframe. It also outlines how these problems were overcome and summarizes some of those application areas likely to benefit from this type of facility.

THE MICROCOMPUTER SYSTEM

In order to effect the operation of a microcomputer as a terminal device, both hardware and software enhancements will usually be necessary. For the PET system, an arrangement of equipment similar to that shown in figure 1 is most often used. Communication with the remote computer installation is achieved over a communications network using a conventional telephone receiver. This is connected via an acoustically coupled modem and a programmable bidirectional interface to the microcomputer system—a 32K Commodore PET (Don80, CBM79). The modem (Tra8) supports transmission speeds up to 300 baud and operates via standard EIA RS-232 (or CCIT V.24) signals.

The Programmable Bidirectional Interface

The external bus structure of the PET is based upon two kinds of Interface:

(a) an implementation of the IEEE-48 standard (Fis80), and
(b) a non-standard User-Port connection (CBM79).

Most peripheral devices are attached to the computer via the IEEE-4888 port. This procedure is adopted because of the availability of a variety of firmware (ROM) routines that service the I/O instructions written in a programming language (BASIC, PASCAL or Assembler). These routines produce synchronized bus signals that control the I/O activity of the devices that are connected to the IEEE Port. Many commonly available peripherals, including modems for terminals, operate according to the RS-232 standard. Consequently, before such devices can be connected to the PET, some means of interconverting between IEEE-488 and RS-232 conventions is necessary. A variety of hardware circuits exist for achieving this (HAM80). We have used a buffered 80 character programmable bidirectional interface constructed by Small Systems Engineering Ltd. (Sma80).

The interface is described as programmable since its communication characteristics can be set up and changed under program control. This is accomplished by sending it a control character (hexadecimal FF) followed by a five byte configuring string. The significance of each of the bytes in this string is as follows:

byte 1: baud rate (50-9600)
byte 2: parity (odd/even)
byte 3: number of stop bits (1 or 2)
byte 4: data input mode (GET or INPUT)
byte 5: code conversion mode (Aor B)

In normal usage the interface is configured for 300 baud operation using even parity, one stop bit, GET input and mode A code conversion.

The Mainframe Computers

Two large mainframe systems have been used to test the communications procedures and terminal software: an IBM 370/168 and a UNIVAC 1110. Each of these supports both conventional batch processing activities and facilities for on-line interactive computing. In addition, both machines are capable of providing user access via ports connected to appropriate com-

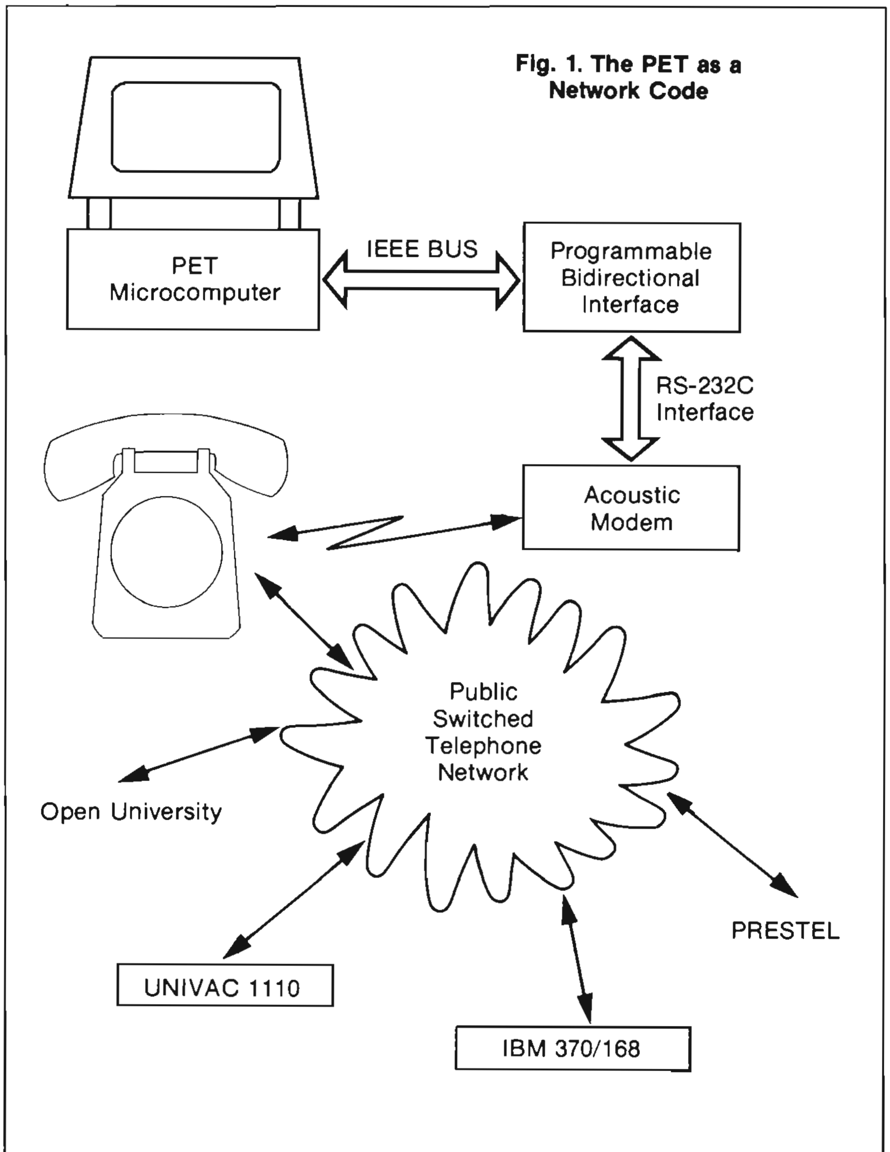


Fig. 1. The PET as a Network Code

munication modems—that can be attached to the communications network. Bulk data storage facilities available on each machine include drums, disks and magnetic tape. A wide range of software exists on both of the computers, including PASCAL, BASIC, PL/I, APL, LISP, COBOL, data base packages, graphics software, word processing systems and files editors. The initial user interface to this software is provided by the operating system that each computer employs.

The IBM 370/168 runs an interactive operating system called MTS—Michigan Terminal System (Bre75, PIR75, NUM78)—which originates from the University of Michigan. This system is highly end-user orientated in that interaction is via a series of English-like commands. These consist of an initial keyword followed by a string of qualifying parameter values. Each of the parameters in a command bears an obvious relationship to the function it performs. Commands exist to create files, edit them and consruct programs to process the data that the files contain. Program development is a relatively simple matter since a variety of useful debugging and testing tools are available.

Like the IBM system, the UNIVAC machine provides a time sharing terminal capability (referred to as 'demand' mode processing) and also supports simultaneous batch work (Bor78, UNI33). It is a manufacturer supplied package but, unlike MTS, is far less end-user orientated. A similar range of software is available. Program development is equally easy even though the file management system and editing facilities are less efficiently organized.

From the point of view of an interactive terminal user there is little difference between the systems with respect to interaction philosophy. However, they do differ significantly in the pragmatics, syntactics and ergonomics of end-user interaction. The net effect of these factors is to burden the remote terminals user with a wide variety of commands, prompt characters, diagnostic messages and peculiarities that derive from particular software packages. Fortunately, if necessary, some of the differences between systems can be appropriately masked from the user by the software that is designed to reside in the microcomputer.

The Microcomputer Software

Before the PET can operate as an interactive terminal a simple software interface is needed. This will take the form of a program that accepts messages typed by the user and transmits these to the mainframe. In addition, it must accept messages transmitted by the remote host and display these on the microcomputer screen. The simplest way in which this can be achieved is when data is transmitted on a character-by-character basis. A simple program to implement this type of data transmission is shown in figure 2.

The code at lines 100 through 140 is executed as an initialization routine and serves the purpose of configuring the programmable interface. Lines 200 through 230 are responsible for detecting character input via the keyboard. As each character typed by the user is detected, it is sent to the mainframe via channel 1. Characters transmitted by the host are stored in the buffer of the interface until they are required for processing by the program. Statements 300 through 330 are responsible for getting characters (hence, GET mode) from the interface buffer (via channel 2) and displaying them on the screen of the microcomputer.

As it is written, the program sets up the mainframe link to function at 300 baud and assumes full-duplex operation—that is, characters typed at the keyboard are echoed back from the mainframe before they are displayed. This is the normal mode of usage with the IBM 370/168 host. When operating in conjunction with the UNIVAC system (see figure 1) the initial byte of the configuring string (line 130 in figure 2) has to set the communication link speed to 110 baud—this is easily achieved by changing the "F" to a "C" in the last item of the print list. In addition, a modification is needed in order to compensate for the fact that characters typed at the keyboard are not echoed back to the terminal by the mainframe. Insertion of the statement

215 PRINT A$;

after line 210 enables this difficulty to be overcome.

