20 Complete, Ready-To-Run Programs Inside

COMPUTE!

$2.95
January
1985
Issue 56
Vol. 7, No. 1
$3.50 Canada
02193
ISSN 0194-347X

The Leading Magazine Of Home, Educational, And Recreational Computing

TURBOTAPE: A MAJOR BREAKTHROUGH!

Save And Load Cassettes At Disk Speeds
Programs Inside For Commodore 64 & VIC-20

Music In The Computer Age: Will Computers Replace Musicians?

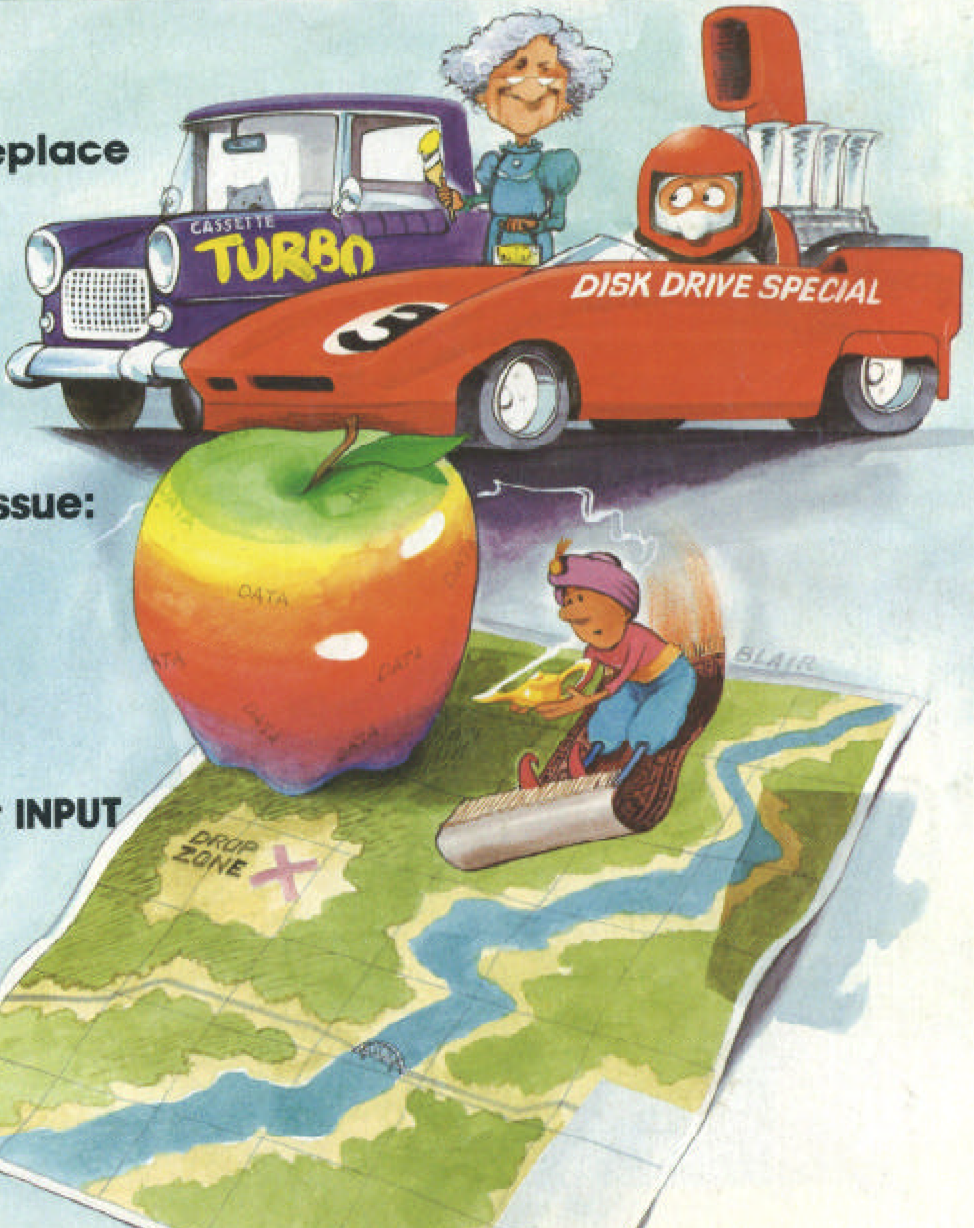
Random Access DATA Statements For Apple

Also Free In This Issue:
A Powerful Terminal Program For Atari

Two Action Games: Rescue Of Blondell, Paratrooper

Enhanced Applesoft INPUT
IBM Chartmaker

And Much More



January 1985 Vol. 7, No. 1

GUIDE TO ARTICLES AND PROGRAMS

FEATURES

31 Music In The Computer Age ..... Richard Mansfield — *
51 Now-Silent Beethovens ..... Richard Mansfield — *
52 MSX Is Coming, Part 2: Inside MSX ..... Tom R. Halfhill — *

EDUCATION AND RECREATION

61 Paratrooper ..... John Goetz — AT/TI/64/V/+4/16/AP/PC/PCjr
80 Rescue Of Blondell ..... Grant Albrecht — AT/64/V
99 Guitar Tuner ..... Christopher Visco — TI/64/+4/16/AT/PC/PCjr

REVIEWS

104 Sequential Circuits Music Sequencer For Commodore 64 ..... Richard Mansfield — 64
108 Sunburst Educational Software ..... Glen M. Kleiman And Susan Keyes — AT/AP/64/PC/PCjr/CC

COLUMNS AND DEPARTMENTS

6 The Editor's Notes ..... Robert Lock — *
10 Readers' Feedback ..... The Editors and Readers of COMPUTE! — *
42 Computers And Society: 1984 Revisited ..... David D. Thornburg — *
44 The World Inside The Computer: Our Computer Handyman ..... Fred D'Ignazio — *
114 The Beginner's Page: Which Computer Language Is Best? ..... Tom R. Halfhill — *
135 INSIGHT: Atari ..... Bill Wilkinson — AT
138 IBM Personal Computing: Music For Amateurs ..... Donald B. Trivette — PC/PCjr
141 Programming The TI: Mixing Graphics And Music ..... C. Regena — TI
158 Machine Language: Multiplication, Part 1 ..... Jim Butterfield — *
160 Telecomputing Today: Smokey & The Modem, Part 8086 ..... Arlan R. Levitan — *

THE JOURNAL

124 TurboTape: High-Speed Tape Utility For Commodore 64 And VIC-20 ..... Harrie De Ceukelaire — 64/V
131 Enhanced Applesoft INPUT ..... Dale W. Woolridge — AP
145 JTERM For Atari ..... Frank C. Jones — AT
152 IBM Pie Chart Maker ..... Michael Posner — PC/PCjr
156 Random Access DATA Statements For Apple ..... Robert Jacques Beck — AP

40 COMPUTE!'s Author Guide
113 CAPUTE! Modifications Or Corrections To Previous Articles
163 COMPUTE!'s Guide To Typing In Programs
166 News & Products
168 Product Mart
176 Advertisers Index

NOTE: See page 163 before typing in programs.

AP Apple, **Mac** Macintosh, **AT** Atari, **V** VIC-20, **64** Commodore 64, **+4** Commodore Plus/4, **16** Commodore 16, **P** PET/CBM, **TI** Texas Instruments, **PC** IBM PC, **PCjr** IBM PCjr, **CC** Radio Shack Color Computer.
*All or several of the above.

TOLL FREE Subscription Order Line
800-334-0868 (In NC 919-275-9809)

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies

One of the ABC Publishing Companies:
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published monthly by COMPUTE! Publications, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919) 275-9809. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408. Domestic Subscriptions: 12 issues, $24. Send subscription orders or change of address (P.O. form 3579) to **COMPUTE!** Magazine, P.O. Box 914, Farmingdale, NY 11737. Second class postage paid at Greensboro, NC 27403 and additional mailing offices. Entire contents copyright © 1984 by COMPUTE! Publications, Inc. All rights reserved, ISSN 0194-357X.

calculating for you. In addition, the image of the sprite pattern is superimposed on the video output of the computer, so the pattern is not actually moved through memory. That means a sprite can seem to move above or beneath other screen images—including other sprites—without disturbing them. What's more, the computer knows when a sprite is touching another object. That's important if you're writing a game, because your program can keep track of these collisions and respond accordingly.

You probably won't find any mention of Atari player/missile graphics in the manuals which came with your 600XL. In fact, player/missile graphics was an undocumented feature when the Atari computer first hit the market in 1979–1980. The first article revealing its existence—written by Atari programmer Chris Crawford—appeared in the January 1981 issue of COMPUTE!*. This issue is out of print, but the article is reprinted in* COMPUTE!'s First Book of Atari. *More detailed information on programming player/missile graphics can be found in* COMPUTE!'s First Book of Atari Graphics *and* COMPUTE!'s Second Book of Atari Graphics.

Future Of The VIC

Will Commodore discontinue the VIC-20? And if so, will the company still make software and hardware for the VIC-20s that are out there?

Paul Fowlie

The Commodore 16, announced in January 1984 and first marketed in October, replaces the VIC-20 as Commodore's entry-level home computer. By last June Commodore had stopped producing the VIC. Although more than two million VICs have been sold worldwide, Commodore obviously feels that the $100 Commodore 16 is a better value for beginners and also helps promote the company's marketing strategy. The Commodore 16 is essentially a Plus/4 with 16K instead of 64K RAM and no built-in software or modem port. It is upwardly compatible with the Plus/4, not true with the VIC and the Commodore 64.

As early as the Winter Consumer Electronics Show (CES) in January 1984, it was apparent that fewer companies were producing software for the VIC. There was even less software at the Summer CES in June. This doesn't mean that everyone is abandoning the VIC overnight. The installed base is still very large. But it will become increasingly difficult to find new products aimed at the VIC-20—and that includes products from Commodore. Because the peripherals are largely compatible, many people have upgraded from the VIC to a 64.

One high-ranking Commodore executive told us that if someone wants to buy a hundred thousand VIC-20s, Commodore could sell them. In other words, there are plenty of VICs still around, but the company is not planning to market them in competition with its own new machines. The same official told us, however, that owners of VICs who need help will be supported by Commodore. "We have spares. We have everything. If people have a problem, we will fix it, repair it—no problem."

COMPUTE! *will continue covering the VIC-20 as long as there is sufficient reader demand. There are still many thousands of VIC users among our readers.*

TI Peripherals

I noticed an inquiry in "Readers' Feedback" in the October 1984 issue of COMPUTE! regarding the availability of the Peripheral Expansion System and its associated plug-ins. Texas Instruments has a toll-free number (1-800-842-2737) for TI users with questions about product availability.

TI also has a list of third-party suppliers available. Some of them even make products that TI never got around to offering.

Randall L. Powell

Thanks for the information. We received numerous letters informing us of various third-party suppliers for TI peripherals, including alternate expansion systems, peripherals that work without any expansion system, and even leftover supplies of TI's own expansion box and cards. These are available mainly through mail-order outlets. In most areas it has become impossible to find any peripherals for the TI-99/4A in local stores.

Tamea Rector, advertising/marketing director of Tenex Computer Express, also sent us a copy of the company's 48-page catalog of TI products. To get a free copy, write to:

Tenex Computer Express
P.O. Box 6578
South Bend, IN 46660

Cool Computing

I own a Commodore 64, and when I use it for a long time—mostly in the summer—funny-looking waves appear on the screen and scroll downward. After that, the waves get bigger and bigger, the computer starts printing characters all over the screen, and the keyboard won't operate. Is there any way to stop these annoying waves?

Paul Mantsch

It sounds like a classic case of overheating. Computer chips are designed to operate within a specified range of temperatures. For example, the VIC-II video chip in your 64 is rated to function normally between 32° and 158°F (0°–70°C). At the high end of their rated ranges, chips can start acting *strangely, and if a particular chip isn't quite up to specs, the bizarre behavior can begin to show up at lower temperatures. While it's unlikely that your room temperature is reaching 150°, it could get that hot inside the computer's plastic housing, since all chips emit heat as they operate.*

There are a couple of possible solutions. First, make sure the ventilation slots on the underside of the computer and the expansion slots on the back panel aren't obstructed. If that's no problem, perhaps you can set up a table fan to keep air circulating over the computer on hot summer days (it'll help keep you cool, too).

