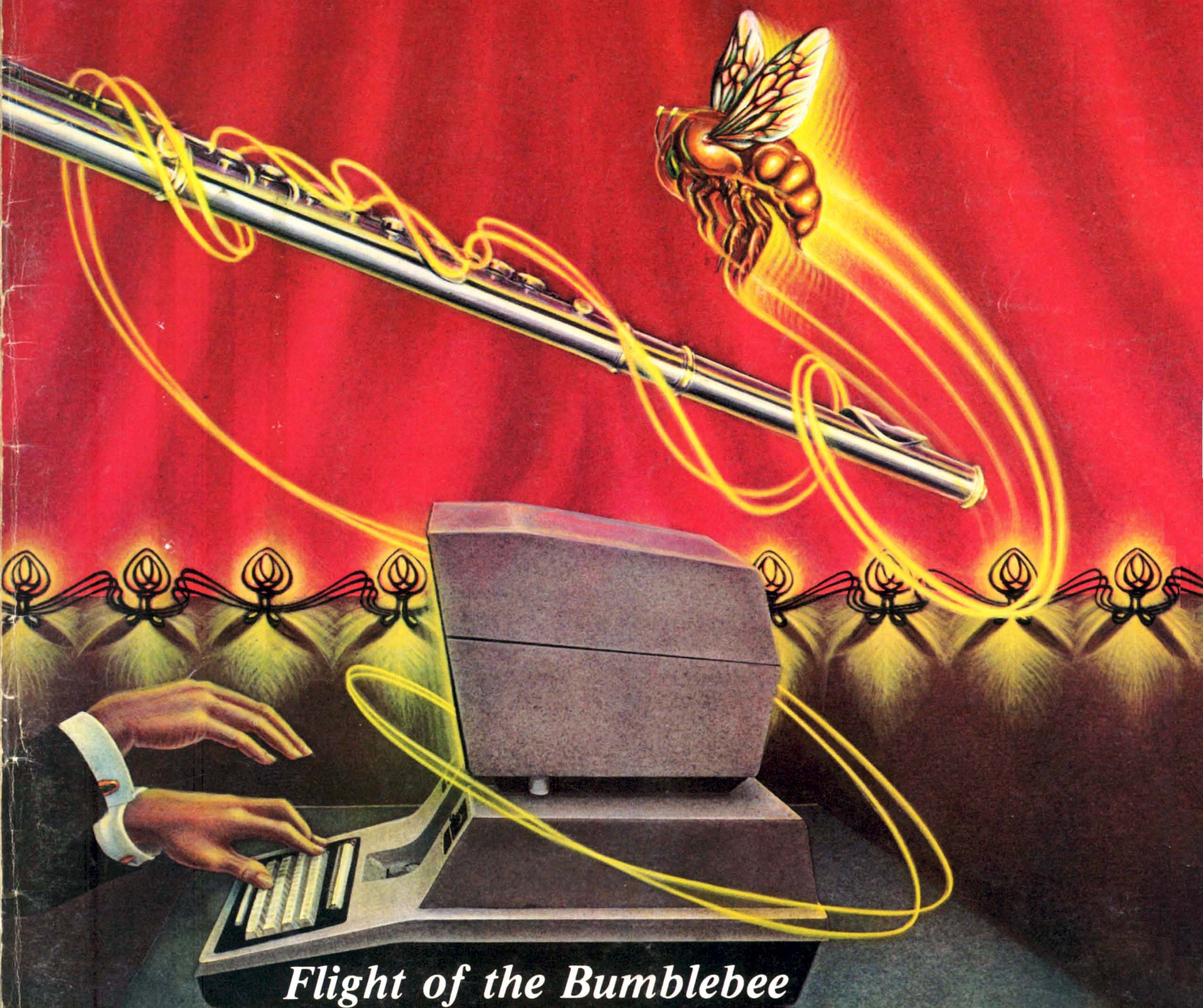


November 1981 Vol. 5, No. 2

SoftSideTM

THREE DOLLARS

Special Issue: Music in the Micro



Flight of the Bumblebee

William A. Giese '81

First it will awe you...

Step beyond TRS-80™ entertainment software as you know it with **diversions thru Envyrn™**, the new bimonthly series. Boot up for the first time and you'll be astonished by what you find:

Experience three scales of graphics — take in a long, wide view of your **Envyrnment™**, then step closer and still closer to zero in on detail.

Travel at your chosen speed — when your life depends on escaping, sprint. When you want to take a closer look at your **Envyrnment™**, stalk.

diversions™ intelligence will challenge you. Only the **Envyrnment™** knows the location of a black hole, whether a bridge will collapse if you're carrying too much weight, or which walls are really secret entryways. It's up to you to uncover the hidden mysteries.

Wait until you discover the magnitude of the area open to your exploration! **diversions™** maps and diagrams are larger than any you've experienced. As you approach a boundary, your **Envyrnment™** may load yet another realm from the disk, much like the unfolding of a road map.

After you've examined the features that make **diversions™** a revolution in computer entertainment, then discover the thrill of experiencing! Imagine yourself commander of the starship Enterprise, a paladin on a medieval quest, or the pilot of a mission flying over the Himalayas.

Subscribe for a full year and receive six **Envyrnment™** modules with **diversions thru Envyrn™** the magazine — not a manual, but a full-sized handbook with history, hints and information that will intensify your experience of **Envyrnment™**.

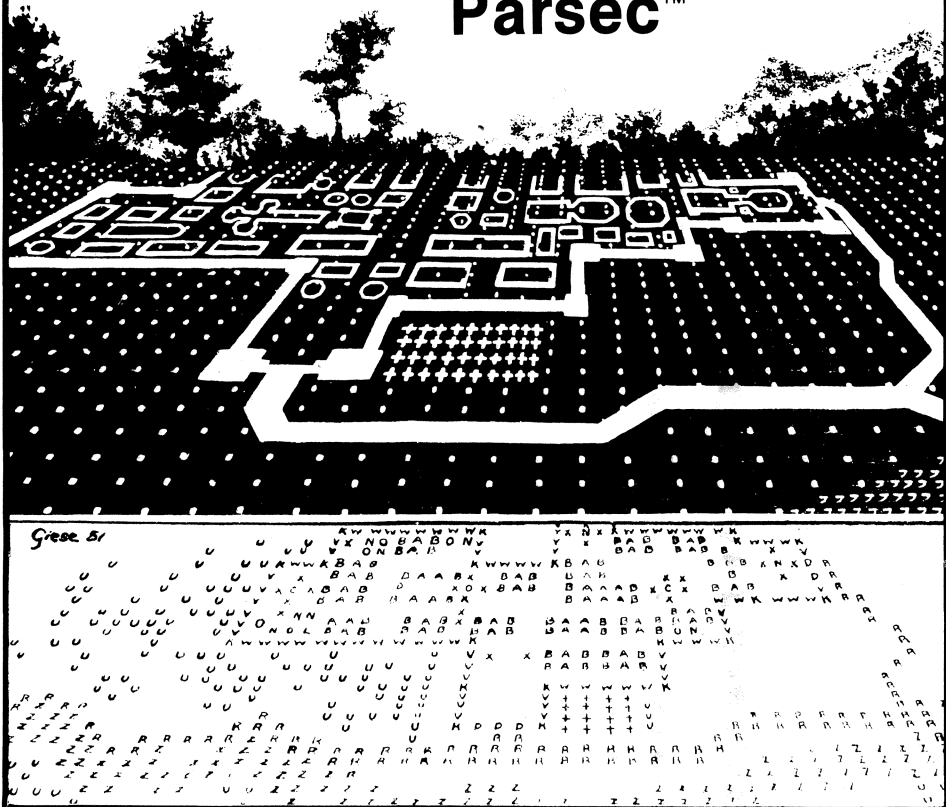
Send \$60 for one year (six modules). You must have a 48K TRS-80™ Model I or III with disk. You may want to take advantage of our special introductory offer: \$20 for the first **Envyrnment™**, **Parsec™**, what *Star Trek* has always tried to be. **diversions thru Envyrn™** Apple edition, will be available in 1982.

Entertainment software will never be the same!

diversions™

thru Envyrn

Premiere Issue Parsec™



Send to:

diversions™

thru Envyrn

6 South Street Milford, NH 03055

- One year subscription (6 modules) I enclose \$60.00
 Sample **Envyrnment™** **Parsec™** Only I enclose \$20.00

MasterCard Visa

Card# _____

Interbank# _____ Exp. Date _____

Name _____

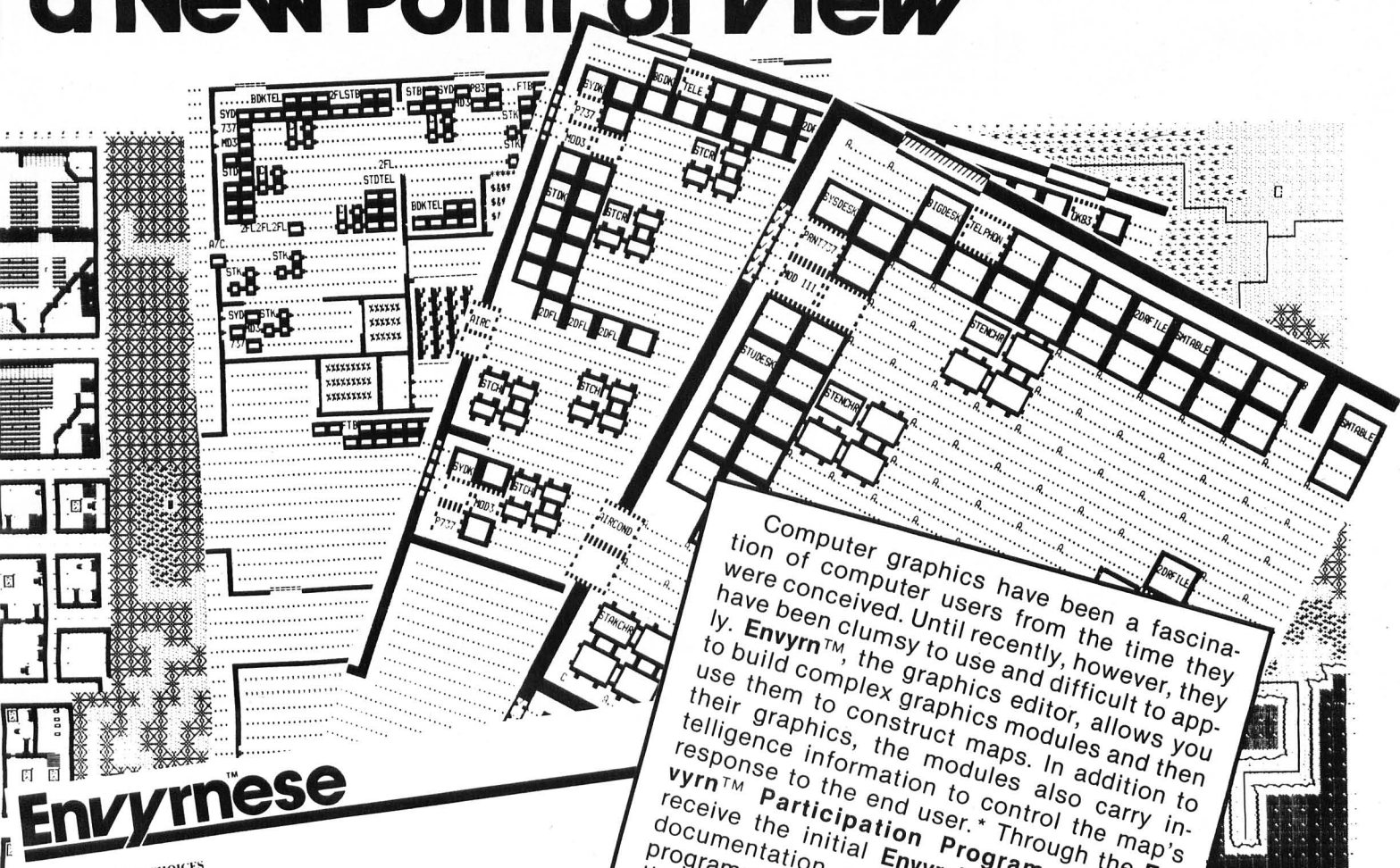
Address _____

City _____

State _____ Zip _____

I own a 48K TRS-80™ Model I Model III

Envyrn™ Graphics Have a New Point of View



Envyrnese

ENVYRN™ SCREEN CHOICES

by Roger W. Robitaille, Sr.

The TRS-80™'s graphics capability is, essentially, a special set of pixel combinations that are treated by a computer just as if they were letters, punctuation, or numbers. Many screen options are available to this group which could not be supported in computers with more complex and exclusive graphics capabilities. In other words, the TRS-80™'s graphics capability is primitive. However, it does present special opportunities, along with its restrictions. Much of the following will only pertain to computers which have graphics character set accessible in text mode.

MAIN VIEWING SCREEN — Almost every Envyrn™ application will make some use of its ability to tie map tiles together into some larger visual image. The Main Viewing Screen (MVS) on the TRS-80™ is one of four configurations by which related map tiles can be displayed together.

SCREEN PAGE — A Screen Page is a user-defined set of information windows which together will define what information will be presented on the screen, and where and how it will be shown.

There are five practical MVS choices in the TRS-80™ graphics structure. They are:

- 0% Text Screen — Small (18 chr x 6 line) MVS in the upper left corner of the screen.
- 10% Peek Screen — Medium size (30 chr x 10 line) MVS in the upper left corner of the screen.
- 30% Travel Screen — Large (48 chr x 16 line) MVS with 75% Look Screen — Large (48 chr x 16 line) MVS with a narrow (16 chr x 16 line) area still available to be assigned other information.
- 95% Full Screen — Nearly the entire useable screen is dedicated to being the MVS. At (@) location ### is the information point of any keyboard activity.

Figure 1

SCREEN KEY
Resolution Table for MVSs

Tile Size	Text	Peek	Travel	Look	Full
1x3	N/A	6x6*	10x10*	16x16*	16x20*
2x6	N/A	3x3*	5x5*	8x8*	8x10*
3x9	N/A	2x2*	3x3(90%)*	5x5(91%)*	6x5(82%)*
4x12	N/A	1x1(47%)*	2x2(63%)*	4x4*	4x5*
5x15	N/A	1x1(69%)*	2x2*	3x3(91%)*	3x4*

* — 100% useful (nn%) — % useful

As I examine the table in Figure 1, I notice that the two tile sizes with the lowest resolution are completely efficient under all MVSs. The other three tile sizes are the most commonly used tile sizes. The larger tile sizes will be selected. I hope that future Envyrn™ will allow the MVSs to be completely efficient under all MVSs.

Figure 2

```
MVS L.A.
18 11 11 11
2EEEEE#####TTT
3EEEEE#####TTT
4EEEEE#####TTT
5EEEEE#####TTT
#####TTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTT
```

Key:
 E — In use under all MVSs.
 T — In use under all except Peek Screen.
 L — In use under Look and Full Screens.
 A — In use as MVS only under Full Screen.
 F — In use as MVS only under Full Screen.
 X — Always in use as Text.
 — Border area where text will reject and graphics will reject graphics window offered.

Computer graphics have been a fascination of computer users from the time they were conceived. Until recently, however, they have been clumsy to use and difficult to apply. Envyrn™, the graphics editor, allows you to build complex graphics modules and then use them to construct maps. In addition to their graphics, the modules also carry intelligence information to control the map's response to the end user.* Through the Envyrn™ Participation Program, you will receive the initial Envyrn™ editor, existing documentation, and a full year of updates, programming notes and suggested applications. In addition, **Envyrnese™**, a new column to appear in **SoftSide™**, a new monthly news of Envyrn's™ development, will provide you only, but will soon also be available for the Apple and Atari microcomputers. The cost to enter the **Envyrn™ Participation Program** is \$200/year. Please refer to the bind-in card in this issue to order.

You've been struggling with graphics applications long enough. Now, with Envyrn™, graphics have a new point of view — one you should have.

For further information concerning Envyrn's™ logic and development, see **SoftSide**, October, 1981.

*TRS-80™ is a registered trademark of the Tandy Corporation.

Ten reasons why your floppy disk should be a BASF FlexyDisk®.



More than four decades of experience in magnetic media – BASF invented magnetic recording tape, the forerunner of today's wide range of magnetic media, back in 1934, and was the first independent manufacturer of IBM-compatible floppy disks.

Tough Tyvek sleeve – no paper dust, no static electricity.

Special self-cleaning jacket and liner help eliminate data errors and media wear and tear.

Packaging to suit your requirements – standard flip-top box, Kasette 10® storage case, or bulk pack.

Center hole diameter punched to more accurate standards than industry specifications, for top performance.

100% certification – every single disk is tested at thresholds 2-3 times higher than system requirements, to be 100% error-free.

Bi-axially oriented polyester substrate – for uniform and reliable performance year after year.

For the name of your nearest supplier, write BASF Systems, Crosby Drive, Bedford, MA 01730, or call 617-271-4030.

Cross-linked oxide coating – for low head wear and long trouble-free media life.

Total capability – one of two manufacturers in the world that makes both 8" and 5.25" models, has tape and disk experience, and manufactures floppy disk drives.

Double lubrication – lubricants both in the formula and on the disk surface, to minimize media wear due to head friction.



BASF

Floppy Disks Mag Cards Cassettes Computer Tapes Disk Packs Computer Peripherals

SoftSide™

Volume V — Number Two

PUBLISHER
Roger W. Robitaille, Sr.

MANAGING EDITOR
Randal L. Kottwitz

PROGRAMMING EDITOR
Jon Voskuil

EDITORIAL DEPARTMENT
Scott Adams
Rich Bouchard
Dean F. H. Macy
Lance Micklus
Mark Pelczarski
Joan Truckenbrod
Ed Umlor
Alan J. Zett

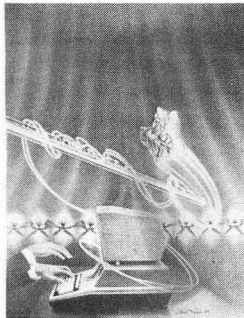
PRODUCTION MANAGER
ADVERTISING
Nancy Lapointe

PRODUCTION DEPARTMENT
Lynda Fedas
Tom Stanton

CUSTOMER SERVICE
Nancy Macy

DEALER SALES
Mary George

STAFF
Kathleen Boucher
Pam Demmons
Stephen Justus
Kathie Maloof
Doris Miller
Christine Spade
Anmar William
Gary Young



The Flight of the Bumblebee is the subject of this month's cover. A musically inclined user controls a whimsical brass bumblebee. The bumblebee, in turn, plays his own tune on the flute. Illustration by Bill Giese.

SoftSide is published each month by SoftSide Publications, 6 South Street, Milford, New Hampshire 03055. Telephone 603-673-0585. Second class postage paid Milford, New Hampshire and pending at additional mailing offices. ISSN: 0274-8630. Subscription rates: USA, \$30.00/12 issues. USA First Class APO, FPO, Canada, Mexico, \$40.00/12 issues. Overseas air mail: \$62.00/12 issues. Media subscription rates: Magazine and cassette, \$75.00/12 months. Magazine and disk, \$125.00/12 months. APO, FPO, Canada, Mexico, (add), \$36.00/12 months. All remittances must be in U.S. funds. Mail subscription inquiries to SoftSide Publications, P.O. Box 68, Milford, New Hampshire 03055. Entire contents copyright 1981 SoftSide Publications. All rights reserved.

POSTMASTER - send address changes to:

SoftSide Publications
515 Abbot Drive
Broomall, PA 19008

If you have not received your November issue of SoftSide by Nov. 6, contact SoftSide Publications, 515 Abbot Drive, Broomall, PA, 19008 or call 1-800-345-8112 (In PA call 1-800-662-2444).

ARTICLES

- 16 Badly Needed — Good Educational Programs**
A guide for building educational software William D. Hedges
- 75 The Tone Envelope**
The micro container for sound Christopher U. Light

REVIEWS

- 76 Orchestra-85** Robb Murray
- 82 Atari Music Composer** Randal L. Kottwitz
- 84 Alf II vs. Musicsystem** Christopher U. Light
- 94 Apple Resource Directories** Jon Voskuil

APPLE™, ATARI™, TRS-80™ PROGRAMS

- 23 Flight of the Bumblebee**
A new buzzword for the micro industry William Morris & John Cope
- 31 Melody Dice**
Roll up your own Scott Joplin medley Gary Cage

SOFTSIDE DV

- 47 APPLE™ — National Anthems** Fred Pence
- 47 ATARI™ — Volleyball** Bradley J. Bell
- 48 TRS-80™ — Mean Checkers Machine** Lance Micklus

TRS-80™ PROGRAMS

- 50 GAMEPLAY/BAS**
The first Envyrn™ interpreter Roger W. Robitaille, Sr.
- 55 Music Editor**
Musical notes for the TRS-80 Richard Lesh

APPLE™ PROGRAM

- 61 Music Machine**
Musical notes for the Apple Jon Voskuil

ATARI™ PROGRAM

- 67 Music Programmer**
Musical notes for the Atari John Rush Elkins

DEPARTMENTS

- 5 Editorial** Dean F. H. Macy
- 6 Input** From Our Readers
- 7 Hints and Enhancements** From Our Readers
- 8 About This Issue** Jon Voskuil
- 10 Calendar** Kathleen Boucher
- 11 Outgoing Mail** Randal L. Kottwitz
- 15 Say Yoho** Scott Adams
- 19 The Sensuous Programmer** "J"
- 21 Bugs, Worms, and Other Undesirables** Editors
- 52 Commanding BASIC** Alan J. Zett
- 95 Hardware Corner** Ed Umlor
- 96 Machine Head** Spyder Webb

*TRS-80™, Apple™, and Atari™, are registered trademarks of The Tandy Corporation, The Apple Computer Company, and Warner Communications.

SoftSide DV, the magazine of the future, is here!

If your computer could pick a magazine, wouldn't it prefer one in its own language? Now there's one available.

SoftSide DV is an enhancement of the **SoftSide** you have in your hands.

SoftSide DV contains not only the complete programs listed in every month's issue of **SoftSide**, but additional programs of every conceivable type, as well — multiple and Machine Language programs, modified languages, ongoing modular programs and software so extensive, it would take an entire issue of **SoftSide** just to print the code. Only the documentation for these programs will appear in **SoftSide Magazine**, **NOT** the code.

Feel as though you're missing something? You are! But, you needn't miss out on another issue. **SoftSide DV** is now available for Apple™, Atari™ and the TRS-80™. The cost to you — \$125 for 12 magazines and 12 disks, packed with some of the best software available, all delivered to your home in the next year. For orders outside the USA, please add \$36. For your convenience, we offer an installment payment plan for VISA and MasterCard holders: You pay only \$32.50 per quarter (a total of \$130, which includes a \$5 billing charge). Please use the bind-in card in this issue to order.

Computerists are offered the rare opportunity of marching into a new frontier. Advance to the front of the parade by subscribing to **SoftSide DV**, the magazine of the future, available today!





Computer Music — The State of the Art

by Dean F.H. Macy (With acknowledgments to J. Pfeiffer)

Riddle: What is the sound of music?
Answer: The music of sound.

Most composers program sound in dots and directions on paper. The common vehicle or computer — arms pulling horse hair over strings, lungs forcing air through and over metal and wooden tubes; fingers shortening and lengthening strings or columns of metal and wood; hands with mallets hitting stretched skins, wooden bars, and metal plates — an orchestra of superbly specialized gymnasts! A hundred or so minds and countless reflexes seek to translate those dots and directions into the sound that will fulfill the composer's score. And in succeeding, they create the music.

Imagine, if you can, sitting in a concert hall; the lights are dimmed and you, with your eyes closed, are absorbing the finale of Wagner's magnificent work, *Tannhauser*.

The orchestra voices shout forth — the strings build to a crescendo, horns mellow, echoing. Cymbals crash, kettle drums roll and the beauty of *Pilgrims Chorus* fills the heart and soul with awe and wonder. Now, 300 male voices lift in harmony, and together, symphony and chorus fill your whole being. As the dying chords reverberate throughout the hall, you stand, tears streaming down your face. You are elevated to the heights of heaven from the performance and you applaud and call, "Bravo! Bravo!"

You open your eyes, and through the mist of tears you see a single man on stage surrounded by electronic equipment, and a microcomputer.

Impossible? Not with artists like Isao Tomita combining the versatility of the microcomputer with other computerized music synthesizers.

Generally, one imagines computer music to be a cacophony of beeps, buzzes and outlandish tones, not the richness of the strings and choral ensembles of a symphony orchestra. Tomita's music of sound is produced by this dignified discipline. Yes, it is tedious: A synergistic process of science and music is a laboratory exercise — analyze, imagine, search, test,

modify, evaluate — carried out in the same ambiance as that surrounding the composer's desk.

Isao Tomita says, "I have struggled with a microcomputer. I say struggled because a computer is beautifully precise, but I wanted to use it to produce musical results — in other words, as a musical instrument. How could a computer keyboard compare with that of a grand piano? I came to realize that those keys could produce an almost limitless number of combinations, each of which is a signal that could determine a characteristic of sound, pitch, attack time, decay time, texture, voice and loudness. And the computer can



WORLD OF CULTURE

be programmed to change any or all these with incredible speed.

"The computer thus produces a sequence of signals that control the sound production of a synthesizer. It is something like millions of little hands rapidly changing all the synthesizer connections to produce a vast variety of sounds. My musical images must be coded by numbers to direct those hands to manipulate the synthesizer."

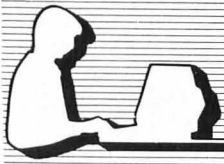
Using his computer, Tomita records a line, a phrase or a measure, on a single track of tape. Then another phrase of different tonal quality, and another. Repeated processes build up layers of harmony and rhythm on suc-

cessive tracks. Then adding more layers, blending, mixing, modifying, to achieve even more possibilities. Then on top of that reverberating; folding sounds with echo; using phase shifters, ring generators and electro-acoustic sound. Finally, a complexity of color — a sound of the composer's music created as directly as he wrote it.

All of Tomita's creations are sonic encounters — composers' musical inventions transcribed into sound colors capable of being generated by conventional instruments. Yes, Tomita imitates with his computer, but with the infinity of resources at his command, he can blend glittering threads of such natural sounds as a human whistle, wind rustling leaves, insects humming, waterfalls, rain, rushing streams, and surf pounding on the beach, with conventional instrument voices producing sounds of music too glorious to be imagined by finite minds.

Tomita explains, "I often use the analogy of an artist's palette to explain about synthesizer music. First an idea comes to my mind, and in order to express that idea in reality I use the synthesizer. This is almost like the painter who mixes his own colors on his palette using paints of some original colors to express the images he has in his mind. I try to create certain four-dimensional images in space, and I imagine in my mind a hall that can hold about 1,000 people. Therefore, space is integral and becomes the basis for my sound images. Microcomputers are being used readily nowadays, and we will soon find ourselves in an age where such computers will order most aspects of existence. Good or evil? Microcomputers are used extensively in the making of my music and suggest that there is hope that civilization will not be destroyed by its own technology."

Indeed, after several hours of listening to the computer music of Tomita I find I have overcome the complexities and realities of daily life. I can go beyond physical limitations and contact my fantasy-imaginings. Through the genius of masters like Tomita, I can reach into limitless space, touch the super intellect, be any object or being and cast myself, all powerful, into the universe.



MICRO TO VIDEO

Dear *SoftSide*,

The following is a suggestion that perhaps you could use to start the ball rolling. As of today, I know of only one firm that sells a board capable of modifying the Apple output signal to acceptable NTSC standard. Of course, by producing a standard NTSC signal, a person could videotape straight through to a VHS recorder. All of a sudden, beautiful titles could be typed-up using "higher text", etc., and added to home VHS recordings.

Adwar Video out of New York currently sells a board for \$300 — a very fantastic and prohibitive price. I don't believe that the price need be over \$100. But they are alone at the moment and no competition means...\$300 it is.

Please explore the possibilities of this board and what it entails. Perhaps you can get someone started on a reasonably priced unit. There are certainly a large number of people who own both Apples and video recorders.

Ronald O'Laughlin
Bay City, MI

Editor's Reply: The cost of manufacturing is frequently minor in comparison to the cost of marketing (finding, selling, and supporting customers). High demand is the surest way to low prices. Such demand is seldom a secret and ultimately results in lower prices. Note the cost of printers, disk drives, etc.

EDUCATIONAL PROGRAMS

Dear *SoftSide*,

As an Apple II owner, I am sending this note to let you know I like my monthly issues of *SoftSide*. The games are good and I have many from the pages of this fine magazine.

One reason I bought my computer was to improve my (and my children's) education. They are learning to type in the programs and I think they are also learning other things that will be needed when they go to work (as adults).

The main point I am making is I would like to see some educational programs in this monthly magazine. I wrote a couple but they aren't polished enough for publication. Maybe others have some that are. How do others feel?

Jim Willis
W. Monroe, LA

Editor's Reply: The problem of educational software scarcity is acute. The educational marketplace is a very difficult one to serve and, in many cases, is not willing to pay the price necessary for software development. See *Badly Needed: Good Educational Programs in this issue for additional information.*

DISK VERSION FOR ADDITIONAL SYSTEMS

Dear *SoftSide*,

Enclosed is my check for *SoftSide-DV*. I am a recent subscriber to *SoftSide*. Formerly, I would buy an issue now and then, and a program or two, for use with my S-80 (three years old). My first issue of *SoftSide* (as a subscriber) was June, 1981. The July and August issues confirmed my choice — **SOFTSIDE IS GREAT!**

The manner in which you cover S-80, Atari and Apple computers helped me to decide to purchase an Atari 800 (as a second computer) for home use, as the S-80 has found a niche down at the office.

(D)isk (V)ersion...Hooray! I have struggled through typing several of the long games, and found "bugs," some of which were never fixed. These were added to a pile of programs that were just filed away. Other games just looked too involved for the time I had, and these were also just filed away.

Here's my problem. I now have two 32K computers with disk drives. The Atari graphics are fantastic! The S-80 is a familiar machine to me. Can my *SoftSide-DV* subscription alternate systems?

Donald C. Hennessy
Massapequa, NY

Editor's Reply: We are now making *SoftSide* Disk Version (disk only, no magazine) available for second and third systems. The cost will be \$100 per additional system when ordered concurrently with the initial subscription.

ROSES AND THORNS

Dear *SoftSide*,

I wish to speak out for those of us who are not experts at programming. There are some of us out there who use home computers for pure enjoyment. We do not seek to downgrade other brands than the one we own, believing everyone is unique in their own interests and abilities.

As the owner of a Level II S-80, I do occasionally wish that it were endowed with color and Hi-Res graphics, but this does not make the machine inferior in my eyes. I made that choice when I purchased the computer. Perhaps in the future I may be able to invest in an Apple or Atari, but I will always give equal time for whatever machine I own. The S-80 may not be perfect, but neither is the Apple; I have spent many hours typing on the Apple at my high school, so I neither condemn nor worship the Apple.

The S-80 and the Apple are both good machines if used for the right purposes and in the right mind. We computer hobbyists should work together and not be at each other's throats constantly.

I think *SoftSide's* new format is a tribute to programmers in that respect. Those readers who feel that *SoftSide* has betrayed any one of the computers it represents should perhaps subscribe to another magazine. On the other hand, I am entering the ministry and some people feel I have a naive view of the world. That may be so, but I still feel that we programmers should try harder to get along.

I hope that you continue to step boldly in the new direction you have taken and that others will someday realize that you were right all along. Keep it up, *SoftSide*.

Kelly Harper
Baytown, TX

Editor's Reply: Amen!

Dear *SoftSide*,

In the *Input* column in your August issue, Mr. Joseph Teller's idea for a minor modification column is an excellent idea! I agree wholeheartedly. Also, I approve of the cover artwork idea. Could you put the mailing label on the back side, not to mess up the picture?

I would like to see a thesaurus column for changing commands from one BASIC to another. For instance, what do the Apple commands VTAB and HTAB correspond to in Atari and Level II BASIC?

I have been a *SoftSide* subscriber for about seven months and this is probably the best software magazine there is. The only thing I have found wrong with it is that every once in a while, you forget to list *Bugs*, *Worms*, and *Other Undesirables* in the table of contents. I could have pulled my hair out looking for the corrections to *Strategy Strike!* Other than that, all I have to say is, keep up the good work!

Tim Gray
Burke, VA


Editor's Reply: You've anticipated many of the changes being made in *SoftSide*. See our new *Hints and Enhancements* column and expanded table of contents. We're sorry we cannot place the labels on the back cover as it might cover important information in a client's advertisement. A series of articles on translations is forthcoming. In the meantime, the second edition of David Lien's *BASIC Handbook* should be a great tool for you.

Dear *SoftSide*,

Congratulations! Your magazine is the best thing to come along since my Atari. I must commend you on your excellent programs, your wide variety, and your commitment to Atari owners.

Ever since the first issue I picked up at the computer store (I later subscribed), I can't wait for the next issue to arrive in the mail. There are not many magazines who recognize the Atari and its capabilities.

Eric Bass
Spring Valley, NY



INPUT POLICY

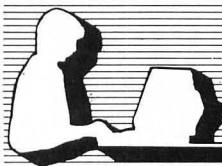
SoftSide Magazine welcomes your comments and thoughts on both the magazine and the field of microcomputing. We try to publish as many of our readers' letters each issue as we can.

For the sake of clarity and legibility, all letters should be typewritten and double-spaced. Send your letters to:

SoftSide Publications,
Input
6 South St.,
Milford, N.H. 03055

We reserve the right to edit any letters prior to publication.





Hints & Enhancements

APPLE PUFF

Here is a short program to demonstrate how one can "puff" a message across the screen in Integer BASIC. Two string variables are dimensioned in Line 10. A\$ is initialized in Line 20 and printed in Line 30. Line 40 is a delay loop, to make the message "crawl" faster; slower would be to change the value of 35 to other values. Line 50 then equates characters 1-39 of B\$ with characters 2-40 of A\$. Character 1 of A\$ is tacked on the end of B\$ in Line 60, completing what was, in effect, a one-character circular shift. B\$ is then copied into A\$. We go back and print the new string and repeat the process forever.

PUFF LISTING

```
10 DIM A$(40),B$(40)
20 A$ = "ANY 40-CHARACTER STRING
GOES HERE"
30 CALL -936:VTAB 10:PRINT A$
40 FOR U=1 TO 35:NEXT U
50 B$ = A$(2,40)
60 B$(40) = A$(1,1)
70 A$ = B$
80 GOTO 30
90 END
```

Randi J. Rost
Fairmont, MN

NEW DUNGEON FOR QUEST 1

I decided to create a new dungeon for *Quest 1* because I liked the program so much. This can be used for all versions of the program. Just type in these new data statements and there you have it.

```
115 DATA 2,41,2,36,11,0,0,1
120 DATA 2,1,8,3,12,1,1,2
125 DATA 1,0,0,4,2,3,2,7
130 DATA 2,0,5,0,3,2,1,2
135 DATA 1,4,6,9,0,6,3,10
140 DATA 2,5,0,0,7,4,1,5
145 DATA 1,0,0,6,8,7,1,6
150 DATA 1,2,0,7,0,5,2,3
155 DATA 1,0,0,10,5,1,3,7
160 DATA 2,0,0,53,9,7,3,6
165 DATA 1,0,12,1,35,2,1,2
170 DATA 2,11,13,2,0,4,2,1
175 DATA 1,12,14,0,0,5,2,4
180 DATA 2,13,0,15,0,7,4,8
185 DATA 1,0,0,16,14,3,1,1
190 DATA 2,0,23,17,15,6,2,7
195 DATA 2,0,19,18,16,8,1,1
200 DATA 1,0,20,0,17,1,2,4
205 DATA 2,17,0,20,0,6,3,8
210 DATA 2,18,21,0,19,4,1,1
215 DATA 2,20,0,0,22,5,2,12
220 DATA 1,0,0,21,23,1,2,3
225 DATA 2,16,0,22,24,7,1,1
240 DATA 1,0,0,23,25,8,1,2
250 DATA 2,0,0,24,26,8,4,9
260 DATA 1,27,0,25,0,2,1,1
270 DATA 2,30,26,0,28,5,2,7
280 DATA 2,29,0,27,45,6,1,2
290 DATA 2,0,28,30,0,7,5,9
300 DATA 1,31,27,0,29,4,2,5
310 DATA 1,32,30,0,0,1,2,3
320 DATA 2,0,31,34,33,5,3,8
330 DATA 1,0,0,32,0,3,1,4
340 DATA 2,35,0,0,32,1,1,1
350 DATA 2,36,34,11,0,3,2,6
360 DATA 2,40,35,37,1,8,5,1
```

```
370 DATA 1,39,0,0,36,2,1,2
380 DATA 2,0,0,0,39,6,3,8
390 DATA 1,46,37,38,0,1,2,5
400 DATA 1,0,36,0,41,7,1,3
410 DATA 1,42,1,40,0,1,1,1
420 DATA 2,43,41,0,0,4,1,2
430 DATA 1,50,42,0,0,5,2,5
440 DATA 2,48,0,46,0,2,6,8
450 DATA 2,0,0,28,47,6,3,8
460 DATA 2,47,39,0,44,3,2,7
470 DATA 1,0,46,45,48,1,2,1
480 DATA 2,49,44,47,0,4,1,2
490 DATA 1,0,48,0,50,1,1,1
500 DATA 2,0,43,49,51,8,3,8
510 DATA 1,0,52,50,0,5,1,3
520 DATA 2,51,54,0,0,2,2,4
530 DATA 2,0,0,54,10,7,4,9
540 DATA 1,52,55,0,53,4,5,7
550 DATA 1,54,0,0,56,7,1,11
560 DATA 2,0,0,55,57,1,2,6
570 DATA 1,0,58,56,0,5,3,7
580 DATA 2,57,0,0,0,8,10,9
```

Tigre Wenrich
Santa Ana, CA

ATARI HINTS

Have you ever wanted to delete entire blocks of statements from your programs and mourned the absence of a DELETE statement in Atari BASIC? If so, you have probably learned that deleting one statement at a time by typing the line number followed by a return is a splendid way to introduce errors into your program. Fortunately, there is a fairly easy way to accomplish this task using the LIST and ENTER commands.

Suppose you have a 1000-line program with a subroutine at Lines 800 to 850 that you want to delete. Using the LIST command, you can write to storage only the portions of the program you wish to save. Using this example, you would use the command: LIST "C":0,799 if using cassette as storage, or LIST "D:=PART 1":0,799 if using a disk drive and you wanted to assign the name 'PART 1' to this part of the program. Since cassette programs cannot be named in Atari BASIC, you would need to use the tape counter to keep track of the program. Next you can save the remainder of the program using the same LIST command, but instead would enter the remaining line numbers, for example, LIST "C":851,1000.

So far, you have managed to store the parts of your program that you wish to retain. You can now splice the two segments back together by simultaneously entering them back into memory. This is done by using the ENTER (rather than the usual LOAD) command. ENTER, unlike LOAD, does not clear programmable memory automatically, so you must first type NEW to clear the RAM. Then you can simply use ENTER to put the stored segments of your program back into memory. The commands would be:
ENTER "C:" for cassette
ENTER "D:PART 1" for disk

Note that in our example you would have to use the ENTER command twice (once for each segment of the stored program). Since ENTER does not clear RAM, it allows the two program segments to be entered into memory simultaneously. Once the segments are reunited, you can save them together using the more common SAVE command.

This feature allows other flexibility, such as joining together two programs, provided that there is no duplication of line numbers between the two programs. Other applications would be for using different sets of data statements for the same program to process, or in writing large programs with different people uniting subroutines or subprograms to be joined together. I'm sure there are uses I haven't even considered, yet that are possible — use your imagination!

There are sometimes circumstances in which you find yourself writing a subroutine in a program which is similar to, but not identical to, an already written subroutine. In this setting, it would be nice to have a function which duplicates a line of code, so that you can then edit it to make the changes you want, rather than having to retype what is very similar or almost identical. Atari BASIC has a feature (which is probably accidental) that you can use for this purpose. If you use the screen editor to change the line number of an existing line, an interesting thing happens — not only do you have a line of code with the new line number, but also the old line of code remains in memory with its original line number. You now have two lines of identical code with different line numbers, and it is fairly simple (although it requires some care) to modify your new line of code to conform to your new subroutine. This provides a handy way to copy a subroutine and then make whatever modifications are necessary. The warning here is to be sure to enter the correct line numbers, or you may end up with a stray line of code playing havoc with your program.

J. Arthur Gleiner
Rochester, NY

LUNAR MISSION MODIFICATIONS

Lunar Mission is an excellent program, however I have added the following changes that might be of interest to other subscribers. The changes add skill levels and can make the program easier for younger players.

Add or change:

```
1 GRAPHICS 0: POKE 752,1:?:?:?
2 ?" LUNAR LANDER":?:?:?
3 ?" Control Speed With "
4 ?" Reverse Thrust":?:?:?
5 ?" Score Is Determined By Skill Level."
6 ??:
7 ?" Input Level Of Difficulty":?:?
8 ?" 1 (Easy) to 10 (Hard)":INPUT N
9 A = N/100
```

At the end of Line 100, make $YS = YS + A$
At the end of Line 410, make $S = 100 + N * 10$
At the end of Line 420, make $S = 50 + N * 10$

Skill level and score are determined by the number from 1 to 10 that is input at Line 8. Line 9 gives the right decimal factor for speed of entry. With the addition of $+N * 10$ to Lines 410 and 420, a score vs. difficulty factor is added.

Other changes could include a change in the amount of starting fuel for a longer game, and a bonus of extra fuel for higher scores.

Kenneth Parsons
Linn Creek, MO

continued on page 9



About This Issue

by Jon Voskuil

❑ You haven't lived until you've worked for a month or so in *SoftSide's* programming department, as the programs for this month's music issue were being evaluated, tested, translated and debugged. The elite few who have had this unique auditory experience will never be the same.

❑ A hymn writer once described the planetary motions as "the music of the spheres;" and ever since Julie Andrews was filmed running atop a grassy Austrian mountain, we've all been accustomed to the hills being alive with the sound of music. But not until very recently have microcomputers joined the orchestra. Hardly as majestic as the "chorus of the morning stars," and perhaps lacking the pastoral splendor of an Austrian mountainside (but see Dean Macy's editorial), microcomputer music is nevertheless a phenomenon which is here to stay.

❑ Not all of us are musically literate. My only musical instrument, until recently, has been the recorder. (No, not the wooden kind — the cassette kind.) But for all of us, however musically talented or inept, the microcomputer opens up some new possibilities for musical expression. Most of us appreciate the tasteful use of sound in computer programs, and in this issue you will find a variety of software and articles which will both tickle your eardrums and help you train your computer to do new musical tricks.

❑ *Flight of the Bumblebee* is our latest musical/graphics fantasy from Morris and Cope, in versions for all three computer systems. The classic Rimsky-Korsakoff tune is accompanied by a lovable animated bumblebee which will win your heart.

❑ From Russian opera we turn to American ragtime, with the music that Scott Joplin never wrote, but might have. *Melody Dice* creates melodies using measures from Joplin's music, allowing your computer to play and display your own (re)arrangements by tossing dice. We think you'll agree that this is a unique idea for a musical game, and will be delighted that it's available for your computer.

❑ Then there are three programs, all variations on a theme, which require a certain amount of diligence and concentration to utilize. *Music Editor*, *Music Programmer*, and *Music Machine* are composer/editor programs for the S-80, Atari, and Apple, respectively. They all allow you to enter, edit, and play tunes to your heart's content (and maybe to your family's despair, depending on your musical aptitude).

❑ If you're interested in doing some experimenting with sound on your S-80, *COMMANDing BASIC* will provide you with a method of enhancing BASIC so that you have a built-in SOUND command, usable in any of your own programs. Alan promises further BASIC enhancements in future issues, as well.

❑ For those who may be considering one of the numerous pieces of software or hardware being marketed to improve your computer's musical talents, the three reviews herein may provide valuable input. Atari's *Music Composer* cartridge, the Alf and Mountain Hardware music boards for the Apple, and *Orchestra-85* for the S-80, are all checked out for their merits and demerits.

❑ For those whose fancy has been captured by the challenging possibilities of *Envyrn™*, as introduced in the October issue, *GAMEPLAY/BAS* makes its first appearance in print. After having a month to think about the *I-string* concept, we hope you're eager to plunge in and begin creating for this *Envyrn™* interpreter.

❑ This month our Atari Disk Version subscribers join our S-80 and Apple DV subscribers in receiving bonus programs on their disks. The Atari extra is an excellent translation of *Volleyball*, an Apple original from last July. Apple DV subscribers are receiving the colorful and tuneful program, *National Anthems*, and the S-80 version of DV contains Lance Micklus' original *Mean Checkers Machine*.

❑ All this, plus regular (and irregular) features, columns, letters, hints, bugs (ugh), and other goodies await you within. ☺

REWARD! TRANSLATION APPEAL

SoftSide will give a \$100 software certificate to the author of the best translation of a past *SoftSide* feature program. Each month we will publish at least one of these translations. Your portfolio will be enhanced to say nothing of your software library!

We will accept entries for all past *SoftSide* programs at any time. However, we suggest you submit translations of recent programs within three months of their original publication date for maximum consideration. Entries must be submitted on tape or disk, accompanied by complete documentation. Please enclose a self-addressed stamped envelope if you would like your entry returned.

The quality of each translation will be judged by the *SoftSide* editorial staff and prizes will be awarded at the time of publication. ☺

ONE LINERS

Scattered throughout the pages of this issue, you will find some very short programs called "One Liners". You can contribute to this department by following three rules and sending your contribution to:

S-80
Atari (pick one)
Apple

ONE LINERS
c/o *SoftSide* Magazine
6 South Street
Milford, New Hampshire 03055

RULES:

1) The programs must be written as a single line of BASIC.

2) The program must be self-contained. Make no assumptions about DIMENSIONS, available string space, current graphics mode, etc.

3) The program can be of any type: graphics, sound, text, utility, etc. ☺

Hints & Enhancements

continued from page 7

NULL WORD = ?

After experimenting with *Word-Search Puzzle Generator* in the June, 1981, issue, I discovered that if you enter a "null" word for any word other than #1, it is considered a word, enters it as spaces and increases the word #. By inserting the following line, the problem is easily corrected:
91 A\$=""

This will prevent the string from retaining the leading space from previous entries.

I have also added a few personal lines which add a few features that I enjoy. I have included them as I realize many people who use *SoftSide*'s programs may not really know how to program or are unfamiliar with some things you can do with a program to personalize it. The additions are all for an S-80 Model III with a disk BASIC. The following lines will alphabetize the word list:
723 DEFINT Q:Q=Z-1
725 CMD"O",Q,W\$(1)

If the line printer that is used will accept CHR\$(12) as a form feed, then the following will help when additional copies are requested:
770 LPRINTCHR\$(12):GOTO 630

I must say, in all honesty, that the most valuable reason for reading *SoftSide* is that a person is able to learn additional programming hints from studying the programming traits of other people. Please keep up the good work.

Carl Bleiweiss
Fort Lauderdale, FL

S-80 HINT

If you have NEWDOS or NEWDOS+, you are probably familiar with the JKL function to dump the contents of the screen to your printer. You may also be familiar with the way all graphic characters are printed as periods on the printer.

For those of you that have printers with graphic capabilities, it's possible to modify JKL for your printer. If your printer uses standard S-80 graphic codes (such as the Microline 80 does), the modification is:
65000 POKE &H43D3,0: POKE &H43D4,0

If you have an MX-80 in non S-80 mode, then use:
65000 POKE &H43D3,198: POKE &H43D4,32

Please note that these POKES must be done every time the disk is rebooted.

Rich Bouchard
Amherst, NH

APPLE HINT

To PEEK or POKE directly into screen memory, use the following formula to convert HTAB and VTAB positions to the proper memory location.

H is the HTAB position (1-40)
V is the VTAB position (1-24)
X is the memory location
 $X = 895 + V * 128 + H - 984 * ((V > 8) + (V > 16))$

One reason for needing a formula such as this one is to determine what character is in a certain position on the Apple's screen. For example, if H is assigned a value of 10, and V a value of 14,

then the formula will give the memory location X. PEEKing into this location will then return a number related to the ASCII value of the character at column 10, line 14 of the screen.

I say "related to" the ASCII value, because characters are stored in screen memory in a somewhat strange way. Flashing characters are stored as numbers 0-63, inverse as 64-127, and normal as 128-191. To further complicate things, the first 32 numbers of each of these ranges represent characters "@" through "_" (including all the letters), and the second 32 represent characters " " (space) through "?" (including numerals and most punctuation).

This information should provide an incentive for you to delve further into the possibilities of direct PEEKing and POKEing into video screen memory. Just make sure that you don't accidentally do any POKEing outside the memory locations 1024-2047, or you may destroy the program in memory and possibly bomb Applesoft.

Jon Voskuil
Milford, NH

POKE YOUR ATARI!

Deep within the mysterious depths of Atari's reserved RAM lies a memory location that controls the display in a sometimes bizarre way.

I'm talking about memory location 755 (2F3H). Depending on the value poked here, a variety of interesting effects can be achieved. Here is a table of numbers for POKEing, with their resulting actions on the screen:

POKE Action 755

- 0 — Reverse inverse video.
- 1 — Black-out inverse video.
- 2 — No effect. Normal characters.
- 3 — White-out inverse video.
- 4 — Display characters upside down. Reverse inverse video.
- 5 — Display characters upside down. Black-out inverse video.
- 6 — Display characters upside down.
- 7 — Display characters upside down. White-out inverse video.

By using different combinations of POKES in conjunction with inverse video characters, the following displays can be achieved:

POKE Action 755

- 0,2 — Flashing text.
- 1,2 — Blinking inverse text.
- 0,1 — Blinking normal text.
- 2,1 — Erase inverse text.
- 0,1 — Erase normal text.
- 2,3 — "Cover" inverse text.
- 0,3 — "Cover" normal text.
- 4,6 — Flashing upside-down text.
- 5,6 — Blinking upside-down inverse text.
- 4,5 — Blinking upside-down normal text.
- 6,5 — Erase upside-down inverse text.
- 4,5 — Erase upside-down normal text.
- 6,7 — "Cover" upside-down inverse text.
- 4,7 — "Cover" upside-down normal text.
- 6,2 — "Flip" text.

Flashing and blinking are attained by alternating between the two values, with a delay loop between each POKE for the rate. Erasing and covering work by doing the first POKE and, when appropriate, following with the second.

Alan J. Zett
Milford, NH

ATTENTION AUTHORS

SoftSide Publications is actively seeking programs, article and review submissions for the TRS-80™, Apple and Atari home computers. This is a chance for programmers as well as users to make some money to help pay for the "computer addiction" and get their efforts out where they can be appreciated.

Programs — *SoftSide* has always been the leader in the field of BASIC software and BASIC remains our specialty. However, with the advent of Disk Version (DV), we can now also offer an outlet for Machine Language and multiple language programs which do not lend themselves to printed versions. Games, utilities and educational software, as well as any other applications for the home computer user are preferred, although we will consider virtually any type of program. Hybrid mixes of articles and programs are also welcomed.

When submitting a program, please be sure to include full documentation of subroutines and a list of variables, as well as a brief article describing the program.

Reviews — Well written, informed reviews of all software for the systems we cover are a regular feature of *SoftSide*. Reviewers should take into consideration all aspects of a particular software package, from speed of execution to programming creativity to the estimated length of time that the product will hold the customer's interest.

Articles — We welcome article submissions of all types, but prefer those specifically geared to the home computer market. We give our readers information as a first priority, but vary our content to include some humor and commentary.

All text, including documentation and descriptive articles for programs should be typewritten and double-spaced. Extra monetary consideration will be given to articles and reviews submitted on machine-readable media (Scriptit, Super-Text II, etc.). Programs should be submitted on a good cassette or disk. TRS-80™ BASIC programs should function under both Level II and Disk BASIC.

Send to:

SoftSide Publications
SUBMISSIONS DEPARTMENT
6 South Street
Milford, NH 03055

We regret that due to the volume we receive, we are unable to return submissions.

Be sure to send for our **FREE Author's Guide**. It further outlines the specifics of our submission procedure.

TRS-80 is a registered trademark of Tandy corporation.

CALENDAR

November 4-18

Seminars on Data Processing and Paperwork Reduction Locations below

Special seminars on implementation of Public Law 96-511, the Paperwork Reduction Act of 1980, are scheduled for six cities during this month. The one-day meetings will provide a thorough update on the impact of this major new legislation for data processing managers and computer specialists. This will provide the opportunity for those outside of Washington to find out how the new law will affect field organization and computer operations. Seminars will be conducted by Robert V. Head, author and contributing editor of *Government Data Systems* magazine; and Dr. Donald W. Fitzpatrick, Director of Corporate Information System Practice for Advanced Technology, Inc. Dates and locations for the seminars are:

November 4 Boston, MA
November 5 Philadelphia, PA
November 6 Atlanta, GA
November 16 Chicago, IL
November 17 Denver, CO
November 18 Dallas, TX

Contact: Seminar Coordinator, U.S. Professional Development Institute, 12611 Davan Drive, Silver Spring, MD 20904 (301)622-0066

November 8-10

Comp-U-Con Trade Show Los Angeles, CA

This show will feature exhibits directed toward the home user/hobbyist as well as the small businessperson. It is the eighth in a series of trade shows that has been produced from coast to coast by Comp-U-Con.

Contact: Jeff Weston, The Weston Research Institute, P.O. Box 175, Waverly, IA 50677, (319)268-1953

November 9-11

Distributed Processing Update: A Management Seminar Westpark Hotel, McLean, VA

This seminar, designed by *DataCommunications* magazine, will encompass a comprehensive overview of the tools, techniques, requirements, and benefits of distributed processing. Participants will receive the latest information on the key elements of a distributed processing system and information on how to best select the

hardware and software for their needs. Featured speakers will be Dr. Marshall D. Abrams, a computer scientist at the National Bureau of Standards; Dr. Ira W. Cotton, a senior associate in the Management Information Systems division of Booz, Allen, and Hamilton, Inc.; Harold C. Folts, a senior engineer at National Communications Systems; and Stuart Wecker, the founder and president of Technology Concepts, Inc. Registration fee for this seminar is \$650. Contact: McGraw-Hill Conference Center, 1221 Avenue of the Americas, Room 3677, New York, NY 10020, (212)997-2855

November 13-15

Los Angeles Computer Showcase Los Angeles Convention Center, Los Angeles, CA

Exhibits and demonstrations will be the focal point of this exposition sponsored by The Interface Group. Small systems for businesspeople, professionals, educators, scientists, corporate executives, and serious computer users will be displayed.

Contact: The Interface Group, 160 Speen Street, Framingham, MA 01701, (800)225-4620, or MA residents (617)879-4502

November 19-21

Comp-U-Con San Francisco, CA

A show for the home and business computerist. Computer representatives from various lines will participate in this trade show and exhibit their computer lines. Daily attendance is expected to range from 20,000 to 35,000 people.

Contact: The Weston Research Institute, P.O. Box 175, Waverly, LA 50677, (319)268-1953-

November 19-22

Comdex Show Las Vegas Convention Center, Las Vegas, NV

The third annual Comdex Conference and Exhibition for independent sales organizations (ISO's) of small computer and word processing systems, peripherals, services, media and supplies. Leading national experts will be featured in 40 sessions, focusing on business, marketing, and financial topics of special relevance

to ISO's. Two "Executive Impact Luncheons" will be separately priced. More than 500 computer-related companies will exhibit.

Contact: Peter B. Young, The Interface Group, 160 Speen Street, Framingham, MA 01701, (800)225-4620, or MA residents (617)879-4502

November 22-24

Microcomputers in Education Guttman Library, Appian Way and Brattle Streets, Cambridge, MA

A series of microcomputer workshops, offered by Technical Education Research Centers (TERC) designed to help educators and administrators in the revolutionary advances in education. The opportunity to choose one to three workshops and the chance to participate in evening sessions with featured speakers and panel discussions are offered. Each workshop is developed by leaders in the field of microcomputers in education and focuses on hands-on experience. You may select from eight topics in the workshops: Pascal, Logo, Software Evaluation, Overview, BASIC I, BASIC II, Science Instruction, and Laboratory Instruments. Registration is limited and is on a first-come basis. Registration fees are \$110 for one day, \$210 for two days and \$300 for all three days. Fees include full-day workshop, evening sessions, manual of curriculum materials, coffee and rolls. Non-participants may attend the general interest evening sessions for a preregistration fee of \$10. Continuing Education Units may be earned by participants; one CEU per workshop. Contact: Ms. Sharon Woodruff, Conference Coordinator, TERC, 8 Eliot Street, Cambridge, MA 02138, (617)547-3890

If you or your organization are sponsoring or know of an event you think would be of interest to *SoftSide* readers, please send complete information to:

SoftSide Publications
Calendar Editor
6 South Street
Milford, NH 03055

Be sure to include complete information concerning dates, location, subject matter and a contact name, address, and phone number.



Outgoing Mail

by Randal L. Kottwitz

PROBLEMS

I've just returned from the fall computer shows and have been delighted to talk with many of you. As with all good things, one must learn to take the sour with the sweet and I'm as grateful for the constructive criticism you offered as I am for the many compliments you paid us. Your comments, combined with the many letters you've sent, have led me to feel there are a few gray areas concerning *SoftSide's* operations which should be cleared up for readers to whom I've not been able to speak in person. We're all forging into new territory in the microcomputer field and we would be fools not to listen to the advice of our friends. All apologies aside, here's why it is the way it is — and what we're doing about it.

The people who have supported *SoftSide* by signing up for our media versions deserve an explanation of the problems that have been plaguing that area.

The main source of our problems is technology. No one has developed a fool-proof method for reproducing media. This fact, compounded by the short production cycle of a magazine, has had us drowning in customer service problems. We are not insensitive to those problems, and we are developing solutions which will solve (or at least minimize) them.

Until recently, there has been no commercial diskette duplication machine available. The software industry has only recently begun to require thousands of reproductions of the same product. It is not yet clear whether these new machines (costing approximately \$100,000) will be price or quality competitive. Obviously, we can't afford such an expense and the machines have yet to actually be produced.

At this time we are reproducing diskettes by exactly the same method you use. Even though we have substantially improved the process and have alternative equipment (in other areas of the company) for emergency use, the volume and time constraints have made the workload situation a case of feast or famine.