When the program is running in the microcomputer, it can be interrupted by pressing the **STOP** key on the PET keyboard. This effect may also be achieved by pressing a user implemented **RESET** button that generates a non-maskable interrupt. This latter facility is useful for interrupting programs written in machine code—provided that the interrupt sectors are set up appropriately. This will be described later. Once the program has been halted it can be listed, modified in various ways and then restarted. If need be, additional programs written in BASIC or in machine code can be loaded from tape or disk without disturbing the mainframe link. Thus, programs to perform particular types of operations (for example, file transfer cross-loading, data conversion, and so on) can easily be loaded, executed and then replaced by other modules that perform different terminal functions.

PROBLEM AREAS

When the BASIC program shown in figure 2 is used to control the PET's operation as a remote terminal, ergonomic limitations become apparent. These arise as a consequence of

(1) The absence of any special communication control keys on the microcomputer keyboard

(2) The limitations imposed by its processing speed

(3) The absence of a screen cursor.

In order that the PET may be used as a more acceptable terminal device, the limitations imposed by each of the above factors need to be overcome. This is easily achieved by using a suitable combination of host processor facilities and local microcomputer software modifications similar to those described below.

Providing Special Control Keys

Most conventional ASCII keyboards usually contain two special keys: **CONTROL** and **BREAK**. These are used, either alone or in combination with other keys, to create codes that have special significance to the software in the remote computer. Such codes are normally employed for control in message editing functions.

Suppose a user is interacting with a host via a terminal device and communication link. From his terminal he initiates the execution of a program so it runs in an interactive mode. While the program is executing, the user depresses the **BREAK** key on his keyboard. This action would normally invoke an interrupt system that would halt the program and then pass control back so that another command could be typed. Similarly, if the terminal user gives a command to list a file containing 200 lines of data and, after seeing the first six, decides that he does not need to see any more, depression of the **BREAK** key would terminate the listing process and set the terminal back into user command mode. The effect of this key is thus analogous to the effect produced by pressing the **STOP** key on the keyboard of the PET while a BASIC program is running.

The key combinations involving the use of the **CONTROL** key are an important means of adding to or extending the functionality of the keyboard. Many keyboards contain special keys for character or line deletion. Thus, when a user is typing a command line, if a mistake is made, the offending characters (or line) can be logically erased. This is achieved by depressing the appropriate 'character delete' or 'line delete' button on the keyboard. If these keys are not present they can be provided by means of a **CONTROL** key combination. Some typical examples might be:

Figure 2. BASIC Program to enable the PET to operate as a remote terminal.

```
10   REM - PET AS A REMOTE TERMINAL
20   GOSUB 100 : REM SET UP MODEM
30   GOSUB 200 : REM GET KEYBOARD CHARACTER
40   GOSUB 300 : REM GET MAINFRAME CHARACTER
50   GOTO 30
100  REM *** CONFIGURE INTERFACE ***
110  OPEN 1,4  :  REM OUTPUT CHANNEL
120  OPEN 2,6  :  REM INPUT CHANNEL
130  PRINT#1, CHR$(255);"FXXGA"
140  RETURN
200  REM *** GET KEYBOARD CHARACTER ***
210  GET A$ : IF A$ = "" THEN: RETURN
220  PRINT#1, A$;
230  RETURN
300  REM *** GET MAINFRAME CHARACTER ***
310  GET#2,A$ : IF ST=2 THEN : RETURN
320  PRINT A$;
330  RETURN
```

CONTROL H	-	delete the previous character typed,
CONTROL X	-	delete the whole of this line,
CONTROL Z	-	produce an attention interrupt at the host computer,
CONTROL C	-	produce an end-of-file sequence at the host.

In the above examples, the notation on the left indicates that two keys are pressed simultaneously. The **CONTROL** key on the terminal is thus operating in the same way in which the **SHIFT** key functions on the PET.

Because these special function keys are not present on the PET's keyboard, some means of producing equivalent effects needs some of the less frequently used PET keys for this purpose. There are two ways to implement these changes: either in the local software contained in the microcomputer or by means of the facilities provided by the remote mainframe. The latter approach is the easier of the two methods and is the one that was used in conjunction with the software shown in figure 2. The other technique will be described later.

Most mainframe systems permit a wide variety of terminal types to be attached to them. To cater for different communication devices, a series of special programs called 'device support routines' are often used. These are software-based conversion procedures resident in the host system which compensate for the fact that different external devices may have/lack certain special features. Their relationship to a computational process running on the mainframe is shown in figure 3. Effectively, the device support routines 'smooth out' the differences between terminal devices so that, as far as the computational process is concerned, they all look alike.

Given that such routines exist, there are two ways in which the mainframe facilities can be used to 'personalize' a PET terminal. One involves issuing commands to the device support routines manually, while the other arranges for these to be issued automatically when the user logs onto the remote computer.

As an example, consider communication with the remote IBM 370/168. The PET establishes a connection with the remote host. Then, the user types a sequence of commands that gives the associated device support routine information about the terminal it is currently servicing. A typical sequence of commands is shown in the box below — appropriately annotated on the right.

```
%ATC=]   -defines the attention interrupt character,
%DPC=←   -specifies the delete previous character code,
%DLC=↑   -specifies the delete line character
%EFC=×   -defines the end-of-file character.
```

The effect of these commands is quite straightforward. They associate special functions with certain of the PET keys. Thus, each time the PET user presses the) key on his keyboard an attention interrupt will be generated in the mainframe. Similarly, depression of the / key will cause and end-of-file sequence to be produced. During a terminal session, the user is free to allocate these special functions to other keys if necessary. For example, having "personalized" the PET with the commands listed above if the user now types the command &ATC = @, this causes the @ key to become the attention interrupt key thereby releasing) for its normal function.

An alternative method of personalizing the remote PET might involve creating a "Personality File" in the file store of the remote host. This file would contain a series of commands (similar to those listed above) which would automatically be invoked as soon as the PET logged onto the system as a remote terminal. This approach could be used to make the existence of device support routines in the host totally transparent to the user.

Limitations Imposed by Processing Speed

Using the outlined scheme, the PET becomes a more usable remote terminal. However, there are still some problems that need to be overcome. The program presented in figure 2 works reasonably well provided that lower case characters are not present in the messages sent to or received from the remote mainframe. Should this not be the case, code conversion problems can arise. Suppose the following message sequence was sent from the host:

```
>""SOURCE"TO DOES NOT EXIST.          ...........(A)
>Enter replacement or "CANCEL".       ...........(B)
```

This would appear on the screen of the PET in the following way:

```
>""SOURCE"TO" DOES NOT EXIST.         ...........(C)
>E.4%2 2%0,!#%-%4 /2 "CANCEL".        ...........(D)
```

In order to understand the somewhat cryptic appearance of line D the ASCII code values for the character of the message need to be examined. A comparison of the codes for the transmitted and received characters in the first work message B is shown in the following table:

Message Character	ASCII Value (Octal)	Ignore High Order Bit	Resultant Character
E	105	05	E
n	156	56	.
t	164	64	4
e	145	45	%
r	162	62	2

It can be seen that the mysterious appearance of line D arises because the high order bit of the ASCII code for lower case characters appears to have been lost. This phenomenon arises because of the special way in which the screen memory of the PET microcomputer drops bit 6 of the standard ASCII value in order to produce a six bit code for its keyboard characters (CBM79). The problem can be overcome by adding some additional statements to the program presented in figure 2. Modifications, similar to those shown below, can be used to provide the lower case capability needed to overcome the encryption problem outlined above.

```
135  POKE 59468, 14
          o
          o
315  IF A$="" THEN:RETURN
316  C=ASC (A$)
317  IF (C>64) AND (C<91) THEN A=128
318  IF (C>96) AND (C<128) THEN A=-32
319  A$=CHR$(C+A)
          o
          o
```

The POKE statement (line 135) sets the PET into alternate character set mode—upper/lower case rather than upper case/graphics. The extra statements at lines 315 through 319 are included in order to compensate for the effect of the POKE statement on the way in which the ASCII code values are interpreted by the PET. →

In POKE 12 mode (normal), the decimal value 65 represents the letter "A." However, in POLE 14 mode this value corresponds to "a" while "A" is produced by the value 193 (that is, 128+65). The modification shown in lines 315 through 319 makes it possible for the PET to interpret both upper and lower cases characters correctly.