Still no results? A more drastic solution is to remove the foil shell which covers the circuit boards of newer 64s. The foil is designed to reduce RF (Radio Frequency) interference, but it also traps heat. Carefully remove the foil shell and see if this solves the problem. (Unfortunately, removing the foil voids your warranty and may also cause more video interference with nearby TV sets.)

Another alternative is to have your computer checked out by a qualified service technician. Perhaps a slightly defective chip is responsible for the overheating.

Named Subroutines In Microsoft BASIC

Microsoft BASIC supports named subroutines. Sort of. The following construction is legal:

GOSUB1200, EVALUATE:IF X=0 THEN PRINT "WHOOPEE!!"

After executing the GOSUB, BASIC returns to the end of the GOSUB line number and looks for the next colon or the beginning of a line. All else is ignored.

This is more useful than a REM, since you can place additional statements on the same line and it saves a byte of memory. It works on the Commodore PET, 64, and VIC computers.

Bill Baldock

Thanks for the tip. This may also work with other machines using Microsoft BASIC, but try it out before embedding it in a crucial program.

Storing Text On Disk

Can a disk drive store text by page?

John B. Gentilucci

Disk files can contain any information you want. However, trying to store a text file by pages would be a time-consuming and inefficient use of disk space. Most word processors allow you to set up limits for page size and also will automatically paginate the printout. You'll find it much easier to store files by chapter or subheadings, and let your computer keep track of the pages when printing the text. This way you'll also be able to make revisions without restructuring your files because of a change in page sizes.

Reading TI Joysticks

I built the joystick adapter presented in "Readers' Feedback" of the August 1983 issue for my TI-99/4A and revised it as suggested in a later issue. I have several questions about the use of joysticks with the TI. First, how do you detect when the fire buttons are being pressed? And second, how do you achieve simultaneous joystick movement?

Matt Phillips

The fire buttons are detected with the CALL KEY statement on the TI. The format is:

CALL KEY(unit,key,status)

where unit is 1 or 2 for the joystick number. When a fire button is pressed, KEY takes on a value of 18. Ordinarily the key value is 0.

You can also detect firing with the STATUS variable. The STATUS variable can have a value of 0, −1, or +1. STATUS is 0 if the fire button is not pressed, −1 if the fire button is still being pressed since the last CALL KEY, and +1 if the fire button was not pressed at the last CALL KEY, but is presently being pressed.

There's no such thing as true simultaneous joystick movement on the TI or any other computer. Instead, you create the illusion of simultaneity by alternately checking the joysticks very quickly. The following sample program demonstrates one method of doing this and also illustrates use of the fire button. This program lets you move two figures around the screen with the joysticks. Joystick 1 moves a stick man figure, while joystick 2 moves a ball-shaped figure. Pressing the fire button changes the color of the respective figures.

```
1Ø REM TWO JOYSTICK DEMO
2Ø CALL CHAR(47,"1818423C183C4242")
3Ø CALL CHAR(48,"ØØ3C7E7E7E7E7E3C")
4Ø X(1)=15
5Ø Y(1)=11
6Ø Y(2)=11
7Ø X(2)=17
8Ø C(1)=13
9Ø C(2)=14
1ØØ CALL COLOR(2,C(1),1)
11Ø CALL COLOR(3,C(2),1)
12Ø CALL CLEAR
13Ø CALL SCREEN(15)
14Ø FOR I=1 TO 2
15Ø CALL JOYST(I,DX,DY)
16Ø CALL KEY(I,K,S)
17Ø IF K<>18 THEN 2ØØ
18Ø C(I)=C(I)+1+(C(I)=16)*15
19Ø CALL COLOR(I+1,C(I),1)
2ØØ CALL HCHAR(Y(I),X(I),32)
21Ø X(I)=X(I)+DX/4
22Ø Y(I)=Y(I)-DY/4
```

```
230 X(I)=INT(32*((X(I)-1)/32-INT((X
    (I)-1)/32)))+1
240 Y(I)=INT(24*((Y(I)-1)/24-INT((Y
    (I)-1)/24)))+1
250 CALL HCHAR(Y(I),X(I),46+I)
260 NEXT I
270 GOTO 140
```

In this program, each joystick is checked for movement (line 150) and firing (line 160) within a FOR-NEXT loop. If a fire button is being pressed (K equals 18), the program executes a routine to change the color of the appropriate figure (lines 180–190). The old figures are then erased (line 200), new positions calculated (lines 230–240), and new figures drawn (line 250).

80-Column VIC?

I own a VIC-20 which I use with a TV set. I have seen ads for monitors with 40 or 80 columns. If I were to buy one of these monitors, would my VIC-20 display 40 or 80 columns? If so, would it change the screen memory?

Allen Murphy

Unfortunately, changing the display format of your computer isn't that simple. A video monitor or TV displays exactly what the computer tells it to display. The VIC generates a video signal for a picture consisting of 23 rows of characters with 22 characters per row, and 22 characters is what you see no matter whether you send that signal to a TV, a monochrome monitor, or a color monitor. The 40- or 80-column figure you mention is only the manufacturer's rating of the number of characters per row that the monitor is capable of displaying clearly—a measure of the resolution *of the monitor.*

A monitor that gives a good 80-column display should give an exceptionally crisp 22-column display when connected to a VIC. To actually get an 80-column display, you'd have to use one of the 80-column video adapter boards available for the VIC. The adapter would indeed change screen memory, and you'd probably be disappointed to learn that little of your favorite software would work with the 80-column adapter.

80-Column Atari?

I have an Atari 1200XL, a Rana 1000 disk drive, and am using a TV set as a monitor. Would I need to expand the text field to 80 columns to accommodate a letter-quality printer?

Shawn Johnson

This isn't necessary. An 80-column video adapter board is nice to have when you're using a word processor to prepare a document because the screen can show how the document will appear on paper. It's not required, however, because the word processor allows you to specify any width for printing—including 80 or even 132 columns (if your software and printer can handle this). The size and format of the video display does not limit your choice of a printer.

You should also be aware that most TV sets cannot adequately display 80 characters per line; the characters will usually be much too fuzzy to read. You would need to buy a monochrome computer monitor. In addition, we haven't heard of any 80-column adapters for the 1200XL, and it's not likely that any will be sold. Unlike other Atari computers, including the 600XL and 800XL, the 1200XL has no expansion slot.

BASIC To Machine Language

I have a VIC and am currently learning machine language. How can I pass BASIC variables to an ML subroutine?

David P. Ballin

One of the easiest ways to transfer numbers between BASIC and machine language is to store them in memory. Safe memory locations can be used like post office boxes—BASIC can POKE the mail into the boxes, and machine language can pick it up, or vice versa. Here's an example:
In BASIC:

```
300 A=57
310 POKE 251,A
320 SYS 4096
```

In machine language:

```
$1000 CLC
$1001 LDA $FB ;get the value POKEd into 251
```

Of course, this assumes that location 251 is unused for anything else. Now, here's the reverse (transferring data back to BASIC):
In machine language:

```
$1C49 STA $FB ;store the accumulator value into loca-
               tion 251 ($FB)
$1C4B RTS
```

In BASIC:

```
500 A=PEEK(251)
```

With a single POKE you can transfer values in the range of 0 to 255 back and forth. If you want to transfer values larger than 255, use the following formula (where N is the number to be stored):

```
NN=INT(N/256):POKE byte1,N-(NN*256):POKE
byte2,NN
```

This method breaks the value of N into two bytes. The value in memory location byte 1 is the remainder after the integer division of N by 256. The quotient is placed in the following memory location, byte2. The bytes are stored low (least significant) byte first, then high (most significant) byte, a 6502 standard for two-byte numbers. Some good areas for temporary data storage on the VIC *are locations 679–767, 828–1019 (the cassette buffer), and 251–254 (free zero page locations). The same locations are available on the Commodore 64, plus 4K of free RAM at 49152–53247.*

You can also load the accumulator, X, and Y registers from BASIC on a VIC or 64 with the POKE statement. The accumulator is stored in 780 ($30C), the X register in 781 ($30D), the Y register in 782 ($30E), and the status register, P, in 783 ($30F).

Before a SYS statement in BASIC passes control to the SYS address, each register is loaded with the value found in the corresponding storage address. After the ML program finishes execution and returns to BASIC with the RTS instruction, the new value of each register is stored in the appropriate location. This is true only of SYS, not the USR function.

A useful application of this would be formatting the screen by using Kernal routines from BASIC. For instance:

```
POKE781,10:POKE782,5:POKE783,0:SYS65520:PRINT
"HELLO"
```

This prints "HELLO" at row 10, column 5. This line will work on both the VIC and 64, as the PLOT routine is entered via the Kernal jump table.

Another, more tricky way to pass a single value back and forth between BASIC and ML is with the USR function. Like any function, it looks for a value in parentheses. This value is passed to the machine language program. And like any function, it returns a value. A=USR(B) would pass the value of B to the machine language program, which can then pass back a value to be stored into A.

For more information, see Mapping the VIC, Mapping the Commodore 64, *or any of the machine language books from COMPUTE! Books.*

TI CALL Destroy?

I own a TI-99/4A computer and have been using the CALL statement to do various tasks. I have heard that certain commands can burn out chips. Is this true? What can I do to avoid damaging my computer?

Robert Brower

We've heard many stories about how various programs or copyright protection schemes are able to destroy monitors, disk drives, and computers by some devious means. It's true that on some late-model Commodore PETs, a certain POKE would sometimes cause an interface chip to race out of control and out of sync, burning itself out. But this small possibility was highly exaggerated. Likewise, it was once said that cranking up the volume too high in Atari BASIC SOUND statements would burn out the sound chip, but our tests failed to validate this rumor.

As a general rule, no program or command can permanently alter or damage your computer. The worst that can happen is a lockup or system crash: The computer refuses to acknowledge any command from the keyboard. To regain control, you must turn off the computer, then turn it back on again. Of course, any program stored in memory is gone. So if there's a chance the program you're typing in or working on could lock up the computer, be sure to save it before running it.

Atari BASIC AUTORUN

How can I automatically run a BASIC program?

David Lanese

The Atari Disk Operating System (DOS 2.0 and 3.0) has a feature that lets you automatically load and run a machine language program from disk whenever the computer is turned on. This feature can be adapted to run a program written in BASIC.

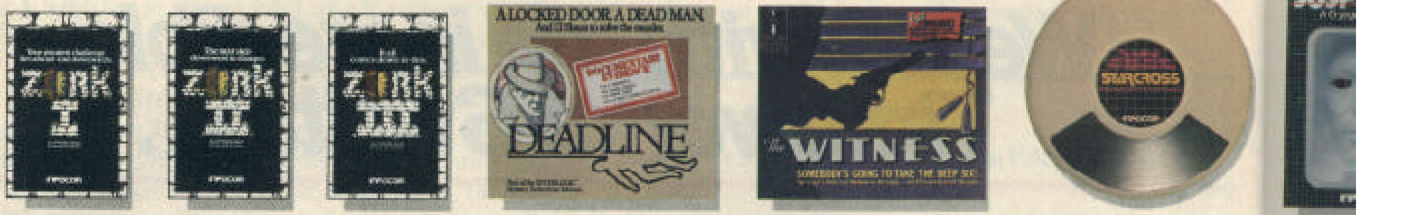
Here's a short BASIC loader for a machine language program which tells the system on powerup to run a BASIC program named AUTORUN.BAS from disk:

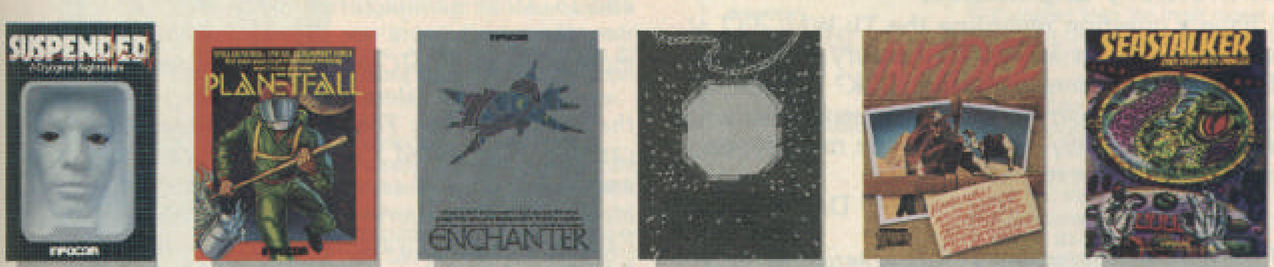
```
CE 10  OPEN #4,8,0,"D1:AUTORUN.SYS"
BA 20  FOR I=1 TO 94
MA 30  READ A
CB 40  PUT #4,A
ON 50  NEXT I
DD 60  CLOSE #4
DO 70  END
AB 80  DATA 255,255,0,6,81,6,216,24
       ,173,48,2,105,4,133,204,173,
       49,2,105,0,133,205,24,160,0,
       177,204,105,162,133,212
OO 90  DATA 160,1,177,204,105,0,133
       ,213,160,32,185,49,6,145,212
       ,136,208,248,169,13,141,74,3
       ,96,0,48,47,43,37,0,24
NI 100  DATA 20,18,12,17,18,26,50,5
        3,46,0,2,36,17,26,33,53,52,
        47,50,53,46,14,34,33,51,2,2
        26,2,227,2,0,6
```

This program, written by Michael E. Hepner, originally appeared in the January 1984 issue of COMPUTE!. *It creates a machine language program on your disk with the filename AUTORUN.SYS. When the computer is turned on, the operating system loads DOS from disk, then runs AUTORUN.SYS if it finds such a program on the disk.*

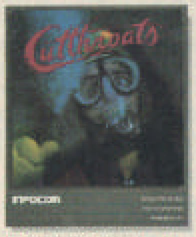
To automatically load and run your BASIC program, store it on the same disk with the filename AUTORUN.BAS. Of course, only one program per disk can be automatically run using this method.