As an example of how Murphy's laws have effected us, I'll cite a recent problem we encountered in reproduc-

ing Atari diskettes. In July, problems developed on all three Atari disk drives used for our duplication process. Attempts to repair the equipment at our own repair facility took three weeks and were unsuccessful due to the in-availability of parts. The drives then were sent to the Atari service contractor. After another three weeks, they returned, ready for duplication and just in time to wait another week and a half for a delayed shipment of diskettes to arrive. Of course, by then there were back orders that would take two weeks under the best of circumstances (the best of circumstances being an entirely theoretical era, yet to be experienced). This work overload on our duplication department caused a large batch of incorrect TRS-80™ diskettes to be produced for the August *SoftSide DV*. The story could go on, but the point should be clear.

Due to unusual circumstances, our magazine suddenly became a second class postage item (i.e., recognized by the Post Office as a true magazine and thus can be mailed faster and less expensively). This may not seem a problem, and in most ways it is not. However, postal regulations will not permit cassettes or diskettes to be mailed with the magazine! We have considered isolating our tape and disk customers, and mailing their magazines with the media. But all that would accomplish would be for it to cost more for you to get your magazine later than anyone else on the list.

Cassette duplication elicits quite another set of woes. They start with the media. A digitally certified C-30 tape is financially out of the question. (They cost approximately two dollars each in lots of 1000.) We have yet to find high-speed tape duplication equipment specifically designed for digital magnetic tape. It's yet to be produced. Frankly, the software industry doesn't represent a large enough marketplace to beget the serious engineering of such a specialized device. The core problem — computer information relies on a different frequency of sound than music or voice. Our duplication equipment has been modified to accentuate those digital characteristics. Yet, it's a compromise. Even though we use ex-

continued on page 13

K-Byters

ANOTHER PROGRAMMING CHALLENGE

Last summer *SoftSide* began inviting its readers to submit "One Liners" — self-contained, single-line programs for the S-80, Apple, or Atari which would provide a continuously changing graphics display. The response has been excellent, and we're still looking for more submissions.

Now we have a new challenge for you as well: "K-Byters." A K-Byter is a BASIC program which fits into 1K (1024) bytes of program memory. There aren't any restrictions on the nature of the program, other than its size. It can be a graphics display, a game, a mini-adventure, or anything your imagination and programming skills can create.

Note that the program does not have to RUN in 1K of memory; it can use as much RAM for arrays, strings, graphics mapping, etc., as you need. We'd prefer that it be able to run in a 16K system, but this is not an absolute limit.

Here, then, are the official rules:

1. The program must be written for the Apple, S-80, or Atari, entirely in BASIC (although it may create and call Machine Language routines).

2. The program must occupy no more than 1024 bytes of memory before running.

3. The program must be submitted on tape or disk, accompanied by your name, address, phone number, and a brief written description of its operation.

4. The tape or disk will be returned only if accompanied by a self-addressed envelope with adequate postage AFFIXED (do not send money).

5. Winners will have their programs published in *SoftSide* and will receive a \$10 software certificate for their programming excellence!

Send submissions to:

K-Byters, c/o *SoftSide*
6 South Street
Milford, NH 03055





THE NATIONAL COMPUTER SHOWS

HAVE WE GOT A PROGRAM FOR YOU IN '81 & '82

Attend the biggest public computer shows in the country. Each show has 100,000 square feet of display space featuring over 50 Million Dollars worth of software and hardware for business, industry, government, education, home and personal use.

You'll see computers costing \$150 to \$250,000 including mini and micro computers, software, graphics, data and word processing equipment, telecommunications, office machines, electronic typewriters, peripheral equipment, supplies and computer services.

All the major names are there including; IBM, Wang, DEC, Xerox, Burroughs, Data General, Qantel, Nixdorf, NEC, Radio Shack, Heathkit, Apple, RCA, Vector Graphic, and Commodore Pet. Plus, computerized video games, robots, computer art, electronic gadgetry, and computer music to entertain, enthrall and educate kids, spouses and people who don't know a program from a memory disk.

Don't miss the Coming Of The New Computers—Show Up For The Show that mixes business with pleasure. Admission is \$5 for adults and \$2 for children under 12 when accompanied by an adult.

Ticket Information

Send \$5 per person with the name of the show you will attend to National Computer Shows, 824 Boylston Street, Chestnut Hill, Mass. 02167. Tel. 617 739 2000. Tickets can also be purchased at the show.

THE MID-WEST COMPUTER SHOW

CHICAGO
McCormick Place
 SCHOESSLING HALL
 23rd & THE LAKE
THURS-SUN
SEPT 10-13, 1981
 11AM TO 7PM WEEKDAYS
 11AM TO 6PM WEEKENDS

THE MID-ATLANTIC COMPUTER SHOW

WASHINGTON, DC
DC Armory/Starplex
 2001 E. CAPITAL ST. SE
 (E CAP. ST. EXIT OFF I 295
 -KENILWORTH FRWY)
 ACROSS FROM RFK
 STADIUM
THURS-SUN
SEPT 24-27, 1981
 11AM TO 7PM WEEKDAYS
 11AM TO 6PM WEEKENDS

THE NORTHEAST COMPUTER SHOW

BOSTON
Hynes Auditorium
 PRUDENTIAL CENTER
THURS-SUN
OCT 15-18, 1981
 11AM TO 7PM WEEKDAYS
 11AM TO 6PM WEEKENDS

THE SOUTHEAST COMPUTER SHOW

ATLANTA
Atlanta Civic Center
 395 PIEDMONT AVE NE AT
 RALPH MCGILL BLVD
THURS-SUN
OCT 29-NOV 1, 1981
 11AM TO 7PM WEEKDAYS
 11AM TO 6PM WEEKENDS

THE SOUTHERN CALIFORNIA COMPUTER SHOW

LOS ANGELES
LA Convention Center
 1201 SOUTH FIGUEROA
THURS-SUN
MAY 6-9, 1982
 11AM TO 7PM WEEKDAYS
 11AM TO 6PM WEEKENDS

Outgoing Mail

continued from page 11

tremely high-quality tape, the certification process adds a tape manufacturing step which, in fact, loads and reads back digital information. In other words, 100% quality control, which is expensive! Each of the computers we support has its own quirks in the manner it expects to read data. The Atari standard is so different that a \$2000 machine has been specially modified to duplicate only Atari cassettes. The adjustment has made that duplicator unsuitable for duplicating any other kind of tapes.

DOSs are another small bucket of worms. There is enough similarity between the TRS-80™ Models I and III that they can — usually — be treated as the same machine. The same is true for Apple II owners with 3.2 and 3.3 DOS.

Cassette and diskette duplication cannot begin until all code for the magazine is finalized. Most translations (required for three-across software) must be continually modified and play-tested. The consequence is that cassette and diskette reproduction can only start when the magazine itself is released to the printer. The printer will mail the magazine within ten days, heading to you with the speed of second class mail. Typically, cassettes and diskettes are still coming off the production line and won't be mailed for another week, heading to your home at the much slower bulk rate speed.

Fulfillment is the trade term for a system which gets your magazine to you every month — on time. Our fulfillment situation is relatively better off than our duplication, but it too has its problems. The job has outgrown the capabilities of our TRS-80™ Model Is. This problem is not new. We have considered a minicomputer and considered converting to TRS-80™ Model IIs. The software we have been using has worked well for us — considering. But, it is woefully inadequate for our present needs and those of the future.

PROPOSED SOLUTIONS!

The fulfillment situation is on its way to being resolved. The answer was quite obvious — do it the way the larger magazines do — engage a fulfillment house. This is a company with big computers and elaborate programs which can point us to problems we

didn't know we had. This solution did not come overnight. However, we feel the situation should be under control by the first of the year — Happy New Year!

The key to most of the other problems is TIME! The roller coaster effect of the magazine printing deadline dictating when production of the media can commence simply has to be turned around. We feel it may take as long as three months to effectively detach the preparation of the software from the preparation of the rest of the magazine by a full month. The problem is, among other things, a shortage of trained personnel — again a problem of not having enough time.

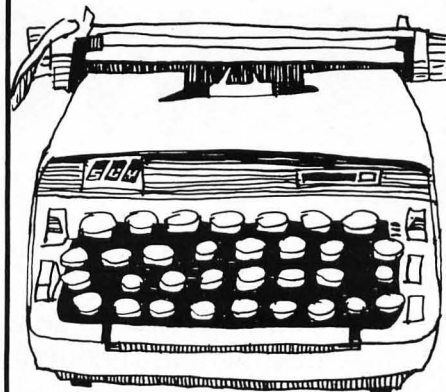
An extra month of preparation time is very important for many reasons. Options outside our walls for duplication and shipping which were out of the question in the past due to preparation time limitations are being investigated in earnest. Instead of attempting the impossible, we will be able to accomplish our work at a much more reasonable pace. This will improve our productivity as well as quality.

ATARI SOFTWARE SHORTAGE

The shortage of Atari programs between our covers has been acknowledged by many of you. Again, this is a situation of which we've long been aware. As we've made clear before, we are heavily dependent on submitted software from our readers for many of the original programs we publish. The Atari programs we receive are, in most cases, repetitions of similar games we've published before. You can only land the Lunar Module in so many different ways! I would like to make an open appeal for more serious Atari software. The machines are out there, the numbers make that very clear. If you are writing utilities, educational programs or entertainment software based on original ideas, please rush them off to us. We will give them every serious consideration possible and, chances are, if you've produced quality pieces of software, you'll soon be getting a contract from us.

That should turn some of those gray areas to black and white. For every critical remark you've sent our way, you've given us ten cheers. Keep rooting for us and we'll do our best to keep you informed and entertained. ☺

Typing Tutor



by Roy Groth

Wish you were a better typist, but don't want to take (or pay for) a class? Teach yourself to type with the aid of your microcomputer. With **Typing Tutor** you will be quizzed and graded, but you set the pace at which you learn. **Typing Tutor** is a set of programs that lets you become as good a typist as you wish, allowing you to advance from one level to the next when you feel comfortable with your skills.

Let "hunt and peck" slip into the past, teach yourself speed and accuracy on the keyboard with **Typing Tutor**.

S-80 Mod I & III

16K Tape \$19.95

32K Disk \$20.95



6 South Street Milford NH 03055

ATARI® SOFTWARE

PIRACY:


THIS GAME IS OVER.

ATARI® has led the industry in the development of video games such as ASTEROIDS™ and MISSILE COMMAND™. The outstanding popularity of these games has resulted from the considerable investment of time and resources which ATARI has made in their development. We appreciate the worldwide response from the videophiles who have made our games so popular.

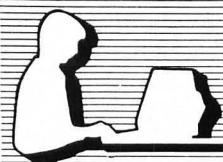
Unfortunately, however, some companies and individuals have copied ATARI games in an attempt to reap undeserved profits from games that they did not develop. ATARI must protect its investment so that we can continue to invest in the development of new and better games. Accordingly, ATARI gives warning to both the intentional pirate and to the individuals simply unaware of the copyright laws that ATARI registers the audiovisual works associated with its games with the Library of Congress and considers its games proprietary. ATARI will protect its rights by vigorously enforcing these copyrights and by taking the appropriate action against unauthorized entities who reproduce or adapt substantial copies of ATARI games, regardless of what computer or other apparatus is used in their performance.

We ask that legitimate software developers cooperate with us to protect our property from any form of software piracy, imitation or infringement. ATARI is currently offering copyright licenses for a limited number of its games to selected software developers. If you happen to be selling a software product which performs a game similar to any ATARI game (such as a game created for a home computer), please contact us immediately. Write to the attention of: Patent Counsel, ATARI, Inc., 1265 Borregas Ave., Sunnyvale, Calif. 94086



 A Warner Communications Company

© 1981, ATARI, INC.



Say Yoho

by Scott Adams

ADVENTURE CONVERSION CONTINUED

In my September column, I discussed problems associated with getting my Adventures up and running on the Atari computers. In trying to transfer files from the Apple II to the Atari, I found I was unable to get the RS232 interface to work with the Atari disk system at all!

The problem turned out to be in the DOS of the Atari. The Atari's peripherals have their drivers built into their ROM. When the system is booted, the drivers are loaded into memory from the different peripherals currently connected.

It turns out that the RS232 driver software overwrote an area of memory that the DOS requires. The solution was to use a new copy of *DOS 2*, the new Atari disk operating system. It has a special preloader which boots automatically and moves the RS232 driver into an area of memory where it no longer conflicts with the DOS.

Now at last I could get the Apple to communicate with the Atari. The next step would be to transfer all the source files from the Apple assembler so I could start converting the interpreter.

The way my Adventures work, all I needed to do was convert the interpreter once and I could then transfer over all the different data bases for each Adventure. Unfortunately, the assembler I was using on the Apple did not store its source files in the simple ASCII form, so I could not just dump the source files over the RS232 link.

Luckily, the way the Apple II is designed I could simply link the RS232 to the screen memory. As items appeared on the screen, they would automatically go over the RS232 link.

Murphy struck! The Atari RS232 linked through Atari's audio serial bus to the main computer (to avoid radio frequency interference problems) and could not keep up with the screen rate on the Apple! So I hooked up an S-80 Model II to the Apple instead and was able to get the entire Adventure source files for the Apple on one 8" double-density disk.

I'll finish the explanation of how I solved my Adventure conversion problems in my next column. Right now I

would like to discuss an interesting new concept on copyright law which Atari has presented.

DEVELOPING COPYRIGHT STANDARDS

The accepted industry copyright standard was, and is, based on source code similarity. If a program used the same, or very similar, source code of a pre-existing program, it was considered to be "pirated" or an infringement of copyright. Atari is in agreement with this concept of software copyright law, and has developed an additional level of copyright protection.

Michael Sherrard of Atari has explained it as follows:

The true value of a software product lies in its "end value": that aspect of the program which is actually interacted with by the end user — in the case of arcade games, the "audio-visual output." Atari has copyrighted its coin-operated arcade games as "audio-visual material." This means the screen of Atari's *Asteroids* coin-operated game is copyrighted for its audio and visual value. When any one of the 50 or so companies imitated *Asteroids* and other pre-existing arcade games, they were borrowing from the audio-visual output (value) of the original version.

This new concept of audio-visual copyright for arcade games, although legally untested, does not seem to be unreasonable. Its legality has not been proven as yet, but it is logical in its concept. A big difficulty will be deciding what the measure is of similar audio/visual values when different computer systems have different capabilities.

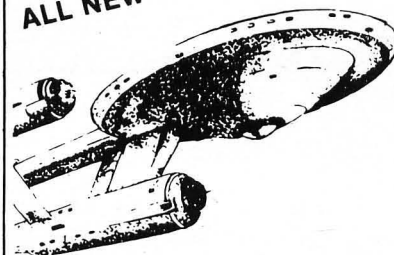
I would like to suggest that anyone writing arcade-type software base it on original ideas. Novel and original arcade games will be great sellers, and who knows, maybe your arcade software will end up on a coin-operated machine!

Until next time...Say Yoho!



SoftSide November 1981

ALL NEW VERSION!



STAR TREK III.5

Now with Sound
Capability and
Increased Speed
of Execution.

by Lance Micklus

You are in command of the starship Enterprise and her complement of 371 officers and crew. You must enter and explore the Omega VI region of the galaxy with its 192 quadrants containing star systems and planets (a few of which are habitable). Astronomical hazards such as pulsars, Class 0 stars, and black holes are known to be present, so the utmost care is needed.

Star Trek III.5 includes, playboard 8 by 8 by 3 quadrants, weapons system of Phasers and Photon Torpedos; Warp and impulse power systems; Science and Ship's computers; Long and Short Range sensors; Damage Control and Status reports; and 20 Klingon battle cruisers, and 100 stars, planets, black holes, and pulsars.

Atari 32K Tape \$19.95
S-80 16K Tape \$14.95
S-80 32K Disk \$15.95



6 South Street Milford NH 03055



Badly Needed: Good Educational Programs

by William D. Hedges

This article is directed to you computer programming wizards out there. How about slowing down on the monster games and turning your creative genius to badly needed educational programs for children and youth in our public and private schools — and for parents to use with their young children?

The need is there. There are few programs that are truly educational and have the appeal of those games. This article describes areas of need and presents some of the learning attributes such programs should possess.

Each school subject has a few concepts that are unusually difficult for many students. Most teachers I know can quickly tell you what these are in their subject. Further, they will tell you, "If I could just get such and such a skill or concept across, the rest would be easy!"

For example, there are meiosis and mitosis in biology, the concepts of base and place value in mathematics, the meaning of time in history, the laws of optics in physics, Kepler's Laws in astronomy, Mendel's in genetics, probability in statistics, ad infinitum. Not to be ignored are the multiplicity of computer programming concepts such as base conversion, sorting and so on.

It is these difficult concepts which are a natural for the microcomputer. They can usually be learned, if insightfully presented, in concise programs requiring only a few K of memory.

As the teacher presents these concepts, we can envision him or her providing student access to selected short computer programs.

Each student can go through the material as many times as needed to master the idea. One of the major findings from research on learning is that students need widely varying amounts of time to attain mastery — and it is mastery we are seeking.

Possessing a small library of such well-designed educational programs on the more difficult concepts of a subject, every teacher could keep a micro busy all day providing individualized assistance for those students needing it.

I am not talking about entire courses. In fact, I think it is generally a mistake to think of entire courses via the microcomputer. Nor am I thinking of the plethora of drill-and-practice software being sold as educationally sound. Too much of it is dull, unimaginative, and not congruent with what we know about how children learn. In short, programmers out there, you are badly needed.

Some attributes of effective programs are:

1. Pilot Testing. One major characteristic of any well-designed, educational program is that it has been carefully field tested with students. Therefore, a practical procedure might be for you to collaborate with classroom teachers. Start with an expressed need of theirs, work with them to get it precisely defined, program it, and then arrange for them to try it out with students.

Doing this a few times, I assure you, can result in your identifying not only program bugs, but needed modifications in the program design.

It is impossible for you and me to anticipate all the problems students will encounter unless we observe them and take notes. Good programs cannot be "armchaired" and published.

With empirical testing in the real world, coupled with revisions, you can include the field testing data in the documentation you send to a publisher; such data carries weight in evaluating the work. You and your teacher-collaborator can jointly author the finished product.

2. Use the medium! One common fault in many educational programs is that they "read to" the student. Reading programs are better left inside the



covers of books. Few of us can remain interested very long as pages of prose scroll by. Most youngsters will be quickly turned off.

3. Make your program interactive. This is a major advantage the microcomputer medium has over TV. TV is a passive medium. CAI is an interactive medium. One of the criticisms of much educational software is that the programs are essentially expensive page turners.

4. Graphics are a must! Certain concepts and relationships can be beautifully and effectively explained using color graphics or shape tables.

For example, I recently saw the internal combustion engine in color. It could be operated slowly and manually, or rapidly and automatically. The student was able to examine the operation as minutely as he or she wished. Unfortunately, missing from this demonstration was any effort to focus student attention, ask questions about the principles involved, clarify wrong answers and the like.

5. Avoid the fluff. It usually only distracts. Be clear about the concept and then present it imaginatively, correctly, clearly and precisely. Ringing bells, funny faces, and such pyrotechnics, while perhaps amusing at first, soon become tiresome and may even interfere with the need for quiet concentration in a classroom setting. Thus, decide in advance the outcomes you are looking for the student to attain. State these outcomes clearly and succinctly. For example, the goal might be: "Mastery of this lesson will enable the student to explain and illustrate Kepler's First Law." Or, "At the end

of this lesson, you should be able to demonstrate the relationship between the circumference and diameter of a circle."

6. Allow the student the chance to return again and again to program components and review them. Most of us have to refer from time to time to something we thought we'd just learned; this quick referral is possible in good educational programs — just as it is possible in our current programs which provide tables of commands whenever we need them.

7. Make it impossible, short of turning the machine off, to get "hung up" not knowing what to do. Well designed programs have built in contingencies that, at the least, get the student back to the menu where a new start may be made. Teachers are not going to be highly impressed by programs which cause the student to constantly interrupt with "Mrs. Brown, what do I do now?"

8. Be clear concerning the assumptions you are making about prerequisite student behaviors. It is of limited educational value to start a student on the multiplication of complex fractions if the skills of whole number multiplication are lacking. This suggests an optional test which should address the following: (a) Has the student already mastered the subject matter and hence need not proceed? (b) Does the student possess the prerequisite skills to go through the lesson?

9. Document that program! Increasingly, good teachers are going to want to understand just what is going on. Adequate documentation can help.

10. Specify the machine for which the

program is suitable, memory required, etc. Without these specs, the result is frustration and anger over a program that will not run — and that instructional budget is shot!

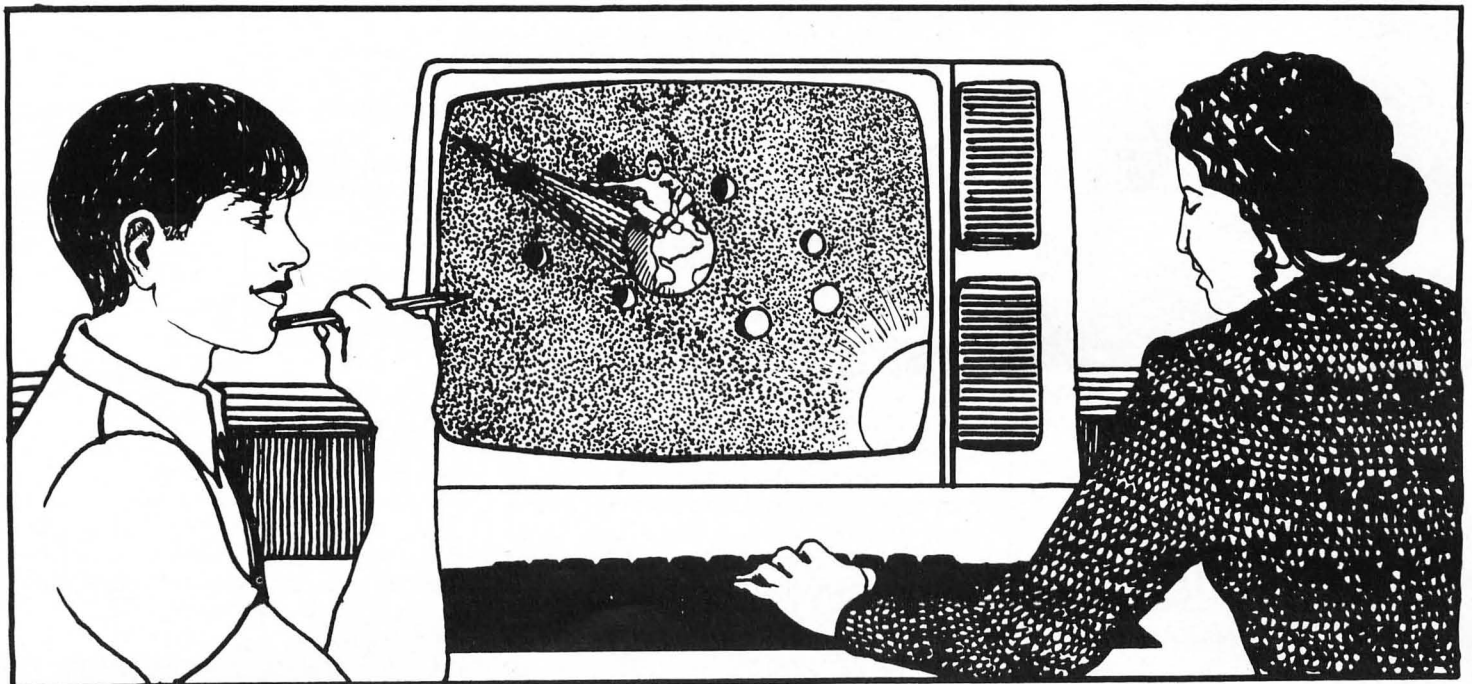
11. Check your facts. Get an expert in the subject field to go over your content. Is it accurate?

12. Watch your level of abstraction. Concepts and ideas can be taught at elementary as well as advanced levels. One of the clues to efficient learning is that the level of abstraction is appropriate for the student population for which you are designing the lesson. Here is another area where pilot testing is invaluable. We cannot know the appropriate abstraction level, but most teachers can soon tell you as they observe youngsters working with it.

13. Periodically reinforce. Effective learning suggests reinforcement that is variable both in intensity and frequency. Slot machines operate on this principle and are highly effective in keeping people busily pumping the handle. The ideas of randomness and variations in intensity of feedback are not difficult to build into educational programs with random number generators and a dictionary of adjectives.

14. Provide feedback on progress. Feedback is crucial and should operate much as the coach reviewing video tapes of a ball game, or as an instructor on a rifle range. In each case, the idea is to enable the learner to understand what he or she is doing wrong as a basis for correcting the behavior.

continued on next page



Nine Games for Preschool Children

by George Blank

Even preschoolers deserve a shot at the wonders of microcomputing. With these nine games, they not only will have a chance to tickle the keyboard, but learn letters and numbers to boot. And if that isn't enough, they'll have a good time doing so. What more could a parent ask for? Here are education and entertainment for the very young in a single package!

S-80, Level II; 16K, Cassette
\$9.95



6 South Street Milford NH 03055

BADLY NEEDED: GOOD EDUCATIONAL PROGRAMS

continued from previous page

The few student responses not anticipated can be handled with the generic "I don't know — see your teacher." It's not such a bad thing for young students to learn that computers can get stuck too.

15. Allow variable response time. Set the time for response, except under certain circumstances, under student control. The idea here is, it's fine for speed to increase, but only as understanding does also. As indicated, the variations in student response time are so tremendous that no one response time is going to work for all students. Even typing teachers reduce speed as they induce accuracy. It is only as accuracy is maintained that they emphasize gradually increased speed. Another possibility is to give choices for slow, medium, or fast.

16. Watch that length! Few of us want to get caught up in something that is going to take an hour when we only have 15 minutes. Periodically provide the student the option of signing off without hitting the reset button.

17. Watch that grammar! Programs should not only be scientifically correct, but also grammatically correct. Schools are not going to be impressed with sloppy or incorrect syntax, spelling, etc.

18. Emphasize quality of learning rather than quantity. A single idea, thoroughly grasped, is far better for the student than a potpourri of ideas, dimly grasped.

19. Place the student into simulated performance situations. You cannot presume something has been learned just because it has been presented. Give the student a chance to practice, preferably under conditions slightly different than those experienced in the initial learning. Going on to something else without such practice is self defeating.

20. Build in branching that enables students to proceed to the goal by different routes, depending on the nature of their responses. Good design allows for variable routes which reflect variations in student background, rate of learning, and learning style.

The time is coming when programs will be designed to be so sensitive to

patterns of student responses that they can branch them forward, backward, or tangentially to provide the students with what they need and then bring them back to the central thread. This is possible now, but we see only the bare bones beginnings in some of the large computer-assisted-instruction projects using main frame systems (as some of the PLATO and TICCIT materials).

It is apparent, from the above, that conceptualizing, designing, programming, documenting, field testing, revising, etc. are not trivial tasks. Perhaps this is why we have such a paucity of really good educational programs. However, one encouraging note is that not all programs need possess all of the characteristics enumerated above.

As an educator, I have been impressed by the bright people with creative ideas characterizing this young field.

And in case you think I am preaching and not practicing, let me brag a little. I am an educator, but a neophyte in this field. I went to work on a program. It took me an inordinately long time. Hopefully, subsequent programs will be progressively easier. But, the first computer journal to which it was sent accepted it!

Summary:

There is virtually no limit to the need and demand for well-designed, imaginative programs which reflect what we know about how children learn. We've only begun to scratch the surface of using the microcomputer medium effectively.

Understandably, our first attempts have too often unconsciously imitated what can be done better with a textbook, a teaching machine, or a demonstration. Just as man initially imitated the horse in making the car by putting the source of power out front where it had been for 5,000 years, we are imitating the practices we have learned from books, films, etc. I repeat, "Let's use this new medium!"

The talent is out there. I see evidence in the journals. I also see evidence of it in the creative writings of seminal minds such as Seymore Papert in *Mindstorms*. If you do a good job, you can put a little jelly on your toast and have the tremendous satisfaction of knowing you have helped many students acquire new depths of understanding.



The Sensuous Programmer

by "J"

The Seventh Secret Meeting

One of the best qualities of a computer is its ability to go around in circles. Not physically, of course, but operationally. Computers are good for doing some kinds of "linear" tasks, but their real forte is doing repetitive ones: tasks that need to be done over and over, usually with just slight variations each time through. Thus it is that little animals called loops infest most computer programs in great numbers.

A loop is simply a section of a computer program written in such a way that it can repeat itself more than once. There is more than one way to construct loops, but most proper loops (as opposed to improper ones?) have four basic parts. These are illustrated in the following simple program:

```
100 N = 1
110 PRINT N, N*N, N*N*N
120 N = N + 1
130 IF N < 21 THEN GOTO 110
140 END
```

The first part of the loop is initialization. "Initialization" is one of those popular computer buzz words that make the uninitiated tremble; but all it means is setting things up the way you want them at the beginning. In this case, the initialization is accomplished in line 100 by assigning the value 1 to the variable N. (There's nothing magical about 1; it just happens to be where I wanted to start.) The importance of this step is fairly obvious. Unless you KNOW what the starting conditions are, the ensuing series of operations will have unpredictable results.

Sometimes the initialization is implied rather than explicit. For example, most BASICs assign the value 0 to all numeric variables upon RUNNING a program. So if the variable N was not used prior to line 100, and you wanted it initialized to 0, you wouldn't have to use an assignment statement to do it. On the other hand, some BASIC dialects (such as Apple Integer) don't zero variables, so you could get into trouble. And you don't always want to initialize a variable to 0, anyway. It's good programming practice ALWAYS to make such initialization explicit, so that there will be no doubt in anyone's



mind about the crucial values. A programmer ought to do everything possible to make the programming clear rather than obscure.

The second part of the loop is the main body. This comprises all the program lines which are to be executed multiple times. In the above program example, line 110 is the main body. Its function is to print the value of N at the beginning of the screen line, followed by the value of N squared at the first standard tab location, followed by N cubed at the next tab location, followed by a carriage return to the beginning of the next line.

The main body of the loop can be as long as the entire program (excluding the other parts of the loop), or as short as — well, as nothing. A common use for a loop with no main body is to cause a delay in the program's execution. If you have the computer do nothing enough times, it will take some noticeable time to do it, simply because of the overhead involved in executing the loop statements.

The third part of the loop is called modification. In this step you change one or more values in order to make the loop do its thing just slightly differently the next time through. Line 120 in the example changes the value of the variable N. It finds the old value of N in memory, adds one to it, and replaces the old value with this new value. The modification can be as simple or as complex as you may want, just as long as something gets modified in some way.

We've seen that it's possible to omit initialization (at the risk of unanticipated results, and loss of clarity) and even to omit the main body of the loop. What happens if the modification step is omitted? Well, if there's nothing that changes, then there's nothing that will cause the loop to do anything differently than it did the last time around. Without a modification step, a loop becomes an infinite loop. If it loops at all, it will loop forever (barring some sort of error). The simplest form of an infinite loop is a line such as this:

```
1000 GOTO 1000
```

All that this loop contains is a main body; there is no initialization and no modification. (If there's nothing to modify, why initialize??) There are at least two common uses of infinite loops. One is to use a statement such as line 1000 to prevent the prompt and cursor from appearing on the screen at the end of a program, such as a graphics display. And the second use is to return to the beginning of a program when it reaches its end, ready to execute again. In this case, a large part of the whole program is the main body of the infinite loop.

continued on next page

continued from previous page

The fourth part of a proper loop (you're beginning to see why I didn't say "EVERY loop," aren't you?) is the exit test. In the example this is accomplished by line 130, which checks the value of N to see if it is less than 21. If this test fails, then the computer exits from the loop by dropping through to line 140 (which, in this case, simply ends the program). If, on the other hand, the test succeeds, then the loop will be reiterated another time. For each iteration the same check is made, until the condition for exiting is met.

Once again, the exit test is not a sine qua non for a loop. (That's the only Greek I know besides "bon jour" and "Volkswagen"; it means that you CAN have a loop without an exit test.) But without the exit test, you again come up with an infinite loop. If there's no test, then there are no conditions under which it will exit — unless it exits after the very first iteration, in which case it's not really a loop at all! The example of line 1000 above, in addition to containing no initialization and no modification, also contains no exit test (a highly improper loop, to be sure).

Initialization, main body, modification, and exit test are the four parts of a proper loop. You may ask, "Do they have to be in that order?" Except for initialization, the answer is "no." By its very nature, initialization must come at the beginning. But the other three elements can be in any order. To illustrate this, here is the sample loop above, rewritten in all five of the other possible orders:

Modification/Body/Test:

```
200 N = 0
210 N = N + 1
220 PRINT N, N*N, N*N*N
230 IF N < 20 THEN GOTO 210
240 END
```

Test/Body/Modification:

```
300 N = 1
310 IF N > 20 THEN GOTO 350
320 PRINT N, N*N, N*N*N
330 N = N + 1
340 GOTO 310
350 END
```

Body/Test/Modification:

```
400 N = 1
410 PRINT N, N*N, N*N*N
420 IF N > 19 THEN GOTO 450
430 N = N + 1
440 GOTO 410
450 END
```

Modification/Test/Body:

```
500 N = 0
510 N = N + 1
520 IF N > 20 THEN GOTO 550
530 PRINT N, N*N, N*N*N
540 GOTO 510
550 END
```

Test/Modification/Body:

```
600 N = 0
610 IF N > 19 THEN GOTO 650
620 N = N + 1
630 PRINT N, N*N, N*N*N
640 GOTO 610
650 END
```

Notice that the exact form of the initialization and exit test may differ from the original when the order of the elements is changed. If modification comes before the main body, then initialization must be adjusted so that the first-time value of the variable N will be correct. If testing comes before modification, then the test must be adjusted to a different value. And if the exit test is not the last step, then the branching is slightly more complex, requiring an additional GOTO statement (when each line has only one statement, as in this example).

Now, there's nothing at all wrong with the loop forms that have been illustrated so far. But the BASIC language provides a loop structure which is generally easier to use than those in the above examples. The loops which use this structure are usually called FOR-NEXT loops, because the words "FOR" and "NEXT" mark their beginning and end. The following example accomplishes the same thing as the previous loops:

```
700 FOR N = 1 TO 20
710 PRINT N, N*N, N*N*N
720 NEXT N
730 END
```

Lines 700 and 720 together perform the three functions of initialization, modification, and exit testing — and in a more compact form than the previous loops. The variable N (or whatever variable you choose to use following the word "FOR") is usually called the "loop variable," and is initialized at whatever value you specify after the equal sign. The body of the loop (everything between the FOR statement and the NEXT statement) is then repeatedly executed, with N being incremented by 1 each time through, until the limit which you specified after the word "TO" is exceeded. At that point execution "falls through" to the statement following the NEXT, which

in this case is the END of the program.

The modification step, then, consists of adding 1 to the value of the loop variable each time. But you don't have to be limited by this default value. Following the FOR...TO... statement, you can add the word "STEP", followed by whatever quantity you want to have added to the loop variable each time. For example, the following loop will do what the previous one did, but only with odd values of N from 1 through 19 (after 19, N will be incremented to 21, which exceeds the limit of 20):

```
750 FOR N = 1 TO 20 STEP 2
760 PRINT N, N*N, N*N*N
770 NEXT N
780 END
```

The STEP quantity need not be positive; if it is negative, the looping will continue until the loop variable is less than the value of the specified limit. (What do you think will happen if you use STEP 0?)

All numbers used in the FOR statement, incidentally, can be replaced by variables or by arithmetic expressions. A loop beginning with the statement

```
FOR K = A TO B STEP (B-A)/100
```

would increment K from A to B in 100 equal steps. Notice that this should cause the loop to be executed a total of 101 times. However, it would not be unusual to have a small rounding error that would bump the value of K slightly over the value of B (rather than exactly equal to it) after the hundredth iteration, causing it to fall through at that point.

Suppose that you program a loop like the following one, where the limit value is smaller than the initial value, and you're not STEPPing negatively from one down to the other:

```
800 FOR J = 2 TO 1
810 PRINT J
820 NEXT J
```

Will the body of the loop (line 810) be executed once, or not at all? (When does the computer check the value of J to see if it exceeds the limit of 1? At the beginning of the loop, or at the end?) And, once the loop has finished, what will be the value of J? Will it be 2? 3? (Does the variable actually get modified before the exit check, or does the computer just check to see if its next value would exceed the limit?) The answers to these questions are the same

for the Apple, Atari, and S-80. I'll leave it to you to check them out for yourself.

There are many occasions when it's desirable to use a structure called "nested loops." Like nested drinking cups, nested loops are simply two or more loops nested together, one completely inside the other. For example:

```
10 FOR I = 1 TO 10
20 FOR J = 1 TO 10
30 PRINT "HI"
40 NEXT J
50 NEXT I
```

This will print "HI" 100 times — which, admittedly, could be done just as easily using a single 1-to-100 loop.

But there are times when nested loops are indispensable. Consider the following nested loops, which result in the printing of the first 50 prime numbers (a number which can be divided evenly only by 1 and itself):

```
1000 YES = 1: NO = 0: NUM = 0
1010 FOR I = 1 TO 50
1020 PRIME = YES
1030 NUM = NUM + 1
1040 FOR DIV = 2 TO SQR (NUM)
1050 IF NUM/DIV = INT
(NUM/DIV) THEN PRIME = NO
1060 NEXT DIV
1070 IF PRIME = YES THEN
GOTO 1090
1080 GOTO 1020
1090 PRINT I; ": "; NUM
1100 NEXT I
1110 END
```

There are really three loops here, which I've shown indented by different amounts. The innermost one spans lines 1040-1060, the middle one lines 1020-1080, and the outermost one lines 1010-1100. (This program doesn't pretend to be the best way to find primes; it's used simply as an illustration.)


The outer loop is a FOR-NEXT type which simply cycles through the program 50 times. In addition to the other two loops nested within it, its main body consists only of the PRINT statement in line 1090 which prints each of the primes, preceded by the number of the prime. Prior to the beginning of this loop, line 1000 initializes three variables. YES and NO are used as constants in conjunction with the "flag" variable PRIME; and NUM is the variable which will step through all

the counting numbers, and be evaluated to see if it is a prime, until 50 are found.


The middle loop does not use the FOR-NEXT structure. Initialization takes place in line 1020, where PRIME is assigned the value of the variable YES. This sets up the program to assume that the number being considered (NUM) is a prime, unless it is proven otherwise. Modification, which consists of incrementing NUM by 1, takes place in line 1030. The main body of the middle loop consists wholly of the innermost loop, whose job it is to "prove otherwise" if possible. Line 1070 is the exit test, which escapes from the loop if a prime has been found. If not, control loops back to line 1020 where PRIME is reset to YES and NUM is incremented by one to try the next number.

The innermost loop is another FOR-NEXT type, which tries dividing NUM by all possible DIVisors, from 2 up to the square root of the NUMBER, and checks to see if the result is a whole number. The test for this is in line 1050, which is the loop's main body: If the NUMBER divided by the DIVisor is the same as its own INTeger value, then NUM is evenly divisible by DIV and is not a prime. In that case PRIME is reassigned the value of the variable NO, which will in turn cause the middle loop's test in line 1070 to fail, and another NUMBER to be tried. If no even divisor is found, then PRIME will retain its YES value and be printed on the screen. In this case the NEXT statement of the outer loop will finally be reached, causing the whole process to start over again, repeating until 50 primes have been found.

Loops — nested and otherwise — can save a tremendous amount of repetitive coding in most programs. I have seen programs which repeat large sections of code, for example, for each of two different players. Often the key to avoiding such repetition is to use a loop in conjunction with subscripted variables, and sometimes with one or more "switches" which flip between two or more different states to keep track of which player is the current one. Subscripted variables will be covered in some detail in a later article, and at that time I'll expound more scholarly thoughts on conserving code through their use.


Well, if last month left you all strung out, perhaps this month has left you going around in circles. Be that as it may, I'll be back next month once again to scrutinize the inscrutable and further fuddle the befuddled. 

Bugs, Worms

and other undesirables 

The S-80 program "ASSAULT IV" appearing in the April 1981 issue has several small bugs. The corrections (sent in by Marc Pierson of Warren, N.J.) are shown below.

```
300 IFL6<250THENL6=250
310 IFL6>890THENL6=890
320 IFL5<192THENL6=192
330 IFL5>832THENL6=832
1230 IF (V=1AND A1=0) OR (V=2AND A2=0) OR (V=
3AND A3=0) PRINT 64, "YOU HAVE NO "; DV$(V)
; " DIVISIONS LEFT! " : GOSUB 113
0: MS=MS+1: GOSUB 2280: GOTO 1160
2160 IFR=6 PRINT 64, "THE PRESIDENT HAS C
ALLED YOU "CHR$(34) "A TURKEY IN THE MI
DST OF EAGLES!"CHR$(34)
```



Imhotep (October, 1980) has a couple of small problems. In the Apple version, lines 790 and 1100 should be deleted since they refer to a non-existent error-handling routine. In both Apple and S-80 versions, line 920 can cause more than the whole population to be wiped out by pestilence. A suggested simple fix would be to change the beginning of each line to:

Apple: 920 Q = RND(1) * P/2: ...etc.

S-80: 920 Q = RND(P/2): ...etc.

Apple One Liner

```
10 DATA 1,1,6,0,12,0,33,63,54,45
,4,0: FOR A = 768 TO 779: READ
B: POKE A,B: NEXT : POKE 232
,0: POKE 233,3: HGR2 : HCOLOR=
3: HPLOT 0,0: CALL 62454:C =
INT ( RND (1) * 8): HCOLOR=
C - (C = 3 OR C = 7): FOR A =
1 TO 100 STEP INT ( RND (1)
* 4) + 1: SCALE= A: ROT= A:
DRAW 1 AT 139,95: NEXT : GET
A$: RUN
```

Roland Gustafsson
San Mateo, CA

SUPERSCRIBE II

THE BEST 250 DOLLAR WORD PROCESSOR ON THE MARKET

and it's only
\$ 129.95

SuperScribe II is the most powerful and easiest to use Word Processor available for your Apple II or II Plus computer. Besides leaping tall buildings in a single bound, it:

- Gives true upper/lower case text on your screen with no additional hardware whatsoever.
- Works with documents larger than the amount of memory in your Apple—transparently to you!
- Edit not only letters but also any text or binary file, or even basic programs!
- Automatically generates up to 4 separate indices for your document!
- Save typing time through a unique ability to designate specified keys as commonly used words, phrases or even commands!
- Globally search for or replace character strings.
- SuperScribe II has a built-in instruction capability such that if you forget how to use a command and the manual is not close by - you may simply ask SuperScribe II.
- Supports multiple disk drives!
- Will support alternate character sets.
- Supports the shift key modification if made to your Apple.
- Lets you work with your text on a screen at a time basis—reducing typos and allowing you to see your document as you edit it.
- Works with any printer!
- Proportional Spacing!
- Hyphenations made
- Supports the language card or any 16K expansion Ram card to keep more of your document readily available in memory.
- 70 column review—see your text formatted in 70 character lines before you print. Again, without additional hardware!

#47-213004D..... \$129.95

by

ON-LINE systems



TSE-HARDSIDE

14 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

Flight of the Bumble Bee

by William Morris and John Cope

Flight of the Bumblebee is a musical animated graphics program for a 16K Apple (with Applesoft), Atari, or S-80.

Those of you without culture may have to be informed that *The Flight of the Bumblebee* is not a beekeeper simulation, but rather a classic tune from Rimsky-Korsakoff's 1900 opera, *The Tale of the Tsar Saltan*. The authors of this program, whose names are familiar to regular *SoftSide* readers, have digitized and animated this great Russian composer's work to suit the music and graphic capabilities of the S-80, Atari, and Apple computers.

No special instructions are needed; you just type in the program of your choice and RUN. The arrangement for generating the sound on the S-80 is the usual one: Connect the cable going to the cassette microphone jack to an amplifier and adjust the volume to suit your tastes. If you have an extension speaker, you can use your cassette recorder itself as the amplifier: Plug the cable into the microphone jack, plug the speaker into the earphone jack, put the recorder in RECORD mode, and you're all set. The Atari version uses the television speaker, and the Apple version uses the built-in computer speaker.

APPLE VARIABLES

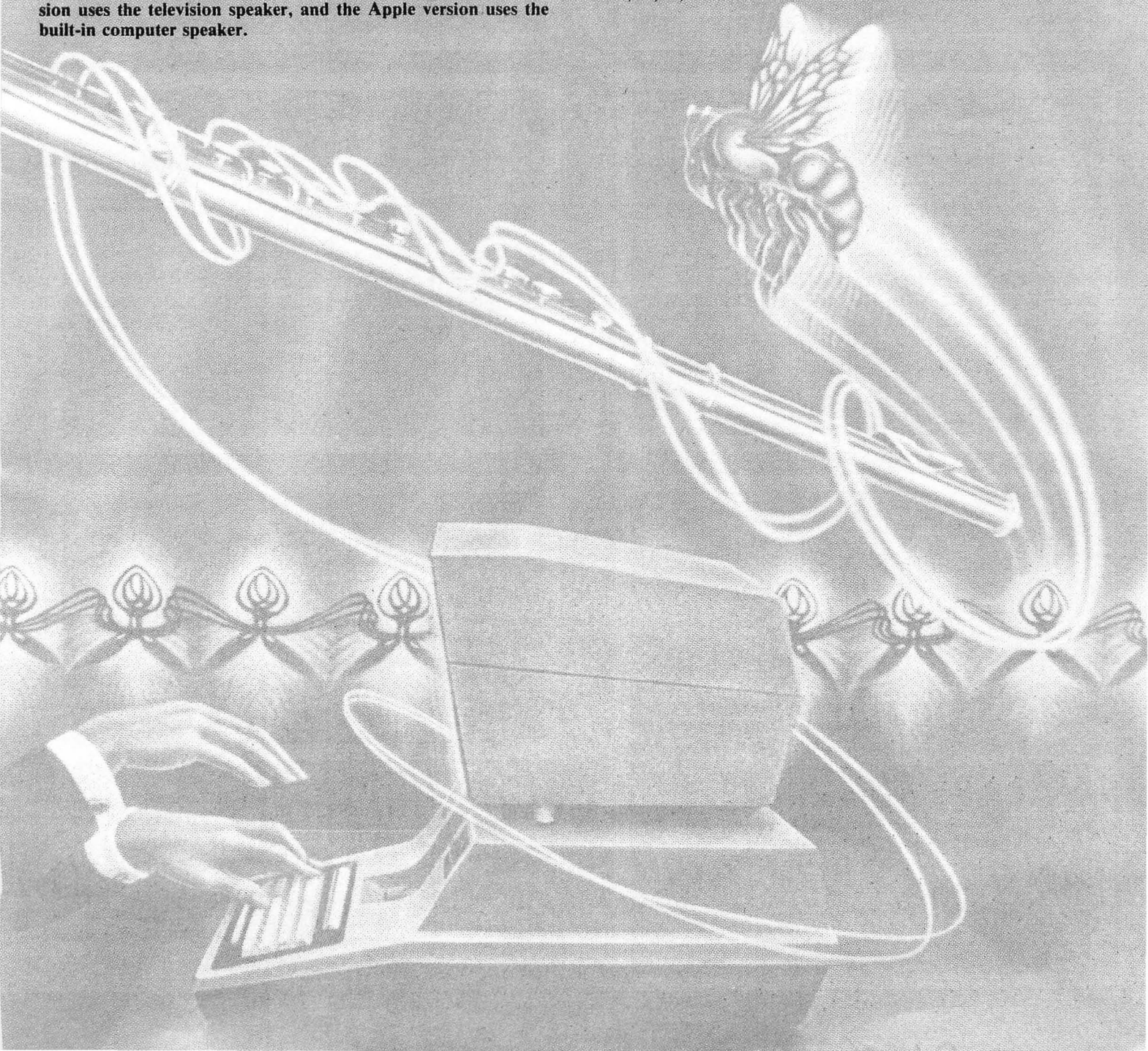
UN: The pitch of the note to be played by the sound routine.
X, Y, Z: Counters.

ATARI VARIABLES

UN: The pitch of the note to be played by the sound routine.
UB: Memory location used in the routine to set the three graphics modes used in the title page.
UC: Memory location used in moving the redefined character set.
UZ, Z: Counters.

S-80 VARIABLES

AS: "Beehive" graphic string.
JMS: "Packed string" used to initialize the sound routine.
U: Memory location used in the sound routine.
UN: The pitch of the note to be played by the sound routine.
UL: The length of the note to be played by the sound routine.
W, X, Y, Z: Counters.



SoftSide Selections

6 South Street Milford NH 03055

NO TELEPHONE ORDERS
NO CREDIT CARDS ACCEPTED
MAIL ORDERS ONLY

ATARI

MUSIC COMPOSER
CARTRIDGE

\$ 46.88

(P&H=\$2.25)

APPLE

ALF AM-II
Music Synthesizer

\$ 123.88

(P&H=\$3.50)

CENTRONICS 739 PRINTER

- * prints up to 132 chr/line
- * proportional spacing
- * handles single sheet paper
- * roll or pin-feed paper
- * 100% duty cycle
- * high resolution graphics
- * descending lower case

\$ 619.88

(P&H=\$6.50)

TRS-80

ORCHESTRA-85
\$ 109.88

ORCHESTRA-90 (Mod-III)
\$ 123.88

ORCHESTRA-80
\$ 72.88

(P&H=\$3.00)

*** STRICTLY LIMITED ***

ATARI/Angle Worms/Game on Tape from A.I.	\$11
ATARI/Deflection/Game on Tape from A.I.	\$11
TRS-80/Lunar Lander/Game on Tape from A.I.	\$10
TRS-80/Missile Attack/Game on Tape from A.I.	\$11
TRS-80/Simutek-I/Game on Tape from A.I.	\$11
TRS-80/Capture & Simon/Games on Tape from A.I.	\$10
TRS-80/Slag/Game on Tape from A.I.	\$11

* QUANTITIES ARE LIMITED *
(P&H=\$1.50 ea)

FAMOUS BRAND DISKETTES

5-1/4" Box of 10

\$ 23.88

(P&H=\$1.50/bx)

TERMS: All sales are final (except for defective merchandise). In the case of LIMITED ITEMS, checks will be returned when supply is exhausted. Orders without proper P&H will be returned. This ad replaces all other SoftSide Selection ads.

ALL PRICES EXPIRE DECEMBER 15th 1981

Apple Version

The text screen is set and cleared while the ONERR function is established as the routine to be used when all of the data have been read.

```
10 TEXT : HOME : ONERR GOTO 199
```

Title display.

```
20 VTAB 2: HTAB 7: PRINT "THE FLIGHT OF THE BUMBLE-BEE": VTAB 13: HTAB 11: PRINT "BY RIMSKY-KORSAKOFF": INVERSE : VTAB 23: HTAB 6: PRINT "(C) WM. MORRIS & J. COPE 1981": NORMAL : FOR Z = 1 TO 2000: NEXT
```

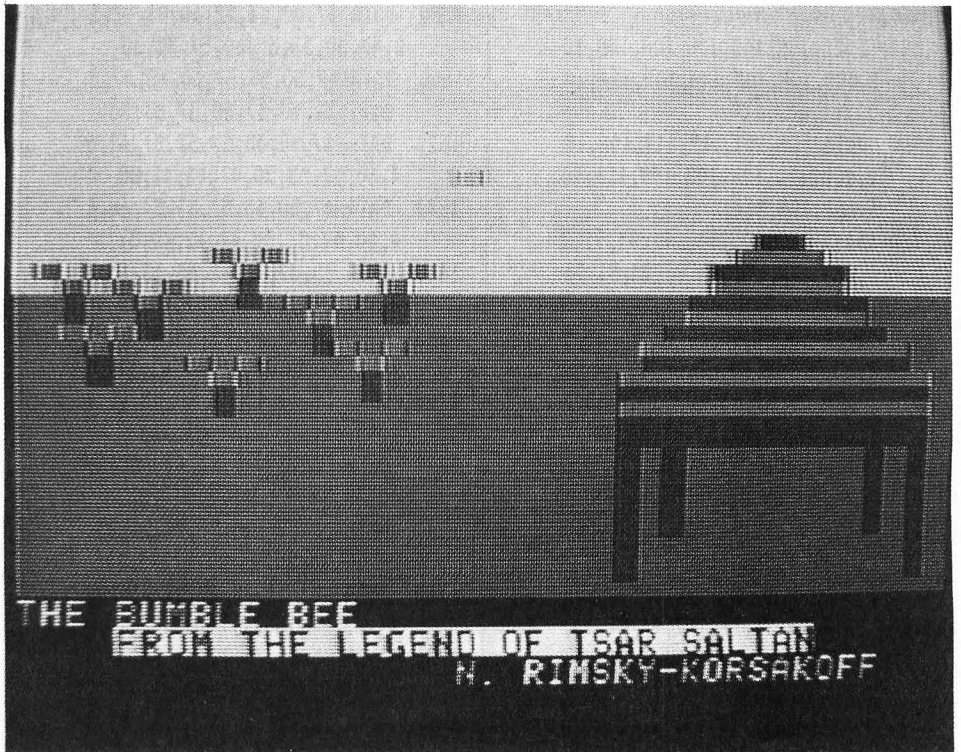
The display for the text window is set for the duration of the song.

```
100 HOME : VTAB 21: PRINT "THE BUMBLE-BEE": HTAB 5: INVERSE : PRINT "FROM THE LEGEND OF TSAR SALTAN": NORMAL : HTAB 19: PRINT "N. RIMSKY-KORSAKOFF":
```

The graphics display is established with lines 110 and 120 setting the "sky" and "grass," while lines 130 and 140 place the "beehive" on the screen. Line 150 is the "flower" drawing routine.

```
110 GR : COLOR= 6: FOR Z = 0 TO 19: HLINE 0,39 AT Z: NEXT
120 COLOR= 12: FOR Z = 20 TO 39: HLINE 0,39 AT Z: NEXT
130 FOR Z = 16 TO 28 STEP 2: READ X,Y: COLOR= 8: HLINE X,Y AT Z : READ X,Y: COLOR= 9: HLINE X,Y AT Z + 1: NEXT
140 COLOR= 8: HLINE 25,38 AT 29: VLINE 30,38 AT 25: VLINE 30,38 AT 3 : VLINE 30,35 AT 27: VLINE 30,35 AT 36
150 COLOR= 9: FOR Z = 1 TO 24: READ X,Y: PLOT X,Y: NEXT : COLOR= 4: FOR Z = 1 TO 8: READ X,Y: PLOT X,Y: PLOT X,Y + 1: NEXT
```

Both sound and graphics display are integrated in line 210. The sound variable UN is read from the data statements and is subsequently utilized both as the musical note and as the means to plot the position of the "bee" on the screen. Note how the algorithm used does not permit the "bee" to appear outside the limits of the predefined "sky."



```
200 GOSUB 25000
210 READ UN: COLOR= 13: PLOT UN / 4,20 - UN / 10: POKE 768,UN: POKE 769,25: CALL 770: COLOR= 6: PLOT UN / 4,20 - UN / 10: GOTO 210
```

The initial graphics data can be found in lines 1000-1020, with line 1030 containing the Machine Language coding used to initialize the sound capabilities of the Apple. The remaining lines contain the data for the music frequencies.

```
1000 DATA 31,32,30,33,29,34,29,34,28,35,28,35,27,36,26,37,26,37,25,38,25,38,25,38,25,38,25,38
1010 DATA 1,18,2,19,3,18,4,19,5,20,6,19,8,17,9,18,10,17,11,20,12,21,13,20,14,18,15,19,16,18,2,22,3,23,4,22,7,24,8,25,9,24,13,23,14,24,15,23
1020 DATA 2,20,5,21,9,19,12,22,15,20,3,24,8,26,14,25
1030 DATA 173,48,192,136,208,5,206,1,3,240,9,202,208,245,174,0,3,76,2,3,96
1040 DATA 96,102,108,114,108,114,121,128,96,102,108,114,108,114,121,128
1060 DATA 96,102,108,114,121,128,136,144,153,144,136,128,121,114,108,102
1080 DATA 96,102,108,114,121,91,96,102,96,102,108,114,121,114,108,102
1100 DATA 96,102,108,114,121,91,96,102,96,102,108,114,121,114,108,102
1120 DATA 96,102,108,114,108,114,121,128,121,114,108,102,96,91,96,102
1140 DATA 96,102,108,114,108,114,121,128,121,114,108,102,96,85,81,76
1160 DATA 72,76,81,85,91,68,72,76,72,76,81,85,91,85,81,76
1180 DATA 72,76,81,85,91,68,72,76,72,76,81,85,91,85,81,76
1200 DATA 72,76,81,85,81,85,91,96,91,85,81,76,72,68,72,76
1220 DATA 72,76,81,85,91,96,102,96,91,85,81,76,72,68,72,76
1240 DATA 72,144,144,144,144,144,144,144,136,153,136,153,136,153,136,153
1260 DATA 144,144,144,144,144,144,144,144,136,153,136,153,136,153,136,153
1280 DATA 144,136,144,153,144,136,144,153,144,136,144,153,144,136,144,153
1300 DATA 144,136,128,121,114,121,128,136,144,136,128,121,114,108,102,96
1320 DATA 108,108,108,108,108,108,108,108,102,114,102,114,102,114,102,114
```

continued on next page

continued from previous page

1340 DATA 108,108,108,108,108,108,
8,108,108,102,114,102,114,10
2,114,102,114

1360 DATA 108,102,108,114,108,10
2,108,114,108,102,108,114,10
8,102,108,114

1380 DATA 108,102,96, 91,85,91,9
6,102,108,102,96,91,85,81,76
,72

1400 DATA 53,57,60,64,68,50,53,5
7,53,57,60,64,68,64,60,57

1420 DATA 53,57,60,64,60,64,68,7
2,68,64,60,57,60,57,53,50

1440 DATA 47,50,53,57,53,57,60,6
4,60,64,68,72,76,81,85,91

1460 DATA 96,91,96,102,96,91,96,
102,96,91,96,102,96,91,96,10
2

1480 DATA 96,91,96,102,96,91,96,
102,96,91,96,102,96,91,96,10
2

1500 DATA 47,47,47,47,47,47,60,6
0,72,72,91,91,72,72,60,60

1520 DATA 47,47,47,47,47,47,60,6
0,72,72,91,91,72,72,60,60

1540 DATA 47,47,47,47,47,47,47,4
7,47,47,47,47,47,47,47

1560 DATA 96,96,96,96,96,91,85,8
1,76,72,68,64,60,57,53,50

1580 DATA 47,50,53,57,60,45,47,5
0,47,50,53,57,60,57,53,50

1600 DATA 47,50,53,57,60,45,47,5
0,47,50,53,57,60,57,53,50

1620 DATA 47,50,53,57,53,57,60,6
4,60,57,53,50,47,45,47,50

1640 DATA 47,50,53,57,53,57,60,6
4,60,57,53,50,47,42,40,37

1660 DATA 35,37,40,42,45,33,35,3
7,35,37,40,42,45,42,40,37

1680 DATA 35,37,40,42,45,33,35,3
7,35,37,40,42,45,42,40,37

1700 DATA 35,37,40,42,40,42,45,4
7,45,42,40,37,35,33,35,37

1720 DATA 35,37,40,42,47,45,42,
40,37,35,33,31,29,27,25,24

1740 DATA 23,24,25,27,29,21,23,
24,23,24,25,27,29,27,25,24

1760 DATA 23,24,25,27,29,21,23,
24,23,24,25,27,29,21,23,24

1780 DATA 85,81,76,72,68,64,60,5
7,53,57,60,64,60,64,68,72

1800 DATA 76,72,68,64,60,57,53,5
0,47,45,47,50,47,45,47,50

1820 DATA 47,81,76,72,68,64,60,5
7,53,57,60,64,60,64,68,72

1840 DATA 76,72,68,64,60,57,53,5
0,47,45,47,50,47,42,40,37

1860 DATA 35,37,40,42,40,42,45,4
7,45,47,50,53,57,60,64,68

1880 DATA 72,76,81,85,81,85,91,9
6,91,96,102,108,114,121,128,
136

1900 DATA 144,136,144,153,144,13
6,144,153,144,136,144,153,14
4,128,121,108

1920 DATA 96,91,96,102,96,91,96
,102,96,91,96,102,96,85,81,7
6

1940 DATA 72,81,85,91,96,102,108
,114,121,128,136,153,144,136
,128,121

1960 DATA 114,108,102,96,91,85,8
1,76,72,68,64,60,57,53,50,47

1980 DATA 35,35,35,35,35,35,35,
35,72,72,72,72,96,96,96,96

Once the final screen position of the "bee" has been set and the last note read from the data, the program recycles itself.