Unfortunately, the additional computational overhead associated with these extra statements introduces a further problem. When long message strings are sent from the host (for example, when listing a file), the speed of the modified program becomes too slow to handle them. Communication between the mainframe and remote terminal is asynchronous (Heb75). Each character transmitted consists of a start bit, seven data bits, a parity bit and one stop bit—that is, ten bits in total. Thus, at 300 baud the mainframe transmits 1 character every 33.33 millisec. If the remote terminal cannot process data quickly enough, then information is likely to be lost unless some form of buffering and/or mechanism for delaying mainframe transmission (flow control) is used.

The programmable interface between the modem and the PET has the capability of buffering 80 characters. Furthermore, when the buffer becomes full, the interface should pass a signal to the mainframe which stops it from transmitting—thereby preventing loss of information. However, if the mainframe chooses to ignore this signal, the interface fails to send it, or, if it is allowed to "float," then information will become lost through buffer overflow. This phenomenon has been observed when the equipment shown in figure 1 is used in conjunction with the modified BASIC program described above.

Suppose R2 is the rate of transmission from the mainframe and R1 the rate at which the PET processes the data in the buffer. An indication of the message size at which information loss will start to occur can be gained from the expression:

$$\text{Message Size} = \frac{80.R2}{R2 - R1}$$

The results of some simple timing experiments are shown in figure 4. The basic time for the original subroutine (lines 200 through 330 in figure 2) to service a character sent from the mainframe is about 31 millisec. The additional overhead added to this routine by the code conversion statements is about 41 millisec/character. It is easy to see that providing a lower case capability more than doubles the time it takes to process each character received from the mainframe.

On the basis that transmission from the mainframe (R2) takes place at a rate of 30 characters/sec and the modified basic program is able to process characters from the buffer at a rate (R1) of 13.83 characters/sec, it is easy to compute an upper limit to the message size that can be handled without information loss. This works out to be about 150 characters. Actual experiments, based upon transferring messages of known size, indicate that information loss starts to take place for messages in excess of 160 characters. Bearing in mind the 'crudeness' of the timing experiments, the agreement between prediction and observation is reasonably good. Indeed, in order to compute the 'correct' answer, the value of R1 only needs to be increased to 15 characters/sec, which means that the difference between estimated and actual values is within 10%. This problem can be easily overcome by speeding up the rate of processing in the PET. Using machine code programs is one way of accomplishing this. Indeed, when the modified program of figure 2 is replaced by an equivalent machine code program, no problems are experienced.

Providing a Cursor

A cursor is commonly used to indicate the position on the screen at which the next character—typed by the user or received from the mainframe—is to appear. There are four basic types of cursors. Different types are distinguished by their shape and whether they are opaque or transparent, static or flashing, hardware or software controlled. A transparent flashing hardware controlled (that is, interrupt driven) cursor is the type which is normally implemented on the PET when it is in direct command or INPUT modes.

Unfortunately, when the PET is programmed to obtain input from the keyboard with a GET statement, no flashing cursor is displayed. However, it is possible to turn it on for this mode of data entry by executing a POKE 167,0 statement, prior to the input transaction (Ham80, Don80). The program listed in figure 2 can thus be modified quite easily in order to provide a flashing cursor facility. Inclusion of the statement

```
21 POKE 167,0
```

is all that is necessary. If such a modification is made, a cursor now appears. But, as user-computer dialogue proceeds, the appearance of the microcomputer screen becomes ergonomically unacceptable. Static images of the cursor ("blobs") remain deposited at what would seem to be randomly selected positions on the screen. In fact, these appear at some of the cursor locations corresponding to the receipt of a carriage return character—from the keyboard or from the mainframe. The particular points at which they occur correspond to instants at which synchronization between the BASIC program and the cursor handling system is lost. There is an easy way of removing the blobs by the addition of some extra BASIC statements, that ensure a space character is deposited at the cursor position when a carriage return code is received. However, like the code conversion routines described in the previous section, the computational overhead of employing such code is prohibitive.

OVERCOMING THE PROBLEMS

The easiest solution to overcoming the problems outlined in the previous sections is to write the software—that listed in figure 2 and the various amendments—in assembler. Bearing in mind what has been said, the basic alogorithm to be implemented is as follows:

Begin:	Set up the non-maskable interrupt vector to handle the RESET button.
Step1:	Get a character from the keyboard.
Step2:	If no character then jump to Step4. If a cursor control character then ignore it (Jump to Step4).
Step3:	Send character to mainframe.
Step4:	Get character from mainframe.
Step5:	If STATUS=2 then jump to Step1.
Step6:	Perform code conversion (upper/lower case).
Step7:	If carriage return character (hexadecimal OD) then over-write the "blob".
Step8:	Jump to Step1 and repeat cycle.
Last:	Reset default Input/Output device codes. Jump back to BASIC Interpreter.

Fig 3. Role of device support routines

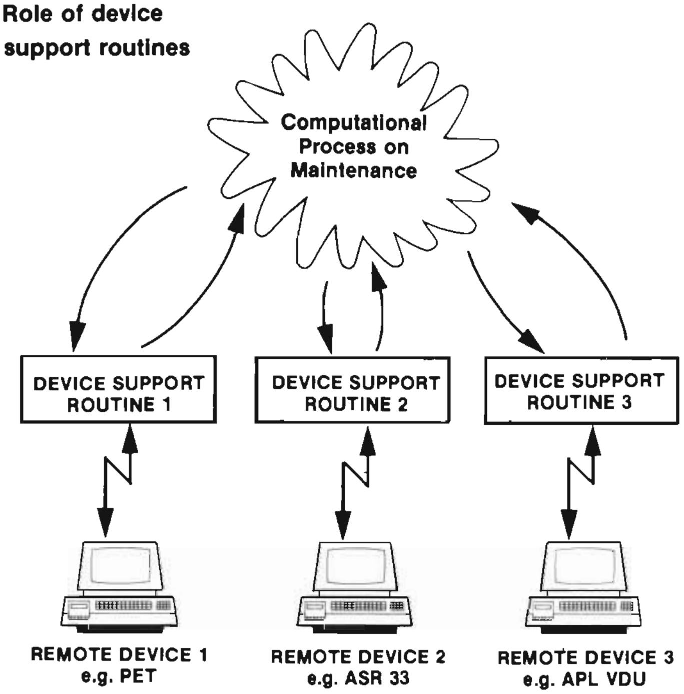


Fig 4. Results of timing experiments.

(1) Driver Routine Blank

```
100  POKE 59468, 14
110  zk=TI
120  FOR I=1 to 100
130  A§=CHR§(65)
140  NEXT I
150  PRINT TI-K
160  PRINT A§
170  END
     Time=39 Jiffies
```

(2) Subroutine Linkage Timing

```
100  POKE 59468, 14
110  K=TI
120  FOR I=1 TO 100
130  A§=CHR§(65)
135  GOSUB 300
140  NEXT I
150  PRINT TI-K
160  PRINT A§
170  END

300  REM *** GET MAINFRAME CHAR ***
310  RETURN

     Time=57 Jiffies
```

(3) Print Statement Overhead

```
lines 100 through 170 of (2)
300  REM *** GET MAINFRAME CHAR ***
310  PRINT A§
320  RETURN

     Time=200 Jiffies
```

(4) GET Timing

```
Lines 100 through 170 of (2)
300  REM *** GET MAINFRAME CHAR ***
310  GET A§ : IF ST=2 THEN : RETURN
320  PRINT A§
330  RETURN

     Time=203 Jiffies (no character input)
         =252 Jiffies (with character input)
     Average=227 Jiffies
```

(5) Code Conversion Timing

```
Lines 100 through 170 of (2)
300  REM *** GET MAINFRAME CHAR ***
310  IF A§=""THEN : RETURN
320  C=ASC (A§)
330  IF (C>64) AND (C<91) THEN A=128
340  IF (C>96) AND (C<128) THEN A=-32
350  A§=CHR§(C+A)
360  RETURN
     Time=303 Jiffies
```

Basic time service a character from mainframe $=\frac{(227-39)}{100} \cdot \frac{1}{60} = 31.3$ millisec
(Results 4 and 1)

Additional overhead in performing code conversion $=\frac{(303-57)}{100} \cdot \frac{1}{60} = 41.0$ millisec
(Results 5 and 2)

The machine code implementation of this algorithm was developed on a cross-assembler for the MCS650X range of microcomputers (Lub80). This was available on one of the backend mainframe machines (IBM 370/168). The development system that was used is similar to that depicted in figure 5. Assembler source language statements were stored in a mainframe file called INPUT. The contents of this file could be modified in various ways by means of the system editor. During an assembly, the cross-assembler read the statements contained in INPUT, checked their validity, and generated appropriate object code which was stored in the file OUTPUT1. At the same time, a listing of the source file (and appropriate diagnostics) was sent to the file OUTPUT2. This could later be listed on a system printer or on a local print device. Alternatively, this output could be produced directly on the screen of the PET. A typical listing of the final version of the assembler program (produced on a local printer) is shown in figure 6.