Another approach using the program above would be to enter the Atari version of "Super Directory" (COMPUTE!, *April 1984) and save it as AUTORUN.BAS on each disk. Then every time you turn on your computer, the boot process ends with Super Directory running and a directory of that disk*

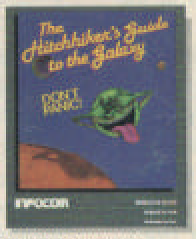

ZORK I
INFOCOM
ZORK II
INFOCOM
ZORK III
INFOCOM
A LOCKED DOOR. A DEAD MAN.
And 12 Hours to solve the murder.
DEADLINE
WITNESS
SOMEBODY'S GOING TO TAKE THE DEEP SIX
STARCROSS
INFOCOM
SUSP



AND NOW FOR SOMETHING INCOMPLETELY DIFFERENT!



In CUTTHROATS,™ the plot involves a motley band of hardbitten salts who get wind of a shipwreck laden with sunken treasure near the remote island where you live. In exchange for your diving skills, they offer you a piece of the action. Your challenge: survive them, the perils of the deep, and escape with the treasure and your life. Good luck!



THE HITCHHIKER'S GUIDE TO THE GALAXY™ by Douglas Adams is the most mind-boggling story we've ever published. In the person of Arthur Dent, you'll chortle as your planet is demolished. You'll yelp with laughter as your life is threatened by a galaxy of horrors. Your sides will positively split as you search the universe for . . . well, you'll find out. Maybe.

In SUSPECT,™ our newest mystery thriller, you're a reporter who gets the scoop on the society event of the year—the murder of a Maryland Blue Blood at a fancy costume ball. Great! Except you're the prime suspect. And if you can't find the real killer, your next by-line could be in the obituaries.

Incomplete, yes. But it's not just because we're always bringing out new stories in the Infocom interactive fiction collection. Nor is it simply due to the fact that with all the writing and re-writing, honing and perfecting that we put into every one of our stories, our work is seemingly never done.

The real reason is: an Infocom work of fiction can never be complete until you become a part of it.

You see, as hard as we work at perfecting our stories, we always leave out one essential element—the main character. And that's where you enter in.

Once you've got Infocom's interactive fiction in your computer, you experience something akin to waking up inside a novel. You find yourself at the center of an exciting plot that continually challenges you with surprising twists, unique characters (many of whom possess extraordinarily developed personalities), and original, logical, often hilarious puzzles. Communication is carried on in the same way as it is in a novel—in prose. And interaction is easy—you type in full English sentences.

But there is this key difference between our tales and conventional novels: Infocom's interactive fiction is active, not passive. The course of events is shaped by the actions you choose to take. And you enjoy enormous freedom in your choice of actions—you have hundreds, even thousands of alternatives at every step. In fact, an Infocom interactive story is roughly the length of a short novel in content, but because you're actively engaged in the plot, your adventure can last for weeks and months.

In other words, only you can complete the works of Infocom, Inc. Because they're stories that grow out of your imagination.

Find out what it's like to get inside a story. Get one from Infocom. Because with Infocom's interactive fiction, there's room for you on every disk.

INFOCOM™

Infocom, Inc., 55 Wheeler Street, Cambridge, MA 02138

For your: Apple II, Atari, Commodore 64, CP/M8*, DECmate, DEC Rainbow, DEC RT-11, IBM PC* and PCjr, KAYPRO II, MS-DOS 2.0* NEC APC, NEC PC-8000, Osborne, Tandy 2000, TI Professional, TI 99/4A, TRS-80 Models I and III.

*Use the IBM PC version for your Compaq, and the MS-DOS 2.0 version for your Wang or Mindset.

CUTTHROATS and SUSPECT are trademarks of Infocom, Inc. THE HITCHHIKER'S GUIDE TO THE GALAXY is a trademark of Douglas Adams.

displayed on your screen. Or you could have the program AUTORUN.BAS chain to any other program you desire.

TI Memory Expansion

I have a question regarding the TI: Why do I always see ads for 32K RAM memory expansion, but never anything more than 32K? Is there any way I could construct a memory expansion with 48K for my TI-99/4A, or does the microprocessor just ignore any extra memory?

David Edwards

Like most microprocessors of its generation, the TI-9900 microprocessor in the TI-99/4 and 99/4A can only address directly a maximum of 64K (65536) memory locations. These locations can't all be used for RAM, since the microprocessor must also have some permanent memory (ROM) to hold its operating system. Still more addresses are required to allow the microprocessor to communicate with the various input/output support chips and peripherals. And the ROM for the built-in BASIC language occupies another large chunk of address space. When all these features are added, only 32K of address space remains free for future memory expansion, which is why no expanders larger than 32K are available.

Note that the 16K of RAM built into the TI-99 console is not directly connected to the microprocessor, and doesn't occupy any of its address space. That memory is part of the VDP (Video Display Processor) chip's address space, and the microprocessor can access it only indirectly, via the VDP. TI's built-in BASIC is designed to access only this VDP memory, which is one of the reasons it's comparatively slow. It also explains why standard TI BASIC can't use any expansion memory connected to the microprocessor. (VDP memory can't be expanded beyond the 16K provided.) To make use of the 32K expanded memory, you need TI Extended BASIC *or some other command module.*

Apple & Atari ML Monitor

I use both an Atari 800XL and an Apple IIe. It's very simple to enter the monitor on the Apple: Just enter CALL −151. Is there a simple method like this on the Atari?

James J. Brennan, Jr.

No, because the Atari does not have a built-in machine language monitor. Few personal computers designed since the late 1970s include ML monitors, since manufacturers feel that only a minority of owners are interested in ML programming and monitors take up valuable ROM space. The Apple IIe and IIc retain an ML monitor because they are enhanced versions of the Apple II, originally designed as a kit-built computer for hobbyists in 1976. The Commodore PET, introduced in 1977, also incorporates an ML monitor. But since then, the only computers introduced for the mass market with a built-in monitor have been the Commodore Plus/4 and 16. Most manufacturers today prefer to eliminate the monitor and use the extra ROM space for a more powerful BASIC or operating system.

Excellent monitors are available separately for the Atari, however. The Atari Assembler Editor *cartridge, Optimized Systems Software's* EASMD *and* MAC/65*, and several other commercial assemblers include monitors.* The Monkey Wrench, *by Eastern House Software, adds several commands to BASIC and includes a Commodore-style monitor that you can call from BASIC. However, it works only in the right cartridge slot of an Atari 800, not with the 800XL.*

POKEing Around

I'm a new ML programmer and would like to know what are the numbers you POKE into memory when entering the machine language parts of some BASIC programs?

Kenny Sumrall

Those numbers are the actual object code (the opcodes and operands) of the machine language program. Each machine language instruction has a value (opcode). This value is what the processor sees and executes.

After you write and debug your machine language program, you can use a utility program to turn the object code into a series of DATA statements. The BASIC program POKEs the numbers into memory, and they can then be executed with a SYS, USR, or CALL statement.

VIC Sound

I own a VIC-20 and use a video monitor instead of a TV. However, the monitor is video only, so I can't hear the sounds in my programs. My monitor cable has an audio output plug, but no one—not even Commodore—has been able to give me exact instructions on how to interface for audio. I have been told I need a high-impedance audio amplifier, but have been given no definition of what that means.

Bob Sterzenbach

If you have a home stereo system, you probably have the high-impedance audio amplifier you need. Simply plug the audio output jack on your monitor cable into the auxiliary input jack of your stereo (use an extension cable if necessary). You might also want to use a Y-adapter, which feeds the single input from the computer into both of your stereo inputs. This should provide superb sound quality. As

6. The
ncor-
only
with a
Plus/4
elimi-
ace for a

ately for
ditor
SMD
l assem-
ch, by
ands to
nitor
works
800, not

ke to
into
guage

Sumrall
the
uage pro-
has a
essor sees

ne lan-
am to
state-
bers into
h a SYS,

instead
only, so I
My mon-
no one—
give me
audio. I
audio
tion of

rzenbach
bably
ou need.
monitor
stereo
might also
single in-
ereo in-
ality. As

not others. Second, legal pressure: MSX licensees must comply with Microsoft's minimum MSX specifications to use the MSX label on their computers. So adherence to the standard seems virtually guaranteed.

Although the MSX hardware seems unlikely to win any awards for advanced technology, the designers have extracted maximum performance with some impressive system software. In fact, MSX BASIC may well be the most powerful BASIC interpreter built into any personal computer at any price.

MSX BASIC is an extension of Microsoft BASIC 4.5 and is patterned after GW-BASIC, a common BASIC on 16-bit computers. It is a close relative to both TRS-80 Color Computer Extended BASIC and IBM PCjr Cartridge BASIC. Unlike the BASICs built into, say, the Atari and Commodore 64—computers with powerful sound and graphics capabilities—MSX BASIC has nearly all the commands you need to access its sound and graphics features without PEEKs, POKEs, or machine language. And that includes the sprites.

This article can't cover every command, statement, and function in MSX BASIC, but here are some highlights:

Besides the usual decimal numbers, constants can be expressed in hexadecimal, octal, or binary with the prefixes &H, &O, and &B. Variables can be any length, two characters significant, and either integer, single-precision, or double-precision. Arithmetic is performed with double-precision accuracy to 14 digits in Binary Coded Decimal (BCD), so the rounding errors common on other home computers are much rarer on MSX machines. There's a full set of relational operators (=, <, >, <>, <=, >=) and bitwise operators (NOT, AND, OR, XOR, EQV, IMP). Line numbers can range from 0 to 65529.

MSX BASIC has full-screen editing similar to Commodore, Atari, and IBM computers. The ten special function keys are preprogrammed with BASIC commands and can be redefined by the user. Auto line numbering and renumbering are built-in. TRON/TROFF commands let programmers trace a program as it executes, and ERROR lets them trap bugs from within programs. MSX BASIC supports DEF FN (defined functions); DEFUSR (jumps to machine language routines); array ERASE; variable CLEAR; LINE INPUT; PRINT USING and LPRINT USING; RESTORE to a line number; RESUME after error; SWAP variable values; conversions between decimal, hex, octal, and binary constants; VARPTR (variable address pointer); numerous string manipulators; KEY, KEY LIST, KEY ON/OFF, and ON KEY GOSUB (for the function keys); STOP ON/OFF/STOP and ON STOP GOSUB (for trapping the STOP key); and INTERVAL ON/OFF/STOP (interrupts from BASIC).