19999 COLOR= 13: PLOT 31,15: POKE
768,144: POKE 769,255: CALL
770: FOR Z = 1 TO 3000: NEXT
Z: RUN

POKEs in the Machine Language sound subroutine.

25000 FOR X = 770 TO 790: READ Z
: POKE X,Z: NEXT
25010 RETURN

Spelling Errors? Does your TRS-80* wordprocessor need help?

PROOF READERTM

ASPEN
SOFTWARE
COMPANYTM

CAN SPELL rendezvous AND mnemonic AND OVER 38,000 OTHER WORDS

Now let TRS-80 and Proofreader by Aspen Software Company check your Scripsit*, Electric Pencil, or other documents for spelling and typographical errors. It has all the features needed to meet your proofreading requirements.

• **Proof-EditTM, optional interactive corrections feature for Model I/III**

- Checks every single word of even your biggest document in under 5 minutes.
- The 38,000 word dictionary is one of the largest available.
- Dictionary can be easily extended to add more words such as technical terms or names.
- All unknown words are listed on the screen and can be saved on a file for printing.
- Works with almost any TRS-80 wordprocessor including Scripsit and Electric Pencil.
- Comes with complete and easy to understand User's Manual.

* Trademark of Tandy Corporation

Model I

Requires 32K RAM, 1 disk drive, TRSDOS or NEWDOS

Proof Reader

#25-281001D.....\$54.00

Proof-Edit

#25-281003D.....\$30.00

Model III


Requires 32K RAM, 1 disk drive, TRSDOS

Proof Reader

#27-281001D.....\$64.00

Proof-Edit

#27-281003D.....\$30.00


TSE-HARDSIDE
14 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

Atari Version

Line 19999 is set to be accessed when all of the data have been read, while also reserving memory space for the redefined character set.

```
10 TRAP 19999:GOSUB 30300
```

Title display.

```
20 GOSUB 30200:POKE 87,2:POSITION 3,1:
? #6;"THE BUMBLE BEE"
30 POKE 87,1:POSITION 1,7:? #6;"by rim
sky korsakoff":POKE 752,1
40 POKE 87,0:POSITION 5,14:? "(c) Wm.
Morris & J. Cope 1981"
100 GOSUB 30310
```

This program segment sets the graphics display for the song. Line 110 establishes the colors to be displayed while line 130 sets the display for the "text window." Line 140 draws in the "grass," with lines 150-160 using the redefined character set to place the "flowers" on the screen. Lines 170-190 complete the display with the "beehive." Note: The COLOR 87, 130, 214, and 215 statements used in this section are NOT misprints; they are the method used to print the redefined characters.

```
110 GRAPHICS 1:POKE 756,UC/255+2:SETCO
LOR 0,15,2:SETCOLOR 1,14,12:SETCOLOR 2
,11,0:SETCOLOR 3,3,0:SETCOLOR 4,8,4
120 POKE 752,1
130 ? " The Bumble Bee ":? " v":? "
      from The Legend of Tsar Saltan":?
      " N. Rimsky-Korsakoff
";
140 COLOR 215:FOR Z=14 TO 19:PLOT 0,Z:
DRAWTO 19,Z:NEXT Z
150 FOR Z=0 TO 9:IF Z=5 OR Z=7 THEN NE
XT Z
160 COLOR 130:PLOT 7,12:COLOR 214:PLOT
7,13:NEXT Z
170 COLOR 4:PLOT 14,8:COLOR 3:PLOT 15,
8:COLOR 5:PLOT 16,8:COLOR 3:PLOT 14,9:
DRAWTO 16,9
180 COLOR 87:FOR Z=11 TO 15 STEP 2:PLD
T 12,Z:DRAWTO 18,Z:NEXT Z:COLOR 4:PLOT
13,9:COLOR 5:PLOT 17,9
190 COLOR 3:FOR Z=10 TO 14 STEP 2:PLOT
12,Z:DRAWTO 18,Z:NEXT Z:COLOR 4:PLOT
12,10:COLOR 5:PLOT 18,10
```

Both sound and graphics displays are integrated in line 200. Note how the simple algorithm, based on the pitch of the note played, will not allow the "bee" to move outside the limits of the "sky." Line 220 erases the "bee" from the screen before returning to line 200 for the next plot position.



```
200 COLOR 1:READ UN:PLOT UN/9,12-UN/15
:SOUND 0,UN,10,8
210 FOR Z=1 TO 15:NEXT Z
220 SOUND 0,0,0,0:COLOR 0:PLOT UN/9,12
-UN/15:GOTO 200
```

The data used to redefine the character set are found in lines 1000-1011. The remaining data lines contain the data for the music frequencies.

```
1000 DATA 0,0,0,0,0,0,0,0
1001 DATA 216,81,115,214,116,30,31,14
1002 DATA 40,146,146,214,254,124,56,16
1003 DATA 170,170,170,170,170,170,170,
170
1004 DATA 0,2,10,42,42,170,170,170
1005 DATA 0,128,160,168,168,170,170,17
0
1010 DATA 17,19,151,222,80,112,16,16
1011 DATA 255,255,255,255,255,255,255,
255
1040 DATA 96,102,108,114,108,114,121,1
28,96,102,108,114,108,114,121,128
1060 DATA 96,102,108,114,121,128,136,1
44,153,144,136,128,121,114,108,102
1080 DATA 96,102,108,114,121,91,96,102
,96,102,108,114,121,114,108,102
1100 DATA 96,102,108,114,121,91,96,102
,96,102,108,114,121,114,108,102
1120 DATA 96,102,108,114,108,114,121,1
28,121,114,108,102,96,91,96,102
1140 DATA 96,102,108,114,108,114,121,1
28,121,114,108,102,96,85,81,76
1160 DATA 72,76,81,85,91,68,72,76,72,7
6,81,85,91,85,81,76
```

```
1180 DATA 72,76,81,85,91,68,72,76,72,7
6,81,85,91,85,81,76
1200 DATA 72,76,81,85,81,85,91,96,91,8
5,81,76,72,68,72,76
1220 DATA 72,76,81,85,91,96,102,96,91,
85,81,76,72,68,72,76
1240 DATA 72,144,144,144,144,144,144,1
44,136,153,136,153,136,153,136,153
1260 DATA 144,144,144,144,144,144,144,
144,136,153,136,153,136,153,136,153
1280 DATA 144,136,144,153,144,136,144,
153,144,136,144,153,144,136,144,153
1300 DATA 144,136,128,121,114,121,128,
136,144,136,128,121,114,108,102,96
1320 DATA 108,108,108,108,108,108,108,
108,102,114,102,114,102,114,102,114
1340 DATA 108,108,108,108,108,108,108,
108,102,114,102,114,102,114,102,114
1360 DATA 108,102,108,114,108,102,108,
114,108,102,108,114,108,102,108,114
1380 DATA 108,102,96,91,85,91,96,102,1
08,102,96,91,85,81,76,72
1400 DATA 53,57,60,64,68,50,53,57,53,5
7,60,64,68,64,60,57
1420 DATA 53,57,60,64,60,64,68,72,68,6
4,60,57,60,57,53,50
1440 DATA 47,50,53,57,53,57,60,64,60,6
4,68,72,76,81,85,91
1460 DATA 96,91,96,102,96,91,96,102,96
,91,96,102,96,91,96,102
1480 DATA 96,91,96,102,96,91,96,102,96
,91,96,102,96,91,96,102
1500 DATA 47,47,47,47,47,60,60,72,7
2,91,91,72,72,60,60
```

continued on next page

continued from previous page

1520 DATA 47,47,47,47,47,47,60,60,72,7
 2,91,91,72,72,60,60
 1540 DATA 47,47,47,47,47,47,47,47,4
 7,47,47,47,47,47,47
 1560 DATA 96,96,96,96,96,91,85,81,76,7
 2,68,64,60,57,53,50
 1580 DATA 47,50,53,57,60,45,47,50,47,5
 0,53,57,60,57,53,50
 1600 DATA 47,50,53,57,60,45,47,50,47,5
 0,53,57,60,57,53,50
 1620 DATA 47,50,53,57,53,57,60,64,60,5
 7,53,50,47,45,47,50
 1640 DATA 47,50,53,57,53,57,60,64,60,5
 7,53,50,47,42,40,37
 1660 DATA 35,37,40,42,45,33,35,37,35,3
 7,40,42,45,42,40,37
 1680 DATA 35,37,40,42,45,33,35,37,35,3
 7,40,42,45,42,40,37
 1700 DATA 35,37,40,42,40,42,45,47,45,4
 2,40,37,35,33,35,37
 1720 DATA 35,37,40,42,47,45,42,40,37,3
 5,33,31,29,27,25,24
 1740 DATA 23,24,25,27,29,21,23,24,23,2
 4,25,27,29,27,25,24
 1760 DATA 23,24,25,27,29,21,23,24,23,2
 4,25,27,29,21,23,24
 1780 DATA 85,81,76,72,68,64,60,57,53,5

7,60,64,60,64,68,72
 1800 DATA 76,72,68,64,60,57,53,50,47,4
 5,47,50,47,45,47,50
 1820 DATA 47,81,76,72,68,64,60,57,53,5
 7,60,64,60,64,68,72
 1840 DATA 76,72,68,64,60,57,53,50,47,4
 5,47,50,47,42,40,37
 1860 DATA 35,37,40,42,40,42,45,47,45,4
 7,50,53,57,60,64,68
 1880 DATA 72,76,81,85,81,85,91,96,91,9
 6,102,108,114,121,128,136
 1900 DATA 144,136,144,153,144,136,144,
 153,144,136,144,153,144,128,121,108
 1920 DATA 96,91,96,102,96,91,96,102,96
 ,91,96,102,96,85,81,76
 1940 DATA 72,81,85,91,96,102,108,114,1
 21,128,136,153,144,136,128,121
 1960 DATA 114,108,102,96,91,85,81,76,7
 2,68,64,60,57,53,50,47
 1980 DATA 35,35,35,35,35,35,35,35,72,7
 2,72,72,96,96,96,96
 Once the final screen position of
 the "bee" has been set and the final
 note read, the program recycles
 itself.
 19999 SOUND 0,144,10,8:COLOR 1:PLOT 15
 ,7:RESTORE :FOR Z=1 TO 64:READ X:NEXT
 Z:TRAP 19999:SOUND 0,0,0

20000 FOR Z=1 TO 400:NEXT Z:GOTO 110

This segment adjusts the screen display to permit three graphics modes to be used in the title display.

30200 GRAPHICS 0:SETCOLOR 2,8,4:SETCOL
 OR 4,8,4:UB=PEEK(560)+PEEK(561)*256+4:
 POKE UB-1,70:POKE UB+2,7:POKE UB+3,7
 30210 FOR UZ=4 TO 8:POKE UB+UZ,6:NEXT
 UZ:POKE UB+22,65:POKE UB+23,PEEK(560):
 POKE UB+24,PEEK(561):SETCOLOR 3,3,0
 30220 SETCOLOR 1,5,0:RETURN

Memory space is allocated for the redefined character set in line 30300, with the remaining lines reading into memory the new character set which includes the graphics figures of the "bee" and the "flowers."

30300 POKE 106,PEEK(106)-5:RETURN
 30310 UC=256+(PEEK(106)+1):FOR UZ=0 TO
 1023:POKE UC+UZ,PEEK(57344+UZ):NEXT U
 Z
 30320 POKE 756,UC/256:FOR UZ=512 TO 55
 9:READ UY:POKE UC+UZ,UY:NEXT UZ
 30330 FOR UZ=944 TO 959:READ UY:POKE U
 C+UZ,UY:NEXT UZ:RETURN

ASPEN SOFTWARE COMPANY T.M.

GRAMMATIK T.M.

BEYOND SPELLING CHECKING

A spelling checker may not be enough! This paragraph contains a number of common errors (indicated by underlining) that will be discovered by Grammatik that would seldom ever be caught by a spelling checker. FOR example, Grammatik checks for improper word usage as identified by a number of writing style manuals (such as "seldom ever"). Grammatik will check for the presence of certain words such as jargon or sexist terms. it also checks for constant punctuation, cap-

italizAtion, balanced quotation marks and parentheses, and and repeated words. In addition, it will produce a list of all unique words found in your document with the number of times each was used. Grammatik comes with a dictionary of commonly misused phrases and a dictionary of sexist terms. It also includes a complete set of utilities to build, sort, and merge phrase and jargon dictionaries of your own. Works with Scripsit, Electric Pencil, and other standard TRSDOS text files.

Model I

Requires 32K RAM, 1 disk drive, TRSDOS or NEWDOS

Grammatik

#25-281002D.....\$49.00

Model III

Requires 32K RAM, 1 disk drive, TRSDOS

Grammatik

#27-281002D.....\$59.00



14 South St., Milford, NH 03055 (603) 673-5144
 TOLL FREE OUT-OF-STATE 1-800-258-1790



S-80 Version

After clearing the screen and defining the integer and string variables, line 1999 is set to be accessed when all of the data lines have been read through.

```
10 CLS: CLEAR1000: DEFSTR A-J: DEFINT V-Z: ON ERROR GOTO 19999
```

Title display.

```
20 PRINTCHR$(23): PRINT@68, "THE FLIGHT OF THE BUMBLE BEE": PRINT@4  
62, "by Rimsky-Korsakoff": PRINT@898, "(c) Wm. Morris & J. Cope 198  
1": FORZ=1 TO 2000: NEXT
```

This program segment sets the graphics display for the music. Line 100 sets the title display, with lines 110-130 setting the display of the "sky," "grass," and "flowers." Lines 140 and 150 complete the graphics display with the construction of the "beehive."

```
100 CLS: PRINTTAB(8) "THE BUMBLE BEE": PRINT: PRINTTAB(12) "from the  
opera, "; CHR$(34); "The Legend of Tsar Saltan"; CHR$(34): PRINTTAB(4  
40) "N. Rimsky-Korsakoff"  
110 FORZ=1 TO 32: A=A+CHR$(44)+CHR$(39): NEXT: Z=648: Y=47  
120 PRINT@Z, LEFT$(A, Y): Z=Z+62: Y=Y+4: IFY<64 THEN 120  
130 PRINT@580, " &&& &&& &&": PRINT@642, "#####  
";  
140 FORZ=18 TO 30 STEP 2: READX, Y: FORW=X TO Y: SET(W, Z): NEXT: READX, Y: FOR  
W=X TO Y STEP 2: SET(W, Z+1): NEXT: NEXT  
150 FORW=50 TO 92: SET(W, 32): NEXT: FORM=33 TO 41: SET(51, W): SET(91, W): N  
EXT: FORM=33 TO 37: SET(55, W): SET(87, W): NEXT
```

Both sound and graphics displays are integrated in line 210. Note how the simple algorithm, based on the pitch of the note played (UN), will not allow the "bee" to move outside the limits of the "sky."

```
200 GOSUB 30000
```

```
210 READUN, UL: SET(UN/2, UL+10): GOSUB 30100: RESET(UN/2, UL+10): GOTO 2  
10
```

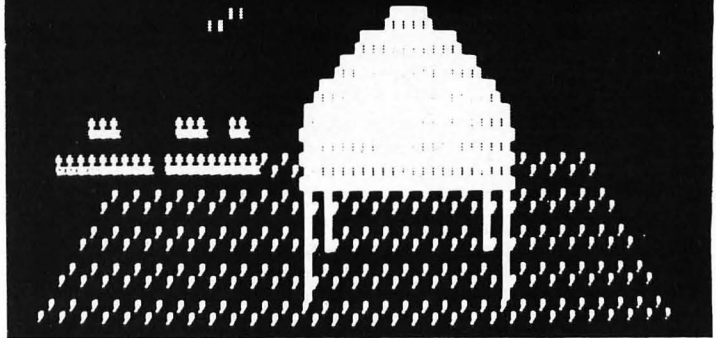
Data used for graphics display purposes, and to initialize the sound routine, are found in lines 1000-1010. The remaining lines contain the data for the pitch and duration of the music notes.

```
1000 DATA 68, 74, 67, 75, 62, 80, 61, 81, 58, 84, 57, 85, 54, 88, 53, 89, 51, 91, 5  
1, 91, 50, 92, 51, 91, 50, 92, 51, 91  
1010 DATA 17, 121, 45, 14, 255, 33, 1, 1, 45, 122, 237, 97, 67, 16, 254, 237, 10  
5, 67, 16, 254, 61, 32, 243, 21, 32, 239, 201  
1040 DATA 96, 7, 102, 7, 108, 7, 114, 7, 108, 7, 114, 7, 121, 6, 128, 6  
1050 DATA 96, 7, 102, 7, 108, 7, 114, 7, 108, 7, 114, 7, 121, 6, 128, 6  
1060 DATA 96, 7, 102, 7, 108, 7, 114, 7, 121, 6, 128, 6, 136, 6, 144, 6  
1070 DATA 153, 6, 144, 6, 136, 6, 128, 6, 121, 6, 114, 7, 108, 7, 102, 7  
1080 DATA 96, 7, 102, 7, 108, 7, 114, 7, 121, 6, 91, 7, 96, 7, 102, 7  
1090 DATA 96, 7, 102, 7, 108, 7, 114, 7, 121, 6, 114, 7, 108, 7, 102, 7  
1100 DATA 96, 7, 102, 7, 108, 7, 114, 7, 121, 6, 91, 7, 96, 7, 102, 7  
1110 DATA 96, 7, 102, 7, 108, 7, 114, 7, 121, 6, 114, 7, 108, 7, 102, 7  
1120 DATA 96, 7, 102, 7, 108, 7, 114, 7, 108, 7, 114, 7, 121, 6, 128, 6  
1130 DATA 121, 6, 114, 7, 108, 7, 102, 7, 96, 7, 91, 7, 96, 7, 102, 7  
1140 DATA 96, 7, 102, 7, 108, 7, 114, 7, 108, 7, 114, 7, 121, 6, 128, 6
```

THE BUMBLE BEE

FROM THE OPERA, "THE LEGEND OF TSAR SALTAN"

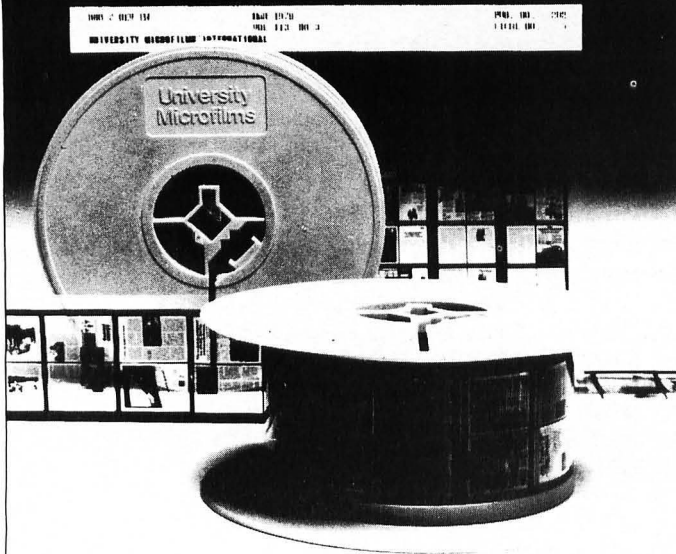
N. RIMSKY-KORSAKOFF



```
1150 DATA 121, 6, 114, 7, 108, 7, 102, 7, 96, 7, 85, 8, 81, 8, 76, 8  
1160 DATA 72, 8, 76, 8, 81, 8, 85, 8, 91, 7, 68, 9, 72, 8, 76, 8  
1170 DATA 72, 8, 76, 8, 81, 8, 85, 8, 91, 7, 85, 8, 81, 8, 76, 8  
1180 DATA 72, 8, 76, 8, 81, 8, 85, 8, 91, 7, 68, 9, 72, 8, 76, 8  
1190 DATA 72, 8, 76, 8, 81, 8, 85, 8, 91, 7, 85, 8, 81, 8, 76, 8  
1200 DATA 72, 8, 76, 8, 81, 8, 85, 8, 81, 8, 85, 8, 91, 7, 96, 7  
1210 DATA 91, 7, 85, 8, 81, 8, 76, 8, 72, 8, 68, 9, 72, 8, 76, 8  
1220 DATA 72, 8, 76, 8, 81, 8, 85, 8, 91, 7, 96, 7, 102, 7, 96, 7  
1230 DATA 91, 7, 85, 8, 81, 8, 76, 8, 72, 8, 68, 9, 72, 8, 76, 8  
1240 DATA 72, 8, 144, 6, 144, 6, 144, 6, 144, 6, 144, 6, 144, 6, 144, 6  
1250 DATA 136, 6, 153, 6, 136, 6, 153, 6, 136, 6, 153, 6, 136, 6, 153, 6  
1260 DATA 144, 6, 144, 6, 144, 6, 144, 6, 144, 6, 144, 6, 144, 6, 144, 6  
1270 DATA 136, 6, 153, 6, 136, 6, 153, 6, 136, 6, 153, 6, 136, 6, 153, 6  
1280 DATA 144, 6, 136, 6, 144, 6, 153, 6, 144, 6, 136, 6, 144, 6, 153, 6  
1290 DATA 144, 6, 136, 6, 144, 6, 153, 6, 144, 6, 136, 6, 144, 6, 153, 6  
1300 DATA 144, 6, 136, 6, 128, 6, 121, 6, 114, 7, 121, 6, 128, 6, 136, 6  
1310 DATA 144, 6, 136, 6, 128, 6, 121, 6, 114, 7, 108, 7, 102, 7, 96, 7  
1320 DATA 108, 7, 108, 7, 108, 7, 108, 7, 108, 7, 108, 7, 108, 7, 108, 7  
1330 DATA 102, 7, 114, 7, 102, 7, 114, 7, 102, 7, 114, 7, 102, 7, 114, 7  
1340 DATA 108, 7, 108, 7, 108, 7, 108, 7, 108, 7, 108, 7, 108, 7, 108, 7  
1350 DATA 102, 7, 114, 7, 102, 7, 114, 7, 102, 7, 114, 7, 102, 7, 114, 7  
1360 DATA 108, 7, 102, 7, 108, 7, 114, 7, 108, 7, 102, 7, 108, 7, 114, 7  
1370 DATA 108, 7, 102, 7, 108, 7, 114, 7, 108, 7, 102, 7, 108, 7, 114, 7  
1380 DATA 108, 7, 102, 7, 96, 7, 91, 7, 85, 8, 91, 7, 96, 7, 102, 7  
1390 DATA 108, 7, 102, 7, 96, 7, 91, 7, 85, 8, 81, 8, 76, 8, 72, 8  
1400 DATA 53, 9, 57, 9, 60, 9, 64, 9, 68, 9, 50, 10, 53, 9, 57, 9  
1410 DATA 53, 9, 57, 9, 60, 9, 64, 9, 68, 9, 64, 9, 60, 9, 57, 9  
1420 DATA 53, 9, 57, 9, 60, 9, 64, 9, 60, 9, 64, 9, 68, 9, 72, 8  
1430 DATA 68, 9, 64, 9, 60, 9, 57, 9, 60, 9, 57, 9, 53, 9, 50, 10  
1440 DATA 47, 10, 50, 10, 53, 9, 57, 9, 53, 9, 57, 9, 60, 9, 64, 9  
1450 DATA 60, 9, 64, 9, 68, 9, 72, 8, 76, 8, 81, 8, 85, 8, 91, 7  
1460 DATA 96, 7, 91, 7, 96, 7, 102, 7, 96, 7, 91, 7, 96, 7, 102, 7  
1470 DATA 96, 7, 91, 7, 96, 7, 102, 7, 96, 7, 91, 7, 96, 7, 102, 7  
1480 DATA 96, 7, 91, 7, 96, 7, 102, 7, 96, 7, 91, 7, 96, 7, 102, 7  
1490 DATA 96, 7, 91, 7, 96, 7, 102, 7, 96, 7, 91, 7, 96, 7, 102, 7  
1500 DATA 47, 10, 47, 10, 47, 10, 47, 10, 47, 10, 47, 10, 60, 9, 60, 9  
1510 DATA 72, 8, 72, 8, 91, 7, 91, 7, 72, 8, 72, 8, 60, 9, 60, 9  
1520 DATA 47, 10, 47, 10, 47, 10, 47, 10, 47, 10, 47, 10, 60, 9, 60, 9  
1530 DATA 72, 8, 72, 8, 91, 7, 91, 7, 72, 8, 72, 8, 60, 9, 60, 9  
1540 DATA 47, 10, 47, 10, 47, 10, 47, 10, 47, 10, 47, 10, 47, 10, 47, 10  
1550 DATA 47, 10, 47, 10, 47, 10, 47, 10, 47, 10, 47, 10, 47, 10, 47, 10  
1560 DATA 96, 7, 96, 7, 96, 7, 96, 7, 96, 7, 91, 7, 85, 8, 81, 8  
1570 DATA 76, 8, 72, 8, 68, 9, 64, 9, 60, 9, 57, 9, 53, 9, 50, 10  
1580 DATA 47, 10, 50, 10, 53, 9, 57, 9, 60, 9, 45, 11, 47, 10, 50, 10  
1590 DATA 47, 10, 50, 10, 53, 9, 57, 9, 60, 9, 57, 9, 53, 9, 50, 10  
1600 DATA 47, 10, 50, 10, 53, 9, 57, 9, 60, 9, 45, 11, 47, 10, 50, 10
```

continued on next page

This publication
is available in microform.



University Microfilms International

Please send additional information
for _____

Name _____
(name of publication)

Institution _____

Street _____

City _____

State _____ Zip _____

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

30-32 Mortimer Street
Dept. P.R.
London WIN 7RA
England

continued from previous page

1605 DATA47, 10, 50, 10, 53, 9, 57, 9, 60, 9, 57, 9, 53, 9, 50, 10
1610 DATA47, 10, 50, 10, 53, 9, 57, 9, 53, 9, 57, 9, 60, 9, 64, 9
1620 DATA60, 9, 57, 9, 53, 9, 50, 10, 47, 10, 45, 11, 47, 10, 50, 10
1630 DATA47, 10, 50, 10, 53, 9, 57, 9, 53, 9, 57, 9, 60, 9, 64, 9
1640 DATA60, 9, 57, 9, 53, 9, 50, 10, 47, 10, 42, 11, 40, 11, 37, 12
1650 DATA35, 12, 37, 12, 40, 11, 42, 11, 45, 11, 33, 12, 35, 12, 37, 12
1660 DATA35, 12, 37, 12, 40, 11, 42, 11, 45, 11, 42, 11, 40, 11, 37, 12
1670 DATA35, 12, 37, 12, 40, 11, 42, 11, 45, 11, 33, 12, 35, 12, 37, 12
1680 DATA35, 12, 37, 12, 40, 11, 42, 11, 45, 11, 42, 11, 40, 11, 37, 12
1690 DATA35, 12, 37, 12, 40, 11, 42, 11, 40, 11, 42, 11, 45, 11, 47, 10
1700 DATA45, 11, 42, 11, 40, 11, 37, 12, 35, 12, 33, 12, 35, 12, 37, 12
1710 DATA35, 12, 37, 12, 40, 11, 42, 11, 47, 10, 45, 11, 42, 11, 40, 11
1720 DATA37, 12, 35, 12, 33, 12, 31, 13, 29, 13, 27, 14, 24, 14, 25, 15
1730 DATA23, 15, 24, 15, 25, 14, 27, 14, 29, 13, 21, 16, 23, 15, 24, 15
1740 DATA23, 15, 24, 15, 25, 14, 27, 14, 29, 13, 27, 14, 25, 14, 24, 15
1750 DATA23, 15, 24, 15, 25, 14, 27, 14, 29, 13, 21, 16, 23, 15, 24, 15
1760 DATA23, 15, 24, 15, 25, 14, 27, 14, 29, 13, 21, 16, 23, 15, 24, 15
1770 DATA85, 8, 81, 8, 76, 8, 72, 8, 68, 9, 64, 9, 60, 9, 57, 9
1780 DATA53, 9, 57, 9, 60, 9, 64, 9, 60, 9, 64, 9, 68, 9, 72, 8
1790 DATA76, 8, 72, 8, 68, 9, 64, 9, 60, 9, 57, 9, 53, 9, 50, 10
1800 DATA47, 10, 45, 11, 47, 10, 50, 10, 47, 10, 45, 11, 47, 10, 50, 10
1810 DATA47, 10, 81, 8, 76, 8, 72, 8, 68, 9, 64, 9, 60, 9, 57, 9
1820 DATA53, 9, 57, 9, 60, 9, 64, 9, 60, 9, 64, 9, 68, 9, 72, 8
1830 DATA76, 8, 72, 8, 68, 9, 64, 9, 60, 9, 57, 9, 53, 9, 50, 10
1840 DATA47, 10, 45, 11, 47, 10, 50, 10, 47, 10, 42, 11, 40, 11, 37, 12
1850 DATA35, 12, 37, 12, 40, 11, 42, 11, 40, 11, 42, 11, 45, 11, 47, 10
1860 DATA45, 11, 47, 10, 50, 10, 53, 9, 57, 9, 60, 9, 64, 9, 68, 9
1870 DATA72, 8, 76, 8, 81, 8, 85, 8, 81, 8, 85, 8, 91, 7, 96, 7
1880 DATA91, 7, 96, 7, 102, 7, 108, 7, 114, 7, 121, 6, 128, 6, 136, 6
1890 DATA144, 6, 136, 6, 144, 6, 153, 6, 144, 6, 136, 6, 144, 6, 153, 6
1900 DATA144, 6, 136, 6, 144, 6, 153, 6, 144, 6, 128, 6, 121, 6, 108, 7
1910 DATA96, 7, 91, 7, 96, 7, 102, 7, 96, 7, 91, 7, 96, 7, 102, 7
1920 DATA96, 7, 91, 7, 96, 7, 102, 7, 96, 7, 85, 8, 81, 8, 76, 8
1930 DATA72, 8, 81, 8, 85, 8, 91, 7, 96, 7, 102, 7, 108, 7, 114, 7
1940 DATA121, 6, 128, 6, 136, 6, 153, 6, 144, 6, 136, 6, 128, 6, 121, 6
1950 DATA114, 7, 108, 7, 102, 7, 96, 7, 91, 7, 85, 8, 81, 8, 76, 8
1960 DATA72, 8, 68, 9, 64, 9, 60, 9, 57, 9, 53, 9, 50, 10, 47, 10
1970 DATA35, 12, 35, 12, 35, 12, 35, 12, 35, 12, 35, 12, 35, 12, 35, 12
1980 DATA72, 8, 72, 8, 72, 8, 72, 8, 96, 7, 96, 7, 96, 7, 96, 7

After all of the data have been read
through, the final graphics display
is established, after which the
program recycles itself.

19999 UN=144:UL=35:SET(71,17):GOSUB30100:PRINT@605," Honey \$1.23
";:FORZ=1TD3000:NEXT:RUN

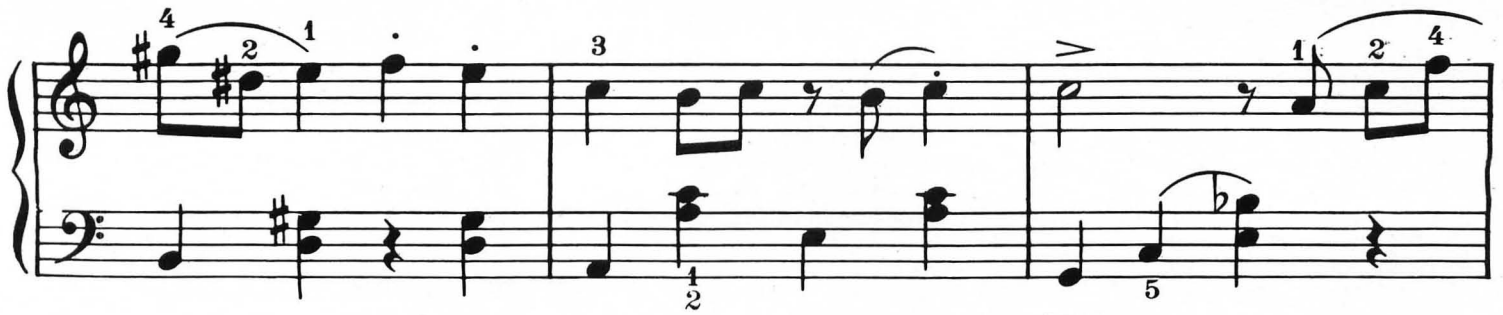
The sound subroutine is initialized
through the use of a "packed
string" procedure. Both Level II
BASIC and Disk BASIC are
accommodated with this routine.

```
30000 JM="":FORZ=1TD27:READY:JM=JM+CHR$(Y):NEXT:IFPEEK(16396)=20
1THEN30030
30010 CMD" T":U=VARPTR(JM):U=PEEK(U+2)*256+PEEK(U+1):IFU>32767THE
NU=U-65536
30020 DEFUSRO=U:RETURN
30030 U=VARPTR(JM):POKE16526,PEEK(U+1):POKE16527,PEEK(U+2):U=PEE
K(U+2)*256+PEEK(U+1):IFU>32767THENU=U-65536:RETURNELSERETURN
```

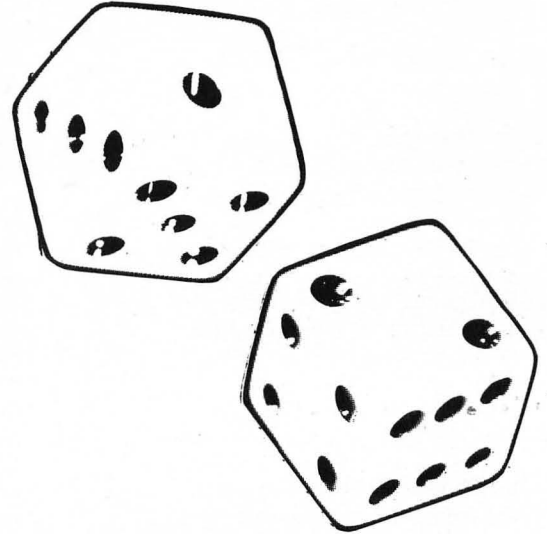
Music routine.

30100 POKEU+1,UN:POKEU+2,UL:US=USR(0):RETURN





MELODY DICE



by Gary Cage

Melody Dice is a musical/graphics game program for the Apple, Atari and S-80. System requirements: Apple with Applesoft and 24K RAM, Atari with 24K RAM, or S-80 Level II 16K or Disk 32K.

This is a computerized adaptation of a game in which there are 60 flash cards, each containing one measure of a Scott Joplin song. Ten of these cards are picked randomly by rolling a pair of dice five times. The numbers rolled are added to the roll number to determine the numbers of the cards. You then take these ten cards to the piano and play the "composition" which Scott Joplin never wrote, but which is nonetheless his music.

I adapted this format to the computer. Basically the program consists of a dice routine to choose the "cards" at random, a music routine to play them, a graphics routine to display the music as it is being played, and routines to save or recall a created song from disk or tape. In this case the cards consist of strings of numbers grouped in sixes; the first three numbers represent the note to be played, and the last three the note length.

The S-80 and Apple programs both include a Machine Language tone generation program. On the S-80 version, this routine is put into line 8000. Therefore, after the program is RUN, line 8000 will look very odd, as the computer is trying to display the Machine Language as BASIC tokens.

The Atari version also includes a Machine Language program, to draw the Hi-Res notes and to pause for the proper note duration. Atari BASIC was too slow to perform this function, so our Alan Zett worked out this routine.

VARIABLES

Note that the variables A\$(*), AA\$(*), and B\$(*) are replaced with A(*), AA(*), and B(*) on the Atari and S-80 versions. If A\$(i) = C\$(x) on the Apple, then A(i) would equal x on these versions.

A\$(*): Holds cards corresponding to die A's throw.
 AA\$(*): Holds A\$ cards while B\$ is being transferred to A\$. (A\$ is the only variable used in the playback routine.)
 AB: Lines up the five cards for each die.
 AN,AN\$: Player's response.
 B\$(*): Holds cards corresponding to die B's throw.
 BH,BV: Horizontal and vertical locations to print numbers on die B.
 C: Card number (1-60).
 C\$(*): String array for music cards.
 D\$: Control-D character for disk operation.
 DA: Number thrown on die A.
 DB: Number thrown on die B.
 DK: Switch to find place where error section should return after printing its message: DK = 0 goes back to SAVE routine; DK = 1 goes back to RECALL routine.
 ER: Error code.
 H: Variable for HTABS.
 HC: Horizontal clear variable for dice routine.
 I,II: General counter variables.
 KEY: Value of key pressed.
 L: Note length variable.
 LL: Cumulative length of notes (used

to determine when a full measure has been reached).

MO: Variable to increment column of numbers thrown by dice.

MS: Measure number.

N: Note variable (pitch).

NA\$: Program author's name; used for title page.

NN,NN%: Hold a value to add to the Y-axis when drawing the notes; i.e., determine where a note should be drawn.

S\$: Name given to saved song.

SW: Switch to test if FOR-NEXT loops have been used more than once (0 = no, 1 = yes).

T: Number of dice toss.

V: Variable for VTABS.

VA: Value to be POKEd into memory.

VC: Vertical clear variable for dice routine.

X,Y: X and Y axis values.

DOCUMENTATION

Lines 0-200: Main Program

On the Apple, the program overlaps the memory area which is used by page 1 of Hi-Res graphics, so the graphics display of the notes uses page 2, beginning at memory location 16384. The shape table which is used to draw the treble clef, time signature, and notes is placed just

continued on next page

continued from previous page

below this in memory, and is then protected by setting HIMEM below that at 16300. This part of the program is essentially the menu, asking if you want to play a song from memory, create a new one, or quit.

On the S-80, lines 12-16 are the graphics note display routine, found at line 4900 on the Apple and Atari. This routine has been put there to increase execution speed.

Lines 1000-1160: Initialization

This loads the shape table (Apple), music routine (Apple and S-80), and note routine (Atari) into memory, and assigns array numbers to the 60 cards, represented by the array C\$. C\$(0) is a bit of music used for the introduction and is not used in the rest of the program.

Lines 2000-2340: Title Page

This is a Lo-Res graphics routine that spells out *Melody Dice*. It then looks for C\$(0) and splits the string of numbers into groups of six. The first three are assigned to variable N (note) and the second three to variable L (length). On the Apple and S-80 they are then POKEd into the music routine, and a CALL (USR for S-80) is issued to play the note. The Atari uses the SOUND command to generate the tone. At the same time (Apple only), in the text part of the screen, one letter of the message "BY G. CAGE" is printed. This happens with each set of six numbers, until the name is completed. You are then returned to the main program and prompted concerning the instructions.

Lines 2500-2820: Instructions

If you answer "Y" to the question "Would you like instructions?" the program will GOSUB to this routine. In the Apple version, the POKE 34,3 sets the top margin of the text screen down three lines, in order to protect the title at the top even when the screen is cleared. Likewise, the POKE 35,22 in line 2470 sets the bottom margin up two lines to protect the message there. There are three pages of instructions. To go to the next page, a subroutine at line 2800 tests to see if the RETURN key has been pressed. If not, nothing happens. This is done by GETting a key and looking at its ASCII value (Apple and Atari) or with the INKEY\$ command (S-80). If this is equal to 13 (RETURN/

ENTER) then the program continues; if not, it waits for another keypress. After viewing the three pages of instructions, you are again returned to the main program where (in the Apple version) a TEXT instruction is given, resetting the text window to its normal dimensions.

Lines 3000-3920: Dice Routine

This is the routine which displays the dice and allows you to stop their spinning, thus creating random numbers. This is done five times. The number of the toss (first try, second try, etc.), and the numbers shown on each die, determine the music cards picked; this is done in lines 3600 and 3620. Die A picks cards 1 through 30, and die B picks cards 31 through 60. In this way it is possible to choose any of the 60 cards using only the six numbers on each die.

After C\$(C) is chosen, it is copied either into the string variable A\$ or B\$ (Apple) or A or B (Atari and S-80) for use by the music routine. The actual order of playback of the cards is not 1,2,3 . . . 10, but rather the following sequence, which makes the tune last twice as long: 1,2,3,4; 1,2,3,5; 6,7,8,9; 6,7,8,10; 1,2,3,5. Note also that the A\$ array is copied into the AA\$ array (or A into AA, Atari and S-80). This is because the actual music-playing routine uses only A\$ (A), and B\$ (B) gets copied into it at a later point. AA\$ (AA) is then used to finish the song.

Most of the dice routine involves randomly changing the number 1-6 for each die, continuing until the space bar is hit. The toss number (T) is then increased by one until it reaches five. You are then asked if you want to try the dice again (not being satisfied with the tosses) or hear the song by playing the cards with the dice picked.

Lines 4000-4360: Music routine

This begins with a GOSUB to 4500 where the music staff lines are drawn. It also sets up the variables which tell the program where to start drawing the notes. The routine at line 4900 (line 12, S-80) calculates where the notes should be drawn. The formula in line 4910 simply lowers the numbers in the NN% array in the following way. Say you want to plot the note D above middle C. This note would occupy the space just below the bottom staff line. If the Y-coordinate of the top

staff line is 10, and the space under that 12, and so on, then the Y-coordinate of the D note will be 28. Using the strange formula above, I would find that the note number of D is $X = 171$, so $NN = NN\%(20)$, which was initialized with a value of 18. Thus the note will be drawn at coordinates $X, Y + NN$ or $X, 28$.

The note length is added into variable LL, and when LL reaches or exceeds 255 (four counts) a bar is drawn to represent the end of a measure. When there are four measures to the line (seven in the S-80 version), X is reset to 27, and 40 is added to Y, thus moving the drawing to the next staff. A CALL 770 (Apple), USR call (S-80), or SOUND command (Atari) plays the note, and the whole process is repeated until the song is finished. You are then returned to the end of the dice routine and given the option of playing the same song again.

Lines 5000-5370: Save Routine

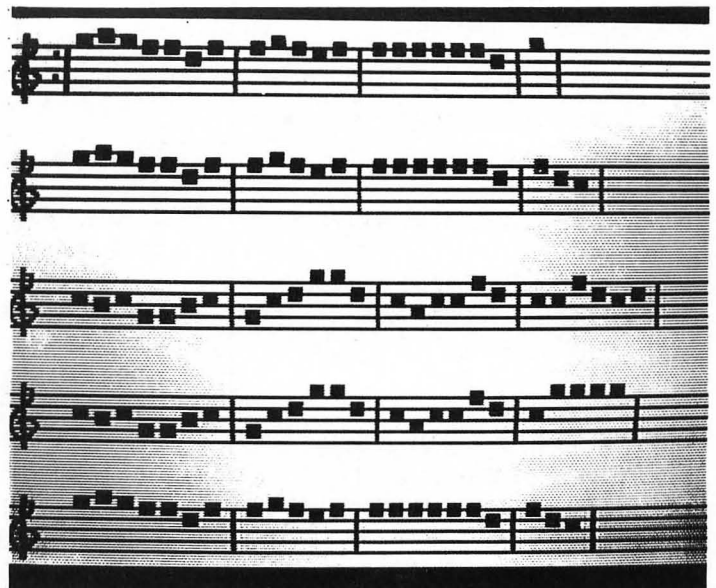
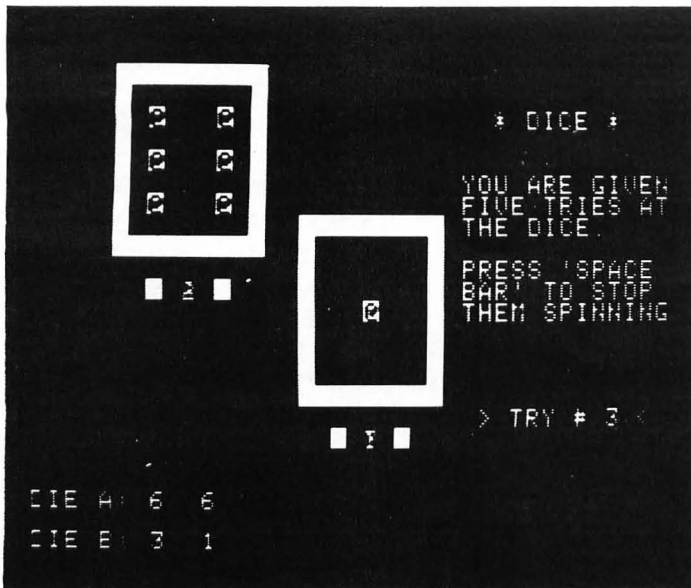
This begins by asking if you want to save the song; if not, you are returned to the main menu. But if so, you will be given the option of saving to either disk or tape. If you choose disk, there is an ONERR (Apple), TRAP (Atari), or ON ERROR (S-80) section which will trap mistakes such as write-protected or full disks which would ordinarily stop the program. The three string arrays A\$, B\$, and AA\$ (or A, B, and AA) are saved in a sequential file using the file name of your choice. You are then returned to the main menu.

Lines 6000-6340: Recall Routine

This is much the same as the Store routine, except that there is a routine which will catch a PROGRAM NOT FOUND disk error and will also DELETE the new file which was created by the OPEN instruction. After a song is loaded, the program GOSUBs to the music routine to play and draw the song. You are then returned to the main menu.

Lines 8000-10300: Data Statements

Lines 8000-8030 contain data for the shape table (Apple) or Atari note drawing routine; line 9000 holds data for the Machine Language music routine; and lines 10000-10300 contain data for the music cards. Typing in a lot of data is a very error-prone process, so check carefully for mistakes.



Apple Version

Main program.

```

10 HIMEM: 16300
20 TEXT : HOME
30 GOSUB 1000
40 GOSUB 2000
50 VTAB 2: PRINT "WOULD YOU LIKE
  INSTRUCTIONS (Y/N) ? ";
60 GET AN$
70 IF AN$ = "N" THEN GOTO 100
80 IF AN$ < > "Y" THEN GOTO 60
90 GOSUB 2500
100 TEXT : HOME
110 VTAB 4: HTAB 5: PRINT "*** W
  OULD YOU LIKE TO:"
120 PRINT : HTAB 9: PRINT "1) PL
  AY A SONG IN MEMORY?"
130 PRINT : HTAB 9: PRINT "2) CR
  EATE A NEW SONG?"
135 PRINT : HTAB 9: PRINT "3) PL
  AY A SONG FROM MEDIA"
140 PRINT : HTAB 9: PRINT "4) QU
  IT?"
150 VTAB 14: HTAB 13: PRINT "NUM
  BER-> ";
160 GET AN$
170 AN = VAL (AN$): IF AN < 1 OR

```

```

AN > 4 GOTO 160
175 IF AN = 1 THEN GOSUB 4000: GOTO
  100
180 IF AN = 2 THEN AB = 1: GOTO
  3000
190 IF AN = 3 THEN 6000
200 HOME : PRINT "OKAY ... BYE":
  END

```

Initialization.

```

1000 FOR I = 16301 TO 16364
1010 READ VA
1020 POKE I,VA
1030 NEXT I
1040 POKE 232,173: POKE 233,63
1050 FOR I = 770 TO 790
1060 READ VA
1070 POKE I,VA
1080 NEXT I
1090 DIM C$(60),A$(5),AA$(5),B$(
  5),NNZ(21)
1100 FOR I = 0 TO 60
1110 C$(I) = "": READ C$(I)
1120 NEXT I
1130 D$ = CHR$(4)
1140 NNZ(0) = -4: NNZ(1) = -2:
  NNZ(2) = 0: NNZ(3) = 0: NNZ(4)
  = 2: NNZ(5) = 4: NNZ(6) = 6: N

```

```

NZ(7) = 6: NNZ(8) = 8: NNZ(10)
  = 10: NNZ(11) = 10: NNZ(12) =
  12: NNZ(13) = 12
1150 NNZ(15) = 14: NNZ(16) = 14: NN
  Z(18) = 16: NNZ(20) = 18: NNZ(
  21) = 18
1160 RETURN

```

Title page.

```

2000 TEXT : HOME
2010 GR
2020 FOR I = 1 TO 3
2030 ON I GOSUB 2120,2140,2260
2040 NEXT I
2050 FOR I = 1 TO 1000: NEXT I
2060 VTAB 23: HTAB 25: PRINT "*"
  ; SPC(12); "*"
2070 NA$ = "BY G. CAGE": H = 27: V =
  23: I = 1
2080 FOR II = 1 TO LEN (C$(0)) STEP
  6: N = VAL ( MID$(C$(0),II,
  3)): L = VAL ( MID$(C$(0),I
  I + 3,3)): HTAB H: VTAB V: PRINT
  MID$(NA$,I,1): POKE 768,N:
  POKE 769,L: CALL 770
2090 H = H + 1: I = I + 1: NEXT II
2100 FOR I = 1 TO 2000: NEXT I
2110 TEXT : HOME : RETURN

```

continued on next page




```

9: HTAB H; PRINT " "; VTAB 1
6: HTAB 11 + H; PRINT " "; NEXT
H
3040 FOR V = 9 TO 1 STEP - 1: HTAB
6: VTAB V; PRINT " "; HTAB 1
7: VTAB 7 + V; PRINT " "; NEXT
V
3050 V = 11; H = 8: VTAB V; HTAB H
: PRINT " "; HTAB H + 4; PRINT
" "; VTAB V + 7; HTAB H + 11
: PRINT " "; HTAB H + 15; PRINT
" "
3060 NORMAL
3070 GOSUB 3360
3080 GOSUB 3430
3090 T = 0; DA = INT ( RND (1) *
6) + 1; DB = INT ( RND (1) *
6) + 1; MD = 0
3100 GOSUB 3470
3110 GOSUB 3500
3120 INVERSE : H = 8; V = 3
3130 BH = 0; BV = 0; ON DA GOSUB 3
190,3210,3230,3250,3270,3290
3140 BH = 11; BV = 7; ON DB GOSUB
3190,3210,3230,3250,3270,329
0
3150 NORMAL
3160 GOSUB 3600
3170 IF T > = 5 THEN FOR I = 1
TO 1000: NEXT I: GOTO 3800
3180 GOTO 3100
3190 HTAB H + 2 + BH; VTAB V + 2
+ BV; PRINT "Q"
3200 RETURN
3210 HTAB H + BH; VTAB V + BV; PRINT
"Q"; HTAB H + 4 + BH; VTAB V
+ 4 + BV; PRINT "Q"
3220 RETURN
3230 FOR I = 0 TO 4 STEP 2: HTAB
H + I + BH; VTAB V + I + BV;
PRINT "Q"; NEXT I
3240 RETURN
3250 GOSUB 3310
3260 RETURN
3270 GOSUB 3310; HTAB H + 2 + BH
: VTAB V + 2 + BV; PRINT "Q"

```

```

3280 RETURN
3290 GOSUB 3310; HTAB H + BH; VTAB
V + 2 + BV; PRINT "Q"; HTAB
H + 4 + BH; PRINT "Q"
3300 RETURN
3310 HTAB H + BH; VTAB V + BV; PRINT
"Q"; HTAB H + 4 + BH; PRINT
"Q"; HTAB H + 4 + BH; VTAB V
+ 4 + BV; PRINT "Q"; HTAB
H + BH; PRINT "Q"
3320 RETURN
3330 FOR VC = 2 TO 8: FOR HC = 7
TO 13: VTAB VC; HTAB HC; PRINT
" "; VTAB VC + BV; HTAB HC +
BH; PRINT " "
3340 NEXT HC; NEXT VC
3350 RETURN
3360 POKE 33,13; POKE 32,26
3370 VTAB 3; HTAB 3; PRINT "DI
CE *"
3380 VTAB 6; PRINT "YOU ARE GIVE
NFIVE TRIES ATTHE DICE."
3390 PRINT ; PRINT "PRESS 'SPACE
BAR' TO STOP THEM SPINNING"
;
3400 VTAB 17; HTAB 2; PRINT "> T
RY #"; SPC( 3); "<"
3410 TEXT
3420 RETURN
3430 POKE 34,20
3440 VTAB 21; PRINT "DIE A:"
3450 VTAB 23; PRINT "DIE B:"
3460 RETURN
3470 T = T + 1
3480 VTAB 17; HTAB 36; PRINT T
3490 RETURN
3500 V = 11; H = 10; VTAB V; HTAB
H; PRINT DA; VTAB V + 7; HTAB
H + 11; PRINT DB
3510 GOSUB 3530
3520 RETURN
3530 KEY = PEEK ( - 16384); IF K
EY = 160 THEN GOTO 3560
3540 DA = INT ( RND (1) * 6) + 1
; DB = INT ( RND (1) * 6) +
1
3550 POP ; GOTO 3500

```

```

3560 GOSUB 3330
3570 POKE - 16368,0; VTAB 21; HTAB
8 + MD; PRINT DA; VTAB 23; HTAB
8 + MD; PRINT DB
3580 MD = MD + 3
3590 RETURN
3600 C = DA * T + (6 - DA) * (T -
1)
3610 A$(AB) = C$(C)
3620 C = (DB * T + (6 - DB) * (T -
1)) + 30
3630 B$(AB) = C$(C)
3640 AA$(AB) = A$(AB)
3650 AB = AB + 1
3660 RETURN
3800 TEXT ; HOME ; VTAB 6; HTAB
5; PRINT "--> DO YOU WISH TO
PLAY THIS"; HTAB 10; PRINT
"SONG OR TRY AGAIN?"
3810 VTAB 9; HTAB 17; PRINT "TYP
E "; INVERSE : PRINT "P";;;
NORMAL ; PRINT " LAY OR ";;
INVERSE : PRINT "A";; NORMAL
; PRINT " GAIN."
3820 VTAB 11; HTAB 24; PRINT "--
> ";
3830 GET AN$
3840 IF AN$ = "A" THEN AB = 1: GOTO
3000
3850 IF AN$ < > "P" GOTO 3830
3860 GOSUB 4000
3870 FOR I = 1 TO 2000: NEXT I
3880 TEXT ; HOME ; PRINT "PLAY T
HE SAME SONG AGAIN ? ";
3890 GET AN$
3900 IF AN$ = "N" GOTO 5000
3910 IF AN$ < > "Y" GOTO 3890
3920 GOTO 3860

```

Music routine.

```

4000 HOME ; GOSUB 4500; SW = 0
4010 FOR I = 1 TO 4
4020 GOSUB 4300
4030 NEXT I
4040 FOR I = 1 TO 3
4050 GOSUB 4300

```

continued on next page



continued from previous page

```
4060 NEXT I
4070 I = 5
4080 GOSUB 4300
4090 IF SW = 1 GOTO 4120
4100 SW = 1: FOR I = 1 TO 5:A$(I)
    = B$(I): NEXT I
4110 GOTO 4010
4120 FOR I = 1 TO 5
4130 A$(I) = AA$(I)
4140 NEXT I
4150 FOR I = 1 TO 3
4160 GOSUB 4300
4170 NEXT I:I = 5
4180 GOSUB 4300
4190 RETURN
4300 FOR II = 1 TO LEN (A$(I)) STEP
    6
4310 N = VAL ( MID$ (A$(I),II,3)
    )
4320 L = VAL ( MID$ (A$(I),II +
    3,3) )
4330 GOSUB 4900
4340 POKE 768,N: POKE 769,L: CALL
    770
4350 NEXT II
4360 RETURN
4500 HGR2 : HCOLOR= 3: SCALE= 1:
    ROT= 0
4510 HPLLOT 0,0: CALL 62454: HCOLOR=
    0
4520 Y = 10:X = 0
4530 FOR I = 0 TO 16 STEP 4
4540 HPLLOT X,Y + I TO X + 279,Y +
    I
4550 NEXT I: IF Y = 50 GOTO 4600

4560 IF Y = 90 GOTO 4610
4570 IF Y = 130 GOTO 4620
4580 IF Y = 170 GOTO 4630
4590 Y = 50: GOTO 4530
4600 Y = 90: GOTO 4530
4610 Y = 130: GOTO 4530
4620 Y = 170: GOTO 4530
4630 X = 6:Y = 6
4640 FOR I = 0 TO 160 STEP 40
4650 HPLLOT X,Y + I TO X,Y + I +
    23
```

```
4660 DRAW 1 AT X,Y + 2 + I
4670 NEXT I
4680 DRAW 2 AT 17,11: DRAW 2 AT
    17,19
4690 LL = 0:MS = 1:X = 27:Y = 10
4700 RETURN
4900 IF N = 1 THEN N = 96: HCOLOR=
    3: DRAW 3 AT X,Y + NN*((N /
    57) * 10 - 10):N = 1: HCOLOR=
    0: GOTO 4930
4910 NN = NN*((N / 57) * 10 - 10)

4920 DRAW 3 AT X,Y + NN: IF N =
    57 THEN HPLLOT X - 4,Y + NN TO
    X + 4,Y + NN
4930 X = X + 8
4940 LL = L + LL
4950 IF LL > = 255 THEN LL = 0:
    HPLLOT X,Y TO X,Y + 16:X = X
    + 8:MS = MS + 1
4960 IF MS > 4 THEN MS = 1:X = 2
    7:Y = Y + 40
4970 RETURN
```