In order to use this program it is necessary to provide a simple prologue routine written in BASIC. An example of such a routine is given below.

```
10  OPEN 1,4   :REM OUTPUT CHANNEL
20  OPEN 2,6   :REM INPUT CHANNEL
21  POKE 167,0 :REM TURN ON CURSOR
30  PRINT#1, CHR$(255);"FXXGA":REM SET UP
    INTERFACE
40  POKE 59468,14 :REM TURN ON LOWER CASE
50  SYS 8192   :REM JUMP TO ASSEMBLER ROUTINE
```

Use of this program combination enables all the previously described problems to be overcome. The prologue code is written in BASIC rather than assembler so that the end-user can easily modify those parts of it which he is likely to want to change—external device addresses, cursor on/off status and interface details. The assembler routine disables all the cursor control keys in order to avoid spurious side effects. The PET's **RUN STOP** and **RVS** keys are not treated in this way. In the system we use, the **RUN STOP** key is used to produce an attention interrupt in the mainframe, while **RVS** is assigned the task of generating an end-of-file character for data entry from the terminal. The **RESET** button on the PET produces a local interrupt causing control to be passed back to BASIC. When this happens, the PET can be made to function as a stand-alone microcomputer and thereby run BASIC (or assembler) programs—provided they do not interfere with the prologue code

→

listed above. For example, suppose the PET contained the following program:

```
1000  FOR I = 1 TO 100
2000    PRINT I, I*I, I↑3
3000  NEXT I
4000  STOP
```

in addition to the original prologue routine (lines 10 through 50). The effect of the following user directives,

(1) Type; RUN
(2) Press: RESET button
(3) Type: RUN 1000
(4) Type: RUN

would be to set up communication with the mainframe (step 1). Then, at a later stage, logically sever the link (step 2), initiate the execution of a local application program running of the PET (step 3) and then re-establish communication with the mainframe (step 4).

The assembler routine is 181 bytes long and could thus easily fit into one of the tape cassette buffers, thereby leaving the whole of the remaining memory space available for other purposes. Although developed for use on a 40-column PET, the programs will also work on the newer 80-column (8000 series) machines (CBM80a, CBM80b). However, in the latter case the return address to BASIC (warm start) would need to be changed from $C389 (BASIC 2.0) to $B3FF (BASIC 4.0). In addition, it would be desirable (although not necessary) to modify the assembler routine to handle the additional keys present on the extended keyboard. When the software outlined above is used in conjunction with the 8000 series CBM a powerful interactive terminal facility can be constructed.

SOME APPLICATIONS

There are many application areas where the ability to combine a local microprocessing capability with a remote back-end mainframe facility can be usefully employed. The desirability, cost and amount of effort involved in implementing this mode of computing will depend critically on the nature of the link between the mainframe and the micro. In the work that has been described here, a communications network has been employed. Direct links or private dedicated lines invariably provide a better quality communication facility but are often expensive unless line utilization is high. The use of lower quality "dial-up" links puts an extra overhead on the processing software—the need to perform appropriate error checks in order to ensure that data transmitted over the link is not perturbed by extraneous noise. Consequently, in each of the following application examples, although it is not explicitly described, extensive error checking procedures would need to be implemented.

The examples that are described in this section represent perfectly general techniques—that is, they could be employed with any microcomputer system. However, for the purpose of illustration, they are described in terms of a particular machine—the Commodore PET.

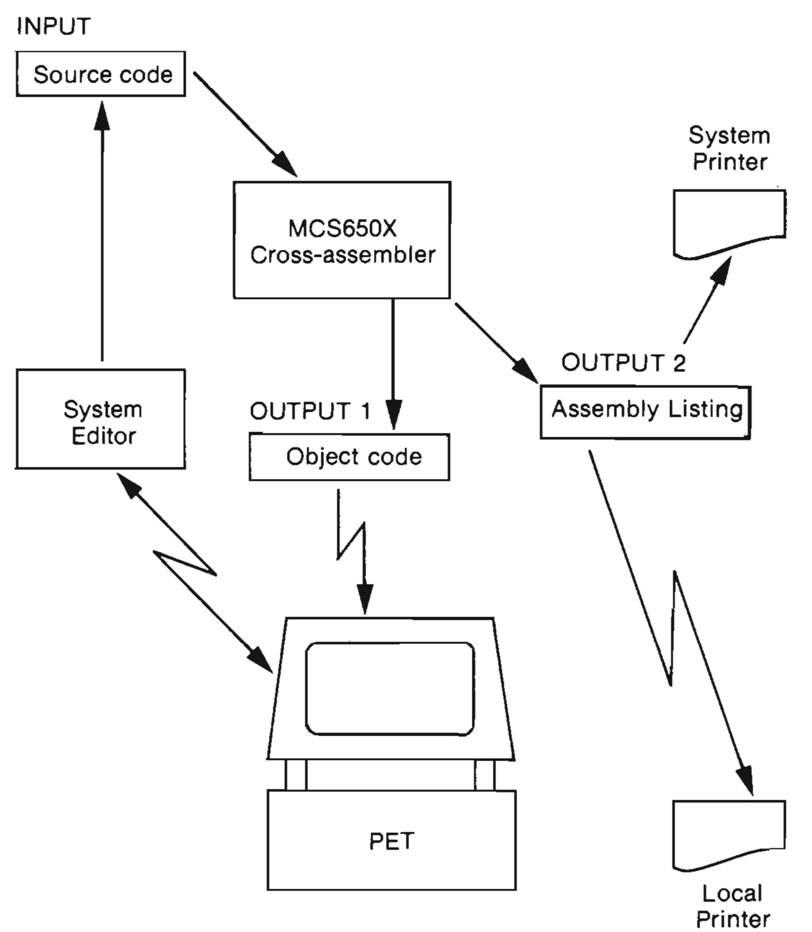
Cross-loading of Machine Code Programs

Applications involving this type of operation fall into two general categories:

(a) those in which the "intelligent" terminal functions as the target machine, and
(b) situations where it acts as intermediate link in a hierarchical chain.

The first of these involves straightforward cross-loading of machine code programs from the mainframe to the microcomputer terminal. This facility can be used to substantially improve its functionality, thereby making it far more useful than a conventional in-

Fig 5 Cross-assembler Development System



teractive terminal. In the second type of application, the intelligent micro-terminal acts as a buffer between the mainframe and the target machine. Code to be cross-loaded is first passed to the micro-terminal where it can be checked for transmission errors and stored. Then, in a subsequent step, it can be sent from the intelligent station across to the target machine.

As far as the Commodore PET is concerned, each of these categories of cross-loading is fairly easy to perform. Consider the case of direct loading. Programs developed using a system similar to that shown in figure 5 are easily transferred from the mainframe. The object code produced by the cross-assembler is stored in a file containing a series of fixed format records. Each record contains:

(a) A value, N, that gives the number of program bytes in the record.
(b) A load address that specifies where these bytes are to be loaded in the memory of the target machine.
(c) The N consecutive program bytes.
(d) A two byte checksum value.

Knowing the format of these records, it is a simple matter to formulate and implement an algorithm to perform the cross-loading operation (Bar81).

Similarly, it is equally easy to implement cross-loading in situations where the PET-based intelligent terminal acts as a mode in hierarchical link. In this context, we have been experimenting with systems in which code that is transferred to the PET is subsequently cross-loaded into other PETs, a Sinclair ZX81, a Texas Instruments TMS 9980A 16-bit machine, and a variety of National Semiconductor SC/MP-based systems.

In-house Viewdata Systems

The simplest form of viewdata systems consists of a series of 'pages' of information stored in the form of a computer data base. In response to queries made by users, appropriate pages of information are transmitted to viewdata terminals distributed over some particular locality or region. Both national and in-house systems currently exist. As business tools, the attractiveness of these systems stems from the ease with which information can be retrieved and the colorful and graphic way in which it can be presented.

A limited form of in-house viewdata system can be designed around the use of the PET as both an editing and retrieval terminal. The data base can reside on a remote mainframe computer, provided that an arrangement similar to that shown in figure 1 is used. In principle, the transmission of pages of information to the PET is similar to the cross-loading of software described in the previous application example. However, in this case, it is screen data that is cross-loaded and not an executable program.

Since the PET must provide page creation, editing and retrieval facilities, a variety of terminal software is needed (Bar81). Pages created locally on the intelligent terminal can be retrieved by means of a variety of simple interfaces. These enable page numbers entered by the user to be transmitted to the mainframe. Retrieval software in the host is used to obtain the required pages from the data base. These are then transmitted back to the PET where a loader program enters them into the screen memory area.