MSX BASIC may be the most powerful BASIC built into any personal computer at any price.

For graphics and sound, MSX BASIC supports SCREEN (for setting the graphics mode and other options), LOCATE (to specify a character position for PRINT), POINT (to determine the color of a specified pixel), COLOR (for setting screen colors), CIRCLE, DRAW, LINE, PAINT (a fill command), SPRITE$ (to define a sprite), SPRITE ON/OFF/STOP, PUT SPRITE, VPEEK and VPOKE (PEEK and POKE video RAM), BEEP, PLAY, and SOUND. Other interesting functions are STICK (read the joystick), STRIG (read the joystick button), PDL (for paddle controllers), and PAD (to interpret input from a touch tablet).

There are many more features, but from this overview it's clear that MSX BASIC is not only more powerful than the BASICs built into other home computers, it's also as powerful as most extended BASICs available at extra cost. There's even a CALL statement which lets manufacturers add their own commands for special features, such as CALL TALK for a voice synthesizer. There's nothing basic about MSX BASIC.

Despite its eight-bit leash, MSX BASIC contains another pleasant surprise: It's lightning fast.

To measure just how fast, COMPUTE! Assistant Editor Philip Nelson ran a series of benchmark tests using a simple bubble sort program. The program was written in plain-vanilla BASIC so it would run unmodified on a variety of popular computers. It creates a numeric array of 150 elements which are then sorted. Although this certainly isn't the most thorough benchmark test that could be devised, it is revealing. Several typical operations are involved, including array dimensioning, looping, and relational comparisons. Here's a listing of the test program:

```
100 PRINT "CREATING
  ARRAY"
110 DIM A(150)
```

```
120 FOR J=1 to 150
130 A(J)=151-J
140 NEXT J
150 PRINT "SORTING"
160 EX=0
170 FOR K=0 TO 149
180 IF A(K)>A(K+1) THEN T=A(K):A(K)=A(K+1):
    A(K+1)=T:EX=1
190 NEXT K
200 IF EX<>0 THEN GOTO 160
```

The only changes made to this program were double colons in line 180 as required for the TI-99/4A. Following are the test results expressed in minutes:seconds.

IBM PC	**5:45**
GoldStar MSX	**6:20**
Apple II Plus	**6:24**
Apple IIc	**6:33**
Commodore VIC-20	**6:34**
IBM PCjr	**6:59**
Commodore 64	**7:02**
Commodore 8032	**7:16**
TRS-80 Color Computer	**8:01**
Commodore 16	**8:35**
Commodore Plus/4	**8:36**
Atari 800XL	**8:55**
Atari 800	**9:00**
TI-99/4A	**12:58**

The specific results of this test aren't as important as the general conclusion. Although an MSX-based computer (and virtually any machine designed earlier than about two years ago) could be termed technologically ancient, the streamlined performance of the MSX is nothing to sneeze at.

Nevertheless, it remains difficult to predict whether or not MSX will succeed in America. Will consumers in 1985 be impressed with its affordable features, or bored by its technology? Both Commodore and Atari are expected to introduce new 16-bit or even 32-bit home computers at the same Winter Consumer Electronics Show where MSX will probably debut in January. Will these machines make MSX look even more tired in comparison? As long as a home computer has sufficient software and power to get the job done, does it matter to the average user if it contains an 8-bit or a 32-bit CPU?

Will MSX succeed because of the compatibility solution it offers? Are consumers tired of new computers that won't run anybody else's software? Or will they prefer the latest hot-technology machines, even if it means waiting for software?

If MSX does prevail, how will competitors react? Will they resist the standard or join it?

After IBM's recent tribulations with the PCjr, and the brick walls that TI, Atari, Mattel, and Coleco ran into in the fast lane, nothing is certain anymore in the home computer market. ©

NEW!

PRINTER ACCESSORIES FROM T.M.

DIGITAL DEVICES

U•PRINT T.M.

PARALLEL PRINTER INTERFACE

Expand your Atari® or Commodore® computer with Digital Devices ***U•PRINT.*** We make it simple to add any printer you choose. ***U•PRINT*** interfaces feature industry standard Centronics parallel connectors to hook up an Epson, Star, NEC, C.Itoh, Okidata, or any other printer.

U•PRINT MODEL A
- EXTRA SERIAL PORT FOR DAISY CHAINING OTHER PERIPHERALS.
- COMPATIBLE WITH ALL ATARI HARDWARE AND SOFTWARE.

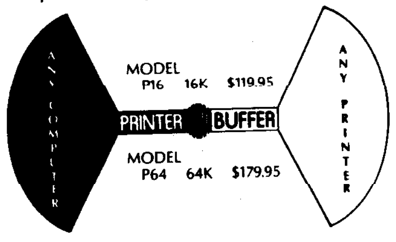
U•PRINT MODEL C
- EMULATION OF COMMODORE PRINTERS, INCLUDING GRAPHICS.
- COMPATIBLE WITH ALL COMMODORE HARDWARE AND SOFTWARE.

Compact, easy to install, and costing only $89.95, ***U•PRINT*** gives you a choice!

PRINTER BUFFER T.M.

CENTRONICS PARALLEL INTERFACE

The ***PRINTER BUFFER*** is the low-cost way to make your computer even faster! The ***PRINTER BUFFER*** takes information from your computer at high speed, stores it in memory and then retransmits it at the slower speeds a printer requires. Your computer is quickly free from the task of printing so you can do other things without waiting. With ***PRINTER BUFFER*** you can print and process *simultaneously.*



call **TOLL FREE (800) 554-4898** for more information on these peripherals from T.M.

DIGITAL DEVICES

430 Tenth Street, Suite N205 Atlanta, Georgia 30318
In Georgia (404) 872-4430;

* ATARI AND COMMODORE ARE REGISTERED TRADEMARKS OF ATARI, INC. AND COMMODORE ELECTRONICS LTD. RESPECTIVELY.

re®
IT. We
choose.
tandard
k up
r any

DAISY
IERALS.
ARI
RE.

DORE
APHICS.

E AND

sting only

T.M.
R
CE
-cost way
he
om your
memory
speeds a
ickly free
do other
BUFFER
usly.

A
N
Y

P
R
I
N
T
E
R

e information
n T.M.
CES
Georgia 30318
;
RADEMARKS OF
TD. RESPECTIVELY.

"Paratrooper" is a game of high responsibility—you control the destiny of ten parachutists, giving the go signal that ejects them from the plane. Their safe landings depend on your ability to judge weight factors, windage, and the all-important crucial moment when they should leap. Originally written for the TI-99/4A (with 16K and Extended BASIC), the program has been adapted for the Commodore 64, unexpanded VIC, Atari (with at least 32K), Apple, IBM PC (with color/graphics adapter and BASICA), PCjr (with Cartridge BASIC), and the Commodore Plus/4 and 16.

Paratrooper

John Goetz

Almost everyone has seen a parachuting exhibition. Perhaps you've wished that you, too, could fall from the sky on the wings of the wind. The plane drones on, cruising at the proper altitude. You peer out the hatch through wispy remnants of clouds as you decide where to land. You can barely see three tiny squares, far below, surrounded by water. These must be the landing pads, your drop zones. An aquatic landing can lead only to disgrace and severe embarrassment, so you know that you must jump at just the right moment.

There are three different-sized landing pads: The smaller pads promise the greatest honor and reward, but allow less room for error. Nearby, graceful sailboats ply the water. You know that soon these tiny features will grow at an alarming rate. You consult with the pilot and estimate the perfect moment for your jump by carefully considering your altitude, the speed of the wind, and your own body weight.

Too many late-night pizzas coupled with a low wind speed, and you'll drop like a stone. But if you're a featherweight, and the wind's kicking up, you'll find yourself drifting quite a way. With all the facts in, you wait for just the right moment. Then you leap out into the cold, crisp wind—with fingers crossed, of course.

If even reading this description makes you nervous, you'll be glad "Paratrooper" is just a computer game. Rarely is such a simple game so fun to play. The single key (or joystick) control and adjustable difficulty levels makes this an easy to learn, yet challenging, game for young children too.

Let Your Fingers Do The Jumping

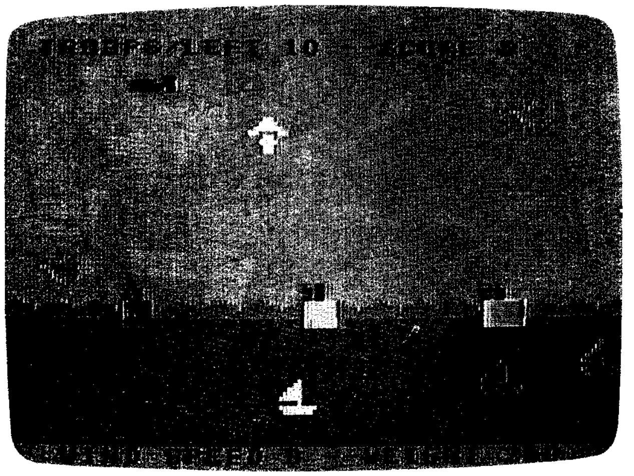
The various versions of Paratrooper differ slightly, but the concept is the same. Your plane continuously flies across the screen at an altitude which changes randomly for each jump. The paratroopers' weights and the wind speed change for each jump, too. All this information is displayed on the screen. You have ten paratroopers: ten chances for glory, or ten chances for dripping disaster. To drop a trooper, press any key (on the TI-99/4A, press Q or the fire button on joystick 1). The three landing pads are worth 25, 50, and 75 points, depending on their size.

All versions have more than one difficulty level. The TI version lets you choose between Novice and Experienced at the start of the game (you must rerun the program to change the level). The plane always moves at the same speed, but the landing pads are smaller in the Experienced level. Versions for the IBM, Atari, Plus/4, Commodore 16, and VIC-20 let you choose between Novice and Expert—again, the plane travels at the same speed, but the landing zones get smaller. The Commodore 64 version adds an Intermediate level. The Apple version has Easy and Hard levels, and the plane flies faster on the Hard level while the landing pads remain the same size.

Special Instructions

After typing in the Atari version (Program 5), it's important to save it on tape or disk before running it for the first time. Before loading the game, clear the computer by turning it off, then on again, and type POKE 128,0:POKE 129,64: NEW and press RETURN. This rearranges memory to make room for a machine language subroutine.

The VIC-20 version is broken into two parts so it works on an unexpanded VIC. Type in Program 3 and save it to tape or disk. If you are using tape, be sure to change the 8 to a 1 in line 40 of Program 3. Type in and save Program 4 as "P2" (for Part 2). Save Program 4 immediately after Program 3 on the tape.


A paratrooper leaps for the landing pads in the TI version of "Paratrooper."