Save routine.

```
5000 TEXT : HOME
5010 VTAB 3: HTAB 7: PRINT "WOUL
    D YOU LIKE TO SAVE THIS SONG
    ": HTAB 7: PRINT "(Y/N) ?";
5020 GET AN$
5030 IF AN$ = "N" THEN GOTO 100

5040 IF AN$ < > "Y" THEN GOTO
    5020
5050 VTAB 8: HTAB 7: PRINT "ON T
    APE OR DISK (T/D) ?";
5060 GET AN$
5070 IF AN$ = "T" GOTO 5260
5080 IF AN$ < > "D" THEN GOTO
    5060
5090 ONERR GOTO 7000
5100 DK = 0
5110 VTAB 15: HTAB 13: PRINT CHR$
    (7);"NAME ->"
5120 VTAB 15: HTAB 20: CALL - 9
    58: INPUT " ";S$
```

```
5130 IF LEN (S$) > 17 THEN VTAB
    20: HTAB 15: PRINT CHR$ (7)
    ;"*** NAME TOO LONG. MUST BE
    "; HTAB 19: PRINT "17 LETTE
    RS OR LESS.": FOR I = 1 TO 2
    000: NEXT I: GOTO 5120
5140 IF S$ = "" GOTO 5250
5150 PRINT D$;"OPEN";S$;".MUSIC"

5160 PRINT D$;"DELETE";S$;".MUSI
    C"
5170 PRINT D$;"OPEN";S$;".MUSIC"

5180 PRINT D$;"WRITE";S$;".MUSIC
    "

5190 FOR I = 1 TO 5
5200 PRINT A$(I)
5210 PRINT B$(I)
5220 PRINT AA$(I)
5230 NEXT I
5240 PRINT D$;"CLOSE";S$;".MUSIC
    "

5250 POKE 216,0: GOTO 100
5260 HOME : HTAB 13: PRINT "* ST
    ORE TO TAPE *"
5270 VTAB 5: PRINT "1. MAKE SURE
    YOUR TAPE IS READY TO": PRINT
    " RECORD."
5280 PRINT : PRINT "2. START REC
    ORDING AND THEN PUSH 'SPACE"
    ": PRINT " BAR". IGNORE TH
    E BEEPS. APPLE WILL": PRINT
    " RETURN YOU TO THE PROGRA
    M WHEN IT IS": PRINT " FIN
    ISHED."
5290 PRINT : PRINT "3. IF YOU GE
    T AN ERROR MESSAGE, SET UP":
    PRINT " TAPE AGAIN, CHECK
    VOLUME, ETC. AND": PRINT "
    TYPE 'GOTO 5260'. PRESS R
    ETURN."
5300 FOR I = 1 TO 2000: NEXT I: VTAB
    8: HTAB 24: INVERSE : PRINT
    "THEN": NORMAL
5310 KEY = PEEK ( - 16384): IF K
    EY < > 160 GOTO 5310
5320 POKE - 16368,0
```



```

5330 VTAB 22: HTAB 15: PRINT "*"
      "; FLASH : PRINT "WORKING";
      : NORMAL : PRINT "*"
5340 STORE A#
5350 STORE B#
5360 STORE AA#
5370 GOTO 100

```

Recall routine.

```

6000 TEXT : HOME
6010 VTAB 8: HTAB 13: PRINT "FROM TAPE OR DISK (T/D) ?";
6020 GET AN#
6030 IF AN# = "T" GOTO 6210
6040 IF AN# < > "D" GOTO 6020
6050 ONERR GOTO 7000
6060 DK = 1
6070 VTAB 15: HTAB 13: PRINT CHR#(7)"NAME ->"
6080 VTAB 15: HTAB 20: CALL - 9
      58: INPUT " ";S#
6090 IF LEN (S#) > 17 THEN VTAB 20: HTAB 15: PRINT CHR#(7)
      "*** NAME TOO LONG. MUST BE"
      "; HTAB 19: PRINT "17 LETTERS OR LESS.": FOR I = 1 TO 20
      00: NEXT I: GOTO 6080
6100 IF S# = "" THEN POKE 216,0
      : GOTO 6200
6110 PRINT D#;"OPEN";S#;".MUSIC"
6120 PRINT D#;"READ";S#;".MUSIC"
6130 FOR I = 1 TO 5
6140 INPUT A$(I)
6150 INPUT B$(I)
6160 INPUT AA$(I)
6170 NEXT I
6180 PRINT D#;"CLOSE";S#;".MUSIC"
6190 HOME : POKE 216,0: GOSUB 4000
6200 TEXT : HOME : GOTO 100
6210 HOME : HTAB 11: PRINT "*" RECALL FROM TAPE #"
6220 VTAB 5: PRINT "1. MAKE SURE

```

```

YOUR TAPE IS SET UP AND": PRINT
" READY AT YOUR MUSIC PROGRAM."
6230 PRINT : PRINT "2. START RECORDING ON PLAY THEN PUSH": PRINT
      " 'SPACE BAR'."
6240 PRINT : PRINT "3. IF ALL GOES WELL, YOU'LL HEAR SOME": PRINT
      " BEEPS. -- JUST IGNORE THEM.": PRINT " APPLE WILL RETURN TO THE PROGRAM."
6250 PRINT : PRINT "4. IF YOU GET AN ERROR MESSAGE, RE-RUN": PRINT
      " PROGRAM AND TRY AGAIN."
6260 FOR I = 1 TO 2000: NEXT I: VTAB 8: HTAB 27: INVERSE : PRINT
      "THEN": NORMAL
6270 KEY = PEEK (- 16384): IF KEY < > 160 GOTO 6270
6280 POKE - 16384,0
6290 VTAB 22: HTAB 15: PRINT "*"
      "; FLASH : PRINT "WORKING";
      : NORMAL : PRINT "*"
6300 RECALL A#
6310 RECALL B#
6320 RECALL AA#
6330 HOME : FOR I = 1 TO 2000: NEXT I: GOSUB 4000
6340 FOR I = 1 TO 2000: NEXT I: GOTO 100

```

Error-handling routine.

```

7000 ER = PEEK (222)
7010 IF ER = 9 THEN VTAB 20: HTAB 11: PRINT CHR#(7)"DISK IS FULL. TRY AGAIN.": GOTO 7090
7020 IF ER = 4 OR ER = 8 GOTO 7080
7030 IF ER = 5 OR ER = 11 THEN VTAB 20: HTAB 11: PRINT CHR#(7)
      "PROGRAM NOT FOUND. TRY AGAIN.": GOTO 7050
7040 VTAB 20: HTAB 11: PRINT "ERROR #";ER: END

```

```

7050 FOR I = 1 TO 2000: NEXT I
7060 PRINT D#;"DELETE";S#;".MUSIC"
7070 GOTO 6080
7080 VTAB 20: HTAB 11: PRINT CHR#(7)"DOS I/O ERROR. CHECK DRIVE OR"; HTAB 11: PRINT "DISK, AND TRY AGAIN."
7090 FOR I = 1 TO 4000: NEXT I
7100 IF DK = 1 GOTO 6080
7110 GOTO 5120

```

Data statements.

```

8000 DATA 3,0,8,0,36,0,46,0
8010 DATA 45,53,54,62,62,62,55,5,5,55,55,62,54,54,53,53,45,45,37,37,37,37,60,60,63,63,62,6,0
8020 DATA 73,49,182,18,36,60,63,39,36,0
8030 DATA 137,63,63,46,45,37,4,3,2,63,63,46,45,53,63,63,48,6,0
9000 DATA 173,48,192,136,208,5,2,06,1,3,240,9,202,208,245,174,0,3,76,2,3,96
10000 DATA "144064096064144064114064096128153064096064153064128064096255"
10010 DATA "068032064064068032064064068032064032","064032064064068032064032064032057032064032"
10020 DATA "076032081032076032064064096032076032064032","064032064064032064073032076032081032076032064032"
10030 DATA "064064068032064064068032064064","064032057032064032076032072032086064076032"
10040 DATA "064032076032064032057064064064096032","076064086032096096086064"
10050 DATA "096032086032076032128064076032086032096032","076

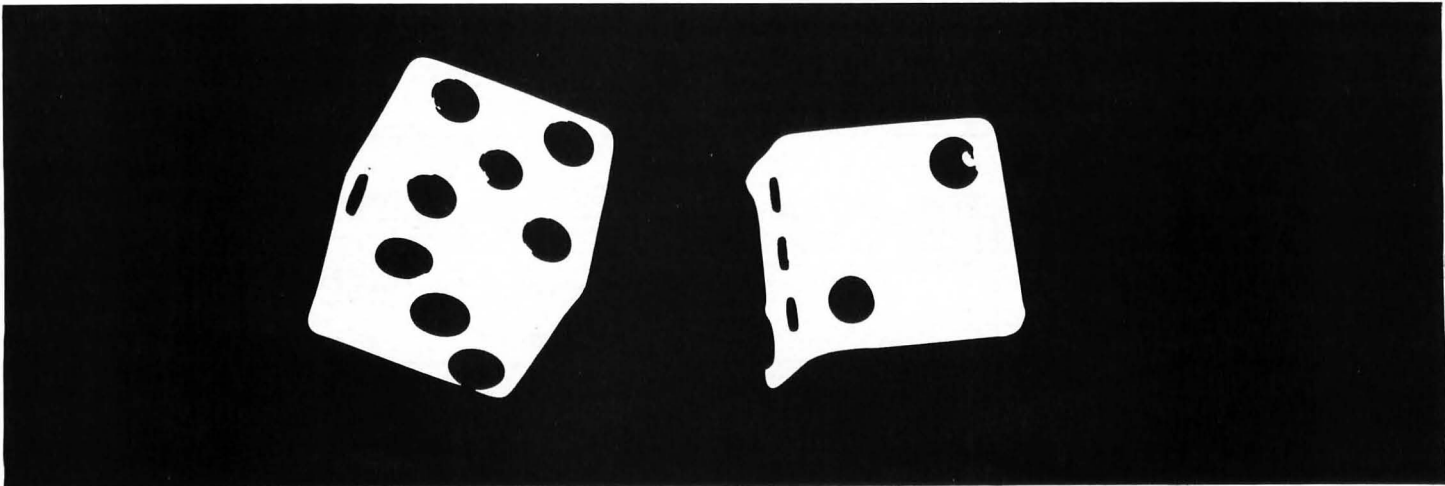
```

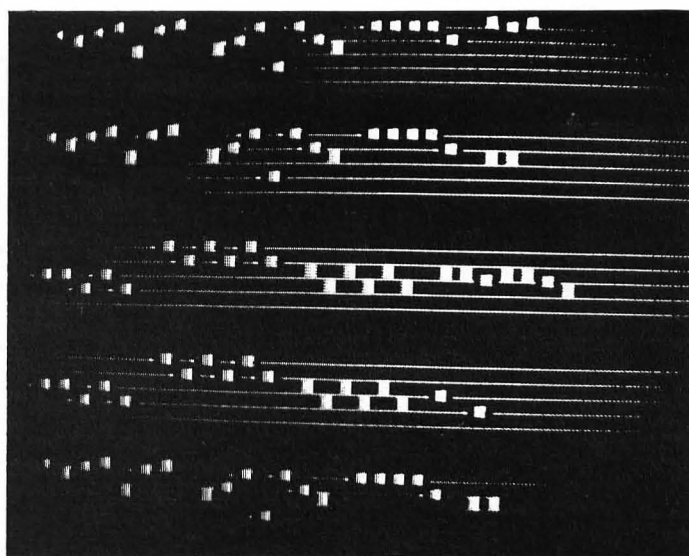
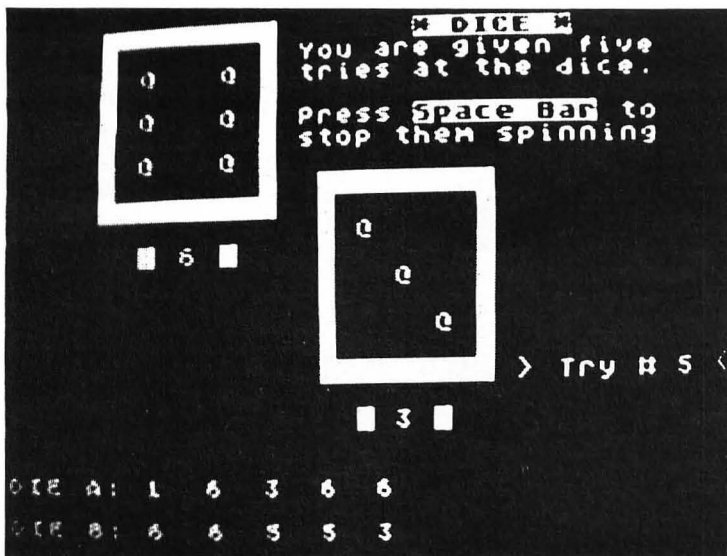
continued on next page





continued from previous page		
160128032096032076032"	064"	8032096064", "096064114064076 032086064096032"
10060 DATA "07606408603209606406 4032076032096032", "076064064 032076064081032076064"	10150 DATA "09606400106412800811 4008102008096040001064", "096 064128064096128"	10230 DATA "09606408603209606410 2032096064", "096064114032096 064114032096032114032"
10070 DATA "08606406403208606406 4032086064", "001032086032057 0320720320640320570320640320 72032"	10160 DATA "10206410203212806410 2032128064", "102032171032128 032114064128032114064"	10240 DATA "00103208603206803208 6032076032068032076032086032 , "0960321140321020320960640 76032086064"
10080 DATA "07203207206407603207 2032072032076032086032", "072 0320570640640320720320760320 72032064032"	10170 DATA "10203210803210203212 8064128032108032102032", "001 0321280320760320860640760320 86064"	10250 DATA "11403211406412103211 4064108064", "096032096064076 032086032096032086064"
10090 DATA "07206407203207206407 2032086064", "064096068032064 032057032064064"	10180 DATA "00103212803207603210 2032086032076032086032102032 , "1020321020641140321280321 36032128032102032"	10260 DATA "09603209606410203209 6032096032102032114032", "121 032114064102032076064096064"
10100 DATA "06416007603207203206 8032", "064064102032096032086 032076032072032064032"	10190 DATA "12803210203208603206 4032064064086064", "102032171 032128032102160"	10270 DATA "09612808603208606409 6032", "086128086064096064"
10110 DATA "064255", "064255"	10200 DATA "12803210206409603208 6064064032086032", "076032086 064076032086032076032086064"	10280 DATA "10206400106412806400 1064", "102160086032076032086 032"
10120 DATA "064096068032064128", "064064064032064032064064064 064"	10210 DATA "12803210203207603208 6096086064", "086032086064096 032102032108032102032086032"	10290 DATA "10206406403206403206 4064064064", "102064001064128 064001064"
10130 DATA "096128096064001064", "076064086064096128"	10220 DATA "09603206806409606406	10300 DATA "102192064064", "10206 4102064086064064064"





Atari Version

```

5 CLR :OPEN #2,4,0,"K:"
10 GOTO 20
12 SOUND 0,N,10,10
14 FOR ZZ=1 TO L/2:NEXT ZZ:SOUND 0,0,0
,0:FOR ZZ=1 TO 5:NEXT ZZ:RETURN
20 GRAPHICS 0
30 GOSUB 1000
40 GOSUB 2000
50 ? :? "Would you like instructions (
Y/N) ? ";
60 GET #2,AN:AN$=CHR$(AN)
70 IF AN$="N" THEN 100
80 IF AN$(">Y") THEN 60
90 GOSUB 2500
100 GRAPHICS 0
110 POSITION 6,3:?"### Would you like
to:"
120 ? :? ,"1) Play a song in memory?"
130 ? ,"2) Create a new song?"
140 ? ,"3) Play a song from media"
142 ? ,"4) Quit?"
150 POSITION 14,13:?"Number-> ";
160 GET #2,AN
170 AN=AN-48:IF AN<1 OR AN>4 THEN 160
175 IF AN=1 THEN GOSUB 4000:GOTO 100
180 IF AN=2 THEN AB=1:GOTO 3000
190 IF AN=3 THEN GOTO 6000

```

```

200 GRAPHICS 0:PRINT "Okay ... Bye":EN
D
1000 REM
1090 DIM C$(2000),C(61),Z$(120),NN(21)
,AN$(13),X$(60)
1092 DIM A(10),AA(10),B(10)
1094 READ Z$:FOR T=1 TO LEN(Z$) STEP 2
:A=ASC(Z$(T,T))-48:IF A>11 THEN A=A-7
1096 A=A*16+ASC(Z$(T+1,T+1))-48:IF Z$(
T+1,T+1)">" THEN A=A-7
1098 X$(LEN(X$)+1)=CHR$(A):NEXT T
1100 RESTORE 1000:FOR I=0 TO 60
1110 READ Z$:C(I)=LEN(C$)+1:C$(LEN(C$)
+1)=Z$
1120 NEXT I:C(61)=LEN(C$)+1
1140 NN(0)=-4:NN(1)=-2:NN(2)=0:NN(3)=0
:NN(4)=2:NN(5)=4:NN(6)=6:NN(7)=6
1142 NN(8)=8:NN(9)=0:NN(10)=10:NN(11)=
10:NN(12)=12:NN(13)=12:NN(14)=0
1150 NN(15)=14:NN(16)=14:NN(17)=0:NN(1
8)=16:NN(19)=0:NN(20)=18:NN(21)=18
1160 RETURN
2000 GRAPHICS 2:POKE 752,1:SETCOLOR 2,
0,0
2020 POSITION 6,3:PRINT #6;"MELODY":PR
INT #6:PRINT #6;" dice"
2050 REM FOR I=1 TO 500:NEXT I
2060 PRINT " By G. Cage & R. Boucha
rd";
2070 A=PEEK(560)+PEEK(561)*256:MEM=PEE

```

```

K(A+4)+PEEK(A+5)*256+200
2080 FOR II=1 TO C(1)-1 STEP 6:N=VAL(C
$(II,II+2)):L=VAL(C$(II+3,II+5)):SOUND
0,N,10,10
2090 A=USR(ADR(X$),MEM,L/8):NEXT II:SO
UND 0,0,0,0
2100 FOR I=1 TO 500:NEXT I
2110 GRAPHICS 0:RETURN
2500 GOSUB 2900
2510 FOR I=1 TO 500:NEXT I
2530 ? " Imagine if you will, that th
ere are 60 cards stored somewhere in A
tari,"
2532 ? "each containing portions of mu
sic stolen from Scott Joplin."
2540 ? :? " Now-what if you were to j
umble up those cards in a random ord
er and play";
2542 ? "them on an instrument. Would
they sound as good as the originals
?"
2550 ? :? "Probably not.. But them aga
in, who knows? At any rate, you wo
uld have"
2552 ? "yourself an original compositi
on inspired by one of the greats
in music."
2560 ? :? "And if you didn't like it,
you could mix up the cards once more
and have"

```

continued on next page



continued from previous page

```
2562 ? "something completely different
."
2570 REM
2580 POSITION 6,23: ? "< Press RETURN t
o continue >";
2590 GOSUB 2800
2600 GOSUB 2900
2610 PRINT " Can't play an instrument
or read music? Well, that's where
Atari comes";
2612 ? "in. You will be shown a pair
of dice.Press the space bar to stop th
em from"
2620 ? "spinning, and the numbers on y
our throw will be displayed. This
is done for a total of 5 tries."
2630 ? :? " Each number of the die co
rresponds to a particular card in mem
ory depen-"
2632 ? "ding on the time it was thrown
(1st try, 2nd try, etc.). Atari"
2640 ? "assembles these together and y
ou are asked if you want to hear the
final product, or try again."
2650 ? :? " If you wish to have it pl
ayed, the music will be accompanied b
y its"
2652 ? "musical notation (of sorts)."
```

```
3020 FOR V=0 TO 8:POSITION 15,V: ? CHR$(
160);:POSITION 26,7+V: ? CHR$(160);:NE
XT V
3030 FOR H=15 TO 7 STEP -1:POSITION H,
8: ? CHR$(160);:POSITION 11+H,15: ? CHR$(
160);:NEXT H
3040 FOR V=8 TO 0 STEP -1:POSITION 7,V
: ? CHR$(160);:POSITION 18,7+V: ? CHR$(1
60);:NEXT V
3050 V=10:H=9:POSITION H,V: ? CHR$(160)
;:POSITION H+4,V: ? CHR$(160);:POSITION
H+11,V+7: ? CHR$(160);
3060 POSITION H+15,V+7: ? CHR$(160);
3070 GOSUB 3360
3080 GOSUB 3430
3090 T=0:DA=INT(RND(0)*6)+1:DB=INT(RND
(0)*6)+1:MO=0
3100 GOSUB 3470
3110 GOSUB 3500
3120 H=8:V=3
3130 BH=0:BV=0:ON DA GOSUB 3190,3210,3
230,3250,3270,3290
3140 BH=11:BV=7:ON DB GOSUB 3190,3210,
3230,3250,3270,3290
3150 REM
3160 GOSUB 3600
3170 IF T>=5 THEN FOR I=1 TO 500:NEXT
I:GOTO 3800
3180 GOTO 3100
3190 POSITION H+3+BH,V+1+BV: ? "a";
3200 RETURN
3210 POSITION H+BH+1,V+BV-1: ? "a";:POS
ITION H+5+BH,V+3+BV: ? "a";
3220 RETURN
3230 FOR I=0 TO 4 STEP 2:POSITION H+I+
BH+1,V+I+BV-1: ? "a";:NEXT I
3240 RETURN
3250 GOSUB 3310
3260 RETURN
3270 GOSUB 3310:POSITION H+3+BH,V+1+BV
: ? "a";
3280 RETURN
3290 GOSUB 3310:POSITION H+BH+1,V+BV+1
: ? "a";:POSITION H+5+BH,V+BV+1: ? "a";
3300 RETURN
3310 POSITION H+BH+1,V+BV-1: ? "a";:POS
ITION H+5+BH,V+BV-1: ? "a";:POSITION H+
```

```
5+BH,V+3+BV: ? "a";
3320 POSITION H+BH+1,V+3+BV: ? "a";:RET
URN
3330 FOR VC=2 TO 6 STEP 2:POSITION 9,V
C: ? " ";:POSITION 9+BH,VC+BV: ? "
";
3340 NEXT VC
3350 RETURN
3360 REM
3370 POSITION 23,0: ? "* DICE #"
3380 POSITION 17,1: ? "You are given fi
ve";:POSITION 17,2: ? "tries at the dic
e.";
3390 POSITION 17,4: ? "Press Space Bar
to";:POSITION 17,5: ? "stop them spinni
ng";
3400 POSITION 28,15: ? "> Try # <"
3410 REM
3420 RETURN
3430 REM
3440 POSITION 2,20: ? "DIE A:"
3450 POSITION 2,22: ? "DIE B:"
3460 RETURN
3470 T=T+1
3480 POSITION 36,15: ? T;
3490 RETURN
3500 V=10:H=11:POSITION H,V:PRINT DA:P
OSITION H+11,V+7:PRINT DB
3510 GOSUB 3530
3520 RETURN
3530 KEY=PEEK(764):IF KEY=33 THEN 3560
3540 DA=INT(RND(1)*6)+1:DB=INT(RND(1)*
6)+1
3550 POP :GOTO 3500
3560 GOSUB 3330
3570 POKE 764,255:POSITION MO+9,20: ? D
A;:POSITION MO+9,22: ? DB;
3580 MO=MO+3
3590 RETURN
3600 C=DA*T+(6-DA)*(T-1)
3610 A(AB)=C
3620 C=(DB*T+(6-DB)*(T-1))+30
3630 B(AB)=C
3640 AA(AB)=A(AB)
3650 AB=AB+1
3660 RETURN
3800 GRAPHICS 0:POSITION 6,5: ? "--> Do
```



```

you wish to play this":POSITION 10,6:
? "song or try again?"
3810 POSITION 18,8:? "Type Play or":PO
SITION 23,9:? "Again."
3820 POSITION 25,10:? "--> ";
3830 GET #2,AN:AN$=CHR$(AN)
3840 IF AN$="A" THEN AB=1:GOTO 3000
3850 IF AN$("<"P" THEN 3830
3860 GOSUB 4000
3870 FOR I=1 TO 1000:NEXT I
3880 GRAPHICS 0:? "Play the same song
again ? ";
3890 GET #2,AN
3900 AN$=CHR$(AN):IF AN$="N" THEN 5000
3910 IF AN$("<"Y" THEN 3890
3920 GOTO 3860
4000 GRAPHICS 0:GOSUB 4500:SW=0
4010 FOR I=1 TO 4
4020 GOSUB 4300
4030 NEXT I
4040 FOR I=1 TO 3
4050 GOSUB 4300
4060 NEXT I
4070 I=5
4080 GOSUB 4300
4090 IF SW=1 THEN 4120
4100 SW=1:FOR I=1 TO 5:A(I)=B(I):NEXT
I
4110 GOTO 4010
4120 FOR I=1 TO 5
4130 A(I)=AA(I)
4140 NEXT I
4150 FOR I=1 TO 3
4160 GOSUB 4300
4170 NEXT I:I=5
4180 GOSUB 4300
4190 RETURN
4300 FOR II=C(A(I)) TO C(A(I)+1)-1 STE
P 6
4310 N=VAL(C$(II,II+2))
4320 L=VAL(C$(II+3,II+5))
4330 IF N>1 THEN SOUND 0,N,10,10
4340 GOSUB 4900
4350 NEXT II
4360 RETURN
4500 GRAPHICS 24:COLOR 1:A=PEEK(560)+P
EEK(561)*256:MEM=PEEK(A+4)+PEEK(A+5)*2

```

```

56
4520 Y=10:X=10
4530 FOR Y=10 TO 170 STEP 39:FOR I=0 T
O 20 STEP 5
4540 PLOT X,Y+I:DRAWTO X+289,Y+I
4550 NEXT I:NEXT Y
4630 X=16:Y=6
4640 FOR I=0 TO 170 STEP 39
4650 PLOT X,Y+I:DRAWTO X,Y+I+28
4660 REM *****
4670 NEXT I
4690 LL=0:MS=1:X=27
4700 RETURN
4900 IF N=1 THEN A=USR(ADR(X$),MEM,L/1
0-2):GOTO 4930
4902 NN=INT(NN(N/5.7-10)*1.25+0.5):A=U
SR(ADR(X$),MEM+(Y+NN+2)*40+INT(X/8),L/
10-2)
4920 IF N=57 THEN PLOT X-4,Y+NN+4:DRAW
TO X+5,Y+NN+4
4930 SOUND 0,0,0,0:X=X+8:LL=L+LL
4950 IF LL>=255 THEN LL=0:PLOT X,Y+4:D
RAWTO X,Y+24:X=X+8:MS=MS+1
4960 IF MS>4 THEN MS=1:X=27:Y=Y+39
4970 RETURN
5000 GRAPHICS 0
5010 POSITION 8,2:? "Would you like to
save this song (Y/N) ?";
5020 GET #2,AN:AN$=CHR$(AN)
5030 IF AN$="N" THEN 100
5040 IF AN$("<"Y" THEN 5020
5050 POSITION 8,7:? "On tape or disk (
T/D) ?";
5060 GET #2,AN:AN$=CHR$(AN)
5070 IF AN$="T" THEN 5260
5080 IF AN$("<"D" THEN 5060
5090 REM
5100 DK=0
5110 POSITION 14,14:? CHR$(253);"Name
->";
5120 INPUT AN$:Z$(1,2)="D":Z$(3)=AN$
5130 REM
5140 REM
5150 OPEN #1,8,0,Z$
5190 FOR I=1 TO 5
5200 PRINT #1:A(I)
5210 PRINT #1:B(I)

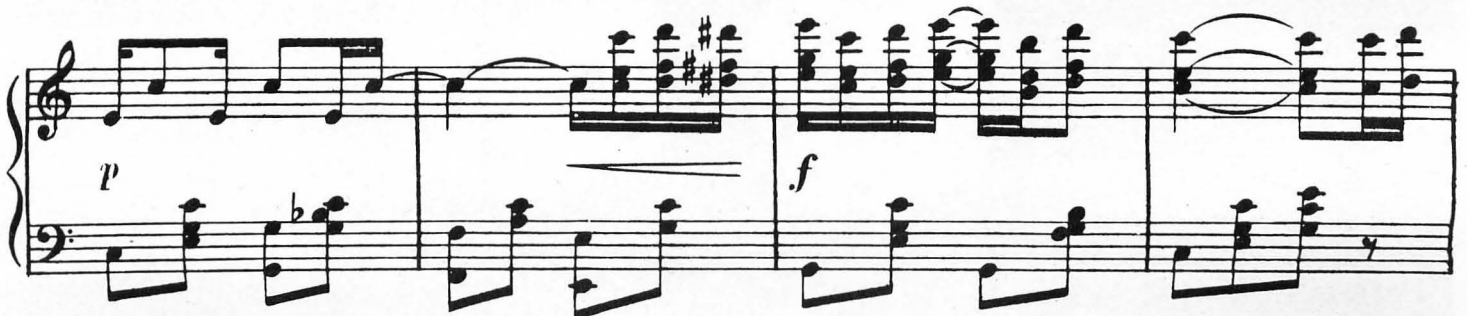
```

```

5220 PRINT #1;AA(I)
5230 NEXT I
5240 CLOSE #1
5250 GOTO 100
5260 GRAPHICS 0:? " * Store to ta
pe #"
5270 POSITION 2,4:? "1. Make sure your
tape is ready to record."
5280 ? :? "2. Start recording and then
push 'Return'. Atari will retur
n you to the program when finished"
5290 ? :? "3. If you get an error mess
age, set uptape again and type 'GOTO 5
260'"
5300 OPEN #1,8,0,"C:"
5310 GOTO 5190
6000 GRAPHICS 0
6010 POSITION 14,7:? "From tape or dis
k (T/D) ?";
6020 GET #2,AN:AN$=CHR$(AN)
6030 IF AN$="T" THEN 6210
6040 IF AN$("<"D" THEN 6020
6050 REM
6060 DK=1
6070 POSITION 14,14:? CHR$(253);"Name
->";
6080 INPUT AN$:Z$(1,2)="D":Z$(3)=AN$
6090 REM
6100 REM
6110 OPEN #1,4,0,Z$
6130 FOR I=1 TO 5
6140 INPUT #1,Z:A(I)=Z
6150 INPUT #1,Z:B(I)=Z
6160 INPUT #1,Z:AA(I)=Z
6170 NEXT I
6180 CLOSE #1
6190 GRAPHICS 0:GOSUB 4000
6200 GRAPHICS 0:GOTO 100
6210 GRAPHICS 0:? " * Recall from
tape #"
6220 POSITION 2,4:? "1. Make sure your
tape is set up and ready at your mus
ic program."
6230 ? :? "2. Start recorder on play a
nd then push 'Return'."
6240 ? :? "3. If all goes well, Atari
will returnto the program."

```

continued on next page





continued from previous page

6250 ? :? "4. If you get an error message, re-run program and try again."

6260 OPEN #1,4,0,"C:"

6270 GOTO 6130

8000 DATA 686885CC6885CB1BA000A97E91CB
986928A8A97E91CR986928A8A97E91CR986928
A8A97E91CB6868ABA514C514F0FC88D0F760

10000 DATA 144064096064144064114064096
128153064096064153064128064096255

10010 DATA 068032064064068032064064068
032064032,0640320640640680320640320640
32057032064032

10020 DATA 076032081032076032064064096
032076032064032,0640320640640730320760
32081032076032064032

10030 DATA 064064068032064064068032064
064,0640320570320640320760320720320860
64076032

10040 DATA 064032076032064032057064064
064096032,076064086032096096086064

10050 DATA 096032086032076032128064076
032086032096032,0761601230320960320760
32

10060 DATA 076064086032096064064032076
032096032,0760640640320760640810320760
64

10070 DATA 086064064032086064064032086
064,0010320860320570320720320640320570

32064032072032

10080 DATA 072032072064076032072032072
032076032086032,0720320570640640320720
32076032072032064032

10090 DATA 072064072032072064072032086
064,064096068032064032057032064064

10100 DATA 064160076032072032068032,06
40641020320960320860320760320720320640
32

10110 DATA 064255,064255

10120 DATA 064096068032064128,06406406
4032064032064064064064

10130 DATA 096128096064001064,07606408
6064096128

10140 DATA 096064001064096064001064,09
6064001064096064001064

10150 DATA 096064001064128032114032102
032096040001064,096064128064096128

10160 DATA 102064102032128064102032128
064,1020321710321380321140641280321140
64

10170 DATA 102032108032102032128064128
032108032102032,0010321280320760320860
65076032086064

10180 DATA 001032128032076032102032086
032076032086032102032,1020321020641140
32128032136032128032102032

10190 DATA 128032102032086032064032064
064086064,103032171032128032102160

10200 DATA 128032102064096032086064064
032086032,0760320860640760320860320760
32086064

10210 DATA 128032102032076032086096086
064,0860320860640960321020321080321020
32086032

10220 DATA 096032068064096064068032096
064,096064114064076032086064096032

10230 DATA 096064086032096064102032096
064,0960641140320960641140320960321140
32

10240 DATA 001032086032068032086032076
032068032076032086032,0960321140321020
32096064076032086064

10250 DATA 114032114064121032114064108
064,0960320960640760320860320960320860
64

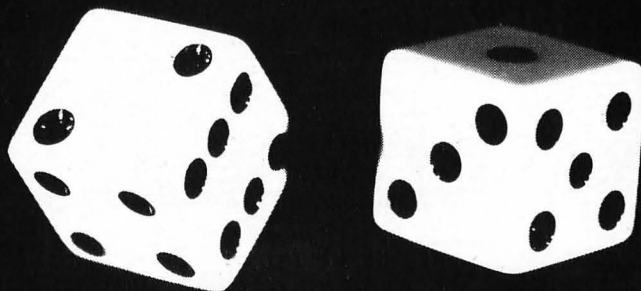
10260 DATA 096032096064102032096032096
032102032114032,1210321140641020320760
64096064

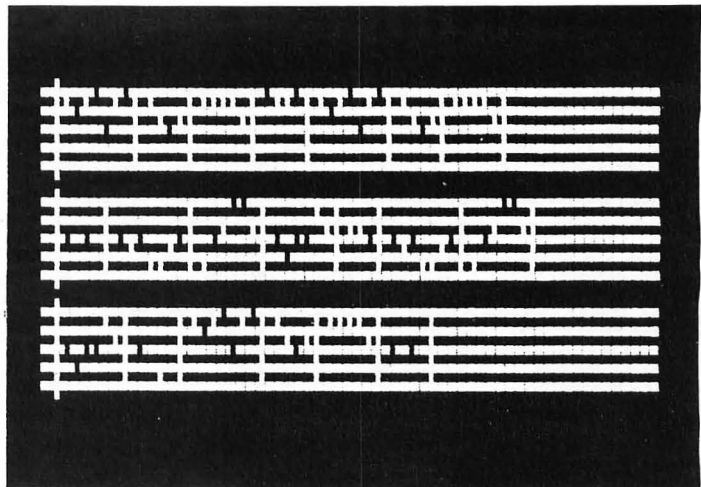
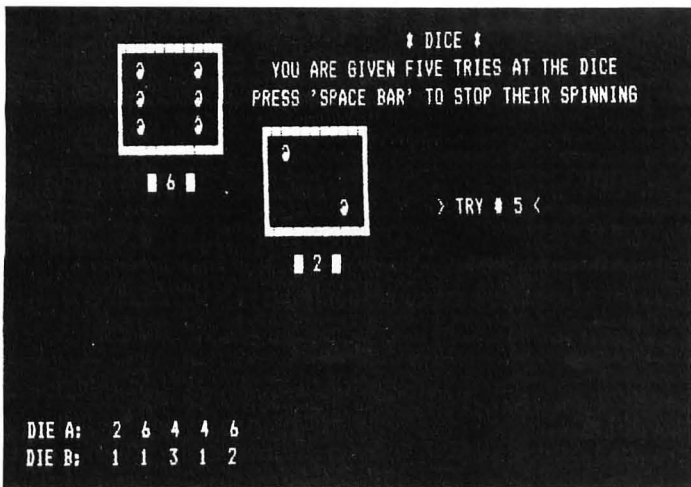
10270 DATA 096128086032086064096032,08
6128086064096064

10280 DATA 102064001064128064001064,10
2160086032076032086032

10290 DATA 102064064032064032064064064
064,102064001064128064001064

10300 DATA 102192064064,10206410206408
6064064064





TRS-80™ Version

```

10 CLEAR 100:DEFINTB-Z:GOTO20
12 IFN(<>)THENN=NN(N/5.7-10)/2:IFINT(NN/2)N=2NTHENSET(X,Y+NN)EL
SERESET(X,Y+NN)
14 X=X+2:LL=L+LL:IFLL>255THENLL=0:FORZ=0TO6STEP2:SET(X,Y+Z):NEXT
Z:X=X+3:MS=MS+1:IFMS>7THENMS=1:X=5:Y=Y+12
16 RETURN
20 CLS:N=0:X=0:Y=0:NN=0:LL=0:L=0:Z=0:MS=0:II=0
30 GOSUB1000
40 GOSUB2000
50 PRINT
WOULD YOU LIKE INSTRUCTIONS (Y/N) ? ";
60 AN%=INKEY%
70 IF AN%="N" THEN 100
80 IF AN%<>"Y" THEN 60
90 GOSUB2500
100 CLS
110 PRINT@330,"*** WOULD YOU LIKE TO: ";PRINT,"1) PLAY A SONG IN
MEMORY",,"2) CREATE A NEW SONG",,"3) PLAY A SONG FROM MEDIA",
,"4) QUIT"
150 PRINT @670,"NUMBER-> ";
160 AN%=INKEY%
170 AN=VAL(AN%):IF AN<1 OR AN>4 THEN 160
180 IFAN=1THENGOSUB4000:GOTO100
190 IFAN=2THENAB=1:GOTO3000
200 IFAN=3THENCLS:GOTO6000ELSEPRINT"OKAY ... BYE":END
1000 READMU#:A=VARPTR(MU#)+1:A1=PEEK(A)+PEEK(A+1)*256
1002 M1=(A1+1)+65536*(A1>32766):M2=(A1+3)+65536*(A1>32764)
1005 IFPEEK(16396)=201THENPOKE16526,PEEK(A):POKE16527,PEEK(A+1):
ELSEDEFUSR=A1+65536*(A1>32767):CMD"t":POKE14308,0
1010 FORA1=A1TOA1+30:READVA
1020 POKEA1+65536*(A1>32767),VA
1030 NEXTA1
1090 DIM C$(60),A(5),AA(5),B(5),NN(21)

```

```

1100 FORI=0TO60
1110 READC$(I)
1120 NEXTI
1130 REM
1140 NN(0)=-4:NN(1)=-2:NN(2)=0:NN(3)=0:NN(4)=2:NN(5)=4:NN(6)=6:N
N(7)=6:NN(8)=8:NN(10)=10:NN(11)=10:NN(12)=12:NN(13)=12
1150 NN(15)=14:NN(16)=14:NN(18)=16:NN(20)=18:NN(21)=18
1160 RETURN
2000 CLS
2010 PRINTCHR$(23);
2020 PRINT@338,"M E L O D Y"
2030 PRINT@470,"D I C E"
2050 REM
2060 PRINT@964,"BY G. CAGE & RICH BOUCHARD";
2080 FORII=1TOLEN(C$(0))STEP6:N=VAL(MID$(C$(0),II,3)):L=VAL(MID$(
C$(0),II+3,3))
2085 POKE M1,L/4-2:POKE M2,N/3:U=USR(0):FORZ=1TOL/10:NEXTZ
2090 NEXTII
2100 FOR I=1 TO 1000:NEXT I
2110 CLS:RETURN
2500 CLS:PRINTTAB(24);CHR$(143);" INSTRUCTIONS ";CHR$(143)
2530 PRINT:PRINT" IMAGINE IF YOU WILL, THAT THERE ARE 60 CARDS
STORED SOMEWHERE IN THE S-80, EACH CONTAINING PORTIONS OF MUSIC
STOLEN FROM SCOTTJOPLIN."
2540 PRINT:PRINT" NOW WHAT IF YOU WERE TO JUMBLE UP THOSE CARDS
IN A RANDOM ORDER AND PLAY THEM ON AN INSTRUMENT. WOULD TH
EY SOUND AS GOOD AS THE ORIGINALS?"
2550 PRINT:PRINT" PROBABLY NOT. BUT THEN AGAIN, WHO KNOWS? AT
ANY RATE, YOU WOULD HAVE YOURSELF AN ORIGINAL COMPOSITION INSP
IRED BY ONE OF THE GREATS OF MUSIC. AND IF YOU DIDN'T LIKE IT
, YOU COULD MIX"
2560 PRINT"UP THE CARDS ONCE MORE AND HAVE SOMETHING COMPLETELY
DIFFERENT."
2590 GOSUB2800
2600 CLS
2610 PRINT:PRINT" CAN'T PLAY AN INSTRUMENT ORREAD MUSIC? WELL

```

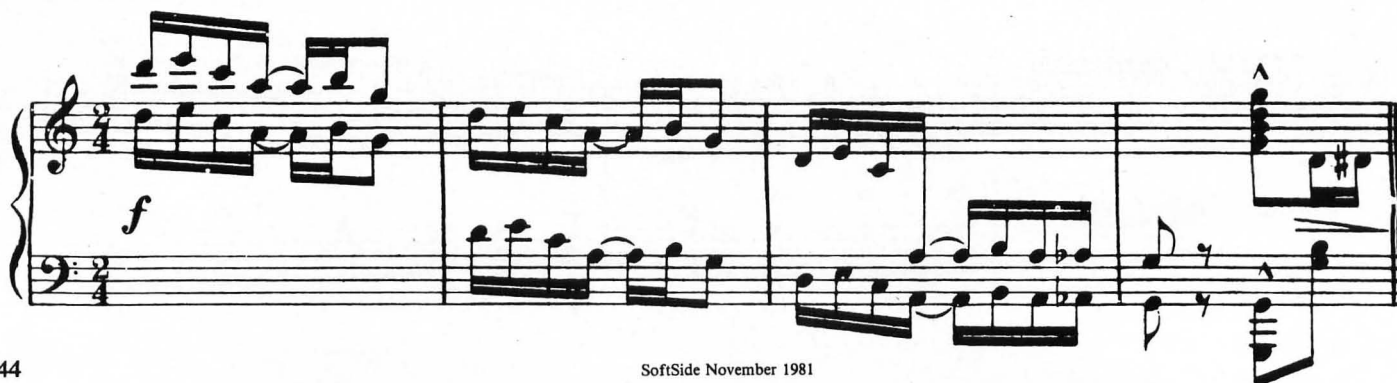
continued on next page



continued from previous page

```
, THAT'S WHERE S-80 COMES IN. YOU WILL BE SHOWN A PAIR OF DICE
. PRESS THE SPACE BAR TO STOP THEM FROM SPINNING, AND THE N
UMBERS ON YOUR"
2620 PRINT"THROW WILL BE DISPLAYED. THIS IS DONE FOR A TOTAL OF
5 TRIES."
2630 PRINT:PRINT" EACH NUMBER ON THE DIE CORRESPONDS TO A PARTI
CULAR CARD IN MEMORY DEPENDING ON THE TIME IT WAS THROWN (1ST
TRY, 2ND TRY,"
2640 PRINT "ETC.). S-80 ASSEMBLES THESE TOGETHER AND YOU ARE AS
KED IF YOU WANT TO HEAR THE FINAL PRODUCTION, OR TRY AGAIN."
2650 PRINT:PRINT" IF YOU WISH TO HAVE IT PLAYED THE MUSIC WILL
BE ACCOMPANIED BY ITS MUSICAL NOTATION (OF SORTS)."
2660 GOSUB2800
2670 CLS
2680 PRINT"AFTER LISTENING TO YOUR COMPOSITION YOU WILL BE ASKED
IF YOU WISH TO SAVE IT ON DISK, TRY AGAIN, OR QUIT ALTOGETHE
R."
2690 PRINT:PRINT" THAT'S ALL THERE IS TO IT."
2700 GOSUB 2800
2710 RETURN
2800 PRINT@980,"PRESS <ENTER> TO CONTINUE";
2810 AN%=INKEY%:IF AN%<>CHR$(13) THEN 2800
2820 RETURN
3000 CLS
3010 FORH=0T020:SET (H+21,1):SET (20,H/2+1):SET (41,H/2+2):SET (H+20
,12)
3020 SET (H+51,10):SET (50,H/2+10):SET (71,H/2+11):SET (H+50,21):NEX
TH
3050 PRINT@333,CHR$(143);" ";CHR$(143);:PRINT@540,CHR$(143);"
";CHR$(143);
3070 GOSUB 3360
3080 GOSUB 3430
3090 T=0:DA=RND(6):DB=RND(6):MO=0
3100 GOSUB3470
3110 GOSUB3500
3120 H=8:V=3
3130 BH=76:ONDAGOSUB3190,3210,3230,3250,3270,3290
3140 BH=283:ONDBGOSUB3190,3210,3230,3250,3270,3290
3150 REM
3160 GOSUB3600
3170 IFT>5THENFORI=1T01000:NEXTI:GOTO3800
3180 GOTO3100
3190 PRINT@BH+67,"@";
3200 RETURN
3210 PRINT@BH,"@";:PRINT@BH+134,"@";
3220 RETURN
3230 FORI=0T02:PRINT@BH+I*67,"@";:NEXT I
3240 RETURN
3250 GOSUB3310
3260 RETURN
3270 GOSUB3310:PRINT@BH+67,"@";
3280 RETURN
3290 GOSUB3310:PRINT@BH+64,"@";:PRINT@BH+70,"@";
3300 RETURN
3310 PRINT@BH,"@ @";:PRINT@BH+128,"@ @";
```

```
3320 RETURN
3330 FORVC=0T02:PRINT@76+VC*64," ";:PRINT@283+VC*64,"
";
3340 NEXTVC
3350 RETURN
3360 REM
3370 PRINT@40,"% DICE %";
3380 PRINT@88," YOU ARE GIVEN FIVE TRIES AT THE DICE";
3390 PRINT@152,"PRESS 'SPACE BAR' TO STOP THEIR SPINNING";
3400 PRINT@427,"> TRY # <";
3410 REM
3420 RETURN
3430 REM
3440 PRINT@896,"DIE A:"
3450 PRINT"DIE B:";
3460 RETURN
3470 T=T+1
3480 PRINT@435,CHR$(T+48);
3490 RETURN
3500 PRINT@334,DA;:PRINT@541,DB;
3510 GOTO3530
3520 RETURN
3530 IFINKEY%=" THEN3560
3540 DA=RND(6):DB=RND(6)
3550 GOTO 3500
3560 GOSUB3330
3570 PRINT@904+MO,DA;:PRINT@968+MO,DB;
3580 MO=MO+3
3590 RETURN
3600 C=DA*T+(6-DA)*(T-1)
3610 A(AB)=C
3620 C=(DB*T+(6-DB)*(T-1))+30
3630 B(AB)=C
3640 AA(AB)=A(AB)
3650 AB=AB+1
3660 RETURN
3800 CLS:PRINT@256," --> DO YOU WISH TO PLAY THIS":PRINTTAB(
10);"SONG OR TRY AGAIN?"
3810 PRINT:PRINTTAB(20);"TYPE <P>LAY OR <A>GAIN."
3820 PRINTTAB(27);"--> ";
3830 AN%=INKEY%
3840 IFAN%="A"THENAB=1:GOTO3000
3850 IFAN%<>"P"THEN3830
3860 GOSUB4000
3870 FORI=1T01000:NEXTI
3880 CLS:PRINT"PLAY THE SAME SONG AGAIN ? ";
3890 AN%=INKEY%
3900 IFAN%="N"THEN5000
3910 IFAN%<>"Y"THEN3890
3920 GOTO3860
4000 CLS:GOSUB4500:SW=0
4010 FORI=1T04
4020 GOSUB4300
4030 NEXTI
4040 FORI=1T03
4050 GOSUB4300
```



```

4060 NEXTI
4070 I=5
4080 GOSUB4300
4090 IFSW=1THEN4120
4100 SW=1:FORI=1TO5:A(I)=B(I):NEXTI
4110 GOTD4010
4120 FORI=1TO5
4130 A(I)=AA(I)
4140 NEXTI
4150 FORI=1TO3
4160 GOSUB4300
4170 NEXTI:I=5
4180 GOSUB4300
4190 RETURN
4300 FORI=1TOLEN(C*(A(I)))STEP6
4310 N=VAL(MID*(C*(A(I)),II,3))
4320 L=VAL(MID*(C*(A(I)),II+3,3))
4340 GOSUB12:IFN(<>)THENPOKEM1,L/4-2:POKEM2,N/3:U=USR(0):ELSEFORZ
=1TDL/10:NEXTZ
4360 NEXTI:RETURN
4500 FORZ=1TO3:PRINTZ#256-128,STRING$(64,179);STRING$(64,140);S
TRING$(64,179);NEXTZ
4510 FORZ=2TO8STEP2:SET(3,7+Z):SET(3,19+Z):SET(3,31+Z):NEXTZ
4520 LL=0:MS=1:X=5:Y=7:RETURN
5000 CLS
5010 PRINT#135,"WOULD YOU LIKE TO SAVE THIS SONG (Y/N) ?";
5020 AN%=INKEY$
5030 IFAN%="N"THEN100
5040 IFAN%("<>")="Y"THEN5020
5050 PRINT#263,"ON TAPE OR DISK (T/D) ?";
5060 AN%=INKEY$
5070 IFAN%="T"THEN5260
5080 IFAN%("<>")="D"THEN5060
5090 ONERRORGOTO7000
5100 DK=0
5110 PRINT#680,"NAME -> ";CHR$(31);
5120 INPUT$
5130 REM
5140 IFS%=""THEN5250
5150 OPEN"O",1,S#+"/MUS"
5190 FORI=1TO5
5200 PRINT#1,A(I)
5210 PRINT#1,B(I)
5220 PRINT#1,AA(I)
5230 NEXTI
5240 CLOSE1
5250 GOTD100
5260 CLS:PRINTTAB(16);"% STORE TO TAPE %"
5270 PRINT#256,"1. MAKE SURE YOUR TAPE IS READY TO RECORD."
5280 PRINT:PRINT"2. START RECORDING AND THEN PUSH 'SPACE BAR'."
5-80 WILL":PRINT" RETURN YOU TO THE PROGRAM WHEN IT IS FINISHE
D."
5310 IFINKEY%("<>") "THEN5310
5330 PRINT#980,"% WORKING %"
5340 FORI=1TO5:PRINT#-1,A(I);";";B(I);";";AA(I)
5360 NEXTI

```

```

5370 GOTD100
6000 CLS
6010 PRINT#263,"FROM TAPE OR DISK (T/D) ?";
6020 AN%=INKEY$
6030 IFAN%="T"THEN6210
6040 IFAN%("<>")="D"THEN6020
6050 ONERRORGOTO7000
6060 DK=1
6070 S%="":PRINT#680,"NAME -> ";CHR$(31);
6080 INPUT$
6090 REM
6100 IFS%=""THEN6200
6110 OPEN"O",1,S#+"/MUS"
6130 FORI=1TO5
6140 INPUT#1,A(I)
6150 INPUT#1,B(I)
6160 INPUT#1,AA(I)
6170 NEXTI
6180 CLOSE1
6190 CLS:GOSUB4000
6200 CLS:GOTD100
6210 CLS:PRINTTAB(17);"% RECALL FROM TAPE %"
6220 PRINT#256,"1. MAKE SURE YOUR TAPE IS SET UP AND READY AT YO
UR MUSIC PROGRAM"
6230 PRINT"2. START RECORDER ON PLAY THEN PUSH 'SPACE BAR'."
6240 PRINT:PRINT"3. IF ALL GOES WELL, S-80 WILL RETURN TO THE PR
OGRAM."
6250 PRINT:PRINT"4. IF YOU GET AN ERROR MESSAGE, RE-RUN PROGRAM
AND TRY AGAIN."
6270 IFINKEY%("<>") "THEN6270
6280 REM
6290 PRINT#980,"% WORKING %"
6300 FORI=1TO5:INPUT#-1,A(I),B(I),AA(I)
6320 NEXTI
6330 CLS:GOSUB4000
6340 FORI=1TO2000:NEXTI:GOTD100
7000 REM
7010 IFERR=122THENPRINT#910,"DISK IS FULL. TRY AGAIN.":GOTD7050
7020 IFERR=128THENPRINT#910,"BAD FILE NAME. TRY AGAIN.":GOTD705
0
7030 IFERR=106THENPRINT#910,"PROGRAM NOT FOUND. TRY AGAIN.":GOT
D7050
7040 PRINT#910,"ERROR #";ER:ONERRORGOTO0
7050 FORI=1TO2000:NEXTI
7060 IFDK=1THENRESUME6050
7062 RESUMRE5092
8000 DATA MACHINE LANGUAGE SOUND ROUTINE.
9000 DATA 22,255,14,255,30,1,27,65,62,1,211,255,27,122,179,200,1
6,250,65,62,3,211,255,27,122,179,200,16,250,24,232
10000 DATA 14406409606414406411406409612814306409606415306412806
4096255
10010 DATA 068032064064068032064064068032064032,0640320640640680
32064032064032057032064032
10020 DATA 076032081032076032064064096032076032064032,0640320640
64073032076032081032076032064032

```

continued on next page

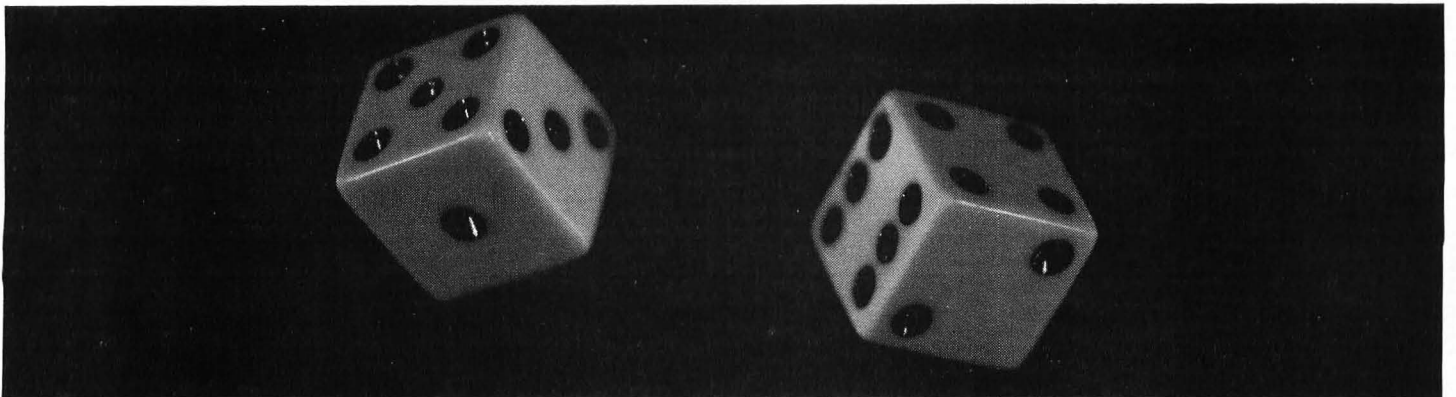


continued from previous page

10030 DATA 064064068032064064068032064064,0640320570320640320760
 32072032086064076032
 10040 DATA 064032076032064032057064064064096032,0760640860320960
 96086064
 10050 DATA 096032086032076032128064076032086032096032,0761601280
 32096096086064
 10060 DATA 076064086032096064064032076032096032,0760640640320760
 64081032076064
 10070 DATA 086064064032086064064032086064,0010320860320570320720
 32064032057032064032072032
 10080 DATA 072032072064076032072032072032076032086032,0720320570
 64064032072032076032072032064032
 10090 DATA 072064072032072064072032086064,0640960680320640320570
 32064064
 10100 DATA 064160076032072032068032,0640641020320960320860320760
 32072032064032
 10110 DATA 064255,064255
 10120 DATA 064096068032064128,064064064032064032064064064064
 10130 DATA 096128096064001064,076064086064096128
 10140 DATA 096064001064096064001064,096064001064096064001064
 10150 DATA 096064001064128012114012102012096040001064,0960641280
 64096128
 10160 DATA 102064102032128064102032128064,1020321710321280321140
 64128032114064
 10170 DATA 102032108032102032128064128032108032102032,0010321280

32076032086064076032086064
 10180 DATA 001032128032076032102032086032076032086032102032,1020
 32102064114032128032136032128032102032
 10190 DATA 128032102032086032064032064064086064,1020321710321280
 32102160
 10200 DATA 128032102064096032086064064032086032,0760320860640760
 32086032076032086064
 10210 DATA 128032102032076032086096086064,0860320860640960321020
 32108032102032086032
 10220 DATA 096032086064096064068032096064,0960641140320960641140
 32096032114032
 10230 DATA 096064086032096064102032096064,0960641140320960641140
 32096032114032
 10240 DATA 001032086032068032086032076032068032076032086032,0960
 32114032102032096064076032086064
 10250 DATA 114032114064121032114064108064,0960320960640760320860
 32096032086064
 10260 DATA 096032096064102032096032096032102032114032,1210321140
 64102032076064096064
 10270 DATA 096128086032086064096032,086128086064096064
 10280 DATA 102064001064128064001064,102160086032076032086032
 10290 DATA 102064064032064032064064064,1020640010641280640010
 64
 10300 DATA 102192064064,102064102064086064064064

5





ATARI-DV

VOLLEYBALL



by Bradley J. Bell
(translation contest winner)
original program by Jim Hilger

Volleyball is a graphics game program for the Atari requiring a 32K disk system and four joysticks. It is included as a bonus program on this month's Atari Disk Version of *SoftSide*.

Volleyball first appeared in the July issue of *SoftSide* as an Apple program.

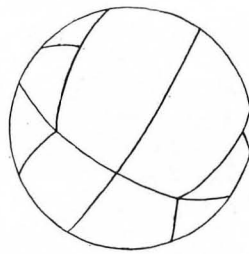
Introductory instructions are first displayed on the screen. Press RETURN, and the playing court will appear with the four "player-missile graphics" players. The message "SERVE" will flash over the team who is serving.