We have been investigating the use of systems in which the full PET keyboard is used to create pages for storage. A much simpler touch sensitive keypad and video display is then used as a retrieval interface. This approach can be used to construct extremely simple inexpensive viewdata systems—particularly where a local mainframe or minicomputer is available to function as a back-end data base machine.

Cross-loading BASIC Programs

There are several approaches to the problem of archiving and retrieving BASIC programs—tape cassette, flexible disk, and the use of a local or remote mainframe computer. Provided suitable data communication facilities are available (figure 1), the major advantages of storing programs in a remote computer are (a) the substantial storage capacity that is available, and (b) the ease with which software can be shared—via appropriate cross-loading techniques. The main disadvantages of this approach are (a) the high telecommunications cost, and (b) the significant program loading time likely to be experienced when low speed lines (110 or 300 baud) are used. Each of these disadvantages can be overcome when attachment is to a local mainframe.

The principles involved in this type of application are very similar to those underlying the previous two examples. The major differences stem from the larger volume of information that is likely to be involved and the more complicated structure of a BASIC program. On the PET, BASIC programs usually occupy two sections of memory. The first of these contains the program code, variables and arrays. The second is used to hold string values. When BASIC programs are transferred between a back-end machine and an intelligent terminal, it becomes necessary to cross-load to two different memory areas. There is one complication to overcome. The load point of the data to be entered into the string area varies with the memory size of the target machine. However, because the loader is resident in each target machine there is no difficulty in determining the available memory space, adjusting pointers in the text as the program is loaded and relocating the string area accordingly.

We are currently investigating the feasibility of connecting multiple PET microcomputers via a common bus to a local mainframe, thereby enabling facile sharing of software and data on an in-house basis. It is likely that the system will employ the standard IEEE bus for data transfer between the intelligent devices and the mainframe. However, it will utilize a separate purpose built address/management bus to control the data flow that takes place.

CONCLUSION

Microcomputer systems have significant potential for use as intelligent interactive terminal devices. As such they have the ability to operate autonomously and may also communicate with computational processes running on local or remote hosts. These hosts may form part of a complex,geographically distributed resource network. As appropriate communication ports become a standard part of microcomputer architecture, it is likely that there will be a substantial increase in 'intelligent terminal' applications. →

Fig. 6. Assembler routine to enable the PET to operate as a terminal

```
                          ;
                          ; CODE TO USE PET AS REMOTE TERMINAL
                          ; ASSEMBLED USING XASM:MCS650XASR AT NUMAC
                          ;
          FFCC            RFILES   EQU  £FFCC   ; REALEASE FILES
          FFE4            GETCHR   EQU  £FFE4   ; GET A CHARACTER
          FFC9            OTFILE   EQU  £FFC9   ; SET UP OUTPUT FILE
          FFC6            INFILE   EQU  £FFC6   ; SET UP INPUT FILE
          FFD2            PRTCHR   EQU  £FFD2   ; PRINT CHARACTER
          C389            BASIC    EQU  £C389   ; RETURN TO BASIC
                          ;
2000                          ORG £2000
                          ;
2000 A0 00                BEGIN LDY #£0
2002 B8 AC20                    LDA LAST,Y    ; NOW SET UP ADDRESS OF
2005 85 84                      STA £94       ; NM1 HANDLER SO THAT
2007 C8                         INY           ; PRESSING THE
2008 89 AC20                    LDA LAST,Y    ; RESET BUTTON PASSES
2008 85 95                      STA £95       ; CONTROL BACK TO BASIC
                          ;
200D 20 00FF              STEP1 JSR RFILES
2010 20 E4FF                    JSR GETCHR    ; GET A KEYBOARD CHARACTER
                          ;
2013 F0 39                STEP2 BEG STEP4     ; IF NO CHAR THEN JUMP TO STEP4
2015 8D AB20                    STA CHAR      ; SAVE CHAR FOR LATER
2018 CS 11                      CMP #£11      ; CURSOR DOWN?
201A F0 27                      BEG NOKEY
201C CS 13                      CMP #£13      ; HOME CURSOR?
201E F0 23                      BEQ NOKEY
2020 CS 14                      CMP #£14      ; DELETE?
2022 F0 1F                      BEQ NOKEY
2024 CS 1D                      CMP #£1D      ; CURSOR RIGHT?
202G F0 1B                      BEG NOKEY
202B CS 8D                      CMP #£8D      ; SHIFT RETURN?
202A F0 17                      BEG NOKEY
202C CS 91                      CMP #£S1      ; CURSOR UP?
202E F0 13                      BEG NOKEY
2030 CS 92                      CMP #£92      ; REVERSE OFF?
2032 F0 0F                      BEG NOKEY
2034 CB 93                      CMP #£93      ; CLEAR SCREEEN?
2036 F0 CB                      BEQ NOKEY
2038 C9 94                      CMP #£94      ; INSERT?
203A F0 07                      BEQ NOKEY
203C C8 9D                      CMP #£9D      ; CURSOR LEFT?
203E F0 03                      BEQ NOKEY
2040 4C 4G20                    JMP STEP3     ; GO SEND CHARACTER
                          ;                     TO MAINFRAME
2043 4C 4E20              NOKEY JMP STEP4     ; IGNOR PET CURSOR
                          ;                     CONTROL KEYS
                          ;
2046 A2 01                STEP3 LDX #£1       ; WRITE CHAR TO MAINFRAME
2048 20 C9FF                    JSR OTFILE
2048 20 D2FF                    JSR PRTCHR
                          ;
204E A2 02                STEP4 LDX #£2       ; GET CHAR FROM MAINFRAME
2050 20 CSFF                    JSR INFILE
2053 20 EFF                     JSR GETCHR
2056 8D AB20                    STA CHAR
                          ;
2059 A5 96                STEPS LDA £96
2058 C9 02                      CMP #£2       ; EXAMINE STATUS
205D F0 AE                      BEQ STEP1
                          ;
205F AD AB20              STEPS LDA CHAR      ; RELOAD CHAR
2062 F0 A9                      BEQ STEP1     ; NO CHAR TO HANDLE
2064 C9 4Q                      CMP #£40      ; IS CHAR GREATER THAN B47
2066 F0 2A                      SEQ STEP7     ; NO - GO PRINT IT
2066 B0 03                      BCS TEST      ; YES
206A 40 B220                    CMP STEP7
206D C9 5B                TEST1 CMP #£60      ; IS CHAR LESS THEN BL?
206F 80 10                      BCC ADD       ; YES THEN ADD ON 123
2071 C9 6C                      CMP #£60      ; IS CHAR GREATER THAN 96
2073 F0 1D                      BEQ STEP7
```