Program 1: Paratrooper For TI-99/4A

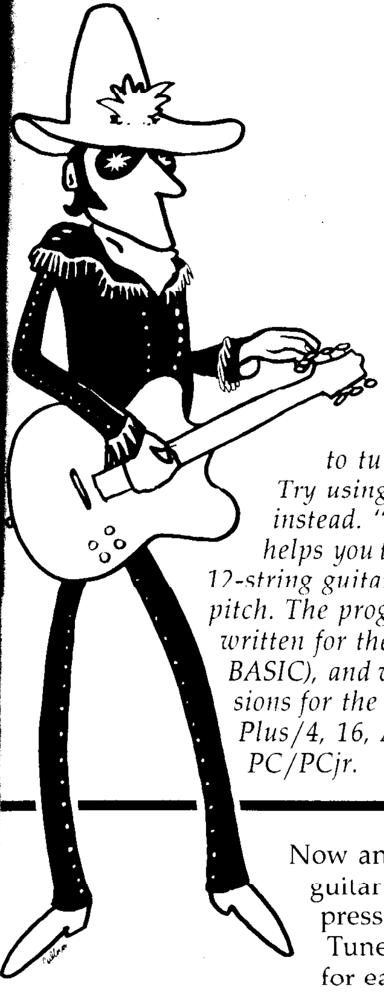
Refer to "COMPUTE!'s Guide To Typing In Programs" before entering this listing.

```
10 REM EXTENDED BASIC REQUIRED
20 CALL CLEAR
30 FOR T=10 TO 14 :: FOR I=10 TO 14
    :: DISPLAY AT(12,9):"PARATROOPE
   R"
40 CALL SCREEN(T):: NEXT I :: NEXT
   T
50 CALL CHAR(131,"183C7EC3183C1818"
   ):: CALL SCREEN(12)
60 FOR T=450 TO 550 STEP 50 :: FOR
   I=9 TO 19
70 DISPLAY AT(14,I):CHR$(131)
80 CALL SOUND(10,T,3):: NEXT I :: N
   EXT T
90 FOR I=1 TO 100 :: NEXT I :: GOSU
   B 920
100 DISPLAY AT(22,2):"NEED INSTRUCT
    IONS?(Y/N)"
110 ACCEPT AT(22,25)VALIDATE("YNyn"
    ):Y$
120 IF (Y$="Y")OR(Y$="y")THEN 750
130 IF (Y$="N")OR(Y$="n")THEN 860
140 CALL CLEAR :: CALL SCREEN(8)
150 CALL CHAR(33,"E7A424E7E781A5E7"
    ,34,"E78585E5E525A5E7")
160 CALL CHAR(37,"F794141727614147"
    ,42,"503D7C7C7C7A0088D")
170 CALL CHAR(43,"183C7DC300000000"
    ,44,"08183878F808FF7E")
180 CALL CHAR(46,"187E5A183C000000"
    ,98,"01031FFFFFFFFFFF")
190 CALL CHAR(99,"80C0FCFDFDFFFFFF
    ",107,"FFFFFFFFFFFFFFFF")
200 CALL CHAR(117,"FFFFFFFFFFFFFFFF
    ",122,"00E0A6E6A6FEBAEE")
210 CALL CHAR(130,"00000173FFFD0000
    ",133,"FFFFFFFFFFFF0000")
220 CALL CHAR(134,"FCFCFCFCFCFC0000
    ",135,"FEFEFEFEFEFE0000")
230 CALL CHAR(137,"183C7E7EFFFF1818
    ",143,"0E5FFE7F3E1C0800")
240 CALL SCREEN(8):: CALL COLOR(9,4
    ,8,10,6,1)
250 CALL HCHAR(16,1,107,256)
260 FOR I=1 TO 31 STEP 2 :: CALL HC
    HAR(16,I,98):: CALL HCHAR(16,I+
    1,99):: NEXT I
270 POINT=0 :: PARA=10
280 RANDOMIZE :: FOR N=22 TO 24 ::
    G=INT(RND*100)+10
290 CALL SPRITE(#N,143,15,G,G+120,0
    ,.60):: NEXT N
300 S=7 :: FOR N=4 TO 6 :: S=S-1 ::
     RANDOMIZE
310 D=INT(RND*5)+1 :: DD=INT(RND*14
    )+3 :: IF (D=OD)+(DD=ODD)+(DD=6
    )THEN 310
```

```
320 OD=D :: ODD=DD :: J=N*1Ø+9Ø+RND
    *1Ø :: CALL SPRITE(#S,44,DD,J,J
    ,Ø,D):: NEXT N
330 IF FL=1 THEN 37Ø ELSE DISPLAY A
    T(15,5):CHR$(37):: DISPLAY AT(1
    5,14):CHR$(34)
340 DISPLAY AT(15,23):CHR$(33)
350 CALL SPRITE(#3,32,1,18Ø,18Ø,Ø,6
    Ø):: REM INVISIBLE OCEAN SPRITE
360 CALL SPRITE(#7,133,1Ø,121,193,#
    8,135,12,121,121,#9,134,14,121,
    49):: REM PADS
370 IF PARA=Ø THEN 63Ø ELSE RANDOMI
    ZE :: U=INT(RND*7Ø)+1Ø :: REM P
    LANE ROW
380 CALL SPRITE(#1,13Ø,2,U,1Ø,Ø,-12
    ,#2,13Ø,16,U,7,Ø,-12):: REM PLA
    NE & TROOPER
390 V=INT(RND*9)+1 :: L=INT(RND*4)+
    1 :: REM WEIGHT & WIND FACTORS
400 DISPLAY AT(1,1):"TROOPS/LEFT";P
    ARA;"--SCORE";POINT
410 DISPLAY AT(24,2):"WIND SPEED";L
    *2;"--WEIGHT";(V*25)+5Ø
420 CALL KEY(1,X,Y)
430 IF X=18 THEN CALL PATTERN(#2,13
    1)ELSE 42Ø
440 CALL MOTION(#2,V,L):: CALL SOUN
    D(3Ø,-6,5,15Ø,5)
450 CALL COINC(#2,#7,Z,C)
460 CALL COINC(#2,#8,Z,CC)
470 CALL COINC(#2,#9,Z,CCC)
480 IF (C=-1)+(CC=-1)+(CCC=-1)THEN
    51Ø
490 CALL COINC(#2,#3,5Ø,R):: IF R=-
    1 THEN 57Ø
500 GOTO 45Ø
510 CALL MOTION(#2,Ø,Ø):: CALL PATT
    ERN(#2,46):: CALL SOUND(-15ØØ,5
    995,4)
520 FOR T=95Ø TO 15ØØ STEP 5Ø :: CA
    LL SOUND(5Ø,T,3):: NEXT T
530 POINT=POINT-25*(C=-1)-5Ø*(CC=-1
    )-75*(CCC=-1)
540 CALL DELSPRITE(#1,#2):: DISPLAY
     AT(13,5):"MISSION ACCOMPLISHED
    "
550 FOR I=1 TO 15Ø :: NEXT I
560 CALL HCHAR(13,5,32,22):: GOTO 3
    7Ø
570 CALL MOTION(#2,Ø,Ø):: CALL SOUN
    D(2ØØ,-4,3):: CALL PATTERN(#2,4
    3)
580 FOR I=1 TO 2ØØ :: NEXT I :: CAL
    L PATTERN(#2,42)
590 CALL DELSPRITE(#1,#2):: DISPLAY
     AT(13,3):"YOU MISSED THE DROP
    ZONE"
600 POINT=POINT-1Ø :: PARA=PARA-1
610 FOR I=1 TO 15Ø :: NEXT I :: CAL
    L HCHAR(13,3,32,26)
620 GOTO 37Ø
630 CALL HCHAR(1,1,32,29):: CALL HC
    HAR(24,1,32,29)
640 FOR I=45Ø TO 85Ø STEP 25 :: CAL
    L SOUND(5Ø,I,3):: NEXT I
650 FOR T=85Ø TO 45Ø STEP -25 :: CA
    LL SOUND(5Ø,T,3):: NEXT T
660 DISPLAY AT(7,1Ø):"GAME OVER"
670 DISPLAY AT(9,6):"YOU HAD ";POIN
    T;"POINTS"
680 DISPLAY AT(12,2):"WANT TO PLAY
    AGAIN? (Y/N)"
690 ACCEPT AT(12,27)VALIDATE("YN"):
    R$
700 IF R$="N" THEN 73Ø
710 CALL HCHAR(12,4,32,26):: CALL H
    CHAR(7,12,32,9):: CALL HCHAR(9,
    6,32,24)
720 FL=1 :: GOTO 27Ø
730 CALL CLEAR :: CALL DELSPRITE(AL
    L):: CALL SCREEN(14):: DISPLAY
    AT(12,1Ø):"GOOD BYE  "
740 GOSUB 92Ø :: END
750 CALL CLEAR :: CALL SCREEN(12)
760 PRINT "LAND YOUR PARATROOPERS O
    N","DROP PADS WORTH 75, 5Ø, OR"
770 PRINT "25 POINTS. RELEASE EACH"
    ,"WITH THE FIRE BUTTON ON","JOY
    STICK #1 OR THE {Q} KEY.": :
780 PRINT "IF YOU MISS, YOU WILL DR
    IFT","INTO THE OCEAN AND LOSE 1
    Ø"
790 PRINT "POINTS. YOU CAN ONLY LOS
    E","1Ø TROOPERS BEFORE THE","GA
    ME ENDS.": : :: PRINT "THE WIND
     SPEED AND WEIGHT"
800 PRINT "OF EACH TROOPER ARE DIS-
    ","PLAYED AT THE BOTTOM OF THE"
    ,"SCREEN. CONSIDER THE SPEED"
810 PRINT "OF DESCENT AND THE DRIFT
    --"
820 PRINT "CHECK THESE BEFORE RELEA
    SING","EACH PARATROOPER.": :
830 PRINT TAB(1Ø);"GOOD LUCK!": :
840 PRINT TAB(4);"PRESS ANY KEY TO
    BEGIN"
850 CALL KEY(Ø,K,S):: IF S=Ø THEN 8
    5Ø
860 CALL CLEAR :: DISPLAY AT(8,6):"
    PARATROOPER RANK ?"
870 DISPLAY AT(11,2):"<N>OVICE OR <
    E>XPERIENCED"
880 ACCEPT AT(8,24)VALIDATE("EN"):C
    $
890 IF C$="E" THEN 91Ø
900 CALL MAGNIFY(2):: Z=1Ø :: GOTO
    14Ø
910 Z=5 :: GOTO 14Ø
920 CALL SOUND(3ØØ,33Ø,3):: CALL SO
    UND(3ØØ,392,3)
930 CALL SOUND(5ØØ,392,3):: CALL SO
    UND(2ØØ,349,3)
940 CALL SOUND(1ØØ,33Ø,3):: CALL SO
    UND(2ØØ,294,3)
950 CALL SOUND(3ØØ,33Ø,3):: CALL SO
    UND(3ØØ,349,3)
960 CALL SOUND(3ØØ,37Ø,3):: CALL SO
    UND(3ØØ,392,3)
970 CALL SOUND(25Ø,44Ø,3):: CALL SO
    UND(15Ø,524,3)
980 CALL SOUND(5ØØ,524,3)
990 CALL SOUND(3ØØ,583,3):: CALL SO
    UND(1ØØ,523,3)
1000 CALL SOUND(2ØØ,44Ø,3):: CALL S
     OUND(3ØØ,392,3)
1010 RETURN
```

Guitar Tuner

Christopher Visco



Need a pitch pipe to tune your guitar? Try using your computer instead. "Guitar Tuner" helps you tune your 6- or 12-string guitar to perfect concert pitch. The program was originally written for the TI-99/4A (either BASIC), and we've added versions for the Commodore 64, Plus/4, 16, Atari, and IBM PC/PCjr.

Now an accurately tuned guitar is just a few keypresses away. "Guitar Tuner" plays a tone for each string on your 6- or 12-string guitar, freeing your hands to adjust the tuning pegs by ear.

To tune a 6-string guitar, run the program and play the tones by pressing the corresponding letter keys: E for the low (bass) E string; A for the A string; D for the D string; G for the G string; B for the B string; and CTRL-E for the high E string. To tune a 12-string guitar, press the SHIFT (or SHIFT LOCK) key for the second set of strings. This raises the tones by one octave (except for the B and high E strings, which are tuned to the same octave, of course).

If you aren't too familiar with the sound capabilities of your computer, you can learn a lot by studying these simple programs. Notice the DATA numbers at the end of each program; these are the tone values for the sound statements. Some programs convert these numbers with a formula to produce the proper tones. All the tones were verified with a quartz guitar-tuning meter calibrated for standard concert pitch.

A Note About Notes

The accuracy of any note produced by a computer tone generator (or synthesizer) is measured in the number of bits of *frequency resolution.* The more bits, the better. (Don't confuse this with the number of bits handled by the computer's main microprocessor—a 16-bit computer might still have a sound chip with only 8-bit frequency resolution, or vice versa.)

For example, the standard pitch for a middle A note is defined by musicians as 440 hertz (cycles per second). Let's say a certain computer's sound chip is limited to 8-bit frequency resolution. The most accurate A note it could generate might be 437.8 hertz. That's close enough to 440 for some people, but it would sound slightly flat to those with a good sense of pitch.

The TI-99/4A, IBM PC, and PCjr have 12-bit frequency resolution (in fact, the TI and PCjr both use the same Texas Instruments sound chip). Twelve-bit resolution is about the minimum required for people with a good sense of pitch. The Commodore 64 has 16-bit frequency resolution, so it's even more accurate. Commodore's new Plus/4 and 16 have 10-bit resolution, which provides passable results. The VIC-20 has only 8-bit frequency resolution, so Guitar Tuner isn't practical on the VIC. The program is easy to write on the VIC, but the tones are too far out of tune for musicians.