The players on each team who are farther from the net are called the "back" players, while those closer to the net are called the "up" players.

The up players have the ability to "spike" the ball by pushing forward on the joystick while pressing the trigger button. The left up player is controlled from slot 2 on the console, the right up player from slot 3, the left back player from slot 1, and the right back player from slot 4. Either the up or back players can serve using their joystick buttons. The back and up players each cover their own zones on the court, and these zones do not overlap.

Normal volleyball rules are generally followed. The game is played to 21, and a team must win by two points. A team can score a point only when it has served. If a team fails to get the ball over the net after three tries, it loses the point. Contrary to regular rules, however, a single player is allowed to hit the ball more than once in succession; this is because of having only two players per side, who cannot leave their respective zones.

Have fun, and may the best team win!



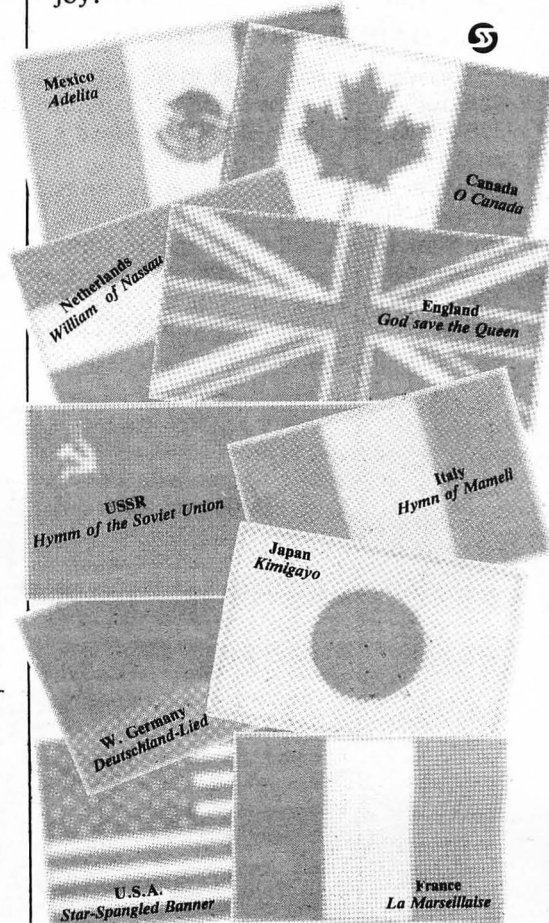
APPLE-DV

NATIONAL ANTHEMS

by Fred Pence

National Anthems is a musical graphics program for the Apple, requiring a 32K disk system and Applesoft. It is included as a bonus program on this month's Apple Disk Version of *SoftSide*.

In keeping with our musical theme this month, Apple Disk subscribers are receiving this colorful and tuneful program which plays the national anthems of ten different countries, while displaying their respective flags on the screen. The countries are U.S.A., France, England, West Germany, Italy, Japan, Mexico, U.S.S.R., Canada and the Netherlands. There's really no explanation needed to use *National Anthems* — just RUN the program, take your pick of countries from the menu, and sit back and enjoy!





S-80-DV

THE MEAN CHECKERS MACHINE™ II

by Lance Micklus

(c) copyright 1980, 81, by Lance Micklus, Inc. All reserved.

Editor's Note: *The Mean Checkers Machine™* is the November TRS-80™ DV selection. Only the documentation is published here with the program contained on the disk.

Welcome to the world of Checkers, a game the whole family can play. Your opponent is *The Mean Checkers Machine™*. It is the product of one of the oldest and most powerful programming languages, FORTRAN. You can beat the computer, but only if you're careful.

The game is played by the American Standard rules. All moves are checked for legality. To win, you must eliminate all of your opponent's pieces from the checkerboard or be the last player who can make a legal move.

LOADING THE PROGRAM

This version requires a minimum of 32K and one disk drive and will run on a S-80 Model I or Model III — it's compatible with both machines. However, earlier production versions of the S-80 Model III have a hardware problem which will cause the machine to bomb out on programs that run in 32-character mode. If your machine is one of these, contact your Radio Shack repair center and arrange to have this

problem fixed as all displays in this program are in 32-character mode. See notes below on how to work around this problem until you get your machine fixed.

The disk is a S-80 Model I disk. If you are using a Model III, put the disk in drive 1 and use the CONVERT utility to transfer the program to your Model III drive 0 disk. The file on the disk supplied is called CHECKERS /CMD. Regardless of whether you are using a Model I or Model III, type the following command from DOS: CHECKERS (ENTER). The program will load from disk and display the title *The Mean Checkers Machine™*.

PLAYING THE GAME

After the title of the program appears on the screen for approximately eight seconds, the screen will clear and the checkerboard will appear. If you are using a Model III and you do not see a checkerboard but instead see a box filled with O* or something similar to that, you probably have an early production version of a Model III with the 32-character fault. Until you can get this fixed, here's what you can do to work around the problem. Reload the program and start running it. As soon as you see the copyright notice and the words *The Mean Checkers Machine™*, press and HOLD DOWN the CLEAR key until the screen is clear. The program will now display in 64-character mode. It is normal for the blinking routines to indicate that the wrong piece is to be moved. The game, however, will still be perfectly usable.

The checkerboard display is as follows: Each square is lettered horizontally from A to H, and numbered vertically from 1 to 8. The upper-left square is A1, and the lower-right square is H8.

Your pieces are O's, and the computer's pieces are X's. Whenever it's

your turn, the computer prompts with YOUR MOVE. To make a move, you must tell the computer the current location of the checker that is to be moved, followed by a dash, then the position it is to move to. The computer will now display the checkerboard with your piece moved per your instructions. The computer will then take its turn.

If the move is illegal, the computer will display the words **INPUT ERROR**, and again prompt you for a move. An example of an illegal move is A6-A4. This will cause an **INPUT ERROR** because such a move is illegal. A good move to start with at the beginning of a game might be B7-A6.

MAKING A JUMP

To make a jump on your turn, enter the move the same way you would for a regular move. For example: C6-A4. To enter an additional jump, if any, you type only the location to move to next. The computer already knows the FROM position. Thus, a second jump to the above example might be just C2 — meaning a jump from A4 to C2.

The Mean Checkers Machine™ can play either regulation Checkers in which all jumps must be taken (JUMP FORCED mode), or a variation of Checkers in which jumps are taken at the discretion of each player (JUMP OPTIONAL mode). The computer begins the game playing in JUMP FORCED mode.

In JUMP FORCED mode, the computer will always prompt for additional jump moves if they can be taken. If no more jumps are possible on your turn, the computer will automatically take its turn.

In JUMP OPTIONAL mode, the computer always assumes that when the first move was a jump, then additional jumps are possible. If you do



not want to take the extra jumps, or none are possible, type the word PASS to let the computer take its turn. The game is set up this way so the computer doesn't tip you off to an extra jump you might otherwise have missed.

SAVE COMMAND

If you want to save a game type the word SAVE instead of entering your move. The program will now prompt for a filespec. Enter a valid TRSDOS file name, then press (ENTER). The program will now store the current checkerboard onto disk, and then prompt for another move.

LOAD COMMAND

You can restore a previously saved game by typing the word LOAD instead of entering your move. The program will now prompt for a filespec. Enter the file name of the disk file which contains the game you wish to restore. Press (ENTER). The program will now read the file and restore the old game. If there are any disk I/O errors, the program will display the words ****INPUT ERROR**** and not restore the game. NO OTHER error messages will appear, such as **ILLEGAL ACCESS TO A PROTECTED FILE**.

OTHER COMMANDS

IQ — The computer normally plays at an IQ of 3. To change the IQ to a level of 2, type IQ=2 instead of entering your move. You may play at any IQ of from 1 to 9. The IQ number is displayed in the upper right-hand corner of the screen. An IQ of 1 plays a very novice game, and an IQ of 9 plays an expert game. At an IQ of 1, the computer makes its moves in a few seconds. At an IQ of 9, the computer takes from four to twelve hours to make a move.

The IQ level determines the number of look-aheads minus 1. For example: An IQ of 3 allows two look-aheads.

UMOV — Normally, the computer lets you go first at the beginning of the game. If you would like the computer to begin the game by making the first

move, type the word UMOV instead of entering your move. This command works any time. It is, however, only legal at the beginning of the game. In the interest of simplifying the program coding, some honesty was assumed on the part of the player.

QUIT — When you are ready to give up, type QUIT instead of entering your move. The computer will automatically start a new game — you go first.

(BREAK) — To return to DOS, press the (BREAK) key.

JO / JF — The computer initially starts the game in JUMP FORCED mode. You may wish to play a common variation of Checkers in which all jumps are optional. In this version of the game, neither player has to make a jump. To play in the JUMP OPTIONAL mode, type the word JO instead of entering your move. Note that the display in the upper right-hand corner of the screen now shows the letter O instead of the letter F. To return to regular JUMP FORCED mode, type JF instead of your move.

KINGS — Besides playing Checkers, you can play still another popular variation of the game called Kings. To play, just type the word KINGS instead of entering your move.

X — This simple one-letter command will exchange your pieces with those of the computer's. Each player still plays the same side of the board, however, and still makes the moves in the same direction. When used with the UMOV command, you can let the computer play against itself.

T — This one-letter command lets you take back your last move. You must issue this command first before any other command, otherwise the results will be unpredictable. Use this command to take back a move entered by mistake. Also, you can use this command to let the computer show you what it thinks your best move is. To do this, use command X then command UMOV. The computer will make its move using your setup. After the computer displays the move and you've had time to study it, type the letter T and hit ENTER. This will now

put the checkerboard back the way it was. If you like the computer's move, then use X followed by UMOV again to put things back to normal and to let the computer figure out a move with what are really its own pieces.

CLEAR — This command clears all of the pieces from the checkerboard leaving you with just the checkerboard. Use this with the ZAP command to set up special situations.

ZAP — This is a complicated command, but it is very powerful. ZAP lets you remove or place pieces on the checkerboard without any checks for legality. When used with the CLEAR command, you can set up special situations to see how the computer will react.

The command contains four characters. The first character is the letter Z. The next character is a number from 1 to 5.

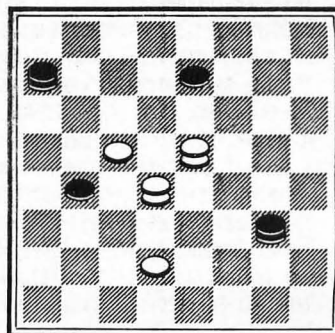
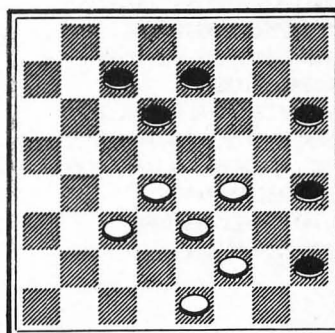
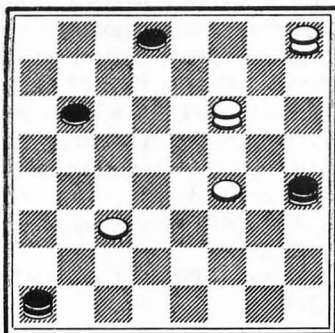
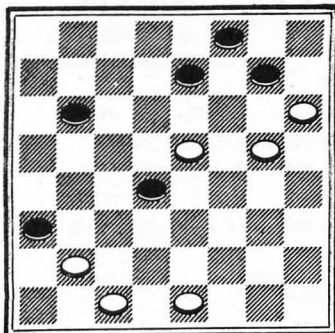
- 1 indicates a computer king
- 2 indicates a computer checker
- 3 indicates a blank square
- 4 indicates a human checker
- 5 indicates a human king

The next two characters are the location of the square to be zapped — like A2. So the command Z5A2 will make square A2 have a player's king on it.

HOW THE PROGRAM WORKS

At the beginning of the computer's turn, the computer begins by searching throughout the checkerboard for every possible move. When a legal move is found, the computer makes the move on an internal board, then looks at the board to see if the move looks promising.

If the IQ is set at 3, the computer sets up a second checkerboard and searches for all of your moves. If it finds an interesting move you might take, it sets up a third checkerboard to see what the computer's best reply is to your move. This process can continue up to nine levels deep, depending on the IQ level of the game. The depth of the current checkerboard is displayed as a blinking number in the top left corner of the screen.



GAMEPLAY is the prototype of the interpreter being prepared for *Diversions™*, the upcoming *Envyrn™*-based magazine. It controls the execution of specially prepared *Envyrnments™* and the player's movement through them. This interpreter is presented expressly for use with the prototype of *Envyrn™* published in the October issue of *SoftSide*.

When the program is run, you will be presented a small menu with three options. The first allows you to enter the *Envyrnment™* currently in memory. The second allows you to save an *Envyrnment™* in progress, and the third loads a new *Envyrnment™* from diskette. In most instances, you'll first want to load the database on disk called ADV/DAT, then enter it.

After doing this, you will see a display showing where you are, as well as displays giving your facing, activity level and energy. Usually, you'll be shown only the tile you occupy (flashing) and the tiles immediately surrounding your position. Once you've viewed a tile, however, you'll be able to see it from any distance, as long as it will fit within the screen parameters. Those with little patience may use the CHEAT command (see below). The function controlling the exposure of tiles will be disabled, and the screen will fill completely with tiles.

These are the operating commands presently active in the program. Their functions are:

- @ = Enter as prefix to long command (see below)
- (<) = Decrease body activity (slower)
- (>) = Increase body activity (faster)
- S = Change body activity to swim
- Left Arrow = Turn facing counterclockwise
- Right Arrow = Turn facing clockwise
- = Return to menu
- U = Use something (such as a door-open/close)
- E = Enter (a building)
- I = Identify the parcel directly in front of you

Long commands

Hitting the @ symbol puts you in a mode in which you can type multiple letter commands..Only the first three letters are significant, although the entire command may be typed, if desired.

- LOO(K): Change display to graphic mode one
- SEE: Graphic mode two
- WAT(CH): Graphic mode three
- IDE(NTIFY): Same as I above
- ENT(ER) or GO: Same as E above
- SWI(M): Same as S above
- CHE(AT): Disable limited display (see above)
- NOR(MAL): Disable CHEAT function

```

5 ' (C)1981 SoftSide Publications -- RWR/RJB
10 CLEAR25000:NE=0:MD=7:I=0:J=0:CH$="":M1$=CHR$(26)+STRIN
  B$(15,8):M$=LEFT$(M1$,10):N$=LEFT$(M$,7):X=0:Z=0:PC=3:PL=3:DIMPM
  $(154),X(8),Y(8),E(10)
21 FORT=1TO24:READZ:AR$=AR$+CHR$(Z+128):NEXTT
22 DATA 24,9,16,3,3,63,-83,-83,-66,48,48,63,9,24,1,63,48,48,-68,
  -83,-83,63,3,3
25 FOR T=1 TO 8:READ X(T),Y(T):NEXTT
26 DATA 0,-1,1,-1,1,0,1,1,0,1,-1,1,-1,0,-1,-1
30 FOR T=0 TO 10:READE(T):NEXTT
32 DATA 1,0,0,0,0,0,-1,-1,-2,-5,-10
50 EN=1000
100 Q=0:AL=5:FA=1:CLS:PRINT@20,"MAIN MENU
Enter Dungeon      1
Save Dungeon       2
Load Dungeon       3
":INPUT"Selection";Q:ONQGOSUB115,10020,10000:GOTO100
115 MS$="STAND":CH$="B":GOSUB150:PC=4:PL=4:GOSUB200:GOTO1000
150 PRINT@980,"New graphics mode (1-3)":X=1004:GOSUB25000:GM=VA
  L(Q$):IF GM<1 OR GM>3 THEN 150 ELSE IF GM=1THENAO=9:DO=147:RL=2:
  RC=2:RETURNELSEIFGM=2THENAO=6:DO=86:RL=3:RC=3:RETURNELSEAO=3:DO=
  19:RL=7:RC=7:RETURN
195 PX$=MID$(MP$(PZ),PX,1):IF(PX$=CHR$(191))OR(PX$=".")OR(PX$="O
  ")OR(PX$="P")OR(PX$="Z")THENSX=1:RETURNELSESX=0:RETURN
199 CLS:PRINT"Saving GAMEPLAY/BAS:0":SAVE"GAMEPLAY/BAS:0":PRINT
  "Saving GAMEPLAY/BAS:1":SAVE"GAMEPLAY/BAS:1":END
200 CLS:GOSUB 1990:X=0:FC=PC-RC:FL=PL-RL:FORI=PL-RLTOPL+RL:FORJ=
  PC-RCTOPC+RC:IFI<1ORJ<1ORI>99ORJ>99THENCH$="+ELSECH$=MID$(MP$(I
  ),J,1):IF CH$<CHR$(128)THENIF(ABS(J-PC)<=1ANDABS(I-PL)<=1)ORC9=1
  THENMID$(MP$(I),J,1)=CHR$(ASC(CH$)+128):ELSECH$="+
201 GOSUB2000:GOSUB2200:X=X+AO:NEXTJ:X=X+DO:NEXTI:RETURN
220 ONGMGOSUB230;240,250:GOSUB2000:GOSUB2200:RETURN
230 X=(I-FL)*128+(J-FC)*6:RETURN
240 X=(I-FL)*128+(J-FC)*6:RETURN
250 X=(I-FL)*64+(J-FC)*3:RETURN
1000 IF AL<6 THEN CH$=INKEY$:CH$="+":I=PL:J=PC:GOSUB220:IFCH$=""
  THENI=PL:J=PC:CH$=MID$(MP$(I),J,1):GOSUB220:GOTO1000:ELSE 1005
1002 FOR Q=1 TO (10-AL)*2:CH$="+":I=PL:J=PC:GOSUB220:CH$=INKEY$:
  IF CH$=""THENI=PL:J=PC:CH$=MID$(MP$(I),J,1):GOSUB220:NEXTQ ELSE
  1005
1003 EN=EN+E(AL):GOTO 1800
1005 CH$=MID$(MP$(I),J,1):GOSUB220:EN=EN+E(AL):IF EN>1000 THEN E
  N=1000
1007 GOSUB 1990:IF CH$=">" AND MS$<>"SWIM"THEN 1200
1008 IF CH$="@" THEN 3000
1010 IFCH$="<"ANDMS$<>"SWIM"THEN1210
1020 IFCH$="-"THEN100
1030 IF CH$<>"U" THEN 1040
1032 CH$=MID$(MP$(PL+Y(FA)),PC+X(FA),1):Z4=1
1034 GOSUB2000:Q$=MID$(MP$(R1+Y-1),R2+TX,1):IF Q$<>". " THEN MID
  $(MP$(PL+Y(FA)*Z4),PC+X(FA)*Z4,1)=Q$:CH$=Q$:I=PL+Y(FA)*Z4:J=PC+X
  (FA)*Z4:GOSUB220:GOTO1000
1036 IF Z4=1THENZ4=0:CH$=MID$(MP$(PL),PC,1):GOTO1034
1038 PRINT@960,CHR$(30);"There is nothing here to use.";
1040 IF CH$="E" THEN 3510
1045 IF CH$="S"THEN3532
1050 IFCH$=CHR$(9)THENFA=-FA*(FA<>8)+1:PRINT@110,"Facing: ";MID$
  (AR$,FA*3-2,3);:GOTO1000
1055 IFCH$=CHR$(8)THENFA=(FA-1)-8*(FA=1):PRINT@110,"Facing: ";MI
  D$(AR$,FA*3-2,3);:GOTO1000
1060 IFCH$="I"THENIF PC+X(FA)>99 OR PC+X(FA)<1 OR PL+Y(FA)>99 OR
  PL+Y(FA)<1 THEN MG$="Nothing there":GOTO1980ELSECH$=MID$(MP$(PL
  +Y(FA)),PC+X(FA),1):GOSUB2000:MG$=MID$(MP$(R1+5),R2,9):GOTO1980
1190 GOTO 1000
1200 IFAL>9THENMG$="At max level":GOTO1980 ELSEAL=AL+1:MG$="DONE
  ":GOTO1950
1210 IFAL<1THENMG$="At min level":GOTO1980 ELSEAL=AL-1:MG$="Done"
  :GOTO1950

```

```

1800 IF PC+X(FA)>99 OR PC+X(FA)<1 OR PL+Y(FA)>99 OR PL+Y(FA)<1 T
HEN MS="Nothing there":GOTO1980
1810 CH=MID$(MP$(PL+Y(FA)),PC+X(FA)):GOSUB2000
1812 Q=MID$(MP$(R1+TY-1),R2+TX-1,1)
1820 IF (Q$="A"ORQ$=".")ANDMS$="SWIM"THENMS$="STAND":AL=5
1830 IF (Q$="A" OR Q$=".")OR(Q$="S"ANDMS$="SWIM") THEN PC=PC+X(F
A):PL=PL+Y(FA):GOSUB2000:GOTO7000
1840 IF Q$<"0" OR Q$="." THEN MS$="I can't!":GOTO1980
1850 Q=ASC(Q$)-47:IF ((QAND1)ANDFA=1)OR((QAND2)ANDFA=3)OR((QAND4
)ANDFA=5)OR((QAND8)ANDFA=7) THENQ$="A":GOTO 1820
1860 PRINT@960,CHR$(30);"Not in this direction";:Q$=" " :GOTO1840
1950 IFAL=0THENMS$="SLEEP"ELSEIFAL=1THENMS$="LIE"ELSEIFAL=2THENM
S$="RECLINE"ELSEIFAL=3THENMS$="SIT"ELSEIFAL=4THENMS$="CROUCH"ELS
EIFAL=5THENMS$="STAND"ELSEIFAL=6THENMS$="STALK"ELSEIFAL=7THENMS$
="WALK"ELSEIFAL=8THENMS$="TROT"
1951 IFAL=9THENMS$="RUN"ELSEIFAL=10THENMS$="FLEE"
1955 PRINT@55,CHR$(30);:PRINT@55,MS$:GOTO1980
1980 FORI1=1TO10:PRINT@560,MS$:FORI2=1TO20:NEXTI2:PRINT@560,CHR
$(30);:NEXTI1:GOTO1000
1990 PRINT@46,"Activity: ";MS$:CHR$(30);:PRINT@110,"Facing: ";MID
$(AR$,FA$3-2,3);:PRINT@174,"Cheat: ";MID$("OffOn ",C9$3+1,3);:PRI
NT@878,"Energy: ";MID$(STR$(EN),2);CHR$(30);
1991 RETURN
2000 '
2005 IF CH$>CHR$(128) THEN CH$=CHR$(ASC(CH$)-128)
2010 E=ASC(CH$)-BC:R1=((INT(E/11)*6)+100:R2=9*(E-(INT(E/11)*11
)+1:IF(R1<100)OR(R1>130) THENR1=106:R2=37:RETURNELSERETURN
2200 ID$=MID$(MP$(R1+4),R2+6,1):ND$=MID$(MP$(R1+5),R2,5):IJ$=MID
$(MP$(R1+4),R2+7,2):ONGMOTO2210,2220,2230
2210 PRINT@X,MID$(MP$(R1),R2,9);M$:MID$(MP$(R1+1),R2,9);M$:MID$(
MP$(R1+2),R2,9);:RETURN
2220 PRINT@X,MID$(MP$(R1+3),R2,6);N$:MID$(MP$(R1+4),R2,6);:REUR
N
2230 PRINT@X,MID$(MP$(R1+3),R2+6,3);:RETURN
3000 PRINT@942,">";CHR$(30);
3010 L=16:GOSUB 5000
3020 Q$=Q$+" " :A$=LEFT$(Q$,3):A=INSTR(Q$, " ") :A1$=LEFT$(MID$(Q$,
A+1,3)
3030 IF A$="LOO" THENGM=1:AQ=9:DD=147:RL=2:RC=2:GOSUB200:GOTO1000
3040 IF A$="SEE" THENGM=2:AQ=6:DD=86:RL=3:RC=3:GOSUB200:GOTO1000
3050 IF A$="WAT" THENGM=3:AQ=3:DD=19:RL=7:RC=7:GOSUB200:GOTO1000
3300 IF A$="IDE" THEN CH$="I":GOTO 1060
3310 IF A$="CHE" THENC9=1:GOSUB200:GOTO1000
3320 IF A$="NOR" THENC9=0:GOTO1000
3500 IF A$<"ENT" AND A$<"GD" THEN 3530
3510 CH$=MID$(MP$(PL),PC,1):GOSUB2000:Q$=MID$(MP$(R1+TY-1),R2+TX
+1,1):IF Q$="X" THEN PRINT@960,CHR$(30);"All the doors are locke
d!";:GOTO1000ELSEIF Q$<"E" THENPRINT@960,CHR$(30);"There's nothi
ng here to enter.":GOTO1000

```

```

3520 GOTO 7028
3530 IF A$<"SWI" THEN 1000
3532 CH$=MID$(MP$(PL+Y(FA)),PC+X(FA),1):GOSUB2000:Q$=MID$(MP$(R1
+TY-1),R2+TX-1,1):IFQ$<"S" THENPRINT@960,CHR$(30);"I can't swim
there.":GOTO1000
3540 AL=7:MS$="SWIM":PL=PL+Y(FA):PC=PC+X(FA):GOSUB1990:GOSUB200:
GOTO 1000
5000 Q$=""
5010 PRINTCHR$(14);:FORI=1TO10:A$=INKEY$:IFA$="" THENNEXTI:PRINTC
HR$(15);:FORI=1TO10:A$=INKEY$:IFA$="" THENNEXTI:GOTO5010
5020 PRINTCHR$(15);:IF A$=CHR$(13) THENRETURN:ELSEIFA$=CHR$(8) THE
NIFQ$="" THEN5010ELSEPRINTA$:Q$=LEFT$(Q$,LEN(Q$)-1):GOTO5010:ELS
EIFA$=CHR$(24) THENPRINTSTRING$(LEN(Q$),8);:Q$="":GOTO5010:ELSEIF
A$<" " THEN5010
5030 IFLEN(Q$)=L THEN5010ELSEQ$=Q$+A$:PRINTA$:GOTO5010
5100 L=16-LEN(Q$):GOSUB5010:GOTO3020
7000 CH$=MID$(MP$(PL),PC,1):GOSUB2000:Q$=MID$(MP$(R1+TY-1),R2+TX
+1,1)
7012 IF PC=3ANDPL=69 THENPC=82:PL=58:GOTO7090
7014 IF PC=82ANDPL=58 THENPC=3:PL=69:GOTO7090
7016 IF Q$="." THEN1000
7020 IF Q$<"J" THEN 7100
7028 I=NR$Y+MH$9+1
7030 FOR Q=1 TO MH
7040 Q$=MID$(MP$(I),Q$9-8,9)
7050 IF VAL(MID$(Q$,1,2))=PC AND VAL(MID$(Q$,3,2))=PL THEN PC=VA
L(MID$(Q$,5,2)):PL=VAL(MID$(Q$,7,2)):GOTO7080
7060 IF VAL(MID$(Q$,5,2))=PC AND VAL(MID$(Q$,7,2))=PL THEN PC=VA
L(MID$(Q$,1,2)):PL=VAL(MID$(Q$,3,2)):GOTO7080
7070 NEXTQ:I=I+1:IFI<NR$Y+MH$9+JR THEN 7030 ELSECH$=MID$(MP$(P
L),PC,1):IFCH$>CHR$(188) ANDCH$>CHR$(189) ANDCH$>CHR$(190) THENP
RINT@960,CHR$(30);"The door is locked";:GOTO1000ELSE1000
7080 CH$=MID$(MP$(PL),PC,1):IFCH$>CHR$(128) THENCH$=CHR$(ASC(CH$)
-128)
7085 IFCH$="("ORCH$=">" THENPL=PL+1ELSEIFCH$=")"ORCH$="<" THENPC=P
C-1
7090 GOSUB200:GOTO1000
7100 GOTO 1000
10000 CLOSE:LINEINPUT"Load filename (ADV/DAT) ";A$:OPEN"I",1,A$
10005 INPUT#1,BC,FM,JR,LC,MH,MW,NE,NG,NR,PV,SG,SY,TX,TY,GC,N1
,N2
10010 FORI=1TO154:IFEOF(I) THENCLOSE:RETURNELSELINEINPUT#1,MP$(I)
:NEXTI:RETURN
10020 CLOSE:LINEINPUT"Save filename ";A$:OPEN"O",1,A$:PRINT#1,BC
,"FM","JR","LC","MH","MW","NE","NG","NR","PV","SG","SX","SY","T
X","TY","GC","N1","N2:FORI=1TO154:PRINT#1,MP$(I):NEXTI:CLOSE:RET
URN
25000 Q$=INKEY$:IFQ$="" THENGOTO25000ELSEPRINT@X,Q$:RETURN

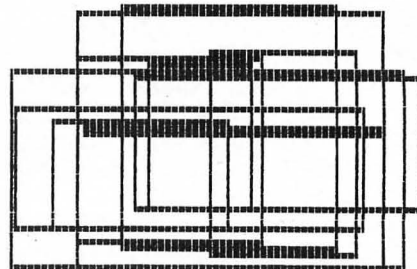
```

S-80 One Liner

```

1 CLS:PRINT@402,"C O M P U T E R A R T ! !":PRINT:DEFINT A-Z:RA
NDOM:FORT=1TO1000:NEXTT:CLS:FORT=1TO10:A=RND(62)-1:B=RND(62)+61:
C=RND(22)-1:D=RND(22)+21:FORX=ATOB:SET(X,D):SET(X,C):NEXTX:FORY=
CTOD:SET(B,Y):SET(A,Y):NEXTY,T:FORT=1TO3000:NEXTT:GOTO1

```



WILLIAM H. TOOKER
499 OAKWOOD DRIVE
GRENA, LA, 70053

by Alan J. Zett

The SOUND command is an enhancement to S-80 Level II and Disk BASIC requiring a minimum of 16K RAM.

Are you tired of making difficult and cumbersome USR calls every time you want to generate sound? Are you frustrated with the limited types of sounds available when using the standard sound routine?

Until now, if you wanted more versatile sound, it meant purchasing one of the many enhanced BASICs on the market. For example, *NEWBASIC* and *ENHBAS* both have sound commands, but depending on where you buy them, can cost quite a bit. And then, when you've written your sound, it can only be used for your personal programs. If you're an experimenter, like I am, you would probably be dissatisfied with that situation.

As long as you're *POKEing* that sound routine into memory, why not make it do something useful as well? Why not **ADD A SOUND COMMAND** to BASIC? Yes, it is possible! Using the same techniques as *NEWBASIC* and *ENHBAS*, I was able to tie into the error-checking relay in BASIC. This is the area that allows Disk BASIC to print longer error messages.

Let me explain. When BASIC comes to an error in a program line, it sets up all the pointers to display an abbreviated error message. But just before the message is displayed, a jump to address 41A6H is done. This is our doorway to BASIC.

One of the pointers set before jumping is the address in memory, minus one, where the error is occurring. By placing the address of our extra command in 41A6H and checking the location of the error, we can perform a check to see if the error matches one of the keywords we have defined. Then, using some helpful ROM routines, the arguments following the keyword can be evaluated and used by our routine. Of course, if the error is a real one and not one of our making, then we want to continue processing the error. To do this, we save the original address contained in 41A6H and jump there to continue.

For those of you who are familiar with Machine Language programming, I've provided a documented source listing with this article. Note that this routine uses ROM calls that may or may not be the same on the Model III.

The format of the new command is "SOUND frequency, duration." SOUND uses all integer arguments, but any single- or double-precision number will be converted by the routine if possible. The "frequency" argument is a value from 0 (highest) to 255 (lowest). Numbers outside this range will repeat the scale. For example, 256-511 will produce the same sounds as 0-255. The duration of the frequency may be from 0 (shortest) to -1 (longest). (Since it is an integer argument, -1 is equivalent to a duration of 65535.) The SOUND command incorporates a BREAK key check for the longer notes that may otherwise tend to lock up the computer for the entire duration of the note.

One of the disadvantages of most of the sound routines that I've seen is that notes of a higher frequency tend to take less time for the same duration value of a note. In the interest of being able to create certain sound effects, I have

left out a routine which would adjust the duration, and instead offer a BASIC formula for those who are interested. Assuming that the note will be in a range of 0-255, the following formula will work:

SOUND frequency, duration * (256 / frequency)

The BASIC program shown below is the routine which will add the SOUND command to Level II or Disk BASIC. The SOUND command will stay in memory until a system reset occurs or until any statement that clears variables is executed. These are the same restrictions that apply to the old USR sound routine. Note: You should change the CLEAR statement in line 60010 to that required by your program. Following the BASIC listing (Figure 1) are a few demonstration sounds (Figure 2) and the unassembled source listing (Figure 3).

I plan to be writing more columns about adding different commands to BASIC. If there is a particular function you would like me to try to work on (or if you have any other comments), write to me at *SoftSide*.

Figure 1

```

1 CLS:GOTO 60000
10 STOP:REM YOUR PROGRAM STARTS HERE
60000 Z=0:FORX=1TO158:READY:Z=Z+Y:NEXT:IFZ<>15204THENCLS:PRINT"D
ATA BASE ERROR IN LINES 60060-60160, CHECK LISTING.":PRINT:LIST6
0060-60160ELSEY=86:X=255:POKE-1,0:IFPEEK(-1)<>0THENX=192:POKE-16
385,0:IFPEEK(-16385)<>0THENX=127
60010 POKE 16562,X:POKE 16561,Y:CLEAR50:A1=PEEK(16561)+2:A2=PEEK
(16562):A=A1+A2*256:Z=A-1:FORX=1TO158:Z=Z+1:Z=Z+65536*(Z>32767)
60020 READY:IFY<0THENY=A1+ABS(Y):POKEZ,Y+256*(Y>255):Z=Z+1:POKEZ
,A2-(Y>255):NEXTELSEPOKEZ,Y:NEXT
60030 IFPEEK(16396)=201POKE16526,A1:POKE16527,A2ELSECMD"T":DEFUS
R=A1+(A2+256*(A2>127))*256:POKE14308,0
60040 IFPEEK(16807)+PEEK(16808)*256<>A+24THENA=USR(0)
60050 SOUND11,11:GOTO10
60060 DATA8,166,65,50,-164,42,167,65,34,-165,62,195,50
60070 DATA166,65,33,-24,34,167,65,201,245,123,254,2,40,4,254
60080 DATA16,32,79,229,213,42,230,64,126,183,32,4,35,35,35,35
60090 DATA215,6,5,17,-156,26,190,32,104,19,35,16,248,43,215
60100 DATA43,34,230,64,241,241,241,241,197,213,215,205,55,35
60110 DATA229,205,127,10,42,33,65,34,-167,225,215,43,34,230,64
60120 DATA35,205,55,35,43,229,205,127,10,42,33,65,58,-167,60
60130 DATA183,87,24,4,24,48,24,44,66,62,1,211,255,16,252,66,62
60140 DATA2,211,255,16,252,58,64,56,230,4,32,7,124,181,40,3,43
60150 DATA24,228,175,50,154,64,225,209,193,215,195,30,29,83,79
60160 DATA85,78,68,209,225,241
    
```

Figure 2

```

10 FORX=1TO50STEP.1:SOUNDX,3:NEXT
20 FORX=0TO50:FORY=XTOX+11:SOUNDY,3-(X/50):NEXT:NEXT
30 Z=0:FORX=50TO0STEP-1:FORY=XTOX+11:SOUNDY,Z:NEXT:Z=Z+.5:NEXT
40 FORX=0TO30:FORY=XTOX+10:SOUNDY,10:SOUND11-Y+70,2:NEXT,Y,X
50 FORX=30TO100:SOUNDX,11:SOUNDX+44,10:SOUNDRND(55),9:NEXT
60 FORX=0TO255:SOUNDX,2:SOUND255-X,2:NEXT
    
```

Figure 3

```

00100      ORG      07F00H ;BASIC PROGRAM IS RELOCATABLE
00110 EVAL  EQU      2337H ;EVALUATE EXPRESSION
00120 CINT  EQU      0A7FH ;CONVERT TO INTEGER
00130 ACCUM EQU      4121H ;MULTI-PRECISION ACCUMULATOR
00140 ERELAY EQU     41A6H ;ERROR MESSAGE RELAY
00150 STMPTR EQU     40E6H ;ENCODED STATEMENT POINTER
00160 ERRCOD EQU     409AH ;ERROR CODE STORAGE
00170 BASINT EQU     1D1EH ;BASIC INTERPRETER
00180 SETUP LD      A,(ERELAY) ;GET ERROR JUMP ADDRESS,
00190      LD      (OUT2),A ;AND SAVE IT.
00200      LD      HL,(ERELAY+1) ;GET THE REST,
00210      LD      (OUT2+1),HL ;AND SAVE THAT TOO.
00220      LD      A,0C3H ;A "JP" CODE
00230      LD      (ERELAY),A ;INTO THE RELAY
00240      LD      HL,START ;POINTING TO OUR
00250      LD      (ERELAY+1),HL ;PROGRAM.
00260      RET      ;BACK TO BASIC.
00270 START PUSH    AF ;SAVE "AF" REGS.
00280      LD      A,E ;GET THE ERROR CODE
00290      CP      2 ;IS IT ?SN ERROR
00300      JR      Z,GOTTEN ;IF SO, CONTINUE.
00310      CP      10H ;IS IT ?BS ERROR
00320      JR      NZ,ROUT1 ;IF IT'S REALLY AN ERROR
00330 GOTTEN PUSH    HL ;SAVE THE OTHER REGS.
00340      PUSH    DE ; " " " "
00350      LD      HL,(STMPTR) ;GET LOCATION OF ERROR-1
00360      LD      A,(HL) ;TEST FOR END OF LINE
00370      OR      A ;(A 00H BYTE)
00380      JR      NZ,NEXT ;IF NOT, GOTO NEXT PART
00390      INC     HL ;BUMP PAST LINE NUMBER
00400      INC     HL
00410      INC     HL ;PAST NEXT LINE POINTER
00420      INC     HL
00430 NEXT  RST      10H ;GET FIRST CHR INTO "A"
00440      LD      B,5 ;# OF BYTE TO COMPARE
00450      LD      DE,CMD ;START OF COMMAND TABLE
00460 COMP  LD      A,(DE) ;GET FIRST CMD CHR.
00470      CP      (HL) ;CMP TO ERROR CHR
00480      JR      NZ,OUTO ;IF NO MATCH, REAL ERROR!
00490      INC     DE ;ELSE BUMP CMD POINTER
00500      INC     HL ;BUMP ERROR POINTER
00510      DJNZ   COMP ;TRY AGAIN
00520      DEC     HL ;BUMP BACK 1 CHR
00530      RST      10H ;FIND 1ST CHR OF ARG.
00540      DEC     HL ;BUMP BACK 1 CHR
00550      LD      (STMPTR),HL ;SAVE IN ERROR POINTER
00560      POP     AF ;GET RID OF EXCESS
00570      POP     AF ;BAGGAGE.
00580      POP     AF
00590      POP     AF
00600      PUSH    BC ;SAVE IMPORTANT VALUES
00610      PUSH    DE
00620 SOUND RST      10H ;GET 1ST CHR OF ARG.

00630      CALL   EVAL ;EVALUATE EXPRESSION
00640      PUSH   HL ;SAVE POINTER TO NEXT ARG
00650      CALL   CINT ;CONVERT ARG. TO INT
00660      LD     HL,(ACCUM) ;GET IT FROM ACCUM.
00670      LD     (TONE),HL ;SAVE IT IN HERE.
00680      POP    HL ;GET NEXT ARG. POS.
00690      RST   10H ;GET 1ST CHR OF ARG.
00700      DEC   HL ;BUMP BACK 1
00710      LD   (STMPTR),HL ;SAVE IN ERROR POINTER
00720      INC   HL ;BUMP UP AGAIN
00730      CALL EVAL ;EVALUATE EXPRESSION
00740      DEC   HL ;BUMP BACK 1
00750      PUSH  HL ;SAVE IT
00760      CALL CINT ;CONVERT TO INT.
00770      LD   HL,(ACCUM) ;GET IT INTO HL
00780      LD   A,(TONE) ;GET BACK THE TONE
00790      INC   A ;ADD 1
00800      OR   A ;CLEAR CONDITION FLAGS
00810      LD   D,A ;GET TONE INTO D
00820      JR   LOOP ;GOTO PLAY LOOP
00830 ROUT1 JR   OUT1 ;ROUTING ADDRESS FOR OUT1
00840 ROUT0 JR   OUT0 ;ROUTING ADDRESS FOR OUT0
00850 LOOP  LD   B,D ;GET FREQUENCY IN B
00860      LD   A,1 ;HIGH SIDE OF CYCLE
00870 LOOP1 OUT (255),A ;OUT TO THE CASSETTE
00880      DJNZ  LOOP1 ;CONT WITH NOTE
00890      LD   B,D ;GET FREQUENCY AGAIN
00900      LD   A,2 ;GET LOW SIDE OF CYCLE
00910 LOOP2 OUT (255),A ;SEND IT OUT!
00920      DJNZ  LOOP2 ;COMPLETE CYCLE
00930      LD   A,(3B40H) ;CHECK FOR A BREAK
00940      AND   4 ;KEY ABORT OF NOTE
00950      JR   NZ,DONE ;IF SO, WE'RE ALL DONE
00960      LD   A,H ;HIGH BYTE OF DURATION
00970      OR   L ;LOW BYTE
00980      JR   Z,DONE ;CHECK FOR ZERO
00990      DEC   HL ;DECREMENT DURATION
01000      JR   LOOP ;CONTINUE
01010 DONE XOR   A ;ZERO A
01020      LD   (ERRCOD),A ;ERASE THE ERROR
01030      POP   HL ;RESTORE REGS.
01040      POP   DE
01050      POP   BC
01060      RST   10H ;GET NEXT CHR AFTER SOUND
01070      JP   BASINT ;CONTINUE PROCESSING LINE
01080 CMD  DEFM 'SOUND' ;COMMAND WORD
01090 OUTO POP   DE ;RESTORE FROM ABOVE
01100      POP   HL
01110 OUT1 POP   AF ;RESTORE FROM ABOVE
01120 OUT2 DEFS 3 ;RESERVED FOR ERROR RELAY
01130 TONE DEFS 2 ;RESERVED FOR FREQUENCY
01140      END

```

5

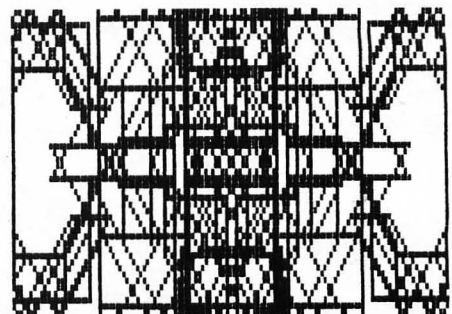
S-80 One Liners

```

1 CLS:DEFINT A-Z:RANDOM:X=32:Y=12:FOR K=1 TO 2560:IF INKEY$="R" THEN
1 ELSE L=RND(25):XD=RND(3)-2:YD=RND(3)-2:FOR A=1 TO L:SET(X,Y):SET(12
7-X,Y):SET(127-X,47-Y):SET(X,47-Y):X=X+XD:Y=Y+YD:X=X-128*INT(X/1
28):Y=Y-48*INT(Y/48):NEXT A,K:REM "R"=RESTART

```

Quentin Barnes
745 Valley St.
Chester, IL, 62233



The Computer Control Center

A NEAT COMPACT UNIT THAT DOES IT ALL . . .

Transforms your micro into a business machine

The Control Center is designed for your convenience, allowing you to set your monitor back to a comfortable viewing and working distance. It has six outlets on the back panel for all your power needs and six lighted switches on the front panel to give you finger-tip control of your computer & peripherals.

There is a specially filtered outlet for your computer, controlled by the 'CPU' switch.

This filter protects your computer from line noise, spikes and other local interference.

A control center

- Finger-tip switches for instant control.
- Lighted switches show which units are on.
- Rugged construction protects your equipment.
- A stable support for your monitor and disk drives.
- No more fumbling around the back for switches.
- Quality wood grained finish to compliment your office.

Price includes shipping in continental U.S.

\$159⁰⁰

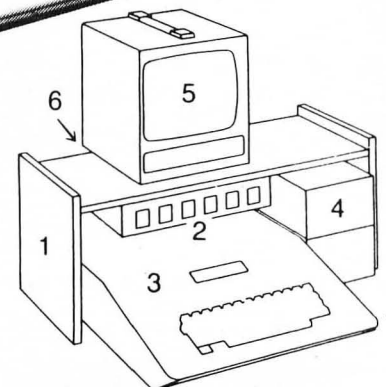


Industrial standards

- Six labeled, lighted switches.
- Advanced in-line filtered supply eliminates spikes, surges and lost data.
- Solid Silver switch contacts for long life and reliability.
- Twin fuse protection.
- Heavy duty power cable.
- The control center is designed for a total load of 1875 Watts.

A professional installation

- Transforms your office desk into a control center.
- Saves space and increases storage by removing clutter.
- Brings your Monitor to a comfortable eye level.
- Power only the units you require.
- Full width back panel hides those vulnerable trailing cables.



- | | |
|--------------------|----------------|
| 1 Control center | 4 Disk Drives |
| 2 Lighted Switches | 5 Monitor |
| 3 Micro Computer | 6 Rear Outlets |



Photo, Hutchins Photography Inc. Belmont



TSE HARDWARE



14 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

Music Editor

by Richard Lesh

enhancements by
Alan J. Zett

Music Editor is a music generator and editor for a 16K S-80, Model I or III.

This program will generate music, in one voice, of up to 600 notes per 16K of RAM in your system. The program consists of two separate parts. The first portion is in BASIC and is used to set up two string arrays that define the notes; this is the Music Editor. The second part of the program is in Machine Language, and is used to actually create the music; it consists of two routines which are called Zero and Music.

The beginning of the Music Editor program initializes variables and POKEs Zero and Music into high memory. You don't have to worry about setting "MEMORY SIZE" because the computer takes care of this for you. The program then proceeds to the main program (command loop). There are ten commands that the program recognizes in order to build a song file. They are as follows:

FILL is the command that is used to generate a new song file or add to the end of an existing file. The song file consists of arrays AD\$ and AF\$. The FILL command automatically begins with the lowest note number that is not used; i.e., if you had three notes in the file (numbers 0, 1, 2), the FILL command would begin with note number 3. To exit this command, simply press the ENTER key in response to the NAME OF THE NOTE inquiry.

The **INSERT** command is used to add a new note anywhere in the song file. If you decide not to insert, simply press ENTER in response to the BEFORE WHICH NOTE inquiry.



The **CHANGE** command will change the name and/or duration of any note. It will ask for the number of the note to be changed. If you wish to abort this procedure, simply press the ENTER key. If not, input the note number. The present name and duration of the note will then be printed, and the program will ask for the new note name. If you wish to change the name, enter the new name. If not, press ENTER to retain the old name. The program will then ask for the new duration. The same procedure for the name changes also apply for changes of duration.

The **DELETE** command is used to delete a specific range of notes. The beginning and ending notes of the segment to be deleted must be specified. The program will prompt you for this information. If you press ENTER in response to the BEGINNING NOTE inquiry, the program will default to a starting note of 0. If you press ENTER in response to the ENDING NOTE inquiry, the program will default to the last note of the file. Therefore if you

press ENTER in response to both inquiries, the entire file will be deleted.

VIEW is the command used to display the song file. If you press ENTER in response to the BEGINNING NOTE inquiry, the program will default to a beginning note of 0. To stop the scrolling of the display, simply press any key. Then press any key to resume the scrolling. If you press ENTER after the scrolling has been stopped, the program will return to the COMMAND inquiry.

SAVE will save the song file onto cassette or disk.

LOAD will load the song file from the cassette or disk.

ASSEMBLE is the command which converts each note name and duration into a number (0-65535) and then POKEs these two numbers into the table used by the Music routine. This command must be executed before the Music routine can generate music. If you press ENTER in response to the BEGINNING NOTE inquiry, the program will start assembling notes begin-

continued on next page

continued from previous page

ning with note number 0. If you press ENTER in response to the ENDING NOTE inquiry, the program will default to the last note of the song file. An ENTER response to the TEMPO inquiry will default to the last tempo entered or to the tempo loaded from a song file on tape or disk.

PLAY relinquishes control to the Music routine. This routine then generates the music, assuming that the notes have been assembled. If you respond to the BEGINNING NOTE inquiry by pressing ENTER, the program will start playing notes beginning with note number 0. You can stop the song by pressing the BREAK key.

The END command ends the program and re-enables the BREAK key.

Zero is the Machine Language routine used by the ASSEMBLE command to zero all memory locations after the ending note specified in the ASSEMBLE command.

Music is the major Machine Language routine. It is responsible for generating the tones. It uses data POKed into the memory area following the end of the Music routine. This is the table created by the ASSEMBLE command. Each note is represented by four bytes in the table. The first and second bytes contain the frequency value of the note; the third and fourth bytes contain its duration value. All data in the table are sequential.

Having finished this basic overview of the program, let's see how to use it. The first step is to find a piece of music that you wish to "teach" to the computer. Once you've done this, you need to build the song file. Run the program, and enter the name of your song when prompted. (This is also the name used when loading from tape or disk.) Now start FILLING the file. To enter the note's name, simply type (in response to the NAME OF THE NOTE inquiry) the note's name, followed by its octave number. If the note is a sharp, then type '#'; if it is a flat, type '/'. Rests are denoted by an 'R'.

The A above middle C is the beginning of the fourth octave. The octaves correspond to the octaves on a piano; hence, the lowest note would be 'A0' and the highest note would be 'C7'.

You can enter notes up to 'G9#', but the accuracy of the generator extends only to about 'E6'. You can enter any note by either of two proper names: 'C5#' could be called 'D5/', 'E5#' could be called 'F5', 'G5#' could be called 'A6/', etc. Following are the names of the notes for the chromatic scale starting with middle C:

- C3 or B3#
- C3# or D3/
- D3
- D3# or E3/
- E3 or F3/
- F3 or E3#
- F3# or G3/
- G3
- G3# or A4/
- A4
- A4# or B4/
- B4 or C4/
- C4 or B4#

To enter the note's duration, simply enter the first letter of the type of note when prompted for NOTE TYPE. The basic note types and their relative lengths are as follows:

T: Thirty-second note —	.125
S: Sixteenth note —	.25
E: Eighth note —	.5
Q: Quarter note —	1.0
H: Half note —	2.0
W: Whole note —	4.0
3E: Eighth-note triplet —	.333333
3Q: Quarter-note triplet —	.666666

If the note is to be dotted (one-half the value of the length of the note is added to it), simply add a period to the duration letter. If the note is double-dotted, add two periods to the duration letter. A dotted quarter note would be 'Q.', and 'H..' would be a double-dotted half note.

If you come across a note that does not conform to any of these standards, what do you do? If you should come across two eighth-note triplets tied together, simply enter '3Q'. That's right: Two eighth-note triplets add up to one quarter-note triplet, as can be seen from the numbers in the preceding table. The standard note is a quarter note, which equals 1; all of the other notes have duration values relative to this. An eighth-note triplet, then, is

equal to .333333; and two of them equal .666666, which is the duration value of a quarter-note triplet. What if you encounter a quarter note tied to an eighth note? Figure it out: $1 + .5 = 1.5$, which is a dotted quarter note ('Q.'). Another example would be a quarter note tied to a dotted eighth note. You would simply enter 'Q.', which is a double-dotted quarter note.

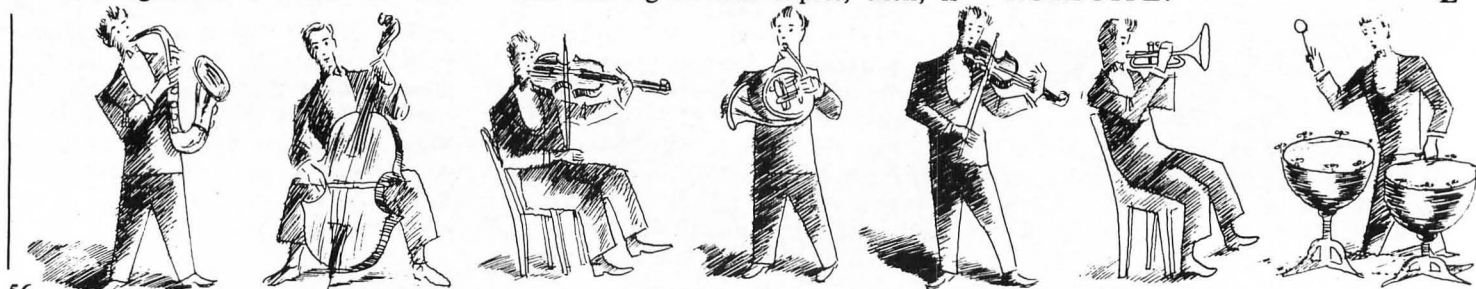
Here's a tough one: Suppose you must enter a sixteenth-note triplet. This is the time to make up your own notes. Just type 'I', for *Improvise*, and the value of the note's duration. Remembering that a quarter note is equal to 1, you would type 'I.1666666' to enter a sixteenth-note triplet. As a further example, 'I2.5' would be a half note tied to an eighth note.

You can modify the note duration by sounding only part of the note and then filling the remainder of the duration with a rest. There are three types of modifiers: staccato, marcato and legato. These modifiers are specified before the note duration letter. Their symbols, and the percentage of the note sounded to the percentage of rest added, are as follows:

Modifier	Symbol	Note #	Rest#
Staccato	.	.25	.75
Marcato	>	.50	.50
Legato	-	.75	.25

The staccato modifier would, of course, be used with staccato notes. The marcato modifier would be used when a distinct space between notes is required; for instance, accents, marcato notes, etc. The legato modifier is very handy when you have two notes with the same name that must be played in succession: The legato modifier will make a slight distinction between the two notes. Suppose you want to play four eighth notes on middle C. You want the passage to be smooth, but not run together. Simply enter the following via the FILL command:

```
NAME OF NOTE 0?           C3
NOTE TYPE?                 -E
NAME OF NOTE 1?           C3
NOTE TYPE?                 -E
NAME OF NOTE 2?           C3
NOTE TYPE?                 -E
```



NAME OF NOTE 3?
NOTE TYPE?

C3
E

If you ASSEMBLE these notes and then VIEW them, you will see that they differ from an assembly of unmodified notes. The ASSEMBLE command will insert an additional note, a rest, for every modified note. The name of each rest will be '?' instead of 'R', to tell you that the note before it is responsible for its presence. The value of the rest will be the numerical value of its duration, just as with improvised note durations. If you want to DELETE a modified note, you must also DELETE the rest following it (the '?' note). If you want to CHANGE a modified note to a different modified note, don't worry about the rest; but if you want to CHANGE the modified note to an unmodified note, you must DELETE the rest. The previous passage would appear as follows, after being ASSEMBLED:

0 NAME:C3	VALUE:-E
1 NAME:?	VALUE:.125
2 NAME:C3	VALUE:-E
3 NAME:?	VALUE:.125
4 NAME:C3	VALUE:-E
5 NAME:?	VALUE:.125
6 NAME:C3	VALUE:E

After you have entered a few measures of your song, ASSEMBLE them. This is advisable because the ASSEMBLE command will take a long time with large files. If you ASSEMBLE the song a few measures at a time, it will reduce the ASSEMBLE time.

The ASSEMBLE command will destroy any data in the table following the last note that you specify to be ASSEMBLED. Suppose that you have 21 notes in the song file. You then ASSEMBLE 0-20 and, after you've played it, you find that you have made a mistake with note 13. You then CHANGE note 13 and ASSEMBLE notes 13 to 13. You PLAY it again, only to find out that the song stops after it plays note 13. You should have ASSEMBLED notes 13 to 20 (just press ENTER for the ending note). The ASSEMBLE command does not affect any data in the table before the notes specified by the ASSEMBLE command, but it does zero all the data after

the specified notes. This is because a zero in the data table terminates the song. If the notes before the range that you want to ASSEMBLE have not already been ASSEMBLED, then the song will not PLAY because the Music routine will encounter a zero in the data table before it gets to the notes that you ASSEMBLED.

After you are satisfied with the first 21 notes, add a few more measures. Then ASSEMBLE the notes, from 22 to the last note entered (press ENTER before the ending note). PLAY the notes that you have ASSEMBLED, and CHANGE them if necessary. Remember that if you CHANGE, DELETE, or INSERT a note, you must reASSEMBLE every note from the CHANGED, DELETED or INSERTED note to the end of the song file. After you are satisfied with these notes, continue the process outlined above until you have completely entered the song.

When you ASSEMBLE a range of notes, the program will also ask you for a tempo at which to ASSEMBLE the notes. This tempo should be in quarter notes per minute. If the time signature is 4/4, 3/4 or 2/4, there is no problem; just enter the tempo marking. Suppose the time signature is 6/8 and the tempo is 90 dotted quarter notes per minute. What now? Simply forget about the time signature. If there are 90 dotted quarter notes (which equal 1.5) per minute, then there are 135 quarter notes per minute. The formula for this is to multiply the tempo by the value of the standard note for that song.

Once you have completed the task of "teaching" the computer a song, you may want to save it. Simply insert a blank tape into the cassette recorder and press the play and record buttons, or insert a system diskette into drive 0. Now enter the SAVE command. It will save the name of the song, the number of notes in the song file, and the last tempo entered in the ASSEMBLE command. Then it will proceed to save the entire song file. Likewise, the LOAD command will load in the song name, the number of notes, the tempo, and the song file. (If you are loading from tape, don't be alarmed if the asterisks don't flash while loading the

song file.) Any song that is already in the buffer before the LOAD command is executed will be destroyed.

Finally, here are two methods for allowing you to hear the music which you have generated.

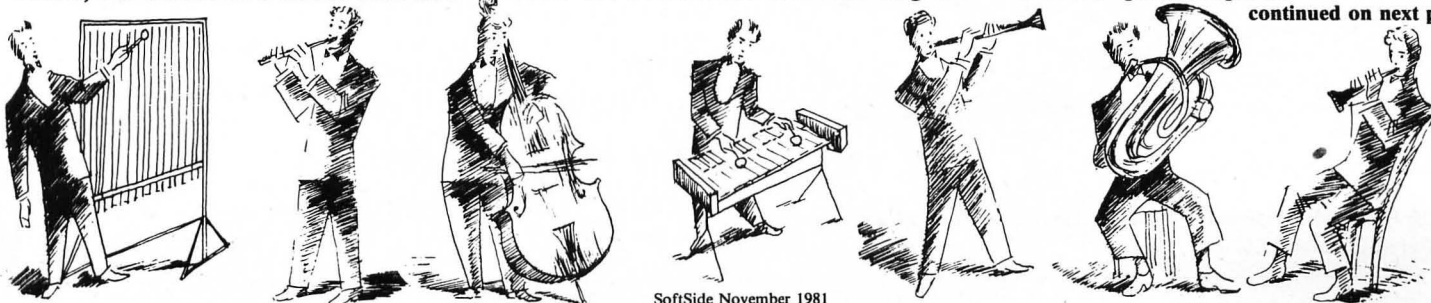
1. You can insert a blank tape into the tape recorder and then press the play and record buttons. Just before you use the PLAY command, remove the remote plug. This will start the tape recorder. The PLAY command will then record the music on the tape, and you can play back the tape to hear it.