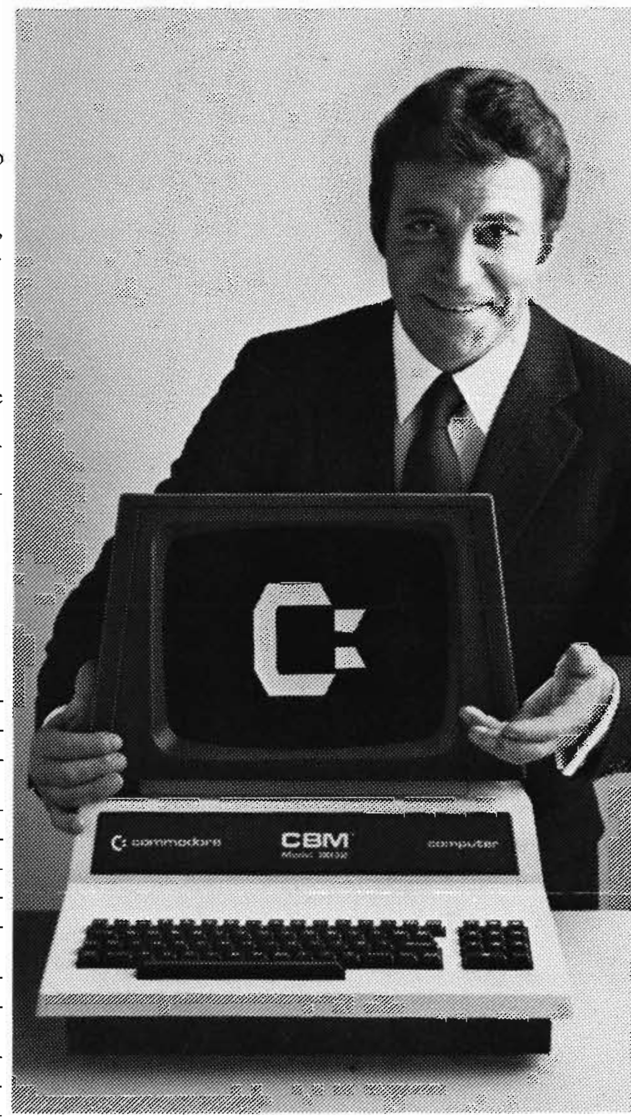
Fig 6 Continued

```
2075 B0 03         74        BCS TEST2
2077 4C 9220       75        JMP STEP7
207A C9 90         76 TEST2  CMP #£SC      ; IS CHAR LESS THAN 12S?
207C 90 0D         77        BCC SUB       ; YES THEN DEDUCT 32
207E 4C 9220       78        JMP STEP 7
2081 D3            79 ADD    OLD
2082 16            80        OLD
2083 69 80         81        ADC #£20      ; ADD ON 12S
2085 8D AB20       82        STA CHAR      ; STORE RESULT BACK
2088 4C 9220       83        JMP STEP7
208B D8            84 SUB    OLD
208C               85        SEC
208D E9 20         86        SBC #£20      ; DEDUCT 32
206F 8D AB20       87        STA CHAR    • ; STORE RESULT BACK AGAIN
                   88 ;
2092 20 CCFF       89 STEP7  JSR RFILES    ; SET DEFAULT DEVICES
2095 AD AB20       90        LDA CHAR      ; GET CHARACTER TO BE PRINTED
2098 CS 0D         91        CMP #£0D      ; IS IT RETURN?
209A DP 09         92        BNE PRINT     ; NO ITS NOT
209C A4 CS         93        LDY £CS       ; STORE A SPACE
209E A9 20         94        LDA #£20      ; IN POSITION OF CURSOR
20A0 B1 C4         95        STA (£C4),Y   ; TO AVOID THE BLCB
20A2 AD AB20       96        LDA CHAR      ; GET CHAR TO BE PRINTED
20A5 20 D2FF       97 PRINT  JSR PRTCHR    ; GO PRINT CHAR IN ACCUMULATOR
                   98 ;
20A8 4C 0D20       99 STEPS JMP STEP1      ; GO BACK AND START LOOP AGAIN
                  100 ;
20AB 00           101 CHAR   DB  £0        ; PLACE TO STORE CHARACTER
20AC AE20         102 LAST   ADDR  *+2     ; DEFNE ADDRESS OF NMI HANDLER
                  103 ;
20AE 20 00FF      104        JSR RFILES    ; SET DEFAULT DEVICES
20B1 4C B9C3      105        JMP BASIC     ; GO BACK TO BASIC WITH "READY"
                  106        END



ADD      2081    79    71
BASIC    0389    10   105
BEGIN    2000    14
CHAR     20AB   101    25    58    64    82    87    90    96
GETCHR   FFE4     6    22    57
INFILE   FFC6     8    36
LAST     20AC   102    15    18
NOKEY    2043    48    27    29    31    33    35    37    39    41
                       43    45
OTFILE   FFC9     7    52
PRINT    20A5    97    92
PRTCHR   FFD2     9    53    97
RFILES   FFCC     5    21    89   104
STEP1    200D    21    62    65    89
STEP2    2013    24
STEP3    2046    51    46
STEP4    204E    55    24    48
STEP5    2059    60
STEP6    205F    64
STEP7    2092    89    37    39    73    75    79    83

STEP8    20AB    99
SUB      2088    84    77
TEST1    206D    70    66


TEST2    207A    76    74

MOS Technology MCS650X Assembler (AN240) done at 16:07:41 on 05-07-91.


0 error(s) detected.

Caradst 106   Symblos:        23   Cost:      £0.09
Punch :   0   References:     46   CPU Time:   0.73
Print : 137   Storage:         6
```

THE COMMODORE COMPUTER

"WHY SPEND $10,000? COMMODORE GIVES YOU WORD PROCESSING PLUS COMPUTING FROM $3,400."*

— William Shatner



You know your office is drowning in paperwork. Letters! Reports! Records! Word processing is the obvious answer. And now, Commodore makes word processing affordable for even the smallest office.

All you need is the Commodore CBM™ 8032 desk-top computer system plus one of our word processing programs. For a modest investment, you get all the features of a sophisticated word processor. You can insert, delete, re-arrange paragraphs, revise as many times as necessary. With no time wasted in hand-typing multiple drafts. Our word processing system displays 80 characters on a line. Text appears on the screen exactly as it will appear in print. And Commodore's screen-editor is the easiest, most versatile to use, with full cursor control.

But that's only the beginning. Because the Commodore CBM is a versatile business computer that can save you more time, work and money than you ever thought possible. Your can use it for Financial Planning, Data Management and Retrieval, Direct Mail Lists, Legal and Medical Accounting and Portfolio Management. And there are programs for Accounts Receivable and Payable. Inventory Control. Payroll. Engineering. Personnel Recordkeeping. And Tax Preparation.

COMPARE THE ALTERNATIVES AND YOU'LL SEE WHY COMMODORE IS THE COMPUTER TO BUY.

	CBM 8032
Price: Under $3,400*	✓
Computer Memory (RAM + ROM): Total = 50K	✓
80 Characters Per Line	✓
Built-in Green Display Screen	✓
Separate Numeric Keypad	✓
Standard Upper & Lower Case Characters	✓
5¼" Dual Disk Storage Capacity — 1,000,000 Characters	✓
National & Local Service/Support	✓
Over 250,000 Desk-Top Computers Sold World-Wide	✓
Over 500 Software Packages	✓
Over 23 Years in Technology	✓

*This chart is based on our standard CBM 8032 Business Series, plus an 8050 Dual Disk Drive. A wide range of printers are available, depending upon your requirements. Commodore offers several Word Processing Packages.

WE WROTE THE BOOK ON SOFTWARE

The Commodore Encyclopedia is a directory of over 500 software programs for business, education, recreation and personal use. Pick up a copy at your local Commodore dealer.

FULL SERVICE, FULL SUPPORT

Commodore dealers throughout the country offer prompt local service. In addition, our new national service contract with TRW provides nationwide support.

Visit your Commodore dealer for a hands-on demonstration of our truly affordable computers that do word processing and so much more.

*This price was configured on our CBM 8032 Business Series and 8050 Dual Disk Drive. A full line of Commodore printers is available, depending upon your requirements.

commodore COMPUTER

CORRECTION!

Converting From 2040 to 4040

In the February issue of this magazine, on page 25, there was a brief article on converting 2040 disks to 4040 format, for use with an upgraded 2040 disk unit or new 4040 disk unit. We have recently been notified of a slight mistake in that article, and would like to make our readers aware of the correction.

In that article, it was stated that: "Please note that once any disk has been formatted by DOS 1.0 it cannot be duplicated, verified or written to, once the DOS 2.1 upgrade kit has been installed. Therefore the conversion to DOS 2.1 must be accomplished before the upgrade kit is installed."

In actuality, a 2040 formatted disk can be read by a 4040 disk drive, but cannot be written to. Sequential and program files can be copied up, however random access files cannot. In any case, any conversion can only be made once the upgrade kit is installed, using the procedure outlined in that article.

WE WANT YOUR FEATURES!

The Commodore Magazine
is seeking features for
use in upcoming issues.
If you have developed
a program, game or
an application that
would be of interest
to our readers
please submit this to:

EDITOR
Commodore Magazine
681 Moore Road
King of Prussia, PA 19406

New Product Developments

In the fast paced microcomputer industry, new products are developed at an incredible rate—sometimes too quickly for even the most informed computer person. To do out part in keeping you abreast of Commodore-related product developments, this section will appear in each issue of *Commodore Magazine*. Its sole purpose will be to announce NEW developments in hardware and software products that are compatible with Commodore equipment. We hope you enjoy this latest addition to the magazine. And, please, if you have a new product that's worth mentioning, don't hesitate to send us the information.

Company: Krell Software Corporation
21 Millbrook Drive
Stony Brook NY 11790
(516) 751-5139

Products (all compatible with PET):

Competency Examination Preparation Series—This program provides a structured, sequential, curriculum for math, reading, and writing instruction.
Price: $1,299.00

College Board S.A.T. 81/82 Preparation Series—An update of Krell's Basic S.A.T. series, from five to 25 programs.
Price: $299.95

Issac Newton: An Introduction to Scientific Logic—An educational game that challenges players to gather evidence and fine the "laws of nature."
Price: $24.95

Odyssey in Time: A Complex Adventure in History —Players must defeat an adversary in combat, while learning about 24 historical time periods.
Price: $39.95

Company: ECX Computer Company
2678 North Main Street #6
Walnut Creek, CA 94596
(415) 944-9277

Products:

MX-910 CBM/PET RAM/ROM — Allows multi ROM protected programs. No need to install protect ROM socket after initial load.
Price: $119.95

MX-232 CBM/PET to RS-232C Interface — Bidirectional, 50 to 19.2K baud rate. Allows for two RS-232C ports. Installs easily.
Price: $149.95

SX-232 CBM/PET RS232C Software — Works with MX-232 to RS-232 Interface and standard RS-232C modem.
Price: $49.95

MX-901 CBM/PET CB2 Sound—Adds sound/music to 2001 and 4001 machines.
Price: $16.95

MX-902 CBM/PET Cassette Sound — Hear programs loading. Finds programs easily.
Price: $16.95

SX-100 IEEE-488 CBM/PET Modem Software — Menu driven.
Price: $49.95

MX-200 IEEE-488 CBM/PET Parity Modem/Software — Allows you to talk to host computer.
Price: $399.95

C101 CBM/PET IEEE-488 To Centronics Parallel Interface — Includes IEEE-488 and parallel cables. Works on Spinwriters and Starwriters.
Price: $129.95

C102 CBM/PET IEEE-488 To Watanabe Plotter Interface — For fast IEEE-488 plotting. Works with "curve" graphics plotting software package.
Price: $295.00

Company: Evans Newton, Inc.
7745 E. Redfield Road, Suite 100
Scottsdale, Arizone 85260

Continued on Page 90.