Atari computers also have 8-bit frequency resolution (the slightly flat A note described above is produced by the Atari). However, the Atari version of Guitar Tuner takes advantage of a little-known feature that lets you combine two of the 8-bit tone generators to make one 16-bit generator. This improves the accuracy of an A note from 437.8 to 439.97 hertz—close enough for almost anybody. (For more information on this technique, see "Perfect Pitch," *COMPUTE!'s Second Book of Atari.*)

Program 1: TI Guitar Tuner

Refer to "COMPUTE!'s Guide To Typing In Programs" before entering this listing.

```
100 DIM PITCH(12)
110 FOR T=0 TO 12
120 READ PITCH(T)
130 NEXT T
140 CALL CLEAR
150 CALL SCREEN(15)
160 PRINT TAB(10);"Guitar Tuner":::
    :
170 PRINT "Release the ALPHA-LOCK k
    ey."::
180 PRINT "Use the E/A/D/G/B/CTRL-E
    "::
190 PRINT "keys for a six-string"::
200 PRINT "guitar."::::
210 PRINT "Depress the ALPHA-LOCK k
    ey"::
220 PRINT "to tune the second set o
    f"::
230 PRINT "strings for a twelve-str
    ing"::
240 PRINT "guitar.":::
250 CALL KEY(0,K,S)
260 IF S=0 THEN 250
270 A$=CHR$(K)
280 A=-(A$="e")-2*(A$="a")-3*(A$="d
    ")-4*(A$="g")-5*(A$="b")-6*(A$=
    CHR$(133))-7*(A$="E")-8*(A$="A"
    )-9*(A$="D")-10*(A$="G")-11*(A$
    ="B")
290 CALL SOUND(1500,PITCH(A),2)
300 GOTO 250
310 DATA 40000,165,220,294,392,494,
    659
320 DATA 330,440,588,784,494,659
```

Program 2: Commodore 64 Guitar Tuner

Version by Gregg Peele, Assistant Programming Supervisor

Refer to "COMPUTE!'s Guide To Typing In Programs" before entering this listing.

```
8 DIMHI(12),LO(12),NO$(12)         :rem 115
10 PRINT"{N}{CLR}{13 RIGHT}GUITAR TUNER":
   FOR T= 0 TO 300:NEXT              :rem 55
15 PRINT"{HOME}{4 DOWN}{7 RIGHT}USE THE E
   /A/D/G/B/CTRL-E"                 :rem 140
16 PRINT"{3 DOWN}{5 RIGHT}KEYS FOR A SIX-
   STRING GUITAR."                   :rem 82
17 PRINT"{3 DOWN}{6 RIGHT}DEPRESS THE SHI
   FT LOCK KEY"                     :rem 161
18 PRINT"{3 DOWN}{7 RIGHT}TO TUNE THE SEC
   OND SET OF"                       :rem 53
19 PRINT"{3 DOWN}{2 RIGHT}STRINGS FOR A T
   WELVE-STRING GUITAR."            :rem 207
20 S=54272:FOR T= 0TO23:POKES+T,0:NEXT:PO
   KES+24,12:POKES+5,17:POKES+6,243
                                     :rem 94
70 FOR T=1TO11:READ HI,LO:HI(T)=HI:LO(T)=
   LO:NEXTT                          :rem 32
80 GET A$:IF A$=""THEN 80           :rem 243
90 A=-(A$="E")-2*(A$="A")-3*(A$="D")-4*(A
   $="G")-5*(A$="B")-6*(A$=CHR$(5))
                                    :rem 171
95 A=A-7*(A$="E")-8*(A$="A")         :rem 62
97 A=A-9*(A$="D")-10*(A$="G")-11*(A$="B")
                                     :rem 34
100 POKES,LO(A):POKES+1,HI(A)       :rem 211
150 POKES+4,17:FORI=0TO2000:NEXTI:POKES+4
    ,16                             :rem 183
175 POKE198,0:GOTO80                :rem 165
200 DATA 10,143,14,24,18,209,25,30,31,165
    ,42,62,21,31,28,49,37,162,50,60,31,16
    5                                :rem 20
```

Program 3: Atari Guitar Tuner

Version by Gregg Peele, Assistant Programming Supervisor

Refer to "COMPUTE!'s Guide To Typing In Programs" before entering this listing.

```
AG 5 DIM A(14)
HG 10 GRAPHICS 2+16
FG 20 POSITION 3,5:? #6;"g▮i▮a▮ T▮
      n▮R"
DO 30 FOR T=0 TO 3000:NEXT T
EJ 40 GRAPHICS 0:POKE 752,1
GM 50 POSITION 6,4:? "Use the E/A/
      D/G/B/CONTROL-E"
GH 60 POSITION 6,7:? "keys for a s
      ix-string guitar."
NN 70 POSITION 9,10:? "Depress the
       shift key "
OE 80 POSITION 3,13:? "to tune the
       second set of strings"
CB 90 POSITION 6,16:? "for a twelv
      e-string guitar."
EB 120 FOR T=0 TO 12
FC 130 READ TUNE:A(T)=TUNE:NEXT T
CP 135 B=PEEK(764)
JF 137 PNTR=1*(B=42)+2*(B=63)+3*(B
       =58)+4*(B=61)+5*(B=21)+6*(B
       =170)+7*(B=106)
HO 138 PNTR=PNTR+8*(B=127)+9*(B=12
       2)+10*(B=125)+11*(B=85)+12*
       (B=234)
KF 139 IF PNTR=0 THEN 135
DN 141 P2=INT((1789790/(2*A(PNTR))
       -7)/256)
FE 142 P1=INT(1789790/(2*A(PNTR))-
       7-256*P2+0.5)
AN 143 POKE 53768,80:POKE 53760,P1
       :POKE 53762,P2:POKE 53763,(
       16*10)+10
GC 156 FOR I=1 TO 3000:NEXT I
NI 157 POKE 764,0:SOUND 0,0,0,0:SO
       UND 1,0,0,0
GM 163 GOTO 135
FP 170 DATA 0,165,220,294,392,494,
       659,330,440,588,784,494,659
```

Program 4: PC/PCjr Guitar Tuner

Version by Gregg Peele, Assistant Programming Supervisor

Refer to "COMPUTE!'s Guide To Typing In Programs" before entering this listing.

```
LG 10 CLS:KEY OFF
AB 20 WIDTH 80:DIM PITCH(12):DEF SEG =
      0:POKE 1047,0
JH 30 LOCATE 1,34:PRINT"Guitar Tuner"
EE 40 LOCATE 4,28:PRINT"Use the E/A/D/
      G/B/CTRL-E"
JF 50 LOCATE 7,26:PRINT "keys for a si
      x-string guitar."
```

CAPUTE!

Modifications Or Corrections To Previous Articles

TI Disassembler

This machine language deciphering aid from the October 1984 issue (p. 159) has a number of shortcomings. First, the article incorrectly stated that the program could easily be translated to standard BASIC. Unfortunately, TI's built-in BASIC lacks the AND operator used throughout the program. The program also fails to properly decode backward jumps and some Format III opcodes, and has several other minor bugs. To correct these problems, the following lines need to be changed as indicated:

```
440 N=(H AND 11)*256 :: J=1792 :: C
    O=(L AND 240)/16 :: WR=(L AND 1
    5):: RESTORE 1040 :: Z=4 :: K=2
    56 :: GOSUB 900
485 IF L>127 THEN L=L-256
680 GOTO 630
700 RESTORE 1080 :: J=12288 :: N=(H
     AND 240)*256 :: Z=12 :: K=4096
     :: GOSUB 900
740 IF TD=12 THEN C$=",*R"&STR$(D)&
    "+" :: GOTO 770
750 IF (TD=8)AND(D=0)THEN C$=","&"@
    "&STR$(O1*256+O2):: A=A+2 :: IF
     TS=32 THEN C$=","&"@"&STR$(O3*
    256+O4)
```

Thanks to Glenn Davis, Henry Satinskas, and others who ferreted out these errors.

Spiders For IBM PC And PCjr

Some punctuation characters were garbled in printing the listing for Program 7 (p. 98) of this game from the November 1984 issue. In line 170, there should be a colon—not a period—between LOCATE 25,1 and PRINT. In line 330, the character between AX$(Y) and CHR$(BX) should be a comma.

TI Reflection

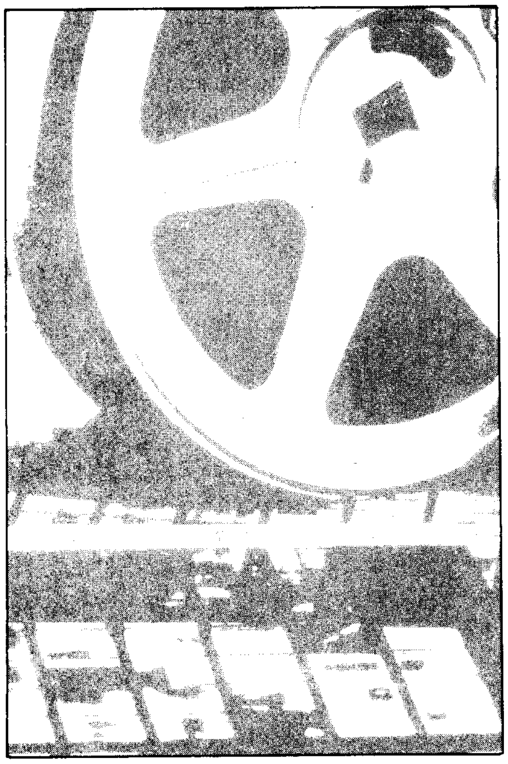
Line 1600 in Program 5 (p. 76) of this game from the November issue is too long to be entered in standard TI BASIC, although it can be entered if you're using Extended BASIC. To use the program with the built-in BASIC, split the line into two parts, as shown below (be sure to include the semicolon at the end of line 1600):

```
1600 PRINT A$,B$,B$,A$&"  'S",B$,B$
     &" UP",A$,B$,B$,A$,B$,B$,A$&"
     SUM",B$,B$&" x:",A$,B$,B$,A$,B
     $&" "&CHR$(128);
1605 PRINT ":",B$,A$,B$,C$;
```

64 Horse Racing

The correction listed in last month's CAPUTE! for the VIC version of "Horse Racing" actually applies to the Commodore 64 version. There are no corrections for the VIC version. ©

This Publication is available in Microform.



University Microfilms International

Please send additional information

for (name of publication)	
Name	
Institution	
Street	
City	
State	Zip

300 North Zeeb Road
Dept. P.R.
Ann Arbor, Mi. 48106

Tom R. Halfhill, Editor

Which Computer Language Is Best?

Most of us have heard the Biblical story about the Tower of Babel—how God made mankind speak in hundreds of different languages for daring to build a tower to heaven.

In the computer age we seem to suffer from a similar problem. We're burdened with scores of different computer programming languages. And like human languages, they're all largely incompatible with each other.

There are, however, definite reasons why we have so many human languages and computer languages. Both were invented because of the need to communicate ideas. The first language for a modern electronic computer was invented in the 1950s for a specific purpose—to make it easier for people to program computers. Today there are scores of different languages and dialects within languages.

Why, then, if computer languages are conscious inventions all conceived for the same reason, do we have so many of them? Why does one language use the word PRINT to put messages on the screen while another uses TYPE? Why weren't words and definitions standardized from the very beginning to eliminate confusion?

A Language For Every Purpose

One answer is that it's no more realistic to expect a single programming language to be suitable for all possible tasks than it is to expect one type of computer to be ideal for every possible application. Another answer is that those who write languages all have their own ideas about how computers should be programmed (or, depending on your point of view, how humans should be programmed to work with computers). In addition, some languages are hard to implement on certain types of computers, especially home computers with relatively small amounts of memory.

That leaves it up to us to sort out the confusion and decide which language to use to get the job done. Generally there are three things to consider: the suitability of the language to the task; the ease of learning and using the language; and the availability of the language on the computer we want to use.