2. Remove the remote and earphone plugs from the tape recorder. Connect an earphone or external amplifier to the earphone jack. You must then push in the tab sensor at the back left corner of the recorder's cassette slot and press the play and record buttons. This will enable you to hear the music when you PLAY it. For you real aficionados, connect the earphone jack to the AUX input of your stereo system and ENJOY!

VARIABLES

- A\$,A1\$,M,N,N1,NN,X,Y,Z: Miscellaneous.
- AD\$(n): Note duration array. Dimensioned to 600 times the number of 16K RAM blocks in the computer.
- AF\$(n): Note frequency array. Dimensioned to 600 times the number of 16K RAM blocks in the computer.
- AN\$: Song title. Used in loading a previously saved file.
- CMS\$: Command string. Contains first letter of command.
- DL: Duration loop counter.
- DM: Duration multiplier.
- DU: Duration.
- EX: Extra notes flag.
- FL: Frequency loop counter.
- FR: Frequency.
- LM: LSB of Music routine.
- LZ: LSB of Zero routine.
- MAX or MA: Maximum number of notes.
- MB: MSB of Music and Zero routines.
- NH: Number of highest note used.
- OC: Octave number.
- ST: Starting location for music table.
- TA: First table location.
- TE: Tempo in quarter notes per minute.
- TV: Tempo multiplier.

continued on next page



continued from previous page

Find top-of-memory, adjust memory size pointers, and disable the <BREAK> key.

```
5 X=3:POKE-1,0:IFPEEK(-1)<>OTHENX=2:POKE-16385,0:IFPEEK(-16385)
<>OTHENX=1
10 POKE16562,X#54+63:POKE16561,54:CLEAR3300:BR(1)=PEEK(16396):BR
(2)=PEEK(16397):POKE16396,175:POKE16397,201
```

Initialization.

```
20 CLS:PRINT@276,CHR$(23)"MUSIC EDITOR":PRINT@340,"VERSION 2.5"
:PRINT@400,"BY RICHARD LESH":PRINT@466,"AUGUST 25,1981":PRINT@5
32,"ENHANCEMENTS":PRINT@592,"BY ALAN J. ZETT":PRINT
30 DEFSTR:A:DEFINTM-D,U-Z:X=0:Y=0:Z=0:A="":A1="":N=0:M=0:N1=0:NN=
0:TE=0:TV=0:EX=0:ST=0:FR=0:DU=0:DL=0:FL=0:NH=-1:DC=0:DM=0:AN=""
40 X=(PEEK(16562)-63)/54:MB=PEEK(16562):MB=MB+256*(MB>128):LZ=PE
EK(16561)+12:TA=MB#256+LZ+98:LM=LZ+12:DIMAF(600*X),AD(600*X):MAX
=600*X-1
```

Poke Machine Language programs "Zero" and "Music"

```
50 DB=0:FORX=MB#256+LZ-9TOTA-1:READY:POKEX,Y:DB=DB+Y:NEXTX
```

Check for error in data lines.

```
60 IFDB<>12010THENCLS:PRINT"DATA BASE ERROR IN LINES 2000-2010,
CHECK LISTING.":PRINT:POKE16396,BR(1):POKE16397,BR(2):LIST2000-2
010
80 PRINT:INPUT"SONG TITLE":AN
90 CLS
```

Command loop.

```
100 PRINT"COMMAND? ";
110 CM#=INKEY$:IFCM#=""THEN10ELSEPRINTCM#
115 IFCM#="F"THEN250
120 IFCM#="D"THEN320
125 IFCM#="I"THEN310
130 IFCM#="C"THEN300
135 IFCM#="V"THEN260
140 IFCM#="A"THEN500
145 IFCM#="P"THEN550
150 IFCM#="S"THEN350
155 IFCM#="L"THEN380
160 IFCM#="E"THENPOKE16396,BR(1):POKE16397,BR(2):CLS:END
200 PRINT"F - FILL","D - DELETE","A - ASSEMBLE","S - SAVE","V -
VIEW","I - INSERT","P - PLAY","L - LOAD","E - END","C - CHANGE":
GOTO100
```

"Fill" subroutine.

```
250 NN=NH+1:IFNN>MATHEN100ELSEGOSUB1000:IFAF(NN)=""THENGOTO100:E
LSENN=NN:GOTO250
```

"View" subroutine.

```
260 IFNH=-1THENPRINT"NO TEXT IN BUFFER":GOTO100:ELSENN=0:PRINT"S
TARTING WITH WHICH":GOSUB1020:PRINTAN
270 GOSUB1030:NN=NN+1:IFNN>NHTHEN100
280 IFINKEY$=""THEN270
290 IN$=INKEY$:IFIN$=""THEN290ELSEIFIN$=CHR$(13)THEN100ELSE270
```

"Change" subroutine.

```
300 PRINT"WHICH":NN=NH+1:GOSUB1020:IFNN>NHTHENGOTO100ELSEGOSUB1
030:GOSUB1000:GOTO100
```

"Insert" subroutine.

```
310 IFNH=MATHENPRINT"CAN'T INSERT":GOTO100:ELSEPRINT"BEFORE WHIC
H":NN=NH+1:GOSUB1020:IFNN>NHTHEN100ELSEFORX=NHTONNSTEP-1:AF(X+1
)=AF(X):AD(X+1)=AD(X):NEXT:GOSUB1000:NH=NH+1:GOTO100
```

"Delete" subroutine.

```
320 NN=0:PRINT"STARTING":GOSUB1020:IFNN>NHTHEN340:ELSEN1=NN:PRI
NT"ENDING":NN=NH:GOSUB1020:IFNN>NHTHEN340ELSEIFNN<N1THEN340
330 Y=NN-N1+1:FORX=NN+1TONH:AF(X-Y)=AF(X):AD(X-Y)=AD(X):NEXT:FOR
Z=NH-Y+1TONH:AF(Z)=""AD(Z)=""NEXT:NH=NH-Y:IFNH=-1THENBOELSE100
340 PRINT"CAN'T DELETE THAT RANGE":GOTO100
```

"Save" subroutine.

```
350 CLS:PRINT"PRESS RECORD AND PLAY BUTTONS ON TAPE RECORDER.":P
RINT"PRESS <ENTER> WHEN READY TO SAVE ";AN."
355 POKE16396,BR(1):POKE16397,BR(2)
360 IFINKEY$<>CHR$(13)THEN360
365 PRINT#-1,NH,AN,TE:FORX=OTONHSTEP15
370 PRINT#-1,AF(X),AD(X),AF(X+1),AD(X+1),AF(X+2),AD(X+2),AF(X+3)
,AD(X+3),AF(X+4),AD(X+4),AF(X+5),AD(X+5),AF(X+6),AD(X+6),AF(X+7)
,AD(X+7),AF(X+8),AD(X+8),AF(X+9),AD(X+9),AF(X+10),AD(X+10),AF(X+
11),AD(X+11),AF(X+12),AD(X+12),AF(X+13),AD(X+13),AF(X+14),AD(X+
14)
375 NEXTX:POKE16396,175:POKE16397,201:GOTO90
```

"Load" subroutine.

```
380 CLS:PRINT"PRESS THE PLAY BUTTON ON THE TAPE RECORDER.":PRINT
"PRESS <ENTER> WHEN READY."
385 POKE16396,BR(1):POKE16397,BR(2)
390 IFINKEY$<>CHR$(13)THEN390
395 INPUT#-1,NH,AN,TE:FORX=OTONHSTEP15
400 INPUT#-1,AF(X),AD(X),AF(X+1),AD(X+1),AF(X+2),AD(X+2),AF(X+3)
,AD(X+3),AF(X+4),AD(X+4),AF(X+5),AD(X+5),AF(X+6),AD(X+6),AF(X+7)
,AD(X+7),AF(X+8),AD(X+8),AF(X+9),AD(X+9),AF(X+10),AD(X+10),AF(X+
11),AD(X+11),AF(X+12),AD(X+12),AF(X+13),AD(X+13),AF(X+14),AD(X+
14)
405 NEXT:POKE16396,175:POKE16397,201:CLS:PRINT"CLEARING MUSIC AR
RAYS":FORX=NH+1TOMA:AF(X)=""AD(X)=""NEXT:GOTO90
490 PRINT"CAN'T ASSEMBLE THAT RANGE":GOTO100
```

"Assemble" subroutine.

```
500 NN=0:PRINT"STARTING":GOSUB1020:IFNN>NHTHEN490:ELSEN1=NN:NN=N
H:PRINT"ENDING":GOSUB1020:IFNN>NHTHEN490
505 INPUT"TEMPO (QUARTER NOTES/MINUTE)":TE:TV=60/TE
510 EX=0:FORX=N1TONN:GOSUB1040:GOSUB1120:GOSUB1130:GOSUB1220:GOS
UB1230:NEXT
520 IFEX=0THENS30ELSEN1=X:NN=N1+EX-1:GOTO510
530 IFBR(1)=201THENPOKE16526,LZ:POKE16527,MB-256*(MB<0)ELSEDEFUS
R0=MB#256+LZ:CMD"T":POKE14308,0
540 ST=TA+4*X:Z=USR(ST):GOTO100
```

"Play" subroutine.

```
550 CLS:NN=0:PRINT"STARTING WITH WHICH":GOSUB1020:
560 IFBR(1)=201THENPOKE16526,LM:POKE16527,MB-256*(MB<0)ELSEDEFUS
R0=MB#256+LM:CMD"T":POKE14308,0
565 ST=TA+NN*4:CLS:PRINT"READY YOUR SOUND SYSTEM. PRESS <ENTER>
WHEN READY TO PLAY.":PRINTAN."
570 IFINKEY$<>CHR$(13)THENS70ELSEZ=USR(ST):GOTO100
```

Note and duration input subroutine.

```
1000 PRINTUSING"NAME OF NOTE ####";NN:INPUTAF(NN):IFAF(NN)=""TH
ENRETURN
```

```
1010 INPUT"NOTE TYPE";AD(NN):RETURN
```

Start and end input subroutine.

```
1020 INPUT"NOTE";NN:IFSGN(NN)=-1THENNN=0:RETURNELSERETURN
```

Display note subroutine.

```
1030 PRINTUSING"#### NAME: % % VALUE: % %";NN,AF(NN),AD(NN):RETURN
```

Frequency calculation subroutine.

```
1040 IFLEN(AF(X))<2THENFR=0:RETURNELSEDC=VAL(MID$(AF(X),2,1)):A=LEFT$(AF(X),1):IFLEN(AF(X))>2THENA=A+RIGHT$(AF(X),1)
1045 IFA="A"THENFR=27.5:GOTO1110
1050 IFA="A#"ORA="B/"THENFR=29.13524:GOTO1110
1055 IFA="B"ORA="C/"THENFR=30.86771:GOTO1110
1060 IFA="B#"ORA="C/"THENFR=32.7032:GOTO1110
1065 IFA="C#"ORA="D/"THENFR=34.64783:GOTO1110
1070 IFA="D"THENFR=36.7081:GOTO1110
1075 IFA="D#"ORA="E/"THENFR=38.89087:GOTO1110
1080 IFA="E"ORA="F/"THENFR=41.20344:GOTO1110
1085 IFA="E#"ORA="F/"THENFR=43.653529:GOTO1110
1090 IFA="F#"ORA="G/"THENFR=46.2493:GOTO1110
1095 IFA="G"THENFR=48.99943:GOTO1110
1100 IFA="G#"THENFR=51.91309:GOTO1110
1105 IFA="A/"THENFR=25.95654
1110 FR=FR*2:GOTO1110
```

"Frequency Loop" calculation subroutine.

```
1120 IFFR=0THENFL=0:RETURNELSEFL=FIX(ABS((1E6/FR-77.7903)/29.31229)+.5):RETURN
```

Duration calculation subroutine.

```
1130 IFAF(X)="?"THENDU=VAL(AD(X))*TV:RETURNELSEA=LEFT$(AD(X),1):A1=LEFT$(AD(X),2):IFASC(A)>64ORA="3"THENDM=1:GOTO1150
1135 IFA="-"THENDM=.75:ELSEIFA=">"THENDM=.5:ELSEIFA="."THENDM=.25:ELSEDM=1
1140 A=MID$(AD(X),2,1):A1=MID$(AD(X),2,2)
1150 IFA="T"THENDU=.125:GOTO1190
1155 IFA="S"THENDU=.25:GOTO1190
1160 IFA="E"THENDU=.5:GOTO1190
1165 IFA="Q"THENDU=1:GOTO1190
1170 IFA="H"THENDU=2:GOTO1190
1175 IFA="W"THENDU=4:GOTO1190
1180 IFA1="3E"THENDU=.333333:GOTO1190
1185 IFA1="3Q"THENDU=.666666:GOTO1190
1189 IFA="I"THENIFDM=1THENDU=VAL(RIGHT$(AD(X),LEN(AD(X))-1))ELSEDU=VAL(RIGHT$(AD(X),LEN(AD(X))-2))
1190 A=RIGHT$(AD(X),1):A1=RIGHT$(AD(X),2):IFA1=".."THENDU=DU*1.75ELSEIFA="."THENDU=DU*1.5
1200 IFAF(X+1)="?"THENAD(X+1)=STR$(DU*(1-DM)):DU=DU*DM*TV:RETURNELSEIFDM=1THENDU=DU*TV:RETURN
1210 IFNH=MATHENPRINT"CAN'T INSERT":GOTO100:ELSEFORV=NHTOXSTEP-1:AF(Y+1)=AF(Y):AD(Y+1)=AD(Y):NEXT:AF(X+1)="?:AD(X+1)=STR$(DU*(1-DM)):DU=DU*DM*TV:EX=EX+1:NH=NH+1:RETURN
```

"Duration Loop" calculation subroutine.

```
1220 IFFR=0THENDL=FIX(ABS((DU*1E6-129.527904)/237.8805)+.5):RETURNELSESDL=FIX(ABS(1E6/(FL*29.31229+77.7903))*DU-.0001240135)+.5):RETURN
```

Poke "Frequency Loop" count and "Duration Loop" count into memory table.

```
1230 ST=TA+4*X:M=FL/256:N=FL-256*M:GOSUB3000:POKEST,N:POKEST+1,M:M=DL/256:N=DL-256*M:GOSUB3000:POKEST+2,N:POKEST+3,M:RETURN
```

Poke data for Machine Language routine.

```
2000 DATAB2,105,99,104,32,76,101,115,104,205,127,10,54,0,35,124,254,0,32,248,201,205,127,10,229,221,225,221,94,2,221,86,3,122,179,200,58,64,56,254,4,200,221,126,0,221,182,1,32,12,27,62,30,71,16,254,122,179,32,246,24,35,221,78,0,221,70,1,62,1,211
2010 DATA255,11,120,177,32,251,221,78,0,221,70,1,62,2,211,255,11,120,177,32,251,27,122,179,32,221,221,35,221,35,221,35,221,35,24,176
```

Check for invalid tempo.

```
3000 IFN>255ORN<0ORM>255ORM<0THENPRINT"TEMPO ERROR, RE-ENTER":GOTO505ELSERETURN
```

Note: For disk systems, change the following lines.

```
350 CLS:PRINT"INSERT A SYSTEM DISK INTO DRIVE 0 WITH A LOT OF FREE SPACE":PRINT"PRESS <ENTER> WHEN READY TO SAVE FILE: ";AN"."
360 IFINKEY<>CHR$(13)THEN360ELSEOPEN"0",1,AN+"/SRC:0"
365 PRINT#1,NH,AN," ",TE:FORX=0TONH
370 PRINT#1,AF(X)," ",AD(X)
375 NEXTX:CLOSE:POKE16396,175:POKE16397,201:GOTO90
380 CLS:PRINT"INSERT THE SYSTEM DISK CONTAINING THE DESIRED FILE INTO DRIVE 0":PRINT"PRESS <ENTER> WHEN READY TO LOAD FILE: ";AN"."
390 IFINKEY<>CHR$(13)THEN390ELSEOPEN"I",1,AN+"/SRC:0"
395 INPUT#1,NH,AN,TE:FORX=0TONH
400 INPUT#1,AF(X),AD(X)
405 NEXT:CLOSE:POKE16396,175:POKE16397,201:CLS:PRINT"CLEARING MUSIC ARRAYS":FORX=NH+1TOMH:AF(X)="":AD(X)="":NEXT:GOTO90
```

WHEN YOU SPEND SO MUCH FOR A PRINTER, YOU SHOULD HAVE ONE THAT YOU CAN USE

Introducing....

THE IBM TOTAL PRINTER/TYPEWRITER



FEATURES:

- 10 and 12 Pitch
- Proportional Space
- Full Typewriter Use
- Auto Correcting
- Sound Cover
- "Smart" Keyboard

SPECIFICATIONS:

- 200 WPM Throughput
- Either Serial OR Parallel
- Self-Test
- Lowest On Site Maintenance
- IBM Backed Printer

Cables stocked for all APPLE, TRS (I, II, III), RS-232 systems.

PRICE:

ONLY

\$1995 (With 30 Day IBM Service Agreement)

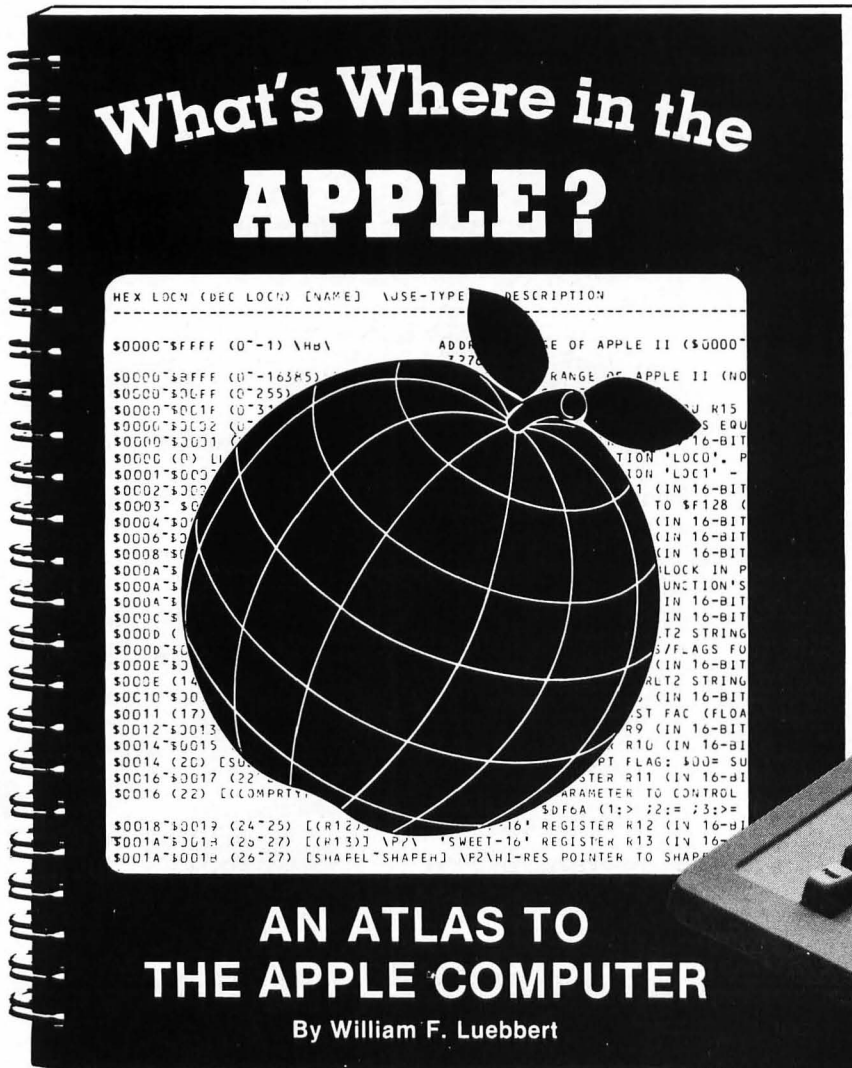
Dealerships Available

CONTACT: ICOM

11 N. MAIN LOMBARD, IL 60148

(312) 932-1766

The most important book ever published for the Apple.



The most comprehensive description of Apple II firmware and hardware ever published — all in one place.

What's Where in the Apple?

- Guides you — with a numerical Atlas and an alphabetical Gazetteer — to over **2,000** memory locations of **PEEKs**, **POKEs**, and **CALLs**.
- Gives names and locations of various **Monitor**, **DOS**, **Integer BASIC**, and **Applesoft** routines — and tells you what they're used for.
- Helps BASIC users to speed up their programs.
- Enables assembly language programmers to simplify coding and interfacing.

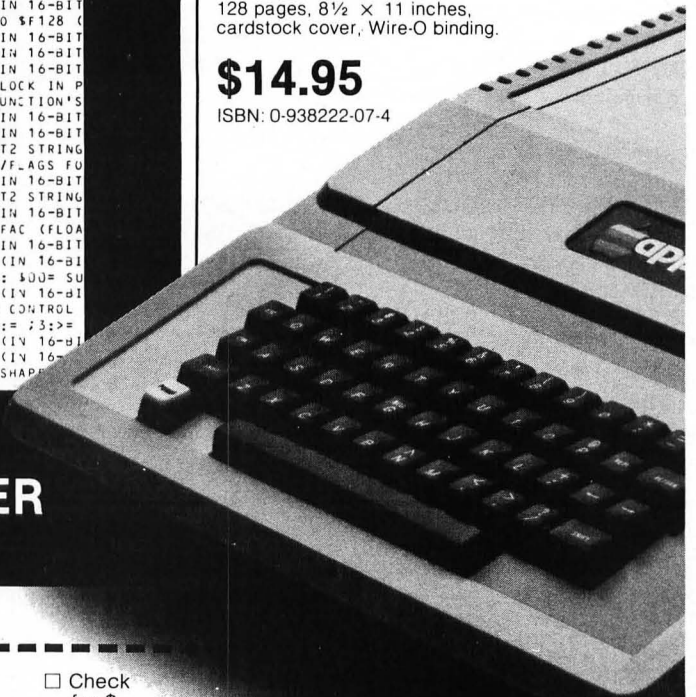
All Apple users will find this book helpful in understanding their machine, and essential for mastering it!

Ask for it at your computer store

128 pages, 8½ × 11 inches, cardstock cover, Wire-O binding.

\$14.95

ISBN: 0-938222-07-4



ORDER TOLL-FREE TODAY **800-227-1617** **EXT. 564**
 (in California 800-772-3545 Ext. 564)

Yes! Please send me _____ copies of *What's Where in the Apple?* at \$14.95 each (in U.S. plus shipping).

Check for \$ _____ enclosed. (Add \$2.00 surface shipping for each copy.) Massachusetts residents add 5% sales tax.

VISA MasterCard

Name _____

Acct. # _____ Expires _____

Address _____

City _____ State _____ Zip _____

Signature _____

MICRO INK, Inc., 34 Chelmsford Street, P.O. Box 6502, Chelmsford, MA 01824

Music Machine

by Jon Voskuil

Music Machine is a music composer/editor program for a 16K Apple with Applesoft.

This simple music editor is one that I originally wrote in order to hear and learn unfamiliar musical tunes. It sings in only one voice, but is easy to use and carries a tune well. A single main menu allows access to all the program's routines. These include: music entry; playing and editing of entered notes; saving and loading finished tunes using tape or disk; and listing of the actual number values which the computer has stored for the notes in memory. The program is entirely self-prompting.

Notes are entered in a shorthand form which specifies both the duration and the pitch of the note. For example, the characters "HC2" denote a half-note C in octave 2, and the characters ".EFS3" denote a dotted-eighth-note F-sharp in octave 3. This is a far less cumbersome entry technique than inputting the pitch and duration separately, although it makes the program more complex because it has to decode the string of characters. As an auditory check on the notes being typed in, each one is played as it is entered. The entire composition can be played at any time by returning to the menu and choosing the appropriate option.

All entered notes can be listed using the list/edit option, and any of them changed at that time. Each note is played (in a shortened form) as its shorthand notation is printed on the screen. If the screen fills with notes, the computer will automatically wait for you to press RETURN before continuing.

In developing other programs which use music, you may find it very helpful to enter the music using *Music*



Machine and then transplant the numbers stored by the computer into your own program. From the menu you can request a listing of the numbers which the computer has calculated for the pitch and duration of each note you have entered; these can then be copied into DATA statements. If the melody you want to transplant is a very long one, you may want to create a routine which will write DATA statements automatically — either printing them to the screen to be copied using the right-arrow key, or writing them to a disk file to be EXEC'd back into the other program.

The ampersand tone routine, which is POKEd into memory by the subroutine at line 8000, is one which has been published several times before in *SoftSide*. It exploits the fact that when Applesoft encounters the ampersand (&), it jumps to memory address 1013, where line 8040 has POKEd an instruction to jump to the tone generator at memory address 768. This is simpler to use than the alternate method of POKeing the pitch and duration values and then CALLing the proper address. Just use the format shown in line 50: the ampersand, followed by the letter T, followed by the note, then a comma, then the duration. The values can be variables, constants, or arithmetic expressions, in the range 0-255. The POKeing subroutine

(lines 8000-8040) is easily transplantable into your own programs. Its method of POKeing the Machine Language instructions is designed so that it won't interfere with any of your own program's DATA statements.

The arrays which hold the note and duration, NT% and D%, are dimensioned using the variable MAX (line 110). If 500 notes seems either excessive or inadequate, you can change the value of MAX accordingly. The size of MAX will influence how long it will take to STORE and RECALL tunes from tape, since the whole of both arrays will be written or read, no matter how few actual notes may be there. Notice that MAX must have the same value when notes are RECALLED from tape as when they were STORED. These comments do not apply to disk storage, where only the notes actually programmed are stored and retrieved, and the DIMensions are uncritical except that they must not be too small.

Many features could be added to *Music Machine*, including such niceties as a graphics display of the notes, an introductory fanfare (though that could get boring very quickly), and provisions for more complex types of notes such as triplets and tied notes. Have fun using and modifying it to suit your applications.

continued on next page

VARIABLES

A\$: String used for inputting and printing note shorthand.
 ADJ: Factor used to adjust octave number in the case of a B sharp or a C flat.
 BELL\$: Two Control-Gs to beep the speaker.
 C: Input variable used in menu selection.
 C\$: String used to contain an individual character of A\$ for analysis.
 D: Numeric value of note's duration.
 D\$: Control-D, for disk commands.
 D%: Used in STORE and RECALL commands to write and read D%array.
 D%(n): Duration value of note n.
 DOT: Factor used in computing note duration when the note is dotted.
 F\$: Name of disk file.
 FLAG: Determines whether note entry subroutine is to return to editing routine or continue with note entries.
 I: Loop variable.
 J: Index variable.
 K: Keeps track of current position in A\$ as it is being analyzed or synthesized.
 M: Loop variable.
 MAX: Maximum number of notes; dimension of NT% and D% arrays.
 MI: Loop variable.
 MOD: Modulo value.
 N: Note number.
 N1: Temporary holder for N.
 NN: Index variable.
 NN\$(n): Shorthand note names.
 NT: Numeric value of note's pitch.
 NT%: Used in STORE and RECALL commands to write and read NT% array.
 NT%(n): Pitch value of note n.
 NT(n): Pitch values corresponding to the 12 notes of the scale in octave 0.
 NUM: Number used as input to modulo subroutine.
 OCTV: Octave number.
 POK\$: Used to POKE in Machine Language sound routine.
 R: Value returned by modulo subroutine.
 T, TB: Tabbing variables.
 TEMPO: Relative speed at which music is played.
 X: Temporary variable.
 X\$: Input variable.

```

10 REM +-----+
11 REM !   MUSIC MACHINE !
12 REM !   !
13 REM ! BY JON VOSKUIL !
14 REM +-----+
15 REM
40 GOTO 100

Subroutine to play a note using the
ampersand to jump to the Machine
Language tone generator.

50 & TNT,D - (D * TEMPO - D) / B
    : RETURN

Mod function subroutine (finds
value of the number NUM modulo MOD,
returning the value as R).

60 R = INT ((NUM / MOD - INT (N
    UM / MOD)) * MOD): RETURN

Initialization.

100 HOME
110 MAX = 500
120 VTAB 5: HTAB 8: PRINT "M U S
    I C M A C H I N E": VTAB
    8: HTAB 13: PRINT "BY JON VO
    SKUIL"
130 GOSUB 8000
140 DIM D%(MAX),NT%(MAX),NT(12),
    NN$(12)
150 D$ = CHR$(4):BELL$ = CHR$(
    7) + CHR$(7)

Read the numerical values for the
notes, and the note name
abbreviations.

160 FOR I = 0 TO 12: READ NT(I):
    NEXT I
170 DATA 0,384,364,342,324,306,
    284,272,255,242,228,216,204
180 FOR I = 0 TO 12: READ NN$(I)
    : NEXT I
190 DATA R,C,CS,D,DS,E,F,FS,G,
    S,A,AS,B
200 VTAB 18: HTAB 8: INPUT "DO Y
    OU WANT INSTRUCTIONS?":X$
210 IF LEFT$(X$,1) < > "N" THEN
    GOSUB 9000

Main control menu.

1000 HOME : PRINT "WOULD YOU LIK
    E TO:"
1010 PRINT : PRINT " 1. PLAY TH
    E TUNE IN MEMORY"
    
```

```

1020 PRINT : PRINT " 2. ADD MOR
    E NOTES"
1030 PRINT : PRINT " 3. LIST/ED
    IT THE NOTES"
1040 PRINT : PRINT " 4. LIST CO
    DED NOTE TABLE"
1050 PRINT : PRINT " 5. START A
    NEW TUNE"
1060 PRINT : PRINT " 6. SAVE TU
    NE TO TAPE OR DISK"
1070 PRINT : PRINT " 7. LOAD TU
    NE FROM TAPE OR DISK"
1080 PRINT : PRINT " 8. SIGN OF
    F"
1090 PRINT : INPUT "(TYPE IN THE
    NUMBER) ";X$:C = VAL (X$)
1100 IF C < 1 OR C > 8 THEN 1090

1110 IF C = 8 THEN PRINT : PRINT
    "YOU MAY RE-ENTER PROGRAM WI
    TH MEMORY INTACT BY TYPIN
    G 'GOTO 1000'." : END
1120 ON C GOSUB 3000,2030,6000,7
    000,2000,4000,5000
1130 GOTO 1000

Routine to accept note input,
decode and check for legal values,
and store note and duration values.

2000 HOME : VTAB 5: PRINT "THIS
    OPTION WILL DESTROY ANY TUNE
    NOW IN MEMORY. DO YOU WA
    NT TO DO THIS?"
2010 PRINT : INPUT "ENTER 'Y' OR
    'N': ";X$: IF LEFT$(X$,1)
    < > "Y" THEN 2670
2020 N = 0

This is the entry point into this
routine if a note table is being
continued, rather than started from
scratch.

2030 HOME
2040 PRINT " 1. (DOT): (.)"
2050 PRINT " 2. DURATION: W, H,
    Q, E, S"
2060 PRINT " 3. NOTE OR REST: A
    -G, R"
2070 PRINT " 4. SHARP, FLAT: S,
    F"
2080 PRINT " 5. OCTAVE: 0-4 (GO
    -FS4)"
2090 PRINT : PRINT "ENTER CHARAC
    TERS WITHOUT SPACES. PRESS
    'RETURN' ALONE FOR MAIN MENU
    ."
    
```


Routine to load a tune from tape or disk into memory.

```
5000 HOME
5010 INPUT "LOAD TUNE FROM TAPE
OR DISK? (T/D) ";X$:X$ = LEFT$(X$,1)
5020 IF X$ = "T" THEN 5140
5030 IF X$ < > "D" THEN 5000
```

Load from disk.

```
5040 PRINT : INPUT "FILE NAME: "
;F$
5050 PRINT : INPUT "INSERT DISK
AND PRESS RETURN. ";X$
5060 PRINT D$;"OPEN";F$
5070 PRINT D$;"READ";F$
5080 INPUT NT$(0): INPUT D$(0):N
= NT$(0)
5090 FOR I = 1 TO N
5100 INPUT NT$(I): INPUT D$(I)
5110 NEXT I
5120 PRINT D$;"CLOSE";F$
5130 GOTO 5180
```

Load from tape.

```
5140 PRINT : INPUT "POSITION TAP
E, START PLAYING, AND
PRESS RETURN. ";X$
5150 PRINT : PRINT "LOADING. . .
(WAIT FOR 4 BEEPS)"
5160 RECALL NT$: RECALL D$
5170 N = NT$(0)
5180 RETURN
```

Routine to list and edit the notes in memory.

```
6000 HOME :TEMPO = 4
6010 T = 1: IF N = 0 THEN 1000
6020 FOR M = 1 TO N
```

Pause if the screen is full.

```
6030 IF M < > 81 AND M < > 161
AND M < > 241 AND M < > 321
AND M < > 401 AND M < > 481 THEN 6060
6040 PRINT : INPUT "PRESS 'RETUR
N' TO CONTINUE LISTING.";X$
6050 HOME
```

Interpret the numerical value of the duration, and start building the string A\$.

```
6060 D = D$(M):NT = NT$(M):NN = N
T
6070 DOT = 2:NUM = D:MOD = 3: GOSUB
60: IF R = 0 AND D < > 255 THEN
DOT = 3
6080 A$ = "": IF DOT = 3 THEN A$ =
","
6090 D = D / DOT
6100 DN LOG (D) / .693 - 1.99 GOTO
6110,6120,6130,6140,6150
6110 A$ = A$ + "S": GOTO 6160
6120 A$ = A$ + "E": GOTO 6160
6130 A$ = A$ + "Q": GOTO 6160
6140 A$ = A$ + "H": GOTO 6160
6150 A$ = A$ + "W"
```

Interpret the numerical value of the note, and continue building A\$.

```
6160 IF NT = 0 THEN A$ = A$ + "R
": GOTO 6220
6170 K = 0
6180 IF NN < 200 THEN NN = NN *
2:K = K + 1: GOTO 6180
6190 FOR I = 1 TO 12:X = ABS (N
/ NT(I)): IF X < 1.02 AND
X > .98 THEN J = I
6200 NEXT I
```

```
6210 A$ = A$ + NN$(J) + STR$(K)
Print the decoded note and play it.
```

```
6220 HTAB T: PRINT "#";M";";A$;
6230 GOSUB 50:T = T + 10: IF T <
41 THEN 6250
6240 T = 1: PRINT
6250 NEXT M
```

Change a note?

```
6260 PRINT : PRINT : PRINT "DO Y
DU WANT TO CHANGE ANY NOTES?
(ENTER)"
6270 PRINT "NOTE #, OR 'RETURN'
FOR NO CHANGE.) ";
6280 INPUT " ";X$
6290 N1 = N
6300 N = VAL (X$): IF N = 0 THEN
N = N1: GOTO 6340
6310 N = N - 1:FLAG = 1: GOSUB 21
10
6320 N = N1:FLAG = 0
6330 PRINT "NEXT CHANGE (NOTE #,
OR RETURN): "; GOTO 6280
6340 PRINT : RETURN
```

List the numerical values of the notes and durations in memory.

```
7000 TB = - 1:MI = 0
7010 HOME : IF N = 0 THEN 1000
7020 PRINT " NT DUR
NT DUR NT DUR";
7030 FOR I = 1 TO N:D = D$(I):NT
= NT$(I)
7040 MI = MI + 1
7050 TB = TB + 1:NUM = TB:MOD = 2
0: GOSUB 60: VTAB 3 + R: HTAB
1 + 14 * INT (TB / 20) + (M
I < 100) + (MI < 10)
7060 PRINT MI;" "; SPC( NT < 10
0);NT;" ";
7070 PRINT SPC( (D < 10) + (D <
100));D
7080 NUM = TB:MOD = 60: GOSUB 60:
IF R < > 59 THEN 7120
7090 TB = - 1
7100 VTAB 24: INPUT "PRESS RETUR
N TO CONTINUE LISTING";X$
7110 HOME : PRINT " NT DUR
NT DUR NT DUR"
;
7120 NEXT I
7130 VTAB 24: INPUT "PRESS RETUR
N FOR MENU";X$
7140 GOTO 1000
```

Subroutine to poke in the Machine Language tone generator.

```
8000 POK$ = "201,084,208,015,032,
177,000,032,248,230,138,072,
032,183,000,201,044,240,003,
076,201,222,032,177,000,032,
248,230,104,134,003,134,001,
133,000"
8010 FOR I = 1 TO 35: POKE I + 7
67, VAL ( MID$( POK$,I * 4 -
3,I * 4 - 1)): NEXT I
8020 POK$ = "170,160,001,132,002,
173,048,192,136,208,004,198,
001,240,007,202,208,246,166,
000,208,239,165,003,133,001,
198,002,208,241,096"
8030 FOR I = 1 TO 33: POKE I + 8
02, VAL ( MID$( POK$,I * 4 -
3,I * 4 - 1)): NEXT I
8040 POKE 1013,76: POKE 1014,0: POKE
1015,3: RETURN
```

Subroutine to print the instructions.

```
9000 HOME : VTAB 2: PRINT "THIS
PROGRAM ENABLES YOU TO INPUT
NOTES AND CREATE TUNES, WHI
CH CAN BE SAVED TO TAPE OR D
ISK."
```

continued on next page

continued from previous page

Set the text window to protect the instruction summary.

2100 POKE 34,9: HOME

Get the user's input.

2110 PRINT "NOTE #";N + 1;: INPUT
": ";A\$

If RETURN alone was pressed, then return to menu.

2120 IF A\$ = "" THEN TEXT: GOTO
2670

Check for valid entry length.

2130 IF LEN (A\$) > 1 AND LEN (
A\$) < 6 THEN 2300

2140 PRINT BELL\$;"INVALID ENTRY;
TRY AGAIN": GOTO 2110

Check for initial dot (.).

2300 DOT = 2: IF LEFT\$ (A\$,1) =
"." THEN DOT = 3

2310 K = DOT - 1: IF LEN (A\$) <
K THEN 2140

Interpret note value: whole, half,
quarter, eighth, sixteenth.

2320 C\$ = MID\$ (A\$,K,1): IF DOT =
3 AND C\$ = "W" THEN PRINT B
ELL\$;"SORRY, DOTTED WHOLE NO
TES NOT ALLOWED": GOTO 2110

2330 IF C\$ < > "W" AND C\$ < >
"H" AND C\$ < > "Q" AND C\$ <
> "E" AND C\$ < > "S" THEN
2140

2340 D = 127.5 * DOT

2350 IF C\$ = "H" THEN D = 64 * D
DT

2360 IF C\$ = "Q" THEN D = 32 * D
DT

2370 IF C\$ = "E" THEN D = 16 * D
DT

2380 IF C\$ = "S" THEN D = 8 * D
T

Interpret note type: C through B.

2390 K = K + 1: IF LEN (A\$) < K THEN
2140

2400 C\$ = MID\$ (A\$,K,1)

2410 NN = 0: IF C\$ = "R" THEN 262
0

2420 IF C\$ = "C" THEN NN = 1

2430 IF C\$ = "D" THEN NN = 3

2440 IF C\$ = "E" THEN NN = 5

2450 IF C\$ = "F" THEN NN = 6

2460 IF C\$ = "G" THEN NN = 8

2470 IF C\$ = "A" THEN NN = 10

2480 IF C\$ = "B" THEN NN = 12

2490 IF NN = 0 THEN 2140

Check for a sharp or flat.

2500 K = K + 1: IF LEN (A\$) < K THEN
2140

2510 ADJ = 0

2520 C\$ = MID\$ (A\$,K,1): IF C\$ <
> "S" AND C\$ < > "F" THEN
2580

2530 IF C\$ = "S" THEN NN = NN +
1: IF NN = 13 THEN NN = 1:AD
J = 1

2540 IF C\$ = "F" THEN NN = NN -
1: IF NN = 0 THEN NN = 12:AD
J = - 1

2550 K = K + 1

Interpret octave.

2560 IF LEN (A\$) < K THEN 2140

2570 C\$ = MID\$ (A\$,K,1)

2580 OCTV = VAL (C\$) + ADJ: IF O
CTV < 0 OR OCTV > 4 THEN 214
0

Calculate note value and check for
legal limits.

2590 NT = NT(NN) / 2 ^ OCTV:NT =
INT (NT + .5)

2600 IF NT > 255 THEN PRINT BEL
L\$;"SORRY, I CAN ONLY GO DOW
N TO G0": GOTO 2110

2610 IF NT < 17 AND NT < > 0 THEN
PRINT BELL\$;"SORRY, I CAN O
NLY GO UP TO FS4": GOTO 2110

2620 N = N + 1

Play the entered note.

2630 D%(N) = D:NT%(N) = NT:TEMPO =
1: GOSUB 50

Print warning if approaching
maximum number of notes.

2640 IF N = MAX - 11 THEN PRINT
BELL\$;"JUST TEN NOTES TO GO.
.."

2650 IF N = MAX THEN 1000

If the variable FLAG is true (=1),
then this subroutine has been called
by the EDIT routine, and control
should return there without further
note entries.

2660 IF NOT FLAG THEN 2110

2670 RETURN

Routine to play the tune in memory.

3000 PRINT : PRINT : INPUT "PLEA
SE CHOOSE TEMPO (1-8) ";X\$:T
EMPO = VAL (X\$): IF TEMPO <
1 OR TEMPO > 8 THEN 3000

3010 FOR M = 1 TO N

3020 D = D%(M):NT = NT%(M): GOSUB
50

3030 NEXT M

3040 RETURN

Routine to save the tune in memory
to tape or disk.

4000 HOME

4010 IF N = 0 THEN 4200

4020 NT%(0) = N

4030 INPUT "SAVE TUNE TO TAPE OR
DISK? (T/D) ";X\$:X\$ = LEFT\$
(X\$,1)

4040 IF X\$ = "T" THEN 4170

4050 IF X\$ < > "D" THEN 4000

Save to disk.

4060 PRINT : INPUT "FILE NAME: "
;F\$

4070 PRINT : INPUT "INSERT DISK
AND PRESS RETURN. ";X\$

4080 PRINT D\$;"OPEN";F\$

4090 PRINT D\$;"DELETE";F\$

4100 PRINT D\$;"OPEN";F\$

4110 PRINT D\$;"WRITE";F\$

4120 FOR I = 0 TO N

4130 PRINT NT%(I): PRINT D%(I)

4140 NEXT I

4150 PRINT D\$;"CLOSE";F\$

4160 GOTO 4200

Save to tape.

4170 PRINT : INPUT "POSITION TAP
E, START RECORDING, AND
PRESS RETURN. ";X\$

4180 PRINT : PRINT "SAVING. . .
(WAIT FOR 4 BEEPS)"

4190 STORE NT%: STORE D%

4200 RETURN

9010 PRINT : PRINT "THE FOLLOWING ARE EXAMPLES OF THE NOTATION USED IN ENTERING NOTES:
"

9020 PRINT : HTAB 9: PRINT "HC2 .EFS3"

9030 PRINT : PRINT "IN THE FIRST EXAMPLE, THE 'H' INDICATES A HALF NOTE, THE 'C' INDICATES THE NOTE"

9040 PRINT "ON THE SCALE, AND THE '2' INDICATES THE OCTAVE."

9050 PRINT : PRINT "THE LEADING PERIOD (.) IN THE SECOND EXAMPLE INDICATES A DOTTED NOTE, IN THIS CASE AN EIGHTH ('E'). THE REMAINING"

9060 PRINT "CHARACTERS INDICATE AN F SHARP IN THE 3RD OCTAVE."

9070 PRINT : PRINT : INPUT "PRESS 'RETURN' TO CONTINUE. ";X\$

9080 HOME

9090 PRINT "HERE IS A COMPLETE NOTATION SUMMARY:"

9100 PRINT : PRINT : PRINT "1. OPTIONAL LEADING DOT"

9110 PRINT : PRINT "2. A LETTER INDICATING NOTE DURATION:"

9120 PRINT " W = WHOLE (DOT NOT ALLOWED)"

9130 PRINT " H = HALF
E = EIGHTH"

9140 PRINT " Q = QUARTER
S = SIXTEENTH"

9150 PRINT : PRINT "3. A LETTER INDICATING THE NOTE, A-G, OR A REST, R. (IF A REST, THIS IS THE LAST CHARACTER TO ENTER.)"

9160 PRINT : PRINT "4. A LETTER FOR SHARP OR FLAT (S,F)"

9170 PRINT : PRINT "5. A NUMBER (0-4) INDICATING THE OCTAVE (RANGE IS GO THROUGH FS4)"; PRINT

9180 PRINT : PRINT "A BRIEF SUMMARY OF THESE INSTRUCTIONS WILL BE DISPLAYED DURING NOTE ENTRY."

9190 PRINT : INPUT "PRESS 'RETURN' TO BEGIN. ";X\$

9200 RETURN



GALACTIC CHASE™

The aliens have swept undefeated across the galaxy. You are an enterprising star ship captain—the final defender of space.

As the aliens attack, you launch a deadly barrage of missiles. Flankers swoop down on your position. Maneuvering to avoid the counterattack, you disintegrate their ships with your magnetic repellers.

As your skill improves, the attackers increase their speed. And as a last resort, the aliens use their invisible ray to slow the speed of your missile launcher.

GALACTIC CHASE provides Atari owners with the most challenging one or two person game in the galaxy.



Atari 400/800 16k. Written in machine language. Requires joysticks.
Payment: Personal Checks—allow three weeks to clear.

American Express, Visa, & Master Charge—include all numbers on card. Please include phone number with all orders. 24.95 for cassette or 29.95 for disk plus 2.00 shipping. Michigan residents add 4%.

Check the dealer in your local galaxy. Dealer inquiries encouraged.
Galactic Chase © 1981 Stedek Software.

SPECTRUM
COMPUTERS

Dept S.
26618 Southfield
Lathrup Village, MI. 48076
(313) 559-5252

WE MIGHT BE GOOD FOR EACH OTHER

SoftSide Publications is constantly on the lookout for those with the special skills necessary to enhance our growing family of publications. A leader in the field of BASIC software publishing for the Apple™, Atari™ and TRS-80™, **SoftSide** is currently expanding its search to include editors, machine language programmers, researchers, librarians and marketing people. The creative environment at **SoftSide** is such that multi-skilled people have every opportunity offered to them for career fulfillment. Computer literacy is a clear advantage (though not a requirement) as is a grasp of communication techniques. **SoftSide Publications** speak to the technical end of microcomputer entertainment, its creation and innovation. We provide the surroundings for people to grow as far as they want. All they have to have is the motivation.

Although New Hampshire wages are not the highest available, we offer a better scale of pay than most of our competitors and have a goal-oriented bonus program. We have excellent fringe benefits and New Hampshire has no personal income or sales taxes.

Milford, New Hampshire, is in the beautiful southern area of the state, only an hour's drive from Boston, the Atlantic Ocean and the White Mountains. The state's two largest cities, Manchester and Nashua are only fifteen minutes away. Ski slopes abound in the region, as do fine restaurants, arts and crafts centers and the beautiful countryside of New England.

If you are a skilled person with experience in the fields of computers, entertainment or magazines, and if you enjoy a relaxed country lifestyle with the convenience and cultural opportunities of major urban centers nearby, let us know. **We might be good for each other!** A formal resume isn't necessary, just send a letter outlining your skills, qualifications and experience to:

Randal L. Kottwitz
SoftSide Publications
6 South Street
Milford, NH 03055



Music Programmer

by John Rush Elkins

Music Programmer is a music editing program for a 24K Atari (32K with disk).

The use of this program will be explained while progressing through an example. The musical scale "DO, RE, MI" is shown being entered as a demonstration program, with eighth notes, in four voices. The last note in the measure is shown being entered into the Enter Music routine.

MUSIC PROGRAMMER

NOTES AND RESTS ARE ENTERED AND EDITED MEASURE BY MEASURE IN 1 TO 4 VOICES USING A ROUTINE CALLING FOR NOTE, DURATION, OCTAVE, AND LOUDNESS. MEASURE LENGTH IS DETERMINED FROM THE TIME SIGNATURE. TIED NOTES, DOTTED NOTES, AND TRIPLETS ARE ENTERED WITH SUBSCRIPTS. TIED MEASURES ARE COMBINED INTO ONE LARGE MEASURE. SHARPS AND FLATS ARE ENTERED WITH THEIR NOTES SINCE THERE ARE NO KEY SIGNATURES. EACH MEASURE MAY BE PLAYED FOR EDITING BEFORE SAVING.

EACH MEASURE IS THEN SAVED BY TRANSFERRING TO DATA STATEMENTS WHICH MUST BE ENTERED INTO THE PROGRAM.

PRESS RETURN TO CONTINUE.

The *Music Programmer* uses standard musical notation to generate music which can be enjoyed on its own or entered into other BASIC programs. No more than minimal musical and programming skills are required. However, the ability to manipulate the editing features of the Atari is required to enter the computer-generated DATA statements.

The *Music Programmer* utilizes musical measures as its "book-keeper" to insure that the notes start and stop at the proper place. If a note needs to be continued into the next measure, the two measures are tied (—) by the note into a double



measure which is entered as one large measure.

WHAT IS THE MAXIMUM NUMBER OF NOTES — UP TO 16 — YOU WILL NEED TO ENTER INTO ANY SINGLE VOICE IN ANY MEASURE? MEASURES TIED TOGETHER WITH CONTINUING NOTES ARE TREATED AS ONE LARGE MEASURE.
?12

This value (V0 in line 818) is required to DIMension the arrays to hold the notes. The maximum number of notes to be entered can be increased (lines 818, 819, 814); but the smaller the value, the more memory is made available for saving the notes in DATA statements.

HOW MANY VOICES (1 TO 4) WILL YOU NEED
?4

The number of voices needed is determined by the number of notes that must be played at any one time. Use of only one voice, playing the melody (top) line of notes, will require less memory and can be entered faster, but will not give the fullness of sound of which the Atari is capable. Musical notation often does not show rests for all voices, so be sure to take care to fill out blank spaces in the music with rests. It may also be necessary to add rests at the beginning of the first measure and at the end of the last measure. The computer requires that a measure of music be entered in each voice selected.

TIME SIGNATURE?
WHICH? 1 FOR 4/4
2 FOR 3/4 OR 6/8
3 FOR 2/4
4 FOR 2/2

?1

The time signature gives the measure length with the 4/4 (or C) and 2/2 (or C) times being equivalent in length to a whole note (four counts). The 2/2 time, as used in the *Music Programmer*, plays twice as fast as the 4/4 time. The 3/4 and 6/8 times are equivalent in length to three-fourths of a whole note (three counts) and the 2/4 time to a half note (two counts).

TEMPO? FAST TO SLOW
WHICH? 8 9 10 11 12
WITH 9 OR 11: NO DOTTED 16th NOTES AND NO 16th OR DOTTED 8th NOTES WITH 2/2 TIME.

WITH 10 OR 12: NO 16th NOTES WITH 2/2 TIME.

WITH 9 OR 12: TRIPLETS ARE ENTERED NORMALLY WITH TEMPOS 9 AND 12 AND AS TWO DOTTED AND ONE NORMAL NOTE OF THE NEXT FASTER SPEED WITH OTHER TEMPOS. NO TRIPLET 16th NOTES WITH 9.
?10

The tempo is precisely controlled by the computer with the values (8-12) representing the duration of a sixteenth note in 60ths of a second. Notes with durations of less than 8/60ths of a second are not possible because of the time required to change notes. The menu indicates which durations are ex-

continued on next page

continued from previous page

cluded, either for being too fast or for giving fractional values with dotted and triplet notes. Care should be taken in selecting a tempo which will fit the music at hand. For music with triplets, you should try to use tempos 9 or 12. For example, triplet eighth notes (♩) with a total duration of a quarter note can be entered normally as three E3 notes in tempos 9 and 12, but must be entered as two dotted sixteenth notes (S.) and one sixteenth note (S) with other tempos.

WHAT IS THE TITLE OF YOUR SONG?

CENTER IN 11th SPACE
WITH FIRST LINE HERE
AND SECOND LINE NEXT.
USE UPPER AND lower
CASE & INVERSE video
? DO,RE
CONTINUE TITLE HERE.
? mi

There are two lines provided for the title of your song. Your title may go on either or both of them. Each line should be centered on the eleventh space. Inverse video and lower-case letters are displayed as upper-case letters with different colors. Be sure to change back to normal upper-case letters.

ENTER TOTAL NUMBER OF COMBINED MEASURES.

HOW MANY? 1 FOR NO COMBINED MEASURES.

2, 3, 4 COMBINED MEASURES.

LIMIT 12 NOTES PER VOICE.

?1

More measures can be combined by increasing V0 in lines 819 and 814 so that more notes can be entered, and increasing CM in lines 997 and 998; but such an increase would require more memory to reserve array space, and it would be more difficult to edit the measure in case of an entry error.

The following entries can be made for the DO,RE,mi demonstration program with eighth notes in four voices:

VOICE 0: C,E,SM,R,Q.,G,E,4,MF,R,Q.

VOICE 1: R,E,D,E,4,MF,R,Q.,A,E,4,MF,R,Q

VOICE 2: SM,E,E,4,MF,R,Q.,B,E,4,MF,R,E

VOICE 3: R,Q.,F,E,4,MF,R,Q.,C,E,5,MF

The last note entered in the measure is shown being entered into the Enter Music routine.

ENTER MUSIC

THIS IS MEASURE 1

NUMBER OF AVERAGE MEASURES LEFT =59

VOICE 3: ENTRY 4

NOTE?

WHICH? AF,A,AS

BF,B

C,CS

DF,D,DS

EF,E,ES

F,FS

GF,G,GS

R FOR REST

ENTER SM FOR SAME NOTE - R,Q.,OCTR,OFF
?C

The desired notes (A-G) and rests (R), with sharps (S) and flats (F), are entered here. The octave on the musical staff is entered from the menu after the next. The range of the Atari is over three full octaves, with one note in a fourth octave, as illustrated in Figure 1. Since the bass range is not very low, it is often necessary to enter notes in a higher octave than that in which they are written.

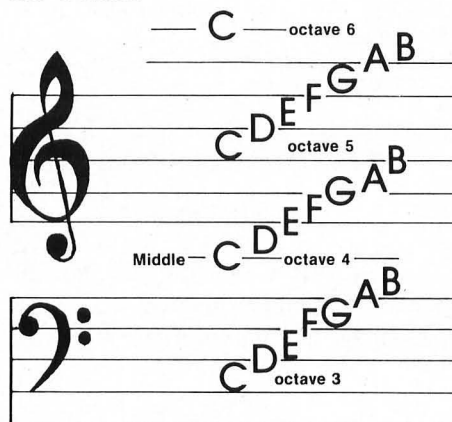


Figure 1

The range of music played by the Atari.

Notice the prompts with the measure number, voice number and entry number which will help you keep track of where you are. The voice will change automatically when the measure is

completed in that voice, as indicated by a bell. You should also pay attention to the bottom prompt, for there are many times when you can enter SM for the same note, especially the same octave and loudness.

VOICE 3: ENTRY 4, NOTE C

TOTAL DURATION IN MEASURE AND PROGRAM

VOICE 0 = 160 160

VOICE 1 = 160 160

VOICE 2 = 160 160

VOICE 3 = 140 140

DURATION OF NOTE OR REST?

WHICH? SIXTEENTH

EIGHTH

QUARTER

HALF

WHOLE

ADD . FOR DOTTED NOTES

ADD 3 FOR TRIPLET NOTES

ADD T FOR TIED NOTES

ENTER ED TO EDIT NOTE C

ENTER SM FOR SAME NOTE - C,Q.,OCTR,OFF

?E

The notations for the allowed durations of notes are S (♩), E (♩), Q (♩), H (♩), W (∞), and of rests are S (♩), E (♩), Q (♩), H (∞), W (∞). Dotted notes such as "E." are 3/2 as long as a normal note and triplets such as E3 are 2/3 as long as a normal note. Triplets can only be entered with tempos 9 and 12; otherwise enter the triplet notes as two dotted and one normal note of the next faster duration. The duration of tied notes such as QT is followed up by the duration of the note to which it is tied, for example, "E.". No T should be added to the last of the tied notes. Tied notes are combined into one note with a total duration equal to the sum of the tied notes. Note that you cannot use SM to enter the note or duration after entering a tied note. The duration of the note entered with SM will be the duration of the tied note and not that of the note in the SM display, which shows only the last duration added to the tied note. You should also pay attention to the ED prompt which allows you to edit the note.

VOICE 3: ENTRY 4, NOTE C, DURATION E

OCTAVE? 4 GOES UP FROM MIDDLE C

WHICH? 3, 4, 5, 6

ENTER ED TO EDIT DURATION E



?SM FOR SAME NOTE - C,E,OCTR,OFF
?5

The octave menu was discussed along with the note menu.

VOICE 3: ENTRY 4, NOTE C, DURATION E, OCTAVE 5

LOUDNESS?
WHICH? PP,P,MP
MF,F,FF
OFF

NORMAL VALUES:
MF FOR MELODY
P FOR ACCOMPANIMENT

ENTER ED TO EDIT OCTAVE 5
?MF

There is much room for experimentation with loudness, but a value of MF for the melody (top) line of notes and P for the accompanying notes is a good starting place. The assigned values of loudness are PP=2, P=4, MP=6, MF=8, F=10, FF=12, and OFF=0. These can be adjusted from 1-15 in lines 1184 and 1188. Loudness can also be controlled with the Play Music routine (lines 30, 91) as explained below. Loudness can only be Edited by editing the entire measure.

If it should happen that you enter a wrong note and cannot get out of that voice, it is possible to BREAK and GOTO 5001 to reenter the entire measure in that voice.

MEASURE FULL

PLAY MEASURE OR
EDIT MEASURE OR
SAVE MEASURE
WHICH?