Everybody Knows . . .

Some things are so well known about the PET/CBM that nobody thinks to say them anymore. In fact, if you're a new PET owner or just didn't happed to be around when everyone else found 'obvious' features, chances are that you still don't know them. These facts are so well known that nobody has bothered telling you about them.

All of the following items apply to all makes of CBM/PET unless specified otherwise.

All PET/CBM computers have both upper/lower case characters (called 'text') mode; and graphics/lower case characters (called 'graphics') mode. You can select text mode with POKE 59468,14 or graphics mode with POKE 59468,12. On newer machines (80 column and Fat 40), you can select text mode with PRINT CHR$(14) and graphics mode with PRINT CHR$(142); in this case the screen display is trimmed up slightly. Not bad when you consider that on some computers, upper/lower case is an expensive extra.

If you buy a 32K PET/CBM computer, you might think that you are getting 32,000 memory locations (called 'bytes'). Wrong on several counts. First of all, a 'K' represents 1,024 bytes of memory, so your total goes up to 32,768. Next, Commodore throws in the screen memory for free—that's an extra 1K on 40-column machines and 2K on 80-column machines that hasn't even been counted. So if you have all that memory, how come a fully fledged 80-column machine with a total of 34,816 bytes tells you '31743 BYTES FREE'? Answer: Because it's telling you how much space is available for your BASIC program; and that doesn't count either screen memory or a number of 'overhead' locations that have been set aside. Final puzzler: how come if you ask for the computer's BASIC free space right away, by typing PRINT FRE(0), you get an answer of 31741? Where did those two bytes go? Hint: BASIC uses a two-byte marker to say, "There's no program beyond this point."

When you press the RETURN key, you go to the next line. More than that: everything on the line you're leaving is received by the computer. That means you can go back to a line on the screen, change it if you wish, and then press RETURN and the whole line will be used. If you want to go to the next line without having the previous line actioned by the PET/CBM, hold down SHIFT as you press the RETURN key.

The PET/CBM has a built-in timer/clock. Type PRINT TI$ and you'll be told how much time has elapsed since you turned the computer on. The TI$ clock works in hours, minutes and seconds, which means you can set the correct time into it. If it's 3:25 pm, just type TI$="152500" and it will keep reasonably good time from that point on. Since it's hard to do arithmetic in hours, minutes and seconds, the computer gives you a second timer that counts in 'juffies': a jiffy is 1/60 of a second. The clock and the jiffy timer are really the same timing device: they both reset to zero when TI$ reaches the 24 hour mark.

Most BASIC commands can be abbreviated. The question mark is a quick substitute for PRINT, but other commands can be abbreviated using a method along the following lines: to print VERIFY the short way, type the V and then hold down the shift key and press the E key. The result may look a little odd, but when you press RETURN, the computer will behave just as if you had typed the full word. Some keywords are not worth abbreviating: IF and TO are too short, for example. Others will give trouble because they start out similarly: GOTO and GOSUB start whith the same two letters, as do NEXT and NEW, POKE and POS, or READ, RETURN, and RESTORE. The computer doesn't know which one you want, so you might pick the wrong one.

The PET/CBM will receive from the keyboard even when there's a program running. It will normally store up to nine characters, holding them in a 'keyboard buffer' until they are needed. If desired, you can pick up keystrokes one at a time while your program is running with a statement like GET X$. X$ will be equal to the key pressed, or to nothing (the null string, written "") if no key has been pressed.

You can clear the screen, home the cursor and/or move the cursor in any direction as part of a PRINT statement. When you type in the statement, the desired screen activity is stored as an odd-looking reversed graphic. Later, when the PRINT statement is executed, the movement take place as planned.

→

New Products cont.

Product (compatible with PET):

The Electronic Classroom Curriculum Orginizer (ECCO) — A program designed to help teachers maintain accurate and detailed reports on the individual progress of students in math and reading. The program was designed by Holt Rinehart and Winston, in conjunction with Evans Newton, Inc.
Price: N/A

Company: Type Share, Inc.
8315 Firestone Blvd.
Downey, CA 90241
(213) 923-9361

Products:
Two copy preparation systems, compatible with the VIC 20 or the CBM 8032, which capture copy on floppy disks (CBM) or cassettes (VIC 20) and code it for typesetting. Users send the disks or cassettes to a Type Share center, where they are input to high speed, computerized phototypesetters to produce finished copy.
Price: Contact Type Share, Inc.

Company: TIS, Inc.
P.O. Box 921
Los Alamos, NM 87544
(505) 455-7049

Product:
"Understanding Your VIC, Volume 1: Basic Programming — A 148-page book that uses a step-by-step approach to help the beginner learn about the VIC 20. The book includes exercises and examples.
Price: $13.95

Company: Robert D. Vernon
RDVDS
P.O. Box 5547
Richmond, VA 23220
(804) 355-7587

Product:
Production System for Architects and Engineers (PSAE) recently licensed RDVDS to distribute "MASTERSPEC 2 — Basic Version." MASTERSPEC is a master specification subscription service for architects and engineers. MASTERSPEC is stored on diskettes that run on a CBM 8032.

Price: The cost is $575 for a subscription, including three updates. The cost thereafter is $185 per year. A demonstration disk for Commodore dealers is available for $25.00. ■

Everybody Knows...cont.

The mechanism is called "programmed cursor" and it's very good for screen animation.

The newer PET/CBM units contain a sounding device, but in any case you can get sound signals from the Parallel User Port. That's the middle of the three connectors at the back, and you should connect pins M and N to your amplifier input and ground respectively. You need an amplifier—your stereo will do—since the signal isn't strong enough to drive a speaker directly. You generate sound from your program in the following manner: POKE 59467,16 turns the sound capability on — before your program stops, you must turn it back off with POKE 59467,0. POKE 59466,15 sets the tone — 15 is a good number but you can try others. POKE 59464,100 sets the pitch — any value from 10 to 255 is audible. Don't forget: when you're finished, turn it off with POKE 59467,0 or you may have trouble with input/output.

All machines, except the first (Original ROM), have a built-in Machine Language Monitor. You can get to it with SYS 4. This should produce about three lines of cryptic information. Return to Basic by typing X and RETURN. Later, you may learn to do marvelous things with the MLM — to start, just know it's there.

It's sometimes useful to know if someone is holding a key down. You can discover this by using the function PEEK(151) ...On Original ROM machines it's PEEK(515). If the value you get is 255, nobody is touching a key at the moment. If it's anything else, a key is being held. Don't try to use the value if it's lower than 255—the numbers will vary from machine to machine—it's better to use GET to find out which key has been pressed.

Are there other features that everybody knows? I'm sure there are lots of others . . .the problem is, they are so obvious I can't think of them . . .

Jim Butterfield, a highly recognized expert on Commodore hardware and software, has been involved in the microcomputing field since 1966. ■

PROJECTIONS & REFLECTIONS

If you fail to plan, you plan to fail. Planning the growth and direction of Commodore's Software Department consumes the time of several individuals. The market for software is continually evolving into the single largest impetus to purchase a personal computer today. It is this evolution of better-informed persons of high expectations that must be nurtured and supported.

Only a few years ago, personal computing was a technological impossibility, and now, if one does not have a micro, they are risking being out of step. What once cost several thousand dollars a month on a lease plan can now be purchased for less than one month's lease payment. The computing power curve is becoming the inverse of the price curve as capacity and functional capability continue to increase at an equal rate. The cost of software in the very near future will become upwards of 72 percent the cost of the hardware because of these curves.

Satisfying the general software needs of the public becomes an ever-increasing task. To address this need, Commodore is going to be the only major microcomputer manufacturer to produce and promote an entire line of vertical market packages along with the more standard horizontal-appeal products. This software runs from statistical business analysis to German agricultural programs for the farming community.