Here's a summary of the most popular languages available today for home and personal computers:

• *BASIC (Beginner's All-purpose Symbolic Instruction Code).* Invented in the early 1960s at Dartmouth College, BASIC was originally designed as a very simple language that beginners could pick up and use with only a few hours of study. Since then, there have been so many extensions and spin-off dialects that BASIC is used to program everything from videogames to powerful business applications. As a result, some people criticize BASIC as a messy, unstructured hodgepodge of commands. Others find it simple, effective, and versatile. Because BASIC has been built into nearly all microcomputers for years, it's by far the dominant language in personal computing. That doesn't seem likely to change in the near future.

• *Logo.* Designed in the 1970s especially for children, Logo is found primarily on home computers and includes *turtle graphics,* a simplified system for drawing pictures on a video screen. You control a small cursor, the turtle, which can be rotated and moved in different directions while leaving behind a colored trail. Series of commands can be grouped into procedures and executed repeatedly to create geometric patterns. Logo also helps teach logical thinking and organization.

• *PILOT (Programmed Instruction, Learning, Or Teaching).* This language resembles Logo and usually includes turtle graphics. It also has flexible word-matching commands that make it easier to write educational programs which ask questions and evaluate answers.

• *Pascal (named after Blaise Pascal, the seventeenth-century French mathematician).* Invented in the 1970s partly as a reaction to the perceived weaknesses of BASIC, Pascal is known as a *structured* language and is widely used to teach programming at the college level. Programs written in Pascal usually have a modular, organized construction. Although Pascal doesn't necessarily force this structure on the programmer, it strongly encourages it.

• *Forth (so-named because it was conceived as a "fourth-generation" language).* Forth is an unusual language, known for its speed of execution, flexibility, and reverse Polish notation arithmetic. It's used for many scientific applications, especially in astronomy, and has a vocal following among microcomputer hobbyists. Forth is one of the few languages that can be readily extended by the programmer. It lets you define your own commands by linking together a series of simpler commands. This new command, in turn, can be used to build commands which are even more powerful.

BASIC: Bread And Butter

For most personal computer programming, BASIC is the first choice. Not necessarily because it's the best language—BASIC certainly has its share of shortcomings. However, it does satisfy the three considerations mentioned above: It's a true general-purpose language which can do a lot of things adequately; it's fairly easy to learn and use; and, perhaps most important, it's widely available. Chances are BASIC is built right into your computer as a standard feature. If not, it's available separately at minimum cost.

BASIC runs on practically every computer because it doesn't require lots of memory. That's partly why it was the first language of its type adapted to microcomputers, back in the days when 4K of RAM was considered luxurious. Even the old Sinclair ZX-81, which came with only 1K of RAM, had a fairly powerful built-in BASIC. What's more, BASIC usually doesn't require you to buy a disk drive or other expensive peripherals. Nearly all BASICs can work with tape storage.

BASIC has other things going for it, too. The vast majority of program listings published in computer magazines and books are in BASIC. It's not that authors don't submit programs written in other languages. It's just that BASIC is the only language which editors can be sure their readers own. Publishing a program in a language like Pascal or Forth renders it useless to 90 percent of the readership. Unavoidably, of course, this policy solidifies BASIC's position and perpetuates its dominance.

BASIC also comes in many flavors. If the BASIC that came with your computer isn't powerful or flexible enough for your purposes, you can probably buy an *extended* or *enhanced* BASIC. For instance, the BASIC built into the Commodore 64 lacks commands to take advantage of the computer's excellent sound and graphics capabilities. If you want to easily write a program using sound and graphics, you can plug in a *Simons' BASIC* cartridge and gain 114 more commands. Similarly, trade-offs made by the designers of Atari BASIC omitted certain features (such as string arrays) which are considered standard in the more common Microsoft BASICs. If this matters, you can buy an extended Microsoft BASIC on cartridge or disk.

Despite all the criticisms leveled at BASIC, for the foreseeable future it's here to stay.

When To Switch

All these reasons don't mean you're pinned down to BASIC by any means. Here are some situations when you might want to make your computer bilingual:

—You're writing a program that simply demands more power, speed, or flexibility than BASIC can deliver.

—You're writing programs only for yourself that won't be shared with other people or submitted to general-interest publications.

—You've run across a program so useful that it's worth your while to buy the language you need to run it.

—You'd like to introduce youngsters to computer programming without bogging them down in the picky details of BASIC.

—You're learning another language at school or work and want to practice writing programs at home with your own computer.

—You want to explore alternatives to BASIC just out of curiosity.

Second languages are available for most computers on cassettes, disks, and cartridges. Cartridges are handiest because you don't have to wait around for a long program (the language) to load—you just plug it in and switch on the computer. Cartridges are also sturdy and generally don't require a disk drive. But because the memory capacity of a cartridge is severely limited (usually no more than 16K), many languages won't fit in a cartridge and are available only on disk.

If you already know one computer language, such as BASIC, you'll find that it's easier to learn a second language—certainly much easier than learning to speak and read a second human language. Human languages have vocabularies of tens of thousands of words, and the rules of syntax are often vague and conflicting. But most computer languages have a total vocabulary of only 50 to 100 words, and the rules for using them are carefully defined. The computer even tells you when you make a mistake. Plus, the fundamental knowledge you gain by learning your first language lets you adjust fairly quickly to the rules of the new language.

The Computer's Native Tongue

You may have noticed one popular computer language missing from the list above: *assembly language* or *machine language* (for now we'll use both terms synonymously).

We deliberately omitted machine language because it isn't quite a language in the same sense as BASIC, Logo, or Pascal. True, machine language is a method of encoding your ideas so that the computer can understand and act on them. In that sense it *is* a language. But with machine language, you're dealing with the computer on a much more intimate level. You're speaking in its native tongue.

The fact is, languages such as BASIC—known as *high-level languages*—were invented for people, not for computers. They were designed for convenience, so people wouldn't have to program computers in machine language. Why? Because machine language programming can be more exacting. Sometimes it takes a dozen or more commands in machine language to do something as simple as display a message on the screen. You might accomplish the same thing in a high-level language with a single command such as PRINT.

But it's important to realize that the computer doesn't understand BASIC or any other high-level language any more than it knows English. A high-level language is really a sophisticated program which itself is written in machine language. When you use a command such as PRINT, the BASIC language translates the command into the proper sequence of machine language commands. In this form, the computer can carry them out.

Despite the extra steps required when programming directly in machine language, it's still very popular. That's because the translation process required by a high-level language takes time, and some programs demand all the speed and power that the computer can deliver. A program written in machine language bypasses this translation step and runs much faster. Sometimes it's the only way to get the job done. However, as technology advances and computers get faster and faster, it's likely that fewer programs will be written directly in machine language.

Questions Beginners Ask

Q I've seen specifications for computers that talk about graphics modes with 320 × 200 pixels, 640 × 200 pixels, etc. But what's a pixel?

A *Pixels* (an abbreviation for *picture elements*) are the tiny dots on the screen that make up the image. If you look very closely at your computer monitor you can see the dots, although they may be too blurred to see clearly on an ordinary color TV.

All video images are composed of pixels, including regular broadcast video pictures. However, there's no standard size for pixels. They can be large or small. Size is important because the smaller the pixel, the more will fit on the screen, and therefore the more detailed the image will be.

For example, a graphics mode of 320 × 200 pixels means the computer can display 320 pixels horizontally and 200 pixels vertically. That's a total of 64,000 pixels. If the computer has a 640 × 200 graphics mode, it can display 128,000 pixels. With twice as many screen dots to work with, the picture can be twice as detailed. In video terms, the more pixels, the greater the *resolution*.

It might seem that creating superdetailed computer images would be as easy as displaying more pixels. But there are several technical obstacles to overcome.

To begin with, the information which defines how each pixel will appear on the screen must be stored in the computer's memory. The computer must know where each pixel will be placed and what color it will be. The more pixels and colors you want to display, the more memory you need. For example, the IBM PCjr has a graphics mode of 640 × 200 pixels with four colors (SCREEN 6 in Cartridge BASIC). It requires 32K of RAM just to store all this information. A 640 × 200 mode with eight colors would require 64K, and a 640 × 200 mode with 16 colors would eat up 128K.

A related problem is computer speed. The more memory it takes to define how the screen will look, the more time it takes the computer to access that memory. An extremely high-resolution screen could bog down the computer so much that it would run programs noticeably slower. (In fact, to reduce this problem, many computers have separate microprocessors just to control the screen display.)

Finally, there's a limit to how sharply a TV set can resolve a pixel. Computers can be designed to work with special monitors (such as the Apple Macintosh), but home computers must be compatible with ordinary TV sets to reach the mass market. ©

C Regena

Mixing Graphics And Music

I've talked about combining graphics with music in a TI program before. This month I'll add a few more ideas and techniques to try to help you in your programming. Remember, there are many ways to do the same thing, and the important idea is to enjoy your computer!

Clear-Screen Effects

The command CALL CLEAR is the usual way to quickly clear the screen. For a different effect, try:

CALL HCHAR(1,1,32,768)

or

CALL VCHAR(1,1,32,768)

These statements tell the computer to start with the first row and first column and fill the screen with 768 spaces (ASCII character 32).

If you want to fill the screen with a color, try the following example. Set the variable C to the desired color number:

100 CALL CLEAR
110 CALL SCREEN(C)

or

100 CALL COLOR(9,C,C)
110 CALL HCHAR(1,1,96,768)

Following is a sample program segment that illustrates another way to clear the screen—by starting at the center and moving outward.

```
100 CALL CLEAR
110 CALL COLOR(9,14,14)
120 C=13
130 T=8
140 U=0
150 FOR R=12 TO 1 STEP -1
160 CALL HCHAR(R,C,96,T)
170 CALL VCHAR(R+1,C,96,U)
180 CALL VCHAR(R+1,C+T-1,96,U)
190 CALL HCHAR(R+1+U,C,96,T)
200 C=C-1
210 T=T+2
220 U-U+2
230 NEXT R
240 GOTO 240
```

Another effect is to change all the spaces to a different color by redefining the color for color set 1:

CALL COLOR(1,2,7)

This definition will retain the default foreground color of black (color 2) for the symbols in set 1, but will change the background color to 7. Since the space character is blank, the background color shines through wherever there's a space.

Making The Invisible Visible

The CALL COLOR statement changes the color of any characters in the specified set on the screen. For example, try writing a program to print a message on the screen, then follow the message with this statement:

200 CALL COLOR(5,10,1)

All the characters in set 5 will change from black to red.

Remember that the number 1 in a color definition means transparency, or the current screen color. Try drawing something on the screen transparently, then use a different CALL COLOR statement to make the object appear all at once. For example:

```
100 CALL CLEAR
110 CALL COLOR(6,1,1)
120 PRINT "HI JIM":::
130 CALL COLOR(6,13,1)
140 GOTO 140
```

Line 100 clears the screen, then line 110 defines the colors for set 6 to be transparent. Line

120 prints a message and scrolls it upward. Line 130 makes the printing visible by changing the color set to dark green. Line 140 keeps the color on the screen until you press CLEAR.

Changing Character Shapes

Another technique you may have fun with is to change a character definition while the character is on the screen. For example, suppose you have a lot of printing on the screen, then you use CALL CHAR to redefine the letter E as a straight line. Wherever there is an E on the screen, it will suddenly appear as a straight line. The following sample program illustrates what happens when you change the definition of the space character. The GOSUB statement is a simple delay loop to pause between definitions.

```
100 CALL CHAR(32,"FF")
110 GOSUB 190
120 CALL CHAR(32,"0102040810204
    08")
130 GOSUB 190
140 CALL CHAR(32,"1010101010101
    01")
150 GOSUB 190
160 CALL CHAR(32,"8040201008040
    201")
170 GOSUB 190
180 GOTO 100
190 FOR D=1 TO 200
200 NEXT D
210 RETURN
220 END
```

Graphics can be a lot of fun. If you like to use graphics, you really need to just sit at the computer and try different things. See what happens if you define the colors first, then display the characters, or if you change the colors after the graphics are on the screen. Try defining the characters before or after printing them on the screen. Look at the difference between using PRINT and CALL HCHAR or CALL VCHAR statements.

A Holiday Greeting

This month I've included a program which is my holiday greeting to you. This program combines sound and graphics using some of the techniques previously discussed. Here's a breakdown of the program.