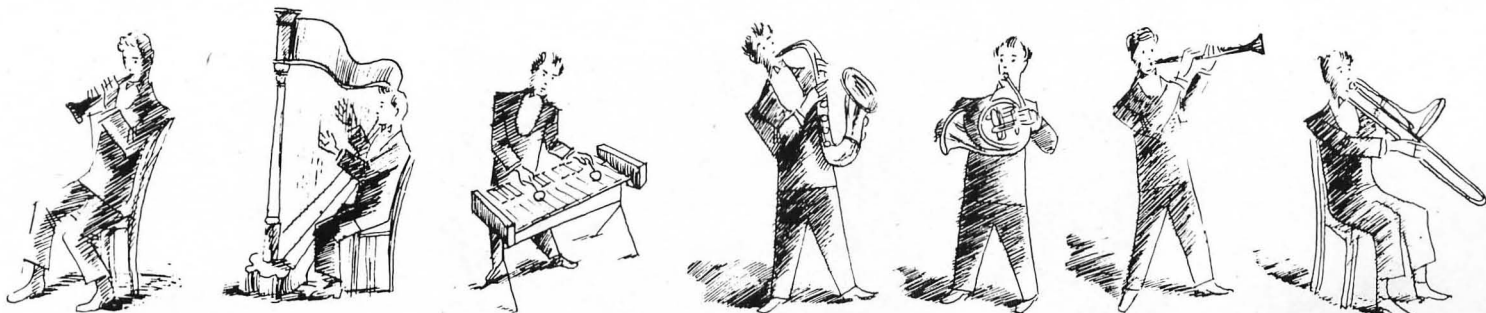
ENTER 1 TO PLAY
2 TO EDIT
3 TO SAVE

The PLAY option lets you listen to the measure of music to check how it sounds before editing or saving it.

EDIT MEASURE OF MUSIC

VOICE?
WHICH? 0,1,2,3
?0

EDIT VOICE 0



SAVE MUSIC

MOVE CURSOR TO LINE NUMBER (LN) AND PRESS RETURN TO ENTER INTO PROGRAM. THEN TYPE CONT AND PRESS RETURN TO CONTINUE.

LN
40 ?#6: ?#6; " DO, RE": ?#6: ?#6; "mi"
STOPPED AT LINE 6514

CONT

The first time through the Save Music routine, the title can be entered into the program for the Play Music routine; and if there are fewer than four voices, certain "blank" lines are entered to remove lines not needed in the Play Music routine. Failure to enter the "blank" lines will result in incorrect play. If you make a mistake in manipulating the edit keys and fail to enter a line number correctly, you can return from the Play Music menu to re-enter the line numbers.

LN
101 DATA 121,8,20,0,0,20,0,0,40,
0,0,60,0,0,60,108,8,20,0,0,60,96,8,20
102 DATA 0,0,60,91,8,20,81,8,20,
0,0,60,0,0,60,72,8,20,0,0,40,64,8,20
103 DATA 0,0,20,60,8,20
104 DATA 256,0,0

ENTER LINE NUMBERS AND CONTINUE
STOPPED AT LINE 7580

The notes have been placed in order of play in DATA statements which **must** be entered into the program for the Play Music routine. Each note READ by the Play Music routine requires three values: note, loudness and duration. Line 104 includes only the "256 flag" in Voice 0 which signals the END of play. Line 104 will be replaced by new DATA when saving the next measure and a new "256 flag" will END the play of both measures, etc. It may be helpful to keep a record of which line numbers correspond to which measures, so that you can manually make changes in the DATA statements if the music doesn't sound right.

PLAY MUSIC

PLAY MUSIC OR
RETURN FOR NEW MEASURE OR
SAVE ON DISKETTE (CASSETTE) OR

RE-ENTER LINE NUMBERS WHICH?

ENTER 1 TO PLAY
2 TO RETURN
3 TO SAVE
4 TO RE-ENTER

?3

The Play Music menu allows you to play all the measures entered, return to enter a new measure, or re-enter line numbers. (The menu for the cassette version is included in parentheses and can be entered into the Music Programmer by replacing lines 31, 8450, 8452, 8453, 8454 and 8469 with the statements in lines 9031, 9450, 9452, 9453, 9454 and 9469.)

The music can be SAVED to disk (or cassette) when completed or when all the memory is used up. The music, once saved, can be compiled with other music (if more than one SAVE is needed) by entering one and then the other, or can be entered into another BASIC program. Be careful not to have line numbers between 39 and 101 end up in the BASIC program since they will be wiped out upon entering the music program. Since the DATA statements are READ faster when they are closer to the beginning of the program, it may be advantageous to GOTO 500 at the program's start and then GOSUB 39 to PLAY MUSIC.

SAVE ON DISKETTE (CASSETTE)

SAVE PLAY PROGRAM AND DATA WITH THE INITIAL GROUP OF MEASURES BUT ONLY DATA IN SUBSEQUENT SAVES.

INSERT BLANK DISKETTE! (Omitted in cassette version)

ENTER LINE NUMBER AND CONTINUE

LN

8460 LIST "D:SONG1.
ENT",39,104(8460LIST"C",39,104)

STOPPED AT LINE 8453

PROGRAM BEING SAVED
(ON BELL, SET RECORDER TO RECORD AND PRESS RETURN.)

ENTER LINE NUMBER TO CLEAR MEMORY FOR NEXT GROUP OF MEASURES.

LN
101
102

continued on next page

103
104

ENTER LINE NUMBERS AND CONTINUE
STOPPED AT LINE NUMBER 8465

For an additional SAVE, GOTO 8450 before entering the "blank" lines to free memory for a new group of measures.

LOAD OR ADD THIS MUSIC PROGRAM AND ADDITIONAL DATA TO ANOTHER PROGRAM WITH ENTER "D:SONG#.ENT" (ENTER "C")

On the second save of the Play program and DATA with the initial group of measures, change 101 to 39 when entering line 8460 with SONG2.ENT.

The Play Music routine (lines 30-91) and the Play/Edit Music routine (lines 700-765, with subroutines at lines 12-20) depend on the ability of the Atari to time itself to 1/60th of a second. The music is timed by POKEing 19 and 20 with 0 (lines 45 and 712), and PEEKing at the timer (PEEK 19*256 + PEEK 20 at lines 55 and 722) each time through the play cycle. All voices are sounded simultaneously (lines 50 and 720) each time through the play cycle, and are turned off with a perceptible pause (lines 81-84 and 751-754) each time there is a note change (lines 55-80 and 722-750). The notes are changed, voice by voice, whenever the time equals or exceeds the time when the previous note should stop playing.

Each voice may be played separately. For example, to listen to Voice 0 (the melody), LIST 50,80 and insert L1=0, L2=0 and L3=0 after the READ statements in lines 60, 70 and 80. Changes in loudness to values other than 0 can also be made.

Though very conservative parameters for the duration of the fastest note have been used in the Music Programmer, it is conceivable that with some songs there may be an occasional mis-ordering of notes because of the extra time required to go through each cycle of the note-ordering routine. If

this happens, the "256 flag" probably will not be READ by Voice 0 and the program will "squawk" instead of ENDing. It can be corrected by hitting the BREAK key, entering END, GOing TO 30, and manually reordering the DATA, using the values for the notes given in the *Atari BASIC Reference Manual*.

Note: Following the program listing are two song files. To save them type LIST "D:FILENAME.EXT" or LIST "C" for disk and cassette respectively. To load, type ENTER "D:FILENAME.ENT" or ENTER "C". To listen to them, first load the Music Programmer, ENTER the desired song, RUN, hit BREAK and type "GOTO 30".

VARIABLES

A\$: Answer name.
B\$: Bell character.
C: Column in array.
C\$: Clear screen character.
CK0-CK3: Checks voices 0-3. 0 = on, 1 = off.
CM: Consecutive measures.
D: Duration number.
D0-D3: Duration of note in voices 0-3.
DL: Duration length.
DM: Duration of measure.
DT: Duration of tied note.
D\$: Musical name of duration.
D(0,0)-D(3,0): Total duration of voices 0-3 in current measure.
D(0,1)-D(3,1): Total duration of voices 0-3.
ENT: Entry number.
H0-H3: Hold a note in voices 0-3. 0 = hold, 1 = no hold.
L: Loudness number.
L0-L3: Loudness of voices 0-3.
LN: Line number.
LN1: Line number at start of DATA.
LNF: Line number at end of DATA.
LNS: Line number at which "saving" should start.

L\$: Musical name of loudness.
ML: Amount of memory left.
MN: Measure number.
MNT: Measure number total.
MV: Measure value.
MVA: Measure value average. Computed in line 1000.
MVT: Measure value total.
N: Note number.
NLV: Note limit value.
NM: Number of measures.
NV: Number of voices.
NS: Number of notes saved.
N0-N3: Notes being played by voices 0-3.
N\$: Musical name of note.
N(N,O): Note value.
N0(V0,0)-N0(V3,0): Order in measure of notes V0-V3 in voices 0-3.
N0(V0,1)-N0(V3,1): Value of note in voices 0-3.
N0(V0,2)-N0(V3,2): Loudness of note in voices 0-3.
N0(V0,3)-N0(V3,3): Duration of note in voices 0-3.
O: Octave number.
ORD: Order of note in measure.
O\$: Musical name for octave.
PES: Play, Edit, Save.
PRS: Play, Return, Save.
P0-P3: Turn off play in voices 0-3. 0 = on, 1 = off.
R: Row in array.
RELN: Re-enter line number.
R0-R3: Row in voices 0-3.
SN: Song number.
ST: Stop from re-entering title. 0 = OK to enter, 1 = not OK.
SV: Save in DATA statements. 0 = off, 1 = on.
T0-T3: Time to stop note in voices 0-3.
TM: Time of play.
TS: Time signature.
TITLE\$: Title of song.
V: Voice number.
V0-V3: Number of note in voices 0-3.
V0E-V3E: Highest number of note in voices 0-3.
VN: Value of notes.
V\$: Name of voice.
WT: Wait for message. Change this value in line 800 to change length of wait.




```

0 REM #MUSIC PROGRAMMER-REVISION 1.1
1 REM # COPYRIGHT 1981
2 REM #
3 REM # JOHN RUSH ELKINS
4 REM # 2100 JEFFERSON STREET
5 REM #BLUEFIELD, WEST VIRGINIA 24701
6 REM # AUGUST 3, 1981
7 REM #
10 GOTO 800
Play/Edit music subroutine.
12 NO=N0(V0,1):IF NO=256 THEN NO(V0,0)
=ORD:POP :GOTO 760
14 L0=N0(V0,2):D0=N0(V0,3):T0=T0+D0:NO
(V0,0)=ORD:ORD=ORD+1:RETURN
16 N1=N1(V1,1):L1=N1(V1,2):D1=N1(V1,3)
:T1=T1+D1:N1(V1,0)=ORD:ORD=ORD+1:RETUR
N
18 N2=N2(V2,1):L2=N2(V2,2):D2=N2(V2,3)
:T2=T2+D2:N2(V2,0)=ORD:ORD=ORD+1:RETUR
N
20 N3=N3(V3,1):L3=N3(V3,2):D3=N3(V3,3)
:T3=T3+D3:N3(V3,0)=ORD:ORD=ORD+1:RETUR
N
Play music.
30 ? C#: ? " **PLAY MUSIC**":? :
? "PLAY MUSIC OR":? :? "RETURN FOR NEW
MEASURE OR"
31 TRAP 30: ? :? "SAVE ON DISKETTE OR":
? :? "RE-ENTER LINE NUMBERS":? :? "WHI
CH?":? :? "ENTER 1 TO PLAY"
32 ? " 2 TO RETURN":? " 3 TO
SAVE":? " 4 TO RE-ENTER":INPUT P
RS
33 IF PRS=1 THEN 39
34 IF PRS=2 AND NM<=0 THEN ? B#: ? " NO
MORE MEASURES!":FOR M=1 TO MT:NEXT M:
GOTO 30
35 IF PRS=2 THEN 990
36 IF PRS=3 THEN 8450
37 IF PRS=4 THEN ST=0:LN1=RELN:GOTO 65
00
38 ? B#:GOTO 30
39 RESTORE 101:GRAPHICS 2+16: ? #6:PRIN
T #6;" Play music"
42 L0=0:N0=0:L1=0:N1=0:L2=0:N2=0:L3=0:
N3=0:P0=0:P1=0:P2=0:P3=0
45 TM=0:T0=0:T1=0:T2=0:T3=0:POKE 19,0:
POKE 20,0
50 SOUND 0,N0,10,L0:SOUND 1,N1,10,L1:S
OUND 2,N2,10,L2:SOUND 3,N3,10,L3
55 TM=PEEK(19)*256+PEEK(20):IF TM>=T0
THEN TM=T0:P0=1:READ NO,L0,D0:T0=T0+D0
:IF NO=256 THEN 90
60 IF TM>=T1 THEN TM=T1:P1=1:READ N1,L
1,D1:T1=T1+D1
70 IF TM>=T2 THEN TM=T2:P2=1:READ N2,L
2,D2:T2=T2+D2
80 IF TM>=T3 THEN TM=T3:P3=1:READ N3,L
3,D3:T3=T3+D3

```

```

81 IF P0=1 THEN SOUND 0,0,0,0:P0=0
82 IF P1=1 THEN SOUND 1,0,0,0:P1=0
83 IF P2=1 THEN SOUND 2,0,0,0:P2=0
84 IF P3=1 THEN SOUND 3,0,0,0:P3=0
85 GOTO 50
90 TRAP 91:FOR V=0 TO 3:SOUND V,0,0,0:
NEXT V:GOTO 30
91 END
Play/Edit music.
700 ? C#: ? " **MEASURE FULL**"
: ? :? "PLAY MEASURE OR":? :? "EDIT MEA
SURE OR":? :? "SAVE MEASURE"
701 TRAP 700: ? :? "WHICH?":? :? "ENTER
1 TO PLAY":? " 2 TO EDIT":? "
3 TO SAVE":INPUT PES
702 IF PES=1 THEN 708
703 IF PES=2 THEN 5000
704 IF PES=3 THEN SV=1:GOTO 708
705 ? B#:GOTO 700
708 GRAPHICS 2+16: ? #6:PRINT #6;"
Play music"
709 ? #6: ? #6;TITLE1#: ? #6: ? #6;TITLE2
#
710 L0=0:N0=0:L1=0:N1=0:L2=0:N2=0:L3=0
:N3=0:P0=0:P1=0:P2=0:P3=0:ORD=0
712 V0=-1:V1=-1:V2=-1:V3=-1:T0=0:T1=0:
T2=0:T3=0:POKE 19,0:POKE 20,0
720 SOUND 0,N0,10,L0:SOUND 1,N1,10,L1:
SOUND 2,N2,10,L2:SOUND 3,N3,10,L3
722 TM=PEEK(19)*256+PEEK(20):IF TM>=T0
THEN TM=T0:P0=1:V0=V0+1:GOSUB 12
725 IF H1=0 THEN 751
730 IF TM>=T1 THEN TM=T1:P1=1:V1=V1+1:
GOSUB 16
735 IF H2=0 THEN 751
740 IF TM>=T2 THEN TM=T2:P2=1:V2=V2+1:
GOSUB 18
745 IF H3=0 THEN 751
750 IF TM>=T3 THEN TM=T3:P3=1:V3=V3+1:
GOSUB 20
751 IF P0=1 THEN SOUND 0,0,0,0:P0=0
752 IF P1=1 THEN SOUND 1,0,0,0:P1=0
753 IF P2=1 THEN SOUND 2,0,0,0:P2=0
754 IF P3=1 THEN SOUND 3,0,0,0:P3=0
755 GOTO 720
760 FOR V=0 TO 3:SOUND V,0,0,0:NEXT V:
IF SV=1 THEN SV=0:GOTO 6500
765 GOTO 700
Initialize music.
800 H1=0:H2=0:H3=0:V1=0:V2=0:V3=0:R0=3
:R1=0:R2=0:R3=0:DIM A*(3),B*(1),C*(1):
B#=CHR*(253):C#=CHR*(125):WT=400
802 ? C#: ? " **MUSIC PROGRAMME
R**":? :? "NOTES AND RESTS ARE ENTERED
AND EDITED";
804 ? "MEASURE BY MEASURE IN 1 TO 4 VO
ICES USING A ROUTINE CALLING FOR NOT

```

```

E, DURATION,OCTAVE,";
806 ? " AND LOUDNESS. MEASURELENGTH IS
DETERMINED FROM THE TIME SIGNATURE
. TIED NOTES, DOTTED NOTES,"
807 ? "AND TRIPLETS ARE ENTERED WITH":
? "SUBSCRIPTS. TIED MEASURES ARE COMBI
NED INTO ONE LARGE MEASURE.";
809 ? " SHARPS AND FLATS ARE ENTERED
WITH THEIR NOTES SINCE THERE ARE
NO KEY SIGNATURES."
810 ? "EACH MEASURE MAY BE PLAYED FOR
EDITING BEFORE SAVING."
812 ? :? "EACH MEASURE IS THEN SAVED B
Y":? "TRANSFERRING TO DATA STATEMENTS
WHICH MUST BE ENTERED ";
813 ? "INTO THE PROGRAM.":? :? "PRESS
RETURN TO CONTINUE.":INPUT A#
814 ? C#: ? "WHAT IS THE MAXIMUM NUMBER
OF NOTES-UP TO 16-YOU WILL NEED TO ENT
ER INTO ANY"
816 ? "SINGLE VOICE IN ANY MEASURE? ME
ASURES TIED TOGETHER WITH CONTINUING N
OTES ARE TREATED AS ONE ";
818 TRAP 814: ? "LARGE MEASURE.":INPUT
V0:NLV=V0
819 IF V0<1 OR V0>16 THEN ? B#: ? " 1 T
O 16 NOTES PER MEASURE.":FOR M=1 TO MT
:NEXT M:GOTO 814
820 TRAP 820: ? C#: ? "HOW MANY VOICES (
1 TO 4)WILL YOU NEED":INPUT NV
821 ON NV GOTO 826,825,824,823
822 ? B#:GOTO 820
823 H3=1:V3=V0:R3=R0
824 H2=1:V2=V0:R2=R0
825 H1=1:V1=V0:R1=R0
826 ? C#: ? :? "TIME SIGNATURE":? "WHI
CH? 1 FOR 4/4":? " 2 FOR 3/4 OR
6/8":? " 3 FOR 2/4"
827 ? " 4 FOR 2/2":TRAP 826:INPU
T TS
828 IF TS<1 OR TS>4 THEN ? B#:GOTO 826
829 IF TS=2 THEN TS=0.75
830 IF TS=3 THEN TS=0.5
833 ? C#: ? "TEMPO? FAST TO SLOW":? "WH
ICH? 8 9 10 11 12":? :? "WITH 9 OR 11:
NO DOTTED 16th NOTES AND NO 16th";
834 ? " OR DOTTED 8th NOTES WITH 2/2
TIME.":? :? "WITH 10 OR 12: NO 16th NO
TES WITH 2/2 TIME.":?
835 ? "WITH 9 OR 12: TRIPLETS ARE ENTE
RED NORMALLY WITH TEMPOS 9 AND 12 A
ND AS TWO DOTTED AND ONE NORMAL";
836 TRAP 833: ? " NOTE OF THE NEXT FAST
ER SPEED WITH OTHER TEMPOS. NO TRIPLE
T 16th NOTES WITH 9.":? :INPUT DL
837 IF DL<8 OR DL>12 THEN ? B#:GOTO 83
3
838 DL=DL*16

```

continued on next page

continued from previous page

```

9452 ? :? "ENTER LINE NUMBER AND CONT
INUE.":?
9453 ? " LN ":? "8460 LIST ";CHR$(34);
"C";CHR$(34);";LNS;";LN1:STOP
9454 ? C#:?:? "ON BELL, SET RECORDER
TO RECORD AND PRESS RETURN."
9469 ? "ADDITIONAL DATA TO ANOTHER PRO
GRAM WITH ENTER "C""

```

The Entertainer

```

39 RESTORE 101:GRAPHICS 2+16:? #6:PRIN
T #6;" PLay muSic"
40 ? #6:? #6;" THE":? #6:? #6;"
ENTERTAINER"
42 L0=0:N0=0:L1=0:N1=0:L2=0:N2=0:L3=0:
N3=0:P0=0:P1=0:P2=0:P3=0
45 TM=0:T0=0:T1=0:T2=0:T3=0:POKE 19,0:
POKE 20,0
50 SOUND 0,N0,10,L0:SOUND 1,N1,10,L1:S
OUND 2,N2,10,L2:SOUND 3,N3,10,L3
55 TM=PEEK(19)*256+PEEK(20):IF TM>=T0
THEN TM=T0:P0=1:READ N0,L0,D0:T0=T0+D0
:IF N0=256 THEN 90
60 IF TM>=T1 THEN TM=T1:P1=1:READ N1,L
1,D1:T1=T1+D1
70 IF TM>=T2 THEN TM=T2:P2=1:READ N2,L
2,D2:T2=T2+D2

```



duplicating service

307 West Main Street
Maple Shade, NJ 08052

(609) 667-1667

- AMP "Data-sette" blank cassettes for digital use
- Cassette Storage Boxes
- Cassette Labels - Custom printing & blank
- Custom Record Album production from your tapes
- Stereo and Spoken Word cassette duplication

Call or write to:

Jerry

for more information.

All cassette work at
AMP R. & D. is custom work
to fit your needs.

```

80 IF TM>=T3 THEN TM=T3:P3=1:READ N3,L
3,D3:T3=T3+D3
81 IF P0=1 THEN SOUND 0,0,0,0:P0=0
82 IF P1=1 THEN SOUND 1,0,0,0:P1=0
83 IF P2=1 THEN SOUND 2,0,0,0:P2=0
84 IF P3=1 THEN SOUND 3,0,0,0:P3=0
85 GOTO 50
90 TRAP 91:FOR V=0 TO 3:SOUND V,0,0,0:
NEXT V:GOTO 30
91 END
101 DATA 0,0,72,0,0,96,0,0,96,0,0,96,1
08,8,12,102,8,12
102 DATA 96,8,12,243,4,24,0,0,24,0,0,1
92,60,8,24,121,4,24,162,4,24,96,8,12
103 DATA 60,8,24,162,4,24,0,0,24,96,8,
12,121,4,24,136,4,24,60,8,72,182,4,24
104 DATA 0,0,24,121,4,24,144,4,24,193,
4,24,0,0,24,60,8,12,53,8,12,121,4,24
105 DATA 162,4,24,50,8,12
106 DATA 47,8,12,162,4,24,0,0,24,0,0,9
6,60,8,12,53,8,12,96,4,24,121,4,24
107 DATA 47,8,24,162,4,24,0,0,24,64,8,
12,53,8,24,91,4,24,128,4,24
108 DATA 60,8,72,96,4,24,121,4,24,0,0,
96,162,4,24,0,0,72,162,4,24,108,8,12
109 DATA 0,0,24,102,8,12
110 DATA 96,8,12,243,4,24,0,0,24,0,0,1
92,60,8,24,121,4,24,162,4,24,96,8,12
111 DATA 60,8,24,162,4,24,0,0,24,96,8,
12,121,4,24,136,4,24,60,8,84,182,4,24
112 DATA 0,0,24,121,4,24,144,4,24,193,
4,24,0,0,48,72,8,12,204,4,24,81,8,12
113 DATA 85,8,12,121,4,36,144,4,36,217
,4,36,72,8,12,60,8,12,47,8,24,121,4,60
114 DATA 144,4,60,217,4,60,53,8,12,60,
8,12,72,8,12
115 DATA 53,8,72,91,4,72,128,4,24,162,
4,24,162,4,24,0,0,72,144,4,24,108,8,12
116 DATA 0,0,24,128,4,24,102,8,12
117 DATA 96,8,12,243,4,24,0,0,24,0,0,1
92,60,8,24,121,4,24,162,4,24,96,8,12
118 DATA 60,8,24,162,4,24,0,0,24,96,8,
12,121,4,24,136,4,24,60,8,72,182,4,24
119 DATA 0,0,24,121,4,24,144,4,24,193,
4,24,0,0,24,60,8,12,53,8,12,121,4,24
120 DATA 162,4,24,50,8,12
121 DATA 47,8,12,162,4,24,0,0,24,0,0,9
6,60,8,12,53,8,12,96,4,24,121,4,24
122 DATA 47,8,24,162,4,24,0,0,24,64,8,
12,53,8,24,91,4,24,128,4,24
123 DATA 60,8,72,96,4,24,121,4,24,0,0,
96,162,4,24,0,0,72,162,4,24,60,8,12
124 DATA 0,0,24,53,8,12
125 DATA 47,8,12,243,4,24,0,0,24,0,0,9
6,60,8,12,53,8,12,162,4,24,193,4,24
126 DATA 47,8,24,136,4,24,0,0,24,60,8,
12,53,8,12,162,4,24,193,4,24,60,8,12
127 DATA 47,8,12,144,4,24,0,0,24,0,0,9

```

```

6,60,8,12,53,8,12,182,4,24,243,4,24
128 DATA 47,8,24,153,4,24,0,0,24,60,8,
12,53,8,12,182,4,24,243,4,24,60,8,12
129 DATA 47,8,12,162,4,24,0,0,24,0,0,9
6,60,8,12,53,8,12,193,4,24,243,4,24
130 DATA 47,8,24,162,4,24,0,0,48,64,8,
12,53,8,24,128,4,24
131 DATA 60,8,72,96,4,72,243,4,24,0,0,
96,0,0,72,0,0,24,0,0,24
132 DATA 256,0,0

```

Row, Row, Row Your Boat

```

39 RESTORE 101:GRAPHICS 2+16:? #6:PRIN
T #6;" PLay muSic"
40 ? #6:? #6;" ROW,row,row":? #6:?
#6;" your BOAT"
42 L0=0:N0=0:L1=0:N1=0:L2=0:N2=0:L3=0:
N3=0:P0=0:P1=0:P2=0:P3=0
45 TM=0:T0=0:T1=0:T2=0:T3=0:POKE 19,0:
POKE 20,0
50 SOUND 0,N0,10,L0:SOUND 1,N1,10,L1:S
OUND 2,N2,10,L2:SOUND 3,N3,10,L3
55 TM=PEEK(19)*256+PEEK(20):IF TM>=T0
THEN TM=T0:P0=1:READ N0,L0,D0:T0=T0+D0
:IF N0=256 THEN 90
60 IF TM>=T1 THEN TM=T1:P1=1:READ N1,L
1,D1:T1=T1+D1
70 IF TM>=T2 THEN TM=T2:P2=1:READ N2,L
2,D2:T2=T2+D2
80 IF TM>=T3 THEN TM=T3:P3=1:READ N3,L
3,D3:T3=T3+D3
81 IF P0=1 THEN SOUND 0,0,0,0:P0=0
82 IF P1=1 THEN SOUND 1,0,0,0:P1=0
83 IF P2=1 THEN SOUND 2,0,0,0:P2=0
84 IF P3=1 THEN SOUND 3,0,0,0:P3=0
85 GOTO 50
90 TRAP 91:FOR V=0 TO 3:SOUND V,0,0,0:
NEXT V:GOTO 30
91 END
101 DATA 121,8,36,0,0,144,0,0,144,0,0,
144,121,8,36,121,8,27,108,8,9,96,8,36
102 DATA 96,8,27,0,0,144,0,0,144,0,0,1
44,108,8,9,96,8,27,91,8,9,81,8,72
103 DATA 60,8,12,121,4,36,0,0,144,0,0,
144,60,8,12,60,8,12,81,8,12,121,4,36
104 DATA 81,8,12,81,8,12,96,8,12,121,4
,27,96,8,12,96,8,12,108,4,9,121,8,12
105 DATA 96,4,36,121,8,12,121,8,12
106 DATA 81,8,27,96,4,27,0,0,144,0,0,1
44,91,8,9,108,4,9,96,8,27,96,4,27
107 DATA 108,8,9,91,4,9,121,8,72,81,4,
72
108 DATA 121,8,36,60,4,12,121,4,36,0,0
,144,60,4,12,60,4,12,121,8,36,81,4,12
109 DATA 121,4,36,81,4,12,81,4,12,121,
8,27,96,4,12,121,4,27,96,4,12,96,4,12
110 DATA 108,8,9,108,4,9,96,8,36,121,4
,12,96,4,36,121,4,12,121,4,12
111 DATA 96,8,27,81,4,27,96,4,27,0,0,1

```

```

839 IF TS=4 THEN TS=1:DL=DL*0.5
840 DIM N0(V0,R0),N1(V1,R1),N2(V2,R2),
N3(V3,R3),N(11,3),D(3,1)
842 DIM TITLE1$(20),TITLE2$(20),V$(1),
0$(2),N$(2),D$(3),L$(3)

849 ? C$?:? :? "WHAT IS THE TITLE OF
YOUR SONG?":? :? "CENTER IN 11th SPAC
E":? "WITH FIRST_LINE HERE"
850 ? "AND SECOND_LINE NEXT.":? "USE U
PPER AND lower":? "CASE & INVERSE vide
o.":INPUT TITLE1$
855 ? :? "CONTINUE TITLE HERE.":INPUT
TITLE2$
910 RESTORE 920:FOR D=0 TO 3:FOR N=0 T
O 11:READ VN:N(N,D)=VN:IF VN=256 THEN
980
920 DATA 243,230,217,204,193,182,173,1
62,153,144,136,128,121,114,108,102,96,
91,85,81,76,72,68,64,60,57,53
930 DATA 50,47,45,42,40,37,35,33,31,29
,0,256
940 NEXT N:NEXT D
980 MVT=44#NV:MN=0:MNT=0:DT=0:SV=0:ST=
0:LN1=101:LNS=39:SN=1
984 FOR R=0 TO 1:FOR C=0 TO 3:D(C,R)=0
:NEXT C:NEXT R
985 ML=FRE(0)
990 FOR V=0 TO V0:FOR R=0 TO R0:N0(V,R
)=0:NEXT R:NEXT V
991 FOR V=0 TO V1:FOR R=0 TO R1:N1(V,R
)=0:NEXT R:NEXT V
992 FOR V=0 TO V2:FOR R=0 TO R2:N2(V,R
)=0:NEXT R:NEXT V
993 FOR V=0 TO V3:FOR R=0 TO R3:N3(V,R
)=0:NEXT R:NEXT V
994 V0=0:V1=0:V2=0:V3=0:V0E=0:CK0=0:CK
1=0:CK2=0:CK3=0:DN=DL#TS:RELN=LN1
995 FOR C=0 TO 3:D(C,0)=0:NEXT C
996 ? C$?: "ENTER TOTAL NUMBER OF COMB
INED MEASURES.":? :? "HOW MANY?
1 FOR NO COMBINED MEASURES."
997 ? " 2,3,4 COMBINED MEASUR
ES.":? :? "LIMIT ";NLV;" NOTES PER VOI
CE.":TRAP 996:INPUT CM
998 IF CM<1 OR CM>4 THEN ? B$:GOTO 996
1000 DN=DN#CM:MN=MN+CM:MNT=MNT+CM:NV=M

```

```

L-FRE(0):ML=FRE(0):MVT=MVT+MV:MVA=MVT/
MN:MN=INT(ML/MVA+0.5)-1
1001 IF H1=0 THEN CK1=1
1002 IF H2=0 THEN CK2=1
1003 IF H3=0 THEN CK3=1
1004 D$="4":D=1:N$="C":N=0:D$="0":D=DL
/4:L$="MF":L=8

```

Enter music.

```

1005 IF CK0=1 AND CK1=1 AND CK2=1 AND
CK3=1 THEN V=0:D=3:N=2:D=0:L=0:GOTO 15
00
1010 ? C$?:? :? " **ENTER MUSI
C**":? :? "THIS IS MEASURE ";MNT:?"NU
MBER OF AVERAGE MEASURES LEFT = ";NM
1015 IF CK0=0 THEN V=0:V$="0":ENT=V0+1
:GOTO 1022
1017 IF CK1=0 THEN V=1:V$="1":ENT=V1+1
:GOTO 1022
1019 IF CK2=0 THEN V=2:V$="2":ENT=V2+1
:GOTO 1022
1021 IF CK3=0 THEN V=3:V$="3":ENT=V3+1
1022 IF ENT=NLV THEN ? B$?: " THIS IS
THE LAST NOTE THE COMPUTER CAN ENTE
R IN THIS VOICE!"
1023 IF ENT>NLV THEN ? C$?: " TOO MANY
NOTES!":GOTO 5001
1030 ? :? "VOICE ";V$;": ENTRY ";ENT
1031 ? :? "NOTE?":? :? "WHICH? AF,A,AS
":? " BF,B":? " C,CS":?
" DF,D,DS"
1032 ? " EF,E,ES":? " F
,FS":? " GF,G,GS":? " R FO
R REST"
1033 ? :? "ENTER SM FOR SAME NOTE-";N
$;";D$;";"OCT";D$;";L$:INPUT A$
1035 IF A$="R" THEN N=1:D=3:L=0:N$="R"
:L$="OFF":D$="R":? C$?: "VOICE ";V$;":
ENTRY ";ENT;"; REST":GOTO 1121
1038 IF A$="SM" THEN 1211
1040 N$=A$:IF N$="C" THEN N=0:GOTO 112
0
1045 IF N$="B" THEN N=11:GOTO 1120
1050 IF N$="BF" OR N$="AS" THEN N=10:G
OTO 1120
1055 IF N$="A" THEN N=9:GOTO 1120
1060 IF N$="AF" OR N$="BS" THEN N=8:GO

```

```

TO 1120
1065 IF N$="G" THEN N=7:GOTO 1120
1070 IF N$="GF" OR N$="FS" THEN N=6:GO
TO 1120
1075 IF N$="F" THEN N=5:GOTO 1120
1080 IF N$="E" THEN N=4:GOTO 1120
1085 IF N$="EF" OR N$="DS" THEN N=3:GO
TO 1120
1090 IF N$="D" THEN N=2:GOTO 1120
1095 IF N$="DF" OR N$="CS" THEN N=1:GO
TO 1120
1105 ? C$;D$;N$="SM":GOTO 1030
1120 TRAP 1120: ? C$?: "VOICE ";V$;": E
NTRY ";ENT;"; NOTE ";N$
1121 ? :? "TOTAL DURATION IN MEASURE A
ND PROGRAM ";
1122 ? " VOICE 0 = ",D(0,0),D(0,1):?
" VOICE 1 = ",D(1,0),D(1,1):? " V
OICE 2 = ",D(2,0),D(2,1)
1123 ? " VOICE 3 = ",D(3,0),D(3,1)
1124 ? :? "DURATION OF NOTE OR REST?":
? "WHICH? SIXTEENTH":? " EIGHTH"
1125 ? " QUARTER":? " HALF
":? " WHOLE":? "ADD _ FOR DOTTED
NOTES"
1126 ? "ADD 3 FOR TRIPLET NOTES":? "AD
D 1 FOR TIED NOTES"
1127 ? :? "ENTER ED TO EDIT NOTE ";N$:
? "ENTER SM FOR SAME NOTE-";N$;";D$;
";"OCT";D$;";L$:INPUT A$
1129 IF A$="ED" THEN ? C$:GOTO 1030
1131 IF A$="SM" THEN 1211
1137 D$=A$:IF D$(1,1)="S" THEN D=DL/16
:GOTO 1152
1142 IF D$(1,1)="E" THEN D=DL/8:GOTO 1
152
1145 IF D$(1,1)="Q" THEN D=DL/4:GOTO 1
152
1147 IF D$(1,1)="H" THEN D=DL/2:GOTO 1
152
1150 IF D$(1,1)="W" THEN D=DL:GOTO 115
2
1151 ? B$:D$="SM":GOTO 1120
1152 IF D$(1,1)=D$ THEN 1158
1154 IF D$(2,2)="T" THEN 1200
1155 TRAP 1156:IF D$(3,3)="T" THEN 120
0
1156 IF D$(2,2)="." OR D$(2,2)="3" THE
N 1158
1157 ? B$:D$="SM":GOTO 1120
1158 IF N$="R" THEN 1200
1160 ? C$?:? :? "VOICE ";V$;": ENTRY ";
ENT;"; NOTE ";N$;";" DURATION ";D$
1161 ? :? "OCTAVE? 4 GOES UP FROM MIDD
LE C":? :? "WHICH? 3,4,5,6"
1162 ? :? "ENTER ED TO EDIT DURATION "
;D$?: "ENTER SM FOR SAME NOTE-";N$;";
;D$;";"OCT";D$;";L$:INPUT A$
1163 IF A$="ED" THEN 1120

```

Continental Adventures

4975 Brookdale Dept. 01
Bloomfield Hills, Mich. 48013
(313) 645-2140

Continental Adventures presents three adventures and one graphics game for the Atari 400 and 800 computer owner

- The Ghost Tower** — Combat with diabolical demons, 16K \$16.95
- Town of Derango** — Avenging the death of a father, 8K \$16.95
- Talisman of Power** — A search for the four keys of Gremlock, 16K.... \$18.95
- Super Shape Builder** — A graphics game for creating your own pictures. Joysticks reqd. 8K \$14.95

```

1164 IF A#="SM" THEN 1200
1166 TRAP 1160:D#A#:D=VAL(D#):D=0-3:I
F D<0 OR D>3 THEN ? B#:C#:GOTO 1160
1177 ? C#:? :? ? "VOICE ";V#;": ENTRY
";ENT;"; NOTE ";N#;"; DURATION ";D#;?
"OCTAVE ";D#
1178 ? :? "LOUDNESS"? :? "WHICH? PP,
P,MP":? " MF,E,FF":? " OFF
"
1179 ? :? "NORMAL VALUES":? " MF FOR
MELODY"
1180 ? " P FOR ACCOMPANIMENT":? :? "
ENTER ED TO EDIT OCTAVE ";D#:INPUT A#
1182 IF A#="ED" THEN 1160
1183 L#A#:IF L#="PP" THEN L=2:GOTO 12
00
1184 IF L#="P" THEN L=4:GOTO 1200
1185 IF L#="MP" THEN L=6:GOTO 1200
1186 IF L#="MF" THEN L=8:GOTO 1200
1187 IF L#="F" THEN L=10:GOTO 1200
1188 IF L#="FF" THEN L=12:GOTO 1200
1189 IF L#="OFF" THEN L=0:GOTO 1200
1192 ? B#:GOTO 1177
1200 IF D*(1,1)=D# THEN 1210
1202 IF D*(2,2)="." THEN D=D#3/2
1204 IF D*(2,2)="T" THEN DT=DT+D?: B#
? " ENTER DURATION OF TIED NOTE!":FOR
M=1 TO WT:NEXT W:GOTO 1120
1206 TRAP 1208:IF D*(3,3)="T" THEN DT=
DT+D?: B#?: " ENTER DURATION OF TIED N
OTE!":FOR M=1 TO WT:NEXT W:GOTO 1120
1208 IF D*(2,2)="3" THEN D=D#2/3
1210 D=D+DT:DT=0:IF D<>INT(D) OR D<8 T
HEN 1940
1211 IF D=3 AND N<>1 AND N<>0 THEN ? B
#?: " NOTE TOO HIGH!":FOR M=1 TO WT:NE
XT W:GOTO 1160
1212 D(V,0)=D(V,0)+D:D(V,1)=D(V,1)+D
1214 IF D(V,0)>DM THEN D(V,0)=D(V,0)-D
:D(V,1)=D(V,1)-D:GOSUB 1940
1216 TRAP 5001:IF V=0 AND D(0,0)=DM TH
EN CK0=1?: B#
1218 IF V=1 AND D(1,0)=DM THEN CK1=1?:
B#
1220 IF V=2 AND D(2,0)=DM THEN CK2=1?:
B#
1222 IF V=3 AND D(3,0)=DM THEN CK3=1?:
B#
1224 IF V=0 THEN 1500
1226 IF V=1 THEN 1600
1228 IF V=2 THEN 1700
1230 IF V=3 THEN 1800
1500 NO(V0,1)=N(N,0):NO(V0,2)=L:NO(V0,
3)=D:IF N(N,0)=256 THEN GOTO 700
1510 V0=V0+1:VOE=V0:GOTO 1005
1600 N1(V1,1)=N(N,0):N1(V1,2)=L:N1(V1,
3)=D:V1=V1+1:V1E=V1:GOTO 1005
1700 N2(V2,1)=N(N,0):N2(V2,2)=L:N2(V2,
3)=D:V2=V2+1:V2E=V2:GOTO 1005

```

```

1800 N3(V3,1)=N(N,0):N3(V3,2)=L:N3(V3,
3)=D:V3=V3+1:V3E=V3:GOTO 1005
1901 ? :? B#:" VOICE UNAVAILABLE!":FOR
M=1 TO WT:NEXT W:GOTO 5000
1940 ? :? B#:" DURATION UNAVAILABLE!":
FOR M=1 TO WT:NEXT W:D#="0":D=DL/4:GOT
O 1120
Edit music.
5000 TRAP 5000: ? C#?: " **EDIT MEAS
URE OF MUSIC**":? :? "VOICE?":? :? "MH
ICH? 0,1,2,3":INPUT V
5001 V0=0:V1=0:V2=0:V3=0
5002 IF V<>0 THEN V0=VOE
5004 IF V=1 AND H1=0 THEN 1901
5005 IF V<>1 AND H1=1 THEN V1=V1E
5006 IF V=2 AND H2=0 THEN 1901
5007 IF V<>2 AND H2=1 THEN V2=V2E
5008 IF V=3 AND H3=0 THEN 1901
5009 IF V<>3 AND H3=1 THEN V3=V3E
5010 IF V<0 OR V>3 THEN ? B#:GOTO 5000
5016 IF V=0 THEN D(0,1)=D(0,1)-D(0,0):
D(0,0)=0:CK0=0
5017 IF V=1 THEN D(1,1)=D(1,1)-D(1,0):
D(1,0)=0:CK1=0
5018 IF V=2 THEN D(2,1)=D(2,1)-D(2,0):
D(2,0)=0:CK2=0
5019 IF V=3 THEN D(3,1)=D(3,1)-D(3,0):
D(3,0)=0:CK3=0
5020 ? :? B#:"EDIT VOICE ";V:FOR M=1 T
O WT:NEXT W:GOTO 1004
Save music in DATA statements.
6500 IF ST=1 THEN 7510
6502 ? C#?: " **SAVE MUSIC**"
: ? :? "MOVE CURSOR TO LINE NUMBER (LN)
AND"
6505 ? "PRESS RETURN TO ENTER INTO PRO
GRAM. THEN TYPE CONT AND PRESS RETUR
N TO CONTINUE."
6508 ? :? "LN"
6510 ? "40 ?#6: ?#6; ";CHR$(34);TITLE#
;CHR$(34);" : ?#6: ?#6; ";CHR$(34);TITLE
2#;CHR$(34)
6511 IF H1=0 THEN ? "60"
6512 IF H2=0 THEN ? "70"
6513 IF H3=0 THEN ? "80"
6514 STDP
7510 ST=1:ORD=0:V0=0:V1=0:V2=0:V3=0:LN
1=LN1-1
7520 ? :? C#?: "LN "
7522 FOR LN=1 TO 6:LN1=LN1+1
7525 ? LN1;" DATA ";
7530 FOR NS=1 TO 9
7532 IF NS=9 AND NO(V0,0)=ORD AND NO(V
0,1)=256 THEN 7540
7535 IF NS=9 THEN ? CHR$(126):GOTO 757
5

```

```

7540 IF NO(V0,0)=ORD AND NO(V0,1)=256
THEN ? CHR$(126):? LN1+1;" DATA ";NO(V
0,1);";";NO(V0,2);";";NO(V0,3):GOTO 75
7545 IF NO(V0,0)=ORD THEN ? NO(V0,1);"
";NO(V0,2);";";NO(V0,3);";";V0=V0+1:
ORD=ORD+1:NEXT NS
7550 IF N1(V1,0)=ORD THEN ? N1(V1,1);"
";N1(V1,2);";";N1(V1,3);";";V1=V1+1:
ORD=ORD+1:NEXT NS
7560 IF N2(V2,0)=ORD THEN ? N2(V2,1);"
";N2(V2,2);";";N2(V2,3);";";V2=V2+1:
ORD=ORD+1:NEXT NS
7570 IF N3(V3,0)=ORD THEN ? N3(V3,1);"
";N3(V3,2);";";N3(V3,3);";";V3=V3+1:
ORD=ORD+1:NEXT NS
7575 NEXT LN
7580 ? :? "ENTER LINE NUMBERS AND CONT
INUE.":STOP
7590 IF NO(V0,1)<>256 THEN GOTO 7520
7600 LN1=LN1+1:GOTO 30
Save music on diskette.
8450 ? C#?: " **SAVE ON DISKETTE
**":? :? "SAVE PLAY PROGRAM AND DATA W
ITH THE"
8451 ? "INITIAL GROUP OF MEASURES BUT
ONLY DATA IN SUBSEQUENT SAVES."
8452 ? :? "INSERT BLANK DISKETTE!":? :
? "ENTER LINE NUMBER AND CONTINUE."
: ?
8453 ? " LN ":? "8460 LIST ";CHR$(34);
"D:SONG";SN;".ENT";CHR$(34);";";LNS;";
";LN1:STOP
8454 ? C#?:? "PROGRAM BEING SAVED."
8461 IF LNS=39 THEN LNS=101: ? C#?:? :?
"ENTER LINE NUMBERS TO CLEAR MEMORY FO
R NEXT GROUP OF MEASURES."
8462 FOR M=1 TO WT:NEXT W:LN=LNS-1
8463 ? C#?: "LN ":LNF=LNS+15:IF LNF>=LN
N1 THEN LNF=LN1
8464 FOR W=LNS TO LNF:LN=LN+1: ? LN:NEX
T W
8465 SN=SN+1: ? :? "ENTER LINE NUMBERS
AND CONTINUE.":STOP
8466 LNS=LNF+1:IF LNS<=LN1 THEN 8463
8468 ? C#?:? "LOAD OR ADD THIS MUSIC
PROGRAM AND"
8469 ? "ADDITIONAL DATA TO ANOTHER PRO
GRAM WITH ENTER "D:SONG#.ENT""
8470 LNS=LN1:FOR M=1 TO WT:NEXT W:GOTO
985
Changes for cassette version.
9031 TRAP 30: ? :? "SAVE ON CASSETTE OR
":? :? "RE-ENTER LINE NUMBERS":? :? "M
HIGH?":? :? "ENTER 1 TO PLAY"
9450 ? C#?: " **SAVE ON CASSETTE
**":? :? "SAVE PLAY PROGRAM AND DATA W
ITH THE"

```



The Tone Envelope

by Christopher U. Light

When using a music synthesizer, it is very useful to have some background in the physics of how "real" music is produced. The following brief explanation should be a helpful introduction.

When a string is plucked or hit, or when a whistle is blown at the end of a tube, air begins to vibrate in a manner that is unique to the instrument; a manner that allows us to say the first is a harp or piano, the second an organ, a tuba, or a flute. The complexity of sounds is known as the instrument's "timbre." Music synthesizers attempt to reproduce the important parts of this timbre artificially and ignore those parts their designers feel to be less important. At least two essential components of timbre have been identified, the instrument's "envelope" and its pattern of overtones.

When a note is struck on a piano, it takes a minute fraction of a second for the string to begin vibrating its full distance, that is, for the volume to build up to its maximum. But a pipe organ reaches full volume almost instantly. The volume of a piano note decays away to zero over a period of time. It becomes perceptively fainter over many seconds. A pipe organ, which is on when the wind blows and off when it doesn't, will not decay at all until the key is released, whereupon it stops abruptly. The pattern of this build-up of a note's volume to its maximum and its fading away to silence is called the note's envelope.

Actually the envelope can be much more complicated, depending on the instrument. Many instruments do not actually have a period when the note's volume is constant. While the wind instruments can be fully sustained, a

piano, for example, cannot. After its note hits full volume, it fades away gradually.

When we pluck a string or blow into a tube, the fundamental note produced has a frequency based on the length of the string or the tube. Additionally the string or tube will also produce, but not as loudly, many other notes whose frequencies are integer multiples of the fundamental note. For example, if on a piano we strike the note A below middle C, which has a frequency of 220 cycles per second, we will hear (more faintly) the same note an octave higher. This is A above middle C, which has a frequency of 2×220 or 440 cycles per second. We will also hear (still more faintly) the note whose frequency is 3×220 cycles per second, which is E, and then with decreasing volume, the A two octaves above middle C, the D above that, etc.

The relative strength of these overtones varies widely from instrument to instrument. A flute has a very sweet and pure tone because it produces the fundamental, the same note an octave higher, a fifth above that and not much else. In comparison, brass instruments get their harshness by producing a very long and loud overtone series.

Many instruments waver in both pitch and volume as the maximum volume builds up and the desired pitch is attained. Although the unassisted or untrained ear cannot always identify exactly what's happening in that split second in which the note is hit, it does realize that there are differences. Thus, one important step in simulating real instruments is that of simulating their envelopes.

```

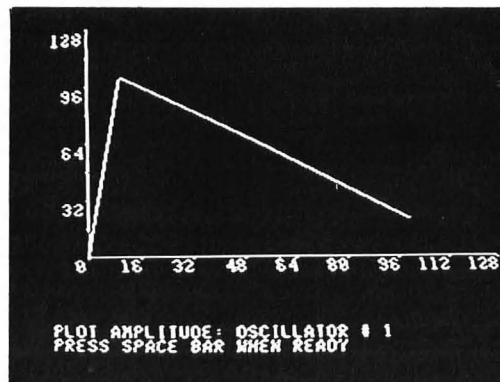
44,108,8,9,91,4,9,108,4,9,96,8,27
112 DATA 96,4,27,96,4,27,91,8,9,108,4,
9,91,4,9,81,8,72,121,4,72,81,4,72
113 DATA 60,8,12,121,4,36,60,4,12,121,
4,36,60,8,12,60,4,12,60,8,12,60,4,12
114 DATA 81,8,12,121,4,36,81,4,12,121,
4,36,81,8,12,81,4,12,81,8,12,81,4,12
115 DATA 96,8,12,121,4,27,96,4,12,121,
4,27,96,8,12,96,4,12,96,8,12,96,4,12
116 DATA 108,4,9,108,4,9,121,8,12,96,4,
,36,121,4,12,96,4,36,121,8,12,121,4,12
117 DATA 121,8,12,121,4,12
118 DATA 81,8,27,96,4,27,81,4,27,96,4,
27,91,8,9,108,4,9,91,4,9,108,4,9
119 DATA 96,8,27,96,4,27,96,4,27,96,4,
27,108,8,9,91,4,9,108,4,9,91,4,9
120 DATA 121,8,72,81,4,72,121,4,72,81,
4,72
121 DATA 0,0,144,60,4,12,121,4,36,60,4,
,12,60,4,12,60,4,12,60,4,12,60,4,12
122 DATA 81,4,12,121,4,36,81,4,12,81,4,
,12,81,4,12,81,4,12,81,4,12,96,4,12
123 DATA 121,4,27,96,4,12,96,4,12,96,4,
,12,96,4,12,96,4,12,108,4,9,121,4,12
124 DATA 96,4,36,121,4,12,121,4,12,121,
,4,12,121,4,12,121,4,12
125 DATA 0,0,144,81,4,27,96,4,27,81,4,
27,91,4,9,108,4,9,91,4,9,96,4,27
126 DATA 96,4,27,96,4,27,108,4,9,91,4,
9,108,4,9,121,4,72,81,4,72,121,4,72
127 DATA 0,0,144,0,0,144,60,4,12,121,4,
,36,60,4,12,60,4,12,81,4,12,121,4,36
128 DATA 81,4,12,81,4,12,96,4,12,121,4,
,27,96,4,12,96,4,12,108,4,9,121,4,12
129 DATA 96,4,36,121,4,12,121,4,12
130 DATA 0,0,144,0,0,144,81,4,27,96,4,
27,91,4,9,108,4,9,96,4,27,96,4,27
131 DATA 108,4,9,91,4,9,121,4,72,81,4,
72
132 DATA 0,0,144,0,0,144,0,0,144,60,4,
12,60,4,12,60,4,12,81,4,12,81,4,12
133 DATA 81,4,12,96,4,12,96,4,12,96,4,
12,121,4,12,121,4,12,121,4,12
134 DATA 0,0,144,0,0,144,0,0,144,81,4,
27,91,4,9,96,4,27,108,4,9,121,4,72
135 DATA 256,0,0
  
```

Atari One Liners

```

0 GRAPHICS 18:POSITION 3,4: ? #6:"MSI
FUTURE BAND":FOR X=10000 TO 0 STEP
-1.5:SOUND 0,0,8,X:POKE 708,INT(RND(
0)*222):NEXT X
  
```

John Niem and ZVL Arifin
Hong Kong



Shown is an example of an instrument's envelope plotted on the Mountain Hardware *Music-System's* Instrument Definer Program.



ORCHESTRA-85

from Software Affair, 858 Rubis Drive, Sunnyvale, CA, 94087. An S-80 hardware/software package. Suggested retail price: \$129.95.

Reviewed by Robb Murray

About a year ago, Software Affair, Ltd. began marketing *Orchestra-80*, a reasonably-priced hardware/software package which was designed to turn the S-80 Model I into a "high-quality musical instrument." The product was impressive, and so easy to use that home-grown musical code for it immediately began sprouting up all over bulletin boards. *Orchestra-80* differed from rival systems for the Apple, Atari, and Pet in that its musical coding scheme didn't resemble the usual music notation. It used a linear symbol which always necessitated translation from standard written music. So while I was excited about *Orchestra-80*, I felt apprehensive that its coding scheme would be its undoing. However, *Orchestra-80* and its new version, *Orchestra-85*, remain the finest music systems available for the S-80 and other units such as the LNW.

Orchestra-90, for the Model III, is already available and there is talk of future products for other S-80 models. Software Affair obviously intends to keep pace with hardware changes, and *Orchestra-85* is only the first in what

may eventually be a large number of progenies.

Orchestra-85 is a more sophisticated product than its predecessor and it has a price tag to match. Its basic price is \$129.95 — roughly \$50 more than *Orchestra-80* — or perhaps you could get one for only \$69.95 if you give *Orchestra-80* a trade-in allowance of \$60.

The *Orchestra-85* coding scheme, while incorporating many new features, is upwardly compatible from *Orchestra-80* so that all old music files will play on the new system. Also it will run optimally only with enhanced clock speed.

Orchestra-85 now offers stereo sound, percussive effects, the ability to change voice quality (including loudness) within the music file, and new editing and other support conveniences.

BETTER PLAYING AMENITIES

Up to five musical voices may now sound simultaneously with *Orchestra-85*. A series of questions, answered by the user during software configuration, allow the selection of three, four, or five voices, and fast or slow clock speed. These answers may be saved so that they needn't be supplied again later.

When faster clock speed is used, the tone registers default to settings that use more harmonic partials, resulting in a much enhanced tone quality. The instruction manual for *Orchestra-85* is frank about this matter; five voices cannot be played satisfactorily at the 1.77 MHz clock speed. This leaves the owners of unmodified Model I's with only the old options of three or four voices.

Not only are more voices now available, but their tone quality has been improved, especially with the faster clock speeds. The advertising flyer reads, "Signal-to-noise ratio improved by 6db!"...And who am I to argue with that?

Stereo sound is also a major new feature of *Orchestra-85* that lends spatial fullness to the music. Channel A will play up to two voices and Channel B up to three. The new "mapping" capability (invoked by the 'Z' symbol) directs individual voices to specified channels, and allows them to be balanced or positioned variably (ping-ponged) during play of a musical selection. Bryan Eggers' *Camptown Races* transcription, one of the sample selections, even simulates a horse galloping from speaker to speaker. Such an effect is partly possible because of the new percussive effects.

Percussive effects can now be created using a new "reverse articu-

MUSIC SAMPLE

The image shows a musical notation sample for a piece in 3/4 time. It features a treble clef staff with a large '3' over a '4' indicating the time signature. The notation consists of several measures of music. The first measure has a single note labeled 'V1'. The second measure has two notes, both labeled 'V1'. The third measure has two notes, both labeled 'V1'. The fourth measure has three notes, labeled 'V1', 'V2', and 'V3'. The fifth measure has three notes, labeled 'V1', 'V2', and 'V3'. The sixth measure has two notes, labeled 'V1' and 'V3'. Below the staff, there are four notes labeled 'V4', which appear to be bass notes or a separate voice part.

lation" clef (consisting of a single underscoring stroke, '—'). The user's manual gives recipes for creating "squeaky" sounds, "scratchy" sounds, and a "woodblock" sound. Unlike the sounds of other synthesized instruments, the percussion does not exist as a system default. Instead, it may be selected through the use of another new feature.

The old *Orchestra-80* let voice registers be set at the time of software configuration and although allowing for some timbral variety and balancing, this scheme "froze" the settings during playback sessions. But *Orchestra-85* allows for timbral resetting within the music file itself, so that voice qualities may undergo transformation while a musical selection unfolds to the listener. In the documentation, the old "Standard Registers" section has been replaced with a new part, "Instrument Definition."

The expression used to define voice registers (it is a string always beginning with 'J') contains as its last element a volume specification parameter. So, *Orchestra-85* also offers the option of changing the loudness of individual parts during a piece, although the method used is somewhat clumsy.

BETTER EDITING AMENITIES

The new "multiple get" ('MULTI') instruction will keep playing a piece (or series of pieces) indefinitely, until interrupted. Such a command is useful for creating continuous background music, such as might be used in a sales demonstration or a media show.

The *Orchestra-85* APPEND command can be used to combine music files. It copies a given file of code in front of another specified file.

The KILL command will now delete a disk file directly from the command mode. To kill a file previously, one had to get out of music code, restore control to the operating system, kill the file, then reconfigure the software to resume playing or editing. These cumbersome middle steps have now been bypassed.

In *Orchestra-80*, the CLEAR key both cleared and deleted a given line. These two functions are now separate. In *Orchestra-85*, CLEAR now only clears the line; SHIFT CLEAR deletes it. Also, global string searching can now be made upwards, not only downwards, in a line.

The user's manual has been newly set with darker type, but most of the new text is virtually the same as the old. The new version is only slightly longer than the old one (43 pages versus 39 pages). New sample percussion code is included.

In future additions of the manual, it would be desirable to supply at least one musical example that incorporates all possible coding features together in one piece. Another addition should be a series of expanded tables that would give register settings for a wide variety of instrumental and percussive sounds. As it is, the user must experiment in order to discover such settings. There is talk of an *Orchestra-80/85* newsletter that might gradually fill some of these gaps. But the new user who has just purchased a system should also have the benefit of this knowledge.