Our fight for market share is not totally dependent upon software, but it certainly figures high in our list or priorities. The crest is forming for the wave of the technological future, and Commodore intends to ride all the way on top. To accomplish this we are committed to bring you possibly the largest and most varied offering of software available today. There are bound to be certain mistakes and set-backs, but to us, commitment is where it's at. There is a certain story about having bacon and eggs for breakfast: the chicken contributed . . . but the pig had what we call "total commitment!"

In addition to our current vertical markets of legal and medical, we are working on dental, veterinary, service firm, trucking, general contractor, insurance, real estate management, agriculture, and off-track betting, to name a few. We are currently soliciting software for critique, or have contracted to have the programming done in all of these areas. The future here at Commodore looks to be a reversal of sorts from having little software available to possibly having more than our distribution network can absorb.

In addition to the verticals that we are handling, new developmental tools are going to be distributed along with quite an offering of "public domain" software. All of these require time, planning, and a little luck.

We thank all of you for the support that we have received so far! Please look for our upcoming series of products.

—Paul Goheen
Software Product Manager

COMMODORE BUSINESS MACHINES, INC.
PRODUCT SPECIFICATIONS

VIC 20
$299.95*

Screen size: 22 char. x 23 lines
Memory: 5K expandable to 32K
User supplied color monitor or TV
4 internal amplifiers including:
- 5 octaves total range
- 3-tone (music) generators
- 1 sound effects generator

Peripherals include: cassette recorder, printer, modem, disk drive, and IEEE-488 interface, RS-232 interface, memory expansion module, 3K-8K-16K expansion cartridges
Features:
- Programmable function keys
- Standard PET BASIC — upwardly compatible to other CBM computers
- Full-size typewriter keyboard
- Graphics character set, upper and lower case letters
- Plug-in program/memory cartridges
- Low-priced peripherals
- Joystick/paddles/lightpen
- Self-teaching materials
- 16 colors
- High resolution graphics (176 x 184 dots)
- Programmable characters
- Full screen editing

*Includes modulator

PET®
$995.00

Screen size: 40 char. x 25 lines
Memory: 16K to 32K
IEEE-488 bus for disk, printer, modem and other intelligent peripherals
Eight-bit parallel user port with "handshake" lines
Supports Commodore C2N cassette and disk unit
Features:
- 128 ASCII plus 128 graphic characters
- 74-key professional keyboard
- Shift key gives 64 graphic characters
- Separate calculator/numeric pad
- Full screen editing capabilities
- 18K of ROM contains BASIC (version 4.0) with 9-digit floating binary arithmetic

Grants:
- 3 for 2 offer available for educational use.

CBM™ 8032
(Business Machine)
$1495.00

Screen size: 80 char. x 25 lines
Memory: 32K expandable to 96K
IEEE-488 bus for disk, printer, modem and other intelligent peripherals
Business software includes:
- Word processing
- VISICALC
- Inventory control
- Accounts receivable
- Accounts payable
- Payroll
- Dow Jones Portfolio Management
- Personnel files

Features:
- 128 ASCII, plus 128 graphics characters
- 73-key professional keyboard
- Separate calculator/numeric pad
- Full screen editing capabilities
- 18K of ROM contains BASIC (version 4.0) with direct (interactive) and program modes
- 9-digit floating binary arithmetic

Grants:
- 3 for 2 offer available for educational use.

SuperPET Computer
$1995.00

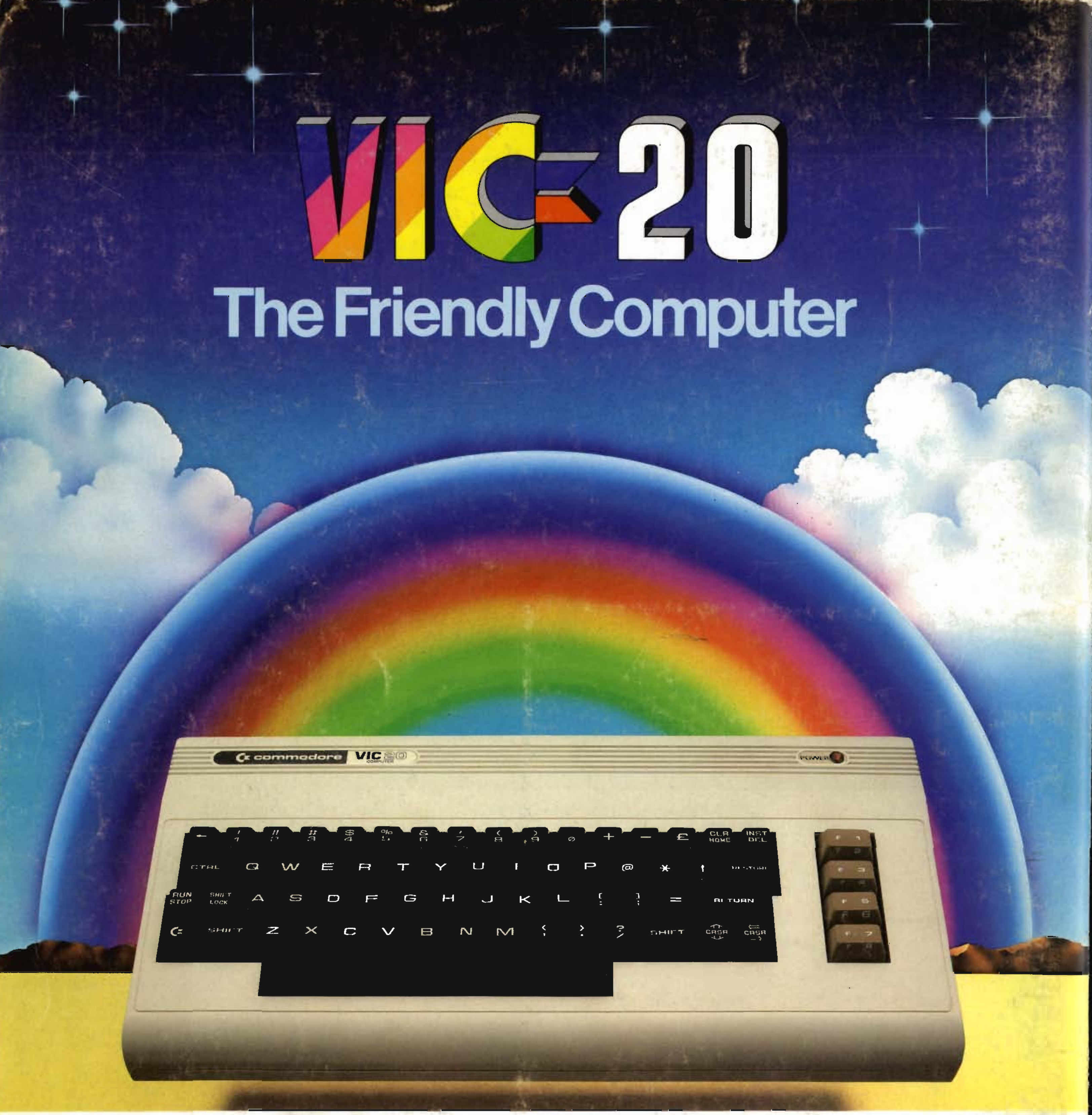
Screen size: 80 char. x 25 lines
Memory: 96K RAM (64K bank switched) IEEE-488 bus for disk, printer, modem and other intelligent peripherals High-speed RS232-C interface for additional compatibility with printer and modem
Communicates to Mainframes*
Compatible with 8032 software
2 microprocessors - 6502 and a 6809
Software includes: language interpreters, editor, operating system (supervisor), and an assembly language development system
- BASIC 4.0
- Waterloo MicroBASIC
- Waterloo MicroPascal
- Waterloo MicroFORTRAN
- Waterloo MicroAPL
- 6809 Assembler Development System, Linker/Loader

Features:
- 128 ASCII, plus 128 graphic characters, IBM/ACM, APL character set
- 73-key professional keyboard
- Separate calculator/number keyboard (line/text editor keypad usage in 6809 mode)
- Screen editing capabilities
- 38K of ROM contains BASIC (version 4.0) with direct (interactive) and program modes and 9-digit floating binary arithmetic

*Program file transfer between suitably equipped Mainframe. (Program upload/ download between M/F and SuperPET). Programs will execute without modification.

commodore
COMPUTER

Commodore Business Machines, Inc., Computer Systems Division, 681 Moore Rd., King of Prussia, Pa. 19406 (215) 337-7100



Commodore Business Machines, Inc.
681 Moore Road
King of Prussia, Pa. 19406

address correction requested

BULK RATE
U. S. Postage Paid
PERMIT No. 267
King of Prussia, Pa.