Line 100 clears the screen, then line 110 changes the screen color to dark blue. The default values of a CALL COLOR statement are black printing on a transparent (screen color) background. Line 140 will change all the spaces to a blue background rather than screen color. The CALL COLOR statements in lines 150–190 change the color sets for graphics to be solid blue squares—the graphics will be drawn invisibly at first. The CALL COLOR statements in lines 200–240 change the printing to white letters on a blue background.

Lines 320–440 print the graphics on the screen. These lowercase letters and symbols need to be typed with the ALPHA LOCK key released. Turn the ALPHA LOCK key back on to type the rest of the program. Since the letters are blue with a blue background on a blue screen, you won't see anything yet.

Line 480 changes the screen color to black. In effect, this puts a black border around the screen (recall that all spaces and other characters are blue). The extra PRINT statements and colons format an attractive left and right margin.

Lines 490–860 define the graphics characters while music is set up and playing. Remember, the graphics are already on the screen, but are invisible because they are blue. Lines 870–890 change the colors of sets 10, 11, and 12 to red with a blue background, making the sleigh appear. Line 920 changes the colors of set 9 so all the reindeer appear instantly.

If you'd like the message to blink, you can add some CALL COLOR statements for sets 5 through 8 among the CALL SOUND statements in lines 930–1170.

Adding The Sound Track

After the graphics in this program were completed, I added the SOUND statements for the music. Line 120 sets a tempo or time of 440. By using the variable T at the beginning of the program and expressing all durations as a function of T, you can change the tempo of the whole song by simply adjusting the value of T in line 120.

When writing the program, I tried only the melody notes of the song first to make sure the graphics did not interfere with the tempo of the music. Later I added two accompaniment notes for each statement.

Sometimes when you have two CALL SOUND statements with the same note and volume, the resulting sound is one long note rather than two shorter notes. To make sure you get distinct notes, you can change the volume numbers slightly. If you want to make two different chords sound like they have a common tied note, keep the frequency and volume the same for that note.

To make the melody heard over the accompaniment, use a louder volume for the melody notes. For example, use a volume of 2 for the melody, 5 for the middle note, and 8 for the bottom note:

CALL SOUND(T,466,2,294,5,175,8)

If you don't want to type this program, you can get a copy by sending a blank cassette or disk, a stamped, self-addressed mailer, and $3 to:

C. Regena
P.O. Box 1502
Cedar City, UT 84720

Please specify the title of the program ("Jolly Old St. Nick") and that you need the TI version. Hope you have fun making your own holiday greeting programs!

Jolly Old St. Nick

Refer to "COMPUTE!'s Guide To Typing In Programs" before entering this listing.

```
100 CALL CLEAR
110 CALL SCREEN(5)
120 T=44Ø
130 CALL SOUND(T,587,2,466,5,17
    5,8)
140 CALL COLOR(1,16,5)
150 CALL COLOR(9,5,5)
160 CALL COLOR(1Ø,5,5)
170 CALL SOUND(T,587,3,466,6,17
    5,8)
180 CALL COLOR(11,5,5)
190 CALL COLOR(12,5,5)
200 CALL COLOR(5,16,5)
210 CALL SOUND(T,587,2,466,5,17
    5,1Ø)
220 CALL COLOR(6,16,5)
230 CALL COLOR(7,16,5)
240 CALL COLOR(8,16,5)
250 CALL SOUND(T,587,3,466,7,17
    5,1Ø)
260 PRINT
270 CALL SOUND(T,523,2,44Ø,5,15
    6,8)
280 PRINT
290 CALL SOUND(T,523,3,44Ø,6,15
    6,8)
300 PRINT
310 CALL SOUND(T*2,523,2,44Ø,5,
    156,8)
320 PRINT :"  `a"
330 PRINT " bcd{3 SPACES}`a"
340 CALL SOUND(T,466,2,392,5,14
    7,8)
350 PRINT "{3 SPACES}ef bcd
    {3 SPACES}`a"
360 CALL SOUND(T,466,3,392,7,14
    7,8)
370 PRINT TAB(9);"ef bcd
    {3 SPACES}`a"
380 CALL SOUND(T,466,2,392,5,14
    7,1Ø)
390 PRINT TAB(14);"ef bcd
    {3 SPACES}hijkl"
400 CALL SOUND(T,466,3,294,5,19
    6,1Ø)
410 PRINT TAB(19);"ef  mnopq"
420 CALL SOUND(4*T,587,2,349,5,
    147,8)
430 PRINT TAB(23);"rstuv"
440 PRINT TAB(23);"wxyz{"
450 PRINT :::TAB(7);"MERRY CHRI
    STMAS"
460 PRINT ::TAB(13);"FROM"
470 PRINT ::TAB(12);"REGENA":::
480 CALL SCREEN(2)
490 CALL CHAR(96,"Ø61F1FØ71F3E7
    CFC")
500 CALL CHAR(97,"3ØF6FF8FF8")
510 CALL CHAR(98,"ØØ1Ø7ØF3F3Ø3
    81")
520 CALL CHAR(99,"FEFFFFFFFF3F"
    )
530 CALL SOUND(T,392,2,311,5,11
    7,8)
540 CALL CHAR(1ØØ,"Ø1FFFFFEFEFE
    7E3F")
550 CALL CHAR(1Ø1,"ØFØ1")
560 CALL SOUND(T,392,3,311,7,11
    7,1Ø)
570 CALL CHAR(1Ø2,"CØCØEØ7Ø18Ø8
    ")
580 CALL CHAR(1Ø4,"ØØØØØØØØØØØØ
    ØØØ3")
590 CALL SOUND(T,392,2,311,5,11
    7,8)
600 CALL CHAR(1Ø5,"ØØØØØØØØØØØØ
    ØØC")
610 CALL CHAR(1Ø6,"ØØØØØ1ØØ86C1
    E8CE")
620 CALL SOUND(T,392,3,311,7,11
    7,1Ø)
630 CALL CHAR(1Ø7,"ØØØCCEE77737
    9Ø9")
640 CALL CHAR(1Ø8,"ØØØØØØØ8D8FØ
    ØØ3")
650 CALL SOUND(T,349,2,294,5,11
    7,8)
660 CALL CHAR(1Ø9,"ØFØFØFØ6ØØ
    ØØ1C")
670 CALL CHAR(11Ø,"FØFC7D3D3C7C
    FEFF")
680 CALL SOUND(T,349,3,294,7,11
    7,1Ø)
690 CALL CHAR(111,"6F67E3FØFF7F
    1E")
700 CALL CHAR(112,"Ø3C787ØFECCØ
    Ø3ØF")
710 CALL SOUND(2*T,466,2,294,5,
    117,8)
720 CALL CHAR(113,"FCFEFØ8Ø1F7F
    FEE")
730 CALL CHAR(114,"7ØCØCØCØ6Ø
    781E")
740 CALL CHAR(115,"7F3F1FØ37Ø3F
    1C38")
750 CALL CHAR(116,"EØFFFFFF7FØ7
    ØØ1E")
760 CALL CHAR(117,"FFFFFFFFFEFC
    ")
770 CALL SOUND(T,44Ø,2,349,5,17
    5,8)
780 CALL CHAR(118,"CØ8Ø8")
790 CALL CHAR(119,"Ø3")
800 CALL SOUND(T,466,2,349,6,17
    5,1Ø)
810 CALL CHAR(12Ø,"F83EØ3")
820 CALL CHAR(121,"Ø7Ø7FF1F")
830 CALL SOUND(T,523,2,349,5,22
    Ø,8)
```

```
840 CALL CHAR(122,"FØ8Ø8ØCØ7F")
85Ø CALL CHAR(123,"ØØCØ6Ø6ØC")
86Ø CALL SOUND(T,587,2,349,7,23
    3,8)
87Ø CALL COLOR(1Ø,1Ø,5)
88Ø CALL COLOR(11,1Ø,5)
89Ø CALL COLOR(12,1Ø,5)
9ØØ CALL SOUND(2*T,523,2,349,8,
    22Ø,1Ø)
91Ø CALL SOUND(2*T,523,2,44Ø,8,
    175,1Ø)
92Ø CALL COLOR(9,11,5)
93Ø CALL SOUND(T,587,2,466,5,17
    5,8)
94Ø CALL SOUND(T,587,3,466,6,23
    3,8)
95Ø CALL SOUND(T,587,2,466,5,17
    5,8)
96Ø CALL SOUND(T,587,3,466,6,23
    3,8)
97Ø CALL SOUND(T,523,2,44Ø,5,17
    5,8)
98Ø CALL SOUND(T,523,3,44Ø,6,31
    1,8)
99Ø CALL SOUND(T*2,523,2,44Ø,5,
    175,8)
1ØØØ CALL SOUND(T,466,2,392,5,1
     96,8)
1Ø1Ø CALL SOUND(T,466,3,392,6,2
     94,8)
1Ø2Ø CALL SOUND(T,466,2,392,5,1
     96,8)
1Ø3Ø CALL SOUND(T,466,3,294,5,2
     33,8)
1Ø4Ø CALL SOUND(2*T,587,2,349,5
     ,22Ø,8)
1Ø5Ø CALL SOUND(2*T,587,2,349,5
     ,262,8)
1Ø6Ø CALL SOUND(T,392,2,311,5,1
     56,8)
1Ø7Ø CALL SOUND(T,392,3,311,6,1
     33,8)
1Ø8Ø CALL SOUND(T,392,2,311,5,1
     56,8)
1Ø9Ø CALL SOUND(T,392,3,311,6,1
     33,8)
11ØØ CALL SOUND(T,349,2,294,5,1
     17,8)
111Ø CALL SOUND(T,349,3,294,6,2
     33,8)
112Ø CALL SOUND(2*T,466,2,294,5
     ,117,8)
113Ø CALL SOUND(T,523,2,311,5,2
     2Ø,8)
114Ø CALL SOUND(T,466,2,294,5,1
     75,8)
115Ø CALL SOUND(T,523,2,311,5,2
     2Ø,8)
116Ø CALL SOUND(T,587,2,349,5,1
     75,8)
117Ø CALL SOUND(4*T,466,2,294,5
     ,175,8)
118Ø CALL KEY(Ø,K,S)
119Ø IF S<1 THEN 118Ø
12ØØ CALL CLEAR
121Ø END
```
©

MAXIMIZE STORAGE CAPACITY ON YOUR ATARI 1050* DISK DRIVE WITH THE HAPPY 1050 MAXIMIZER™

Now you can store **twice** as much data on your **ATARI 1050** disk drive with this easy to install high quality plug in adapter. Requires no soldering and no permanent modifications. Runs all popular true double density programs, utilities, and operating systems.

You can upgrade your **HAPPY 1050 MAXIMIZER** to a **WARP SPEED HAPPY 1050 ENHANCEMENT™**. Improves reading and writing speed 500% and comes with the **HAPPY COMPUTERS WARP SPEED SOFTWARE™** package. Makes your **ATARI 1050** the most powerful disk drive available. Easy plug in installation lets you upgrade your **HAPPY 1050 MAXIMIZER** to **WARP SPEED at any time.**

Take COMMAND with the HAPPY 1050 CONTROLLER™

When used with the **ENHANCEMENT** or **MAXIMIZER** allows writing on the flip side of disks without punching holes. Selects protection from writing on valuable disks. Selection can be made both from software commands and a three position switch. When used with the **ENHANCEMENT** allows both switch and software control of reading and writing speeds. Plug in installation requires no soldering. May be used without **ENHANCEMENT** or **MAXIMIZER** with manual control of write protection.

Discount prices through Dec. 31, 1984:

HAPPY 1050 MAXIMIZER complete	$124.95
MAXIMIZER to **ENHANCEMENT UPGRADE** (You must already have a Happy 1050 Maximizer)	$129.95
HAPPY 1050 MAXIMIZER with factory installed **MAXIMIZER** to **ENHANCEMENT** upgrade, same as **WARP SPEED HAPPY 1050 ENHANCEMENT**	$249.95
HAPPY 1050 CONTROLLER	$49.95
WARP SPEED HAPPY 810 ENHANCEMENT™ for 810 disk drive (supports high speed single density)	$249.95

Price above include free delivery in the USA. California residents add 6.5% sales tax.

*Note: ATARI 1050 is a trademark of Atari, Inc.

HAPPY COMPUTERS, INC.
P.O. Box 1268, Morgan Hill, CA 95037
(408) 779-3830