FURTHER DESIRABLE ENHANCEMENTS

In general, the disadvantages of the *Orchestra-80* system noted in my review in the October, 1981, issue of *SoftSide*, have been carried over to

Orchestra-85. Among them are: a coding scheme that is unlike traditional written music; overtone distortion, especially in the higher musical registers (although this problem has been somewhat reduced); interference among voice lines that can cause breaks in the flow of sounds that should be continuous; a lack of sustainable holds during playback; and a lack of fine-tuning capability. However, at least one of the problems noted earlier — that of dynamic control within music files — is solved in *Orchestra-85*. Several more improvements should be added to future releases in the *Orchestra-80/85* series.

But before I get to those, I would like to offer a retraction and an apology. I was incorrect in one of my former criticisms of *Orchestra-80* when I stated that the playable (versus codable) octave range of *Orchestra-80* was only four (versus six) octaves. Since individual voices may be transposed, and not just entire measures, *Orchestra-80* and *85* can actually play within their full six octave ranges at all times. If a voice climbs above (or sinks beneath) its codable range, then transposing the voice up (or down) an octave, for example, would restore an octave's worth of usable symbols. My apologies to Software Affair for this lapse in understanding.

And now, the suggestions:

1. SCROLLING DURING PERFORMANCE — At least one tablature-coded micromusic system now on the market reads through a music file as it is being played so that the user may follow. Such a feature would be a useful addition to *Orchestra-85*. Besides being visually interesting, it would greatly aid in

continued on next page

MUSIC SAMPLE

The image shows a musical score sample with four staves labeled V1, V2, V3, and V4. The top staff (V1) contains a series of notes with a slur over the first two. The lower staves (V2, V3, V4) contain block chords. The key signature has two sharps (F# and C#).

continued from previous page

testing. As it is, the user must always follow a hard-copy version of code during testing and keep track of where particular musical events (such as mistakes!) occur. An enhanced graphics tracking system, on the other hand, could show instantly the code that is producing musical sound, measure by measure.

2. BETTER INSTRUMENT SIMULATION — At this stage of development, it is almost meaningless to state, as part of the product promotion, that the registers of *Orchestra-85* give a "spectrally" accurate reproduction of particular instrumental timbres. None of the instrumental settings sound like anything other than a reed organ. For this situation to change, note attacks and decays will also need to conform to those of the desired instruments. *Orchestra-85* is a much more flat and boring performer that it deserves to be.

More hardware or software will be needed if successors to *Orchestra-85* are to produce more genuine instrument sounds in the future. Perhaps plans for such improvements are already in the works.

3. MORE CONTOURED TEMPO AND DYNAMIC CONTROL — *Orchestra-85* should provide a more nearly analog method of speeding up music, slowing it down, getting louder, and getting softer. Now, the only way to change tempo and volume is the clumsy, "terraced" one of resetting voice registers and tempo parameters at every measure. Such a process is time-consuming and tiring. Why couldn't it be provided for as an automated part of the system? At least, *Orchestra-85* documentation should give several examples of well-handled crescendos, decrescendos, accelerandos, and ritardandos within coded music.

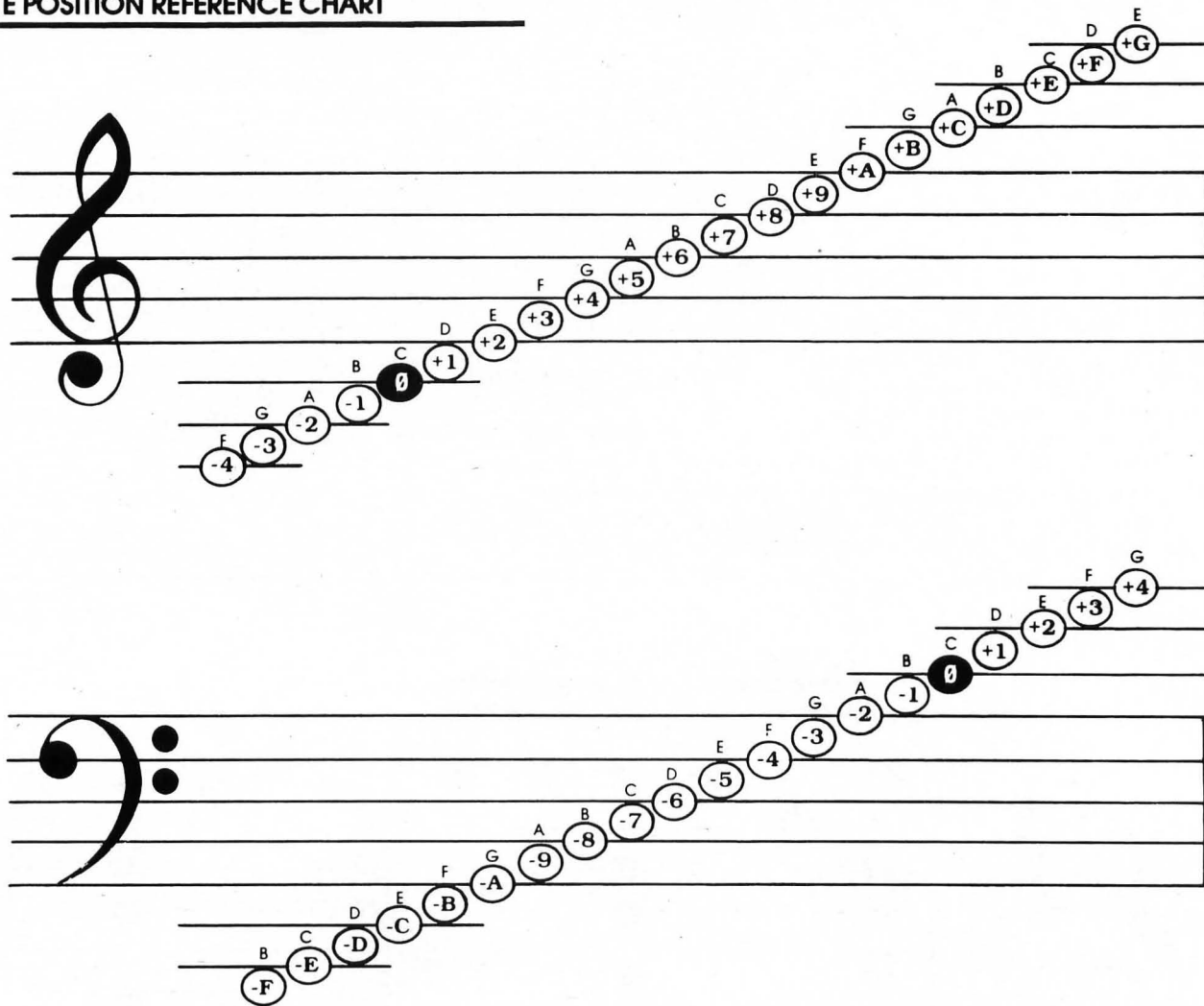
TIME FOR AN HONOR ROLL

The esthetic potential of the *Orchestra-80/85* system is great, and has not yet been fully exploited. Its capabilities still go well beyond what anyone has taken the trouble to code for it yet. To encourage the best, most creative use of musical coding, I want to suggest that some enterprising marketer sponsor an *Orchestra-80/85* contest and give cash awards for the best entries. Contestants could mail their coded music on cassette, or diskette, or submit it via bulletin board. The best 20 or 30 entries could be sold as a sample audio cassette or code file. There could be money, fun, and a lot of good music in it. Any takers?

STILL A GOOD BUY

As time passes, it seems probable that the *Orchestra-80/85* series will not only accommodate, but demand, more

NOTE POSITION REFERENCE CHART



Note: The symbols inside the notes represent the **Orchestra-85** scale. The small letters above the notes represent the musical scale and are for reference only.

powerful computers — in particular, processors with faster clock speeds. During product development, Software Affair is continually faced with a somewhat arbitrary decision of where to stop upgrading and to define the

next model for sale. For any user not committed to buying all the successive versions, the old question remains of how long to wait before investing in a new version of still-developing software.

Now, at least, the price of *Orchestra-85* is still low, and the decision to buy it is not difficult to reach. Unquestionably, if you own a 16K S-80, along with a stereo speaker system, you will find *Orchestra-85* an amazing and delightful way to produce music.

MORE SAMPLE MUSIC

Note: Remember to check the Key signatures, all samples are in 4/4 time.



*I6Q4S5'6'15'Q.5
@V2H52



*WB
V2I0247420-3



*H.BQD
V2I\$24264QS
V3W-5



*I.\$S7'7'I7'S7'I8Q6S8'8'
V2Q5'54%4
V3Q2'2-1'-1
@V4Q2'25'5



*TA'9'I.AQAQAS9'7#'8#'9'
V2H1#0#



*SS3'6&'8'B'8'6'3'\$4'6&'7'I:945
V2H-30



NOVEMBER ADVENTURE OF THE MONTH AROUND THE WORLD IN EIGHTY DAYS

Try to repeat the feat of the classic novel, complete with a balloon and other exciting features of the original adventure. Are you ready to take the challenge?

A new obsession is sweeping the country as members of **SoftSide's Adventure of the Month Club** rush to their mailboxes each month. Where will they be taken this month — Off into space? Back in time? To the bottom of the sea?

You too can look forward to an exciting new spree, full of secrets, every month. Your program budget won't be destroyed, either. Membership in **SoftSide's Adventure of the Month Club** costs only \$27 for six months on tape (\$4.50 per adventure) or \$45 for six months on disk (\$7.50 per adventure).

You can sample these delights with one month's edition. The tape is only \$6 and the disk \$9.

To join this unique crew, use the order card bound into this issue. Fill it out and send with payment to:

**Adventure of the Month Club
Department 681
6 South Street
Milford, NH 03055**

Sit back and wait for your first program to arrive. **SoftSide** will select a high quality, original Adventure in BASIC. Watch for the announcement of each month's Adventure here in the pages of **SoftSide Magazine**.

SYSTEM REQUIREMENTS:

Apple 24K Tape 32K Disk
Atari 32K Tape 40K Disk
S-80 16K Tape 32K Disk

UPPER/LOWER CASE for your Apple

MPC Peripherals introduces the new AP-96 UPPER/lower case ROM. Now your Apple's text mode can include upper and lower case letters featuring a full 96 character ASCII set with true descenders.

The AP-96 plugs in with no modifications to the Apple and is compatible with most word processing systems. For REV.7 or later Apple computers only, **Lifetime Warranty**.

Available direct or at your computer dealer.

#07-223047H..... **\$39.50**



TSE-HARDSIDE

14 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

*Apple is a trademark of Apple Computers, Inc.

SoftSide™

BACK ISSUES

SOFTSIDES FROM THE PAST

If you like what this issue of **SoftSide** has to offer, you should see what's waiting for you in **SoftSide** Back Issues! You may feel that you've missed out on many of our programs that appeared before you became a subscriber. It's not too late to do something about it. You won't have to miss a thing because we still have back issues available to complete your **SoftSide** library! But, order now, as some of our more popular

issues are already out of stock and others are dwindling quickly.

Listed below are all of our past issues with their feature articles and the systems they're for. Each issue costs \$3.00 for the magazine only. Those issues marked with an asterisk are also available on cassette for \$9.95 or disk for \$14.95. September 1981 is the first enhanced Disk Version and costs \$19.95.

SoftSide S-80 Edition:

October 1978 — Not Available
November 1978 — "End Zone"
December 1978 — "Santa Paravia En Fiumaccio"
*January 1979 — "Round the Horn"
*February 1979 — "Form 1040"
*March 1979 — "Personal Finance"
*April 1979 — "Safari"
*May 1979 — "Dog Star Adventure"
*June 1979 — "Atlantic Balloon Crossing"
*July 1979 — "All Star Baseball"
*August 1979 — "Meltdown"
September 1979 — Not Available
*October 1979 — "Westward 1847"
November 1979 — Not Available
*December 1979 — "Oil Baron"
*January 1980 — "Moving Maze"
February 1980 — Not Available
*March 1980 — "Broadway"
April 1980 — Not Available
*May 1980 — "Star Trek"
*June 1980 — "Micro Millionaire"
*July 1980 — Adventure Issue

PROG 80 (S-80 only)

March 1979 — Not Available
May 1979 — "Clock Routines"
July 1979 — "Histogram and BASIC Statistics"
September 1979 — "DOS-How to stop lost data errors"
December 1979 — "Computer Telephone Dialing"
February 1980 — "Hexmem"
April 1980 — "Redefining Level II Keyboard"
June 1980 — "Z-80 Disassembler"
August 1980 — "Varlist"—A program to list variables

SoftSide: Apple Edition

*January 1980 — "Dog Star Adventure"
*February 1980 — "Connection"
*March 1980 — "Treasure Hunt"
*April 1980 — "Jig-Saw"
*May 1980 — "Invaders"
June 1980 — Not Available
*July 1980 — "Pork Barrel"

SoftSide: Apple, Atari and S-80 Combined Edition

*August 1980 — "Caribbean Cruising" — Apple
 "Master's Golf" — Atari
 "Sailplane" — S-80
September 1980 — Not Available
*October 1980 — "Developing Data Base II" — All Three
 "Moonlanding" — Apple
 "World Series" — Atari
 "Earth-Port II" — S-80

*November 1980 — "Developing Data Base III" — All Three
 "Collision" — Apple
 "Trench" — Atari
 "Kriegspiel" — S-80
*December 1980 — "Developing Data Base IV" — All three
 "Baseball" — Apple
 "Speedello" — Atari
 "Kidnapped" — S-80
*January 1981 — "Developing Data Base V" — All three
 "Convoy" — Apple and S-80
 "Angle Cannon" — Atari
 "Ship Destroyer" — S-80
*February 1981 — "Developing Data Base VI" — All three
 "Miner" — All three
 "Mini-Golf" — Atari and S-80
 "Long Distance" — S-80
*March 1981 — "Developing Data Base VII" — All three
 "Strategy Strike" — Apple and S-80
 "Flags" — Atari
 "Volcano" — S-80
*April 1981 — "Battle At Sea" — Apple
 "Convoy" — Atari
 "Dominoes" — S-80
*May 1981 — "Galaxia" — Apple
 "Dodge" — Atari
 "Orienteering At Jacques's Coulee" — S-80
*June 1981 — "Old Glory" — All three
 "Word-Search Puzzle Generator" — All three
 "Anallist" — S-80
*July 1981 — "Chemistry Drill" — All three
 "Kidnapped" — Apple and Atari
 "Magic Paper Calculator" — S-80
*August 1981 — "Quest 1" — All three
 "Battlefield" — All three
 "Compu-Sketch" — S-80
*September 1981 — "Flip-It" — All three
 "Word Challenge" — All three
 "Exterminate" — S-80
 (Enhanced Disk Version — \$19.95)

Use the bind-in card in this issue to order or send a list of the back issues you'd like, with payment of \$3.00 per magazine to:

SoftSide Publications
515 Abbott Drive
Broomall, PA 19008

For the magazine/media combination, send \$9.95 per cassette and magazine, or \$14.95 per disk and magazine (\$19.95 for September 1981) to:

SoftSide Publications
6 South Street
Milford, NH 03055



THE ATARI MUSIC COMPOSER



Reviewed

from Atari, Inc., Consumer Division, 1195 Borregas Ave., Sunnyvale, CA 94086. Suggested retail price: \$59.95.

Reviewed by Randal L. Kottwitz

Teaching a novice, of any age, to read music can be quite a chore. Most introductory music classes utilize a rather clumsy method of associating the note names with a piano keyboard or its equivalent. This leaves many students with a good understanding of the note names and the piano keyboard. However, they lack comprehension when confronted with traditional musical notation. The *Atari Music Composer* ROM cartridge could be used as an interactive tool to simplify lengthy explanations of the interrelationships of sound, note names and musical scoring. The program and its accompanying 20-page manual assume the user has some ability to read music. However, with the proper supervision, its effectiveness as an "audio blackboard" cannot be denied. As I explored this program, I could see my elementary school teachers to my college music professors crying out for just this sort of tool. I emphasize the word "tool" for education as I found

the program lacking when it came to using it for serious composition or being a programmable musical instrument.

The utility is operated through one main menu and four submenus. Entry of a single letter and RETURN will change any function. The user's first step is to enter the music in the Edit Music mode. The *Music Composer* can record ten phrases of as many measures as the memory will allow. This is indicated in the Edit Music mode as the number of notes that can still be stored.

Notes are entered in a *note letter, octave number, note duration* format over a range of three octaves and one note (as opposed to a piano's seven and one-half octaves). Note durations from thirty-second through whole notes may be entered, and tied and dotted notes are allowed. Accidentals may be input individually and will carry through the measure or may be included in the key signature.

Any of the standard music key signatures may be entered before the music to make it play in other than the default C Major (A Minor) key. Changing the key signature after the music has been entered will make it only display in the new key, but play in the old. The user may enter his own meter indication and ask the computer to check his placement of measure

markings or may use the default 4/4 and turn off the program's Check Measure status, allowing as many beats per measure as he cares to enter. Any of nine tempo settings are available, but their calibration is not equal in the range and only three are varied enough to be useful. All of this leads one to a positive perspective of the program's versatility.

As I proceeded to enter the sample piece in the manual *Row, Row, Row Your Boat*, the shortcomings of the program began to show. Phrase 1 (the first two measures) was easy to enter and I was surprised at the tone quality the machine returned. I was putting it to the test as I had the audio signal from my television coming through my stereo equipment. The manual states, "The first measure of **PHRASE 2** is a problem," and they don't lie. In the traditional version of the song, the measure should be entered as four eighth-note triplets. As the *Music Composer* cannot accept triplets (a serious problem for a great deal of music), the manual offers one alternative in its illustration and another in the suggested entry codes. Both are wrong. The illustration shows the substitution for each triplet as a sixteenth note, followed by a dotted sixteenth note and another sixteenth note. This leads to a measure of three and three-quarters beats in a meter requir-

ATARI MUSIC

EDIT MUSIC
ARRANGE MUSIC
SAVE
RETRIEVE
DOS
LISTEN

COPYRIGHT ATARI 1979

WHICH?

EDIT MUSIC

PHRASE
METER 4/4
KEY SIG. G5
TEMPO 2
CHECK MEASURES OFF
STOP

WHICH?



ing four beats per measure. The suggested entry codes are for three dotted sixteenth notes per triplet, leading to a measure of four and one-eighth beats.

I didn't discover the discrepancy until I had arranged the phrases into four voices and had developed a nervous twitch as I heard the round go astray. I changed each triplet to an eighth note followed by two sixteenth notes, giving *Row, Row...* something of the flavor of an English madrigal. The awkwardness of the editing function became apparent as each insertion or deletion of a note caused the measure to play again. This repetition is a minor annoyance to the user experienced in music and could be discouraging to a novice composer who must hear a measure played wrong at least twice before being able to fix it completely.

Once music is entered, the user can proceed to the Arrange Music function. In this mode, each of the phrases composed in the Edit mode can be arranged in any of four voices to play simultaneously. Each voice is arranged in a manner similar to a 20-step program, including loops, volume changes and transpositions. (The program's transpositions are accomplished by half-step changes in the tonic and are not true transpositions in the musical sense.) It is also in this mode that you choose which one of the four voices will display during playback. This

seems no problem until you hear a discrepancy in one of the voices and have to change the arrangement program four times to display each of the voices before you can find the problem.

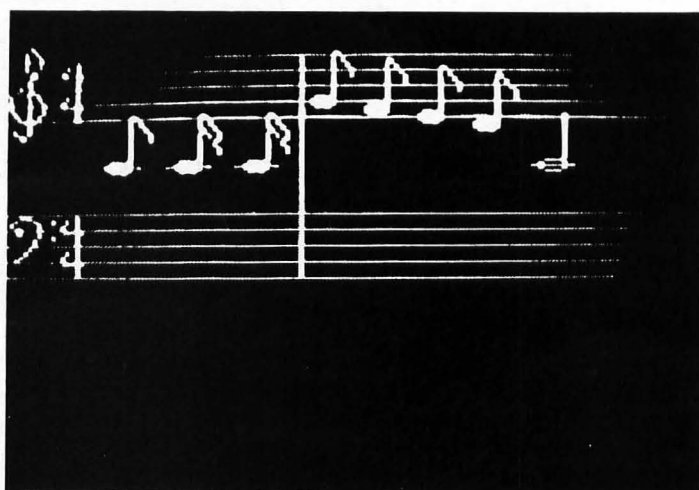
After music has been arranged, the user can listen to it, save it to tape or disk, and recall it for further work. The Save and Retrieve functions are easy to operate and allow many options as to the portion of the music you wish to save and in what form. You may save everything, all phrases, all voices, a single phrase, or a single voice. If everything is saved, any portion of it may be exclusively retrieved.

My background includes several years of vocal study in college and serious instrumental studies in piano and the French horn. Although I do not consider myself a serious composer, the *Music Composer* did leave me wanting. The limitations imposed by the confined range, lack of variance in tempo and unavailability of triplets were enough to send me on a serious hunt through my music library to find a sheet of music capable of being input to *Music Composer* without serious deleterious effects.

Although I cannot recommend the *Music Composer* to the serious musician, I do heartily promote it as an introductory "tool" for those who wish to learn musical notation or simply

"play" with music. The graphics are well done and supportive to the structure of the program's mechanics. Certainly, any music theory classes whose tuition could be paid with the *Music Composer's* \$59.95 price could not come close to educating with the program's thoroughness or individualized rate of instruction. I hope that the two methods of instruction at the introductory level (traditional classroom and individualized computer) are or soon will be combined for the greater good of the musical world.

The *Music Composer* was released two years ago in 1979. In most industries, a product with that short a history would be a long way from the restructuring point. However, considering the rapid rate of software development, especially in the area of audio output, perhaps it is time Atari consider a serious enhancement of this program — maybe a "professional" version. The current version of *Music Composer* cannot create music files to be merged into BASIC programs. This addition could elicit a great deal of entertainment software utilizing much more of the Atari's capabilities. We are becoming increasingly aware of the sound potential of the Atari. It would seem as though a program should be developed to showcase that potential and initiate a handshake with budding musicians.



ALF II

from Alf Products, Inc., 1448
Estes, Denver, CO 80215. Sug-
gested retail price: \$198.

MUSICSYSTEM

from Mountain Computer, Inc.,
300 El Pueblo, Scotts Valley,
CA 95066. Suggested retail
price: \$545.

Reviewed by Christopher U. Light

One of the things I have always wished I could do is play a musical instrument (nothing exotic, mind you; a piano or violin would do just fine) or even compose great symphonies (dreamer that I am). Although I had an obligatory year or two of piano lessons when I was nine or ten years old, almost none of what I learned stayed with me. I cannot play a piano or any other serious musical instrument, and I cannot really read music, although I do understand most of the notation.

Ten years ago my wife gave me an Appalachian dulcimer, which is probably the easiest instrument there is to learn (outside of a kazoo). It took me half an hour to learn how to play a couple of simple folk tunes whose melodies I already knew. Now, a decade later, I can play a couple of dozen melodies, but like the first ones, I must play them by ear. Although I can pick out the pitches from written music, I can never get the rhythm right unless I've heard the tune played several times.

A year ago at the Midwest Computer Show in Chicago, I heard music blaring across the lower exhibition hall. Although it sounded overly bright and was definitely synthesized rather than an actual orchestra, I could recognize Hayden's *Surprise Symphony*. I pushed my way through the crowd and stood entranced, listening to one of Bach's three-part inventions, a Scarlatti sonata and some songs from *My Fair Lady*...all played on an Apple II. There was no phonograph or tape player — just an Apple, a stereo amplifier, and a pair of very large speakers.

I learned from one of the salesmen running the booth that the musician inside the Apple measured just 3" by 4",

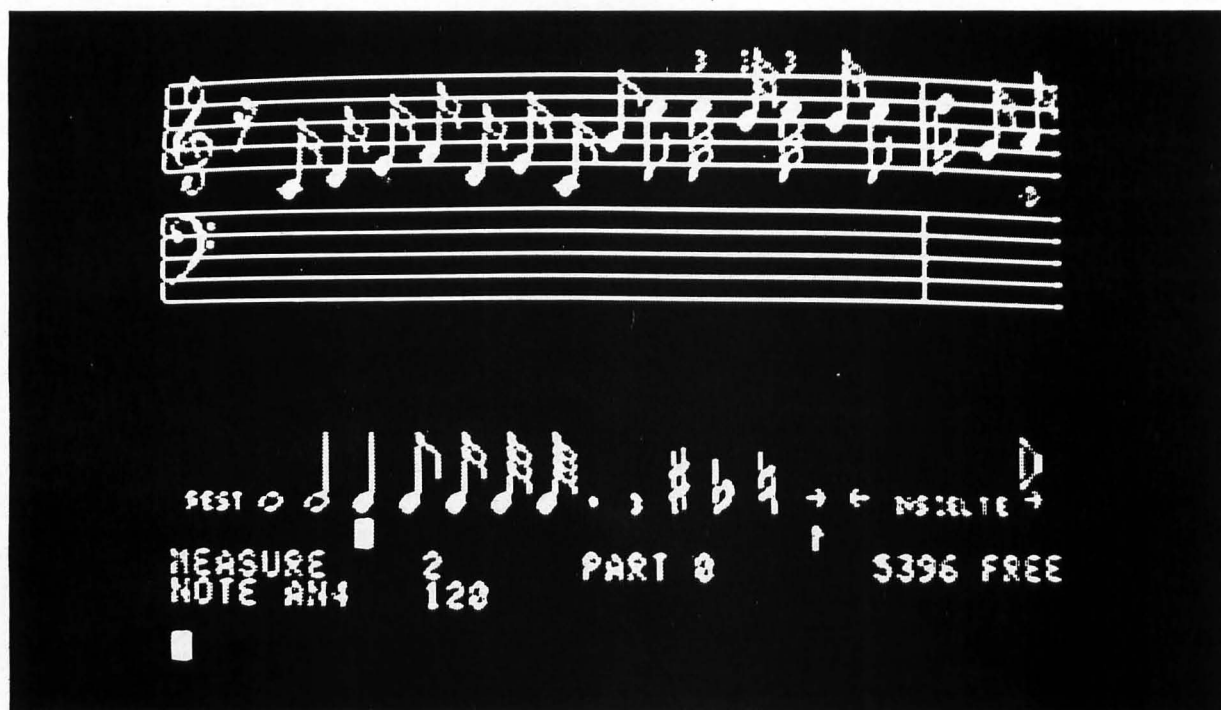
was plugged into the Apple's slot 4, and was called an *Alf*. Maybe this could be the solution to my dulcimer dilemma I thought. I could enter unfamiliar tunes into my Apple's memory, via *Alf*, again and again until I had entered them into my own memory. Then I could play them by ear on my dulcimer. And maybe *Alf* could even play the great symphonies I would compose.

While I listened, I pestered the salesman with questions. There were actually two different *Alfs*, he told me. *Alf I*, which can play more than three voices simultaneously, costs \$250, while the nine-voice *Alf II* sells for \$198. Before I could ask, he explained the apparent discrepancy: *Alf II's* tonal accuracy becomes increasingly poor as it goes up the scale, so that it is limited to six octaves and only 16 different volume levels with a range of 28 decibels. On the other hand, the salesman claimed that *Alf I*, with eight octaves and a 78-decibel volume range, has pitches accurate enough to be used for voice training. He also said that three *Alf I's* can be put into three Apple slots to provide nine voices.

I bought the *Alf II* — the less expensive one with nine voices on one board. At last I could listen to the music that I

Figure 1.

The first measure of the treble part of Bach's *Invention Number One* as it appears in *Alf II's* music editor. While six octaves can be entered on the double staff, and the bass clef can be used, only one part can be entered at a time. Below the staff is *Alf's* only menu.



The screenshot displays a music editor interface. At the top, a double staff is shown with a treble clef on the upper staff and a bass clef on the lower staff. The upper staff contains a sequence of musical notes, including eighth and sixteenth notes, with some triplets indicated by a '3' above the notes. Below the staff is a menu with several options: 'REST', a musical note icon, a treble clef icon, a bass clef icon, a sharp sign, a flat sign, a key signature icon, and a double bar line icon. Below the menu, the text 'MEASURE 2' and 'NOTE AM4 128' is visible. To the right, 'PART 8' and '\$396 FREE' are displayed. The entire interface is rendered in white text and symbols on a black background.

couldn't hear in my mind when I only saw it written on paper. And I might even try composing — if not great symphonies, maybe some simple songs.

The price of this miracle? \$545. I gulped. But if I had one, I wouldn't be limited to synthesizer sounds, and if I wrote a great piano and violin sonata (dreamer), I could hear it played as if on a piano and violin rather than on the indeterminate instruments of the *Alf II*. I convinced myself that my ten-year-old typewriter would last another decade and that I could do without the word processor I had actually come to buy that day (and which, as a writer, I would not only use but could also write off on my tax returns.) Instead I bought the *MusicSystem* (Version 2.0) and on the way home, stopped at an office supply store to pick up a new typewriter ribbon.

Before inserting the boards in my Apple, I called Mountain Hardware in California to make sure that two competing music synthesizers in the same Apple wouldn't cause shorts, smoke, fire, blown fuses, hung systems, or other malfunctions. They assured me that while the two systems are not compatible, they are neutral.

In comparing the *Alf II* and *MusicSystem* (hereafter *MS*) perhaps the first thing one notices is the contrast in their packaging, a contrast that typifies the differences between the synthesizers themselves.

Alf II's box and the cover of its 92-page manual are rather plain. The *MS* box and 232-page manual come with a California sunset printed in full color on both.

Even the ads in national computer magazines give a very prophetic sense of the major differences between these synthesizers. *MS* has been promoted through full-page, full-color ads while *Alf II* is advertised by one-half or full-page black and white ads.

Could it be that, like their advertisements and packaging materials, the *Alf II* will turn out to be an unexciting but steady and reliable Plain Jane and the *MS* a young Marianne full of razzmatazz and promise — a beige Chevrolet Impala versus a Masarati Quattroporte?

THE PHYSICAL SYSTEMS

Because the Apple's speaker is designed only to provide beeps and clicks to prompt the user, any serious music

synthesizer is connected to an external stereo system by two cables, one for each channel. Therefore, both the *Alf II* and the *MS* consist of software on disk and output interface cards that generate tones of the frequencies and durations specified by the software and send these tones to a stereo system that plays them just as if they had come from a phonograph or tape player.

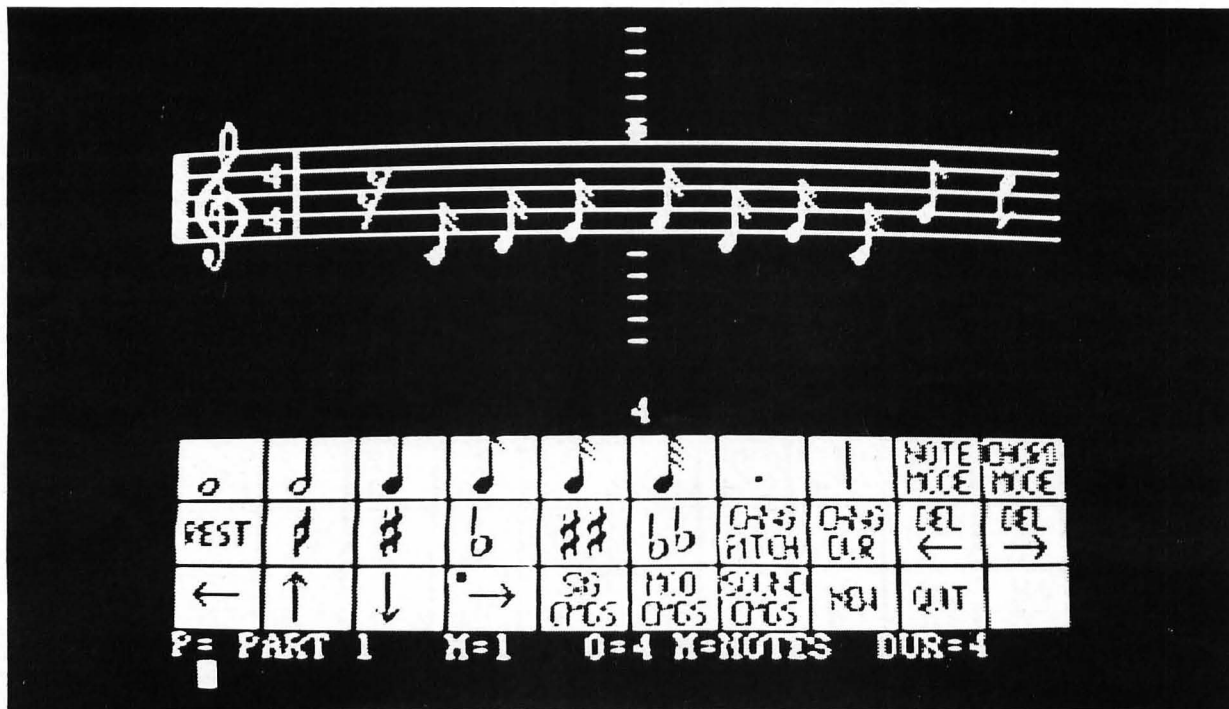
Alf II's hardware is relatively simple. Because it is limited to synthetic sounds, does not attempt to simulate real instruments, and its tones are increasingly inaccurate in the higher ranges, *Alf II* needs only one 3" by 4" card. *Alf II's* software is all written to send its output to slot 4. However, if the board is in any other slot (not 0) changing Line 10 in the underlying programs to 10 SLOT = N (where N is the actual slot number) configures to the system.

MS, if we can take their ads at face value, offers the sounds of every instrument in the orchestra. It requires two boards, each 3" high by 7 1/2" long. Because they are connected by a cable at the top, they must be in adjacent slots (not 0). Since the boards send

continued on next page

Figure 2

Showing the first ten notes of Bach's first *Invention*, the editor used by Mountain Hardware's *MusicSystem* can handle eight octaves, but will display only four octaves at a time. The number 4 between the note cursor and the main menu indicates that octave number 4 is currently at the center of the screen. Its main menu, which, like *Alf II's*, is operated by game paddle and button, can also be used to call three supplementary menus.



continued from previous page

a signal to the software telling it which slots they're in, there are no default slots, nor is there any need to change the programs if they are moved to other slots.

Alf II's board is connected to the stereo by a pair of cords. *MS's* output goes through a rubber strain relief that fits into one of the slits on the back of the Apple to the phono jack receptacles about three inches behind the Apple.

Although both synthesizers come with two disks of software, *Alf II* can only use one, while *MS* effectively uses four. One of *Alf II's* disks contains programs that use Integer BASIC, while the other disk is for systems using Applesoft. *MS* operates with either Integer or Floating Point BASIC automatically. Its two master disks are recorded on both sides, so you'll need four disks for your working copies.

Before going into detail about their respective systems, both manuals lead the reader step-by-step through the entering, editing, and playing of a simple tune. This may take the first-time user an hour or two, but it's excellent pedagogy and prepares the user for the complicated sections that follow. It takes some time to learn these systems (especially *MS*, which is incredibly

complex), but both of the instruction manuals are intelligently organized, clear, and accurate. Any user who knows the rudiments of musical notation and has reasonable proficiency with the elementary Apple commands (loading, saving, calling for the catalog, etc.) should have no trouble using either of them.

AN OVERVIEW OF THE SYSTEMS

Both *MS* and *Alf II* contain a music editing program and a program that plays music. *MS* also provides a program that merges song files to allow the player to play pieces that are too long for the editor to handle in one pass, a program that allows the user to enter the parameters needed to simulate the various instruments, and a program that takes the "instruments" that have been created and loads them into the player independently of the piece they will play.

Alf II offers in addition to its main player program, two that will play several tunes in succession, one in random order, the other in specified order. *Alf II's* editor produces a single tune file, named by the user, that can either be reloaded into the editor for further work, or played by the player

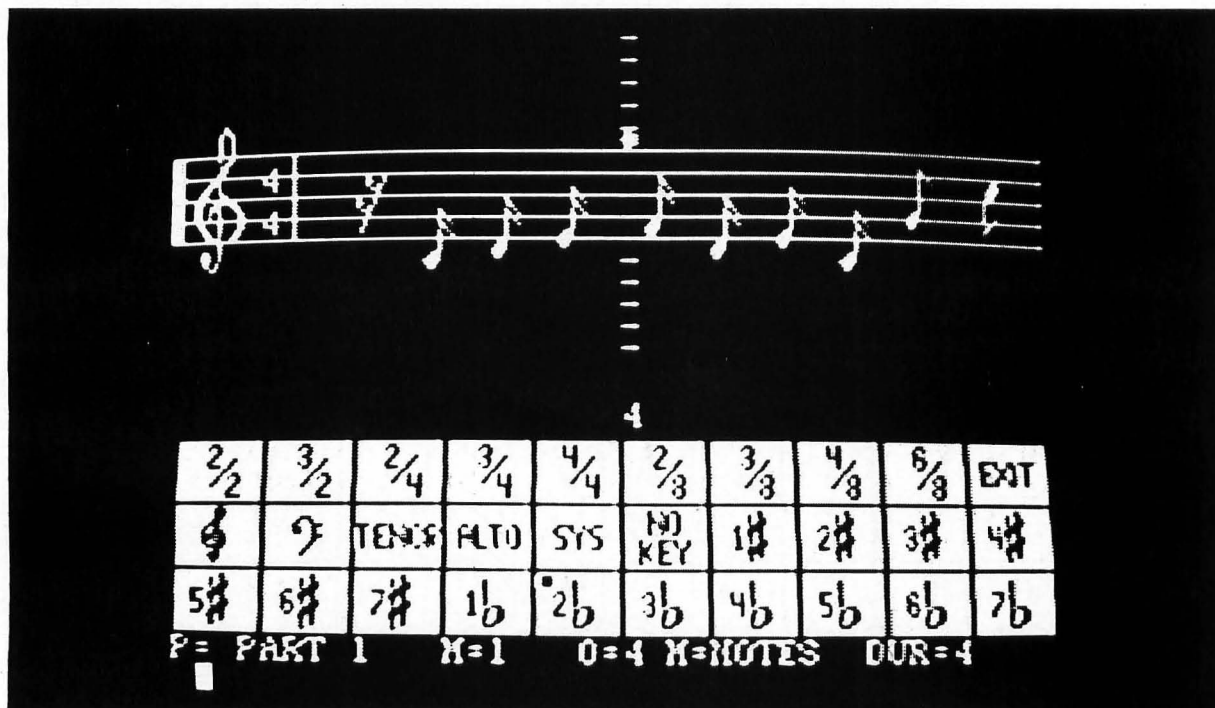
program. *MS's* editor creates a COMP (composition) file which also can be reloaded into the editor for alterations or additions, but it cannot be played directly. Its player requires that the COMP file be saved and then compiled into a PLAY file before it can be played. The PLAY file can be saved with either the same or different name. Although a PLAY file can be made from a COMP file, the reverse is not true. If you have only a PLAY file, there's no way you can alter it, although you can change its sound on playback by specifying different simulated instruments.

To help the user keep track of and properly use its programs and their many options, *MS* is operated through a series of menus which the user selects using the keyboard, the game paddles and their buttons, or a light pen supplied with the system. *Alf II* has only one menu, which is used by the music editor.

MS seems to be almost completely protected against accidental catastrophe. Before you can load a COMP file, which would erase any other COMP file in the editor's memory, you are asked to verify the order with a Y or an N. The same is true if the file name you are saving is

Figure 3

The *MusicSystem* has a signature commands menu for entering time and key signatures, while *Alf II* uses the keyboard.



already used on the disk and will be overwritten. *Alf II* has no such protections. But this is no different than Apple's standard operating procedure.

Loading *Alf II* is very simple. First the monitor asks you to list the slot the *Alf II* is in, and you respond with a number from 1 to 7. Then it instructs you to "Please relax for a moment" while the first program loads. The entire procedure takes just over 60 seconds. To use the music editor, you must first press ESC which will return you to BASIC, and then RUN ENTRY, *Alf II*'s name for its editor program. This adds another 15+ seconds.

Booting, loading, and running are inherently more complicated with *MS*, but are simplified by the menus and many prompts. First you must choose the correct disk out of four. Generally this will be the system disk 1, which contains the editor, the player, and the merger programs. After quietly booting (there is no annoying tone through the speakers like with the *Alf II*) the menu on the screen offers a choice of three programs. You can select the one you want by pushing the keyboard buttons 1, 2, or 3; by setting the cursor controlled by paddle 0 inside one of the three numbered boxes and pushing its button; or by aiming the

light pen at the correct box. I use a color television set rather than a monitor, and often have trouble with the light pen. Total time from boot to music editor is about 45 seconds.

THE MUSIC EDITORS

At the heart of a computer-driven music synthesizer is the music editor. While the player program is essential so that you can hear what you have written, the player alone would be all but useless; even a moderately priced tape player or phonograph will outshine a synthesizer in the reproduction of music. In any computer system the music editor is to the composer what a word processor is to a writer, except that a music editor's instructions go to a stereo (or a player piano) which performs the music. Like the word processor, a good music editor allows full provisions for making any changes the composer wants at a later time. Word processors use the computer's typewriter-like keyboard for input. Some music editors also do but are laborious to use. One could, of course, enter the note A above middle C as "440:1/4", meaning a pitch of 440 cycles per second and a duration of 1/4 note. Or a system might use the notation A4Q for note A in the fourth octave above some reference point and

held for a quarter note. In fact, some synthesizers do use this type of notation but it is tedious.

Both *Alf II* and *MS* (and a number of others) have a much more satisfactory entry system that uses standard musical notation as closely as possible. They display the standard five-line musical staff on the monitor and then let the user enter musical notes on the staff by manipulation of the game paddles and their buttons.

To test the two entry systems, I decided to put the same piece into each — J.S. Bach's little two-part *Invention Number One* in C major. It's only 22 measures long (although these are full of 1/16 notes) and lasts nearly a full minute when played at the usual speed.

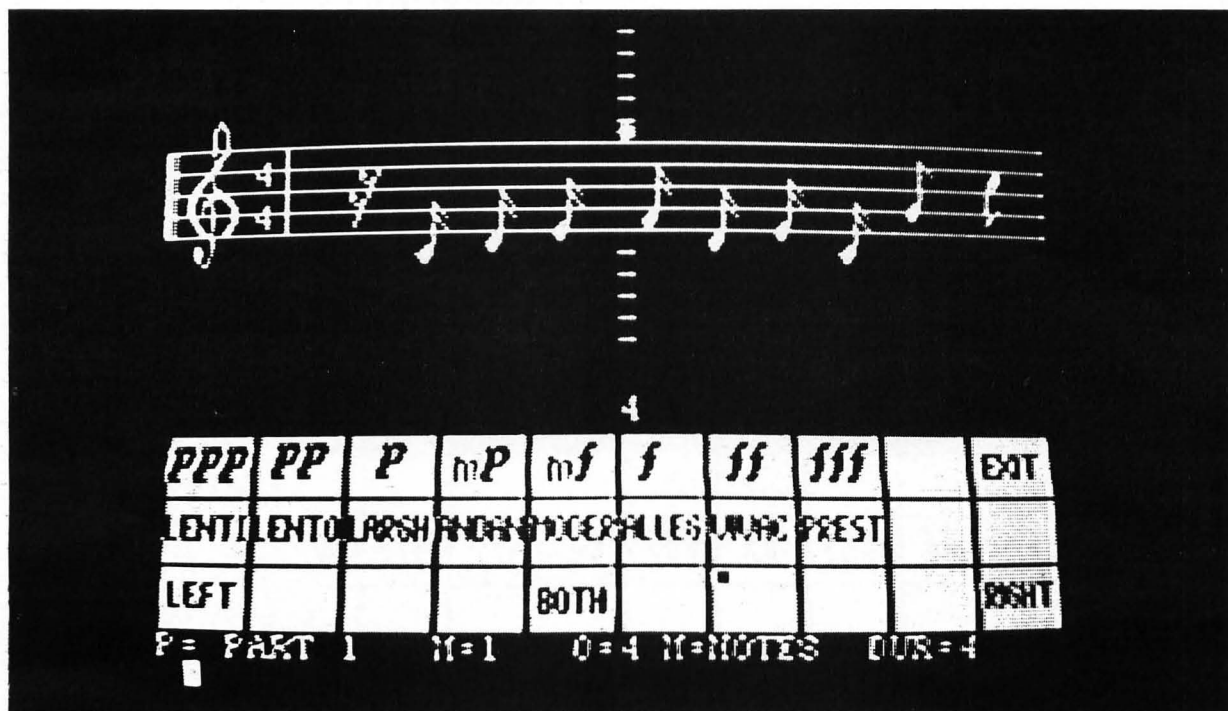
Entering *Invention Number One* with *Alf II* was easy. It took me an hour and a quarter, including error corrections, and another 15 minutes adjusting the volume and experimenting. It used 510 bytes of memory and a byte meter in the lower right-hand corner of my screen informed me that I still had an additional 5396 free bytes.

By contrast, it took me parts of three days to finally get this short, two-part work entered in the *MS*. My procedure was to work for an hour or so, get

continued on next page

Figure 4

While *Alf II* requires keyboard commands to control volume, speed, and choice of stereo speakers, *MusicSystem* allows the use of either the keyboard or standard musical terms selected by this menu.



continued from previous page

frustrated and go on to something else, and return later. I lost track of the time that I spent, but I think it was about six hours. There is no byte meter in *MS* so I had no idea how much memory was available, but I ran out of it the first time in measure 11. Then, because I wanted to save a few bytes for error correction, I deleted the notes in this measure and saved measures 1-10 on disk. I then entered and saved measures 11-16, then measures 17-22. Using the merger program, I then merged 1-10 with 11-16 and, on a second pass, 1-16 with 17-22.

Alf II's much greater entry speed comes from five differences:

1. Relative simplicity. Because *MS* has so many more options, its music editor uses four menus, which must be called, selected from, and exited. *Alf II's* editor uses only one menu, which is always displayed.

2. Audible prompts. When you're using *Alf II*, you hear a click each time your paddle moves the vertical cursor one note on the staff. When you push that button to enter the selected note, *Alf II* plays that tone through your stereo speakers. It's possible to enter a string of notes without taking your

eyes from the score. *MS's* editor is silent, so you must shift your attention from the score to the screen for each note.

3. Error correction. If you realize that you've entered a wrong note (in either system), you set the menu cursor over a backspace arrow to return the note entry cursor to that note. (*MS* also allows you to use the light pen or a keyboard control.) Once there, *Alf II's* cursor lies on top of the wrong note, and entering the correct note deletes the wrong one automatically. *MS*, on the other hand, inserts the correct note ahead of (or behind) the wrong one, and a second step is needed to delete the old note.

4. Cursor control. *MS's* note entry cursor moves only vertically. Horizontal movement from note to note (only ten of them are displayed at one time) involves shifting all of the notes one place at a time. Each of these one-note shifts requires that all the notes on the staff be repainted on the screen, a procedure that takes almost two seconds even when the repeat key is held down. It doesn't seem like a lot of time, but it adds up. *Alf II*, has a note cursor that will move across the screen (displaying 18 notes) does not repaint the notes, and moves just as fast as the game pad-

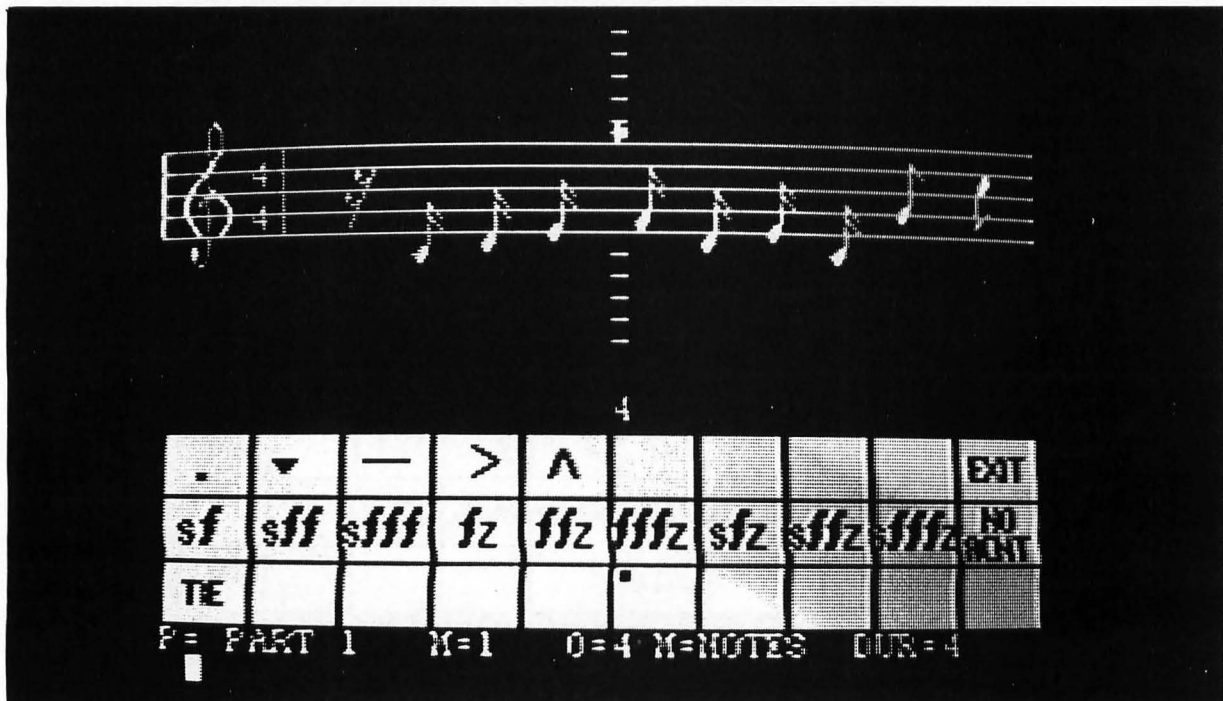
dle button can be pushed.

5. Measure bars. *Alf II* provides automatic measure control while *MS* does not. Let's say you've set the time signature for 3/4 time. At the end of three 1/4 notes, or six 1/8 notes, or the appropriate combination, *Alf II* will automatically enter a measure bar on the staff. If you missed a note, entered one too many, or kept on with 1/16 notes after the composer switched to 1/8 notes, you'll know it right away. With *MS* you must keep your own count and enter the bars either by the editor's main menu or by M on the keyboard. As a result, it's possible to have a part whose ending measures look right, but which will play out of synchronization with the other parts because of an incorrect note several measures before. If (and when) that happens, the only way to find the wrong note is to start at the beginning of the piece and either check every note against the score, or to count beats in each measure over again.

In order to have a range of eight octaves and still show a display that's large enough to see, *MS* puts only four octaves at a time on the screen. Thus an octave number 3 indicates that, although the screen appears to display two octaves below and two above mid-

Figure 5

Although the *MusicSystem* will play the staccato, staccatissimo, tenuto, etc., accents entered from the top row of the sound command menu, the sforzando series in the second row can be entered into the program, but cannot be played back by the current version of the player program.



dle C, it actually shows three below and one above. Notes outside the displayed range are shown as arrows pointing to them off the screen. If you accidentally turn paddle 0 too far in either direction (to change the octave number) and fail to notice the new number, all subsequent notes will be transposed one to three octaves higher or lower than you want. It is difficult to catch the mistake until you proof-read the piece or play it. This happened to me three or four times, and I had to reenter several measures. This octave shift feature is useful, but its control via the same game paddle that's also used to control the note cursor is extremely frustrating. Either keyboard or menu control of the octave shift would be much better.

Entering trills also posed a problem with *MS*. I tried entering three 1/32 notes played as triplets (in the duration of two 1/16 notes). Unfortunately *MS* doesn't allow triplets. I tried playing four 1/32 notes, but the trill sounded awkward so I let it go.

MS does allow us to enter several notes together in the "chord mode," but all must be of the same duration, which won't work for contrapuntal music. To do this you must add keyboard command ADDP (add new part) PART 2.

Another problem, as any veteran *Space Invaders* player knows, is in the Apple's push buttons. They just don't always register and *MS* has no audible signals of any kind to let you know whether the note has been missed.

Unfortunately, with *MS* you cannot hear the piece while the editor is in use. *MS*'s editor cannot call either its player or the disk's catalog. While you probably don't care to listen to a Bach piece every measure or two while you're entering it, you certainly will want to when you compose that great symphony.

MS creates a file the player can use out of the COMP file made on the editor. The player menu allows us to change instruments (organ is the default) and also which speaker each part will play through. Following a prompt from the screen, you swap the instrument files disk for the song files disk and wait while the instrument file is loaded and the piece is played. Either before or after playing you could have saved it as a compiled and defined play file, which would thereafter play almost without delay. If, however, you want to change even one note, you must reload the COMP file into the editor, make the change, recompile it, reload the instrument file disk, etc.

By contrast to *MS*, *Alf II*'s music editor is a joy to use. *Alf II* has only a six-octave range, so all the pitches can appear at once. Like *MS*, there is a choice of the order many of the preliminary instructions are entered.

Alf II clicks for each note to help you find the correct one quickly and sounds the note through the stereo speakers when it's entered. Also, *Alf II* has triplets and can play them correctly, so you have the ability to do trills. If you wish to hear a measure you have just entered you can do so immediately. *Alf II*'s editor can call the player directly by the command PLAY.

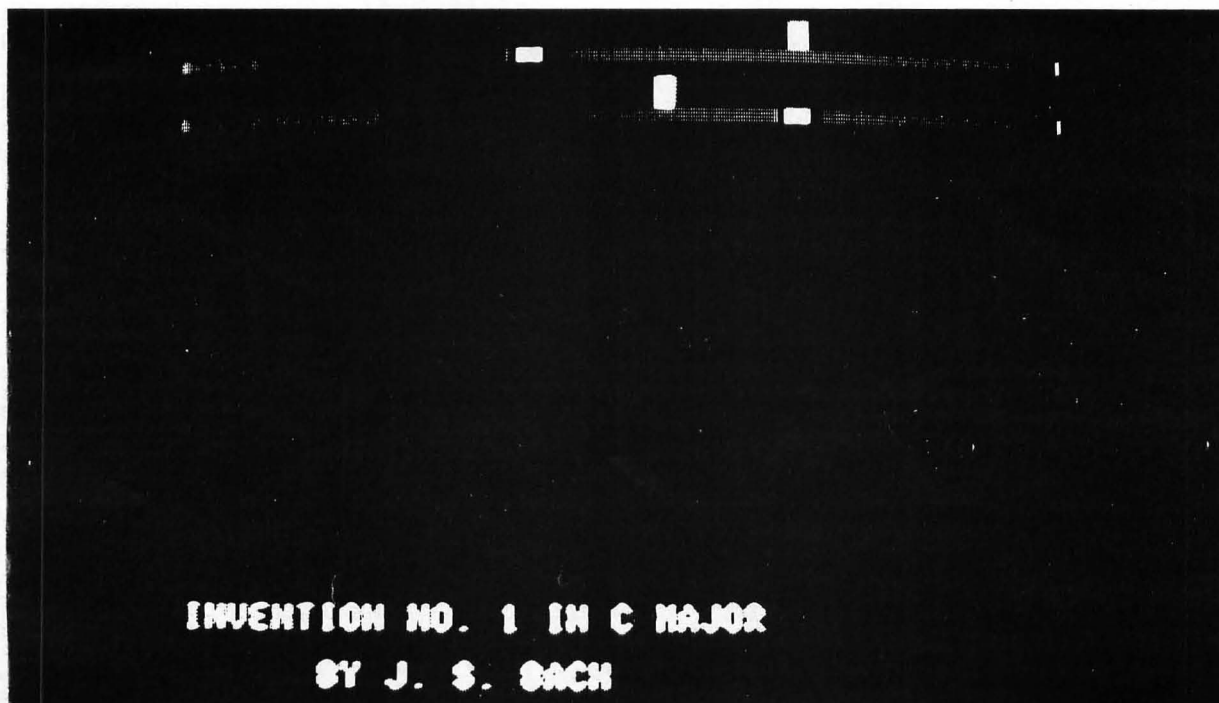
Alf II has two other very useful features that *MS* lacks. Its accidentals follow conventional musical usage, and it utilizes BASIC subroutines. Unlike *MS*, which does not recognize measure bars, *Alf II* will continue an accidental (a sharp, flat, or natural not in the key signature) for the remainder of the measure. *MS* requires that you enter the accidental sign for each note because standard printed music shows the accidental only the first time it is used in a measure. This is another potential source of error.

Alf II's ability to enter music into a subroutine saves both entry time and storage space when portions of a piece

continued on next page

Figure 6

While a piece is playing, *Alf II* displays in color a stylized keyboard for each part with a small rectangle to represent middle C and a bouncing square to show the note being played.



continued from previous page

are repeated. If you want to program a round, for example, you can enter the tune (up to nine parts) just once in a subroutine. The main program then will simply call that subroutine at the desired intervals as many times as you wish. The command is CALL:n, where n is the subroutine number; 99 of them are available. And a subroutine can call itself, which gives you an infinite loop that will force your audience to listen to your opus magnus forever. And *Alf II* offers 1/64 notes, while *MS* is limited to 1/32 notes. Neither has quintuplets (like triplets, but five notes played in the space of one for more elaborate trills).

THE MUSIC PLAYERS

Both *MS* and *Alf II* have a music player program that takes the instructions entered into the music editor and uses them to operate one or more wave-generating oscillators, whose output goes to a stereo amplifier just as if it had come from a phonograph or tape player. That's why both have boards inside the Apple. *Alf II*'s player will generate a maximum of nine notes at a time, while *MS*'s can play 16. Like its editor, *MS*'s player is loaded via the main menu. *Alf II* also lets you load

the player via a menu but I copied the play program onto a disk by itself, so my command is RUN PLAY. When they are running, both ask for the name of the song file you want. This is then loaded from the disk and played.

While the music is playing, *MS* displays the name by which the piece was saved. *Alf II*, on the other hand, shows the four title lines entered by the user into the editor, i.e., Beethoven's Ninth Symphony Conducted By (Your Name). *Alf II* also gives you something to watch during playback. For each voice in the piece, *Alf II* displays a Low-Res line, in color, on the screen. Middle C is shown on the line by a white rectangle, while a colored square, representing the note being played, bounces back and forth along the line. It's cute and it helps distract the listener from the fact that the music is definitely synthesized and is not played by recognizable instruments.

With certain limitations, both *Alf II* and *MS* will allow you to direct each part to either the right or left speaker. *MS*'s editor also allows intermediate positions — say, 40% right and 60% left — to give a true stereo effect. Unfortunately this feature cannot be used by the player in the current version, although future versions may imple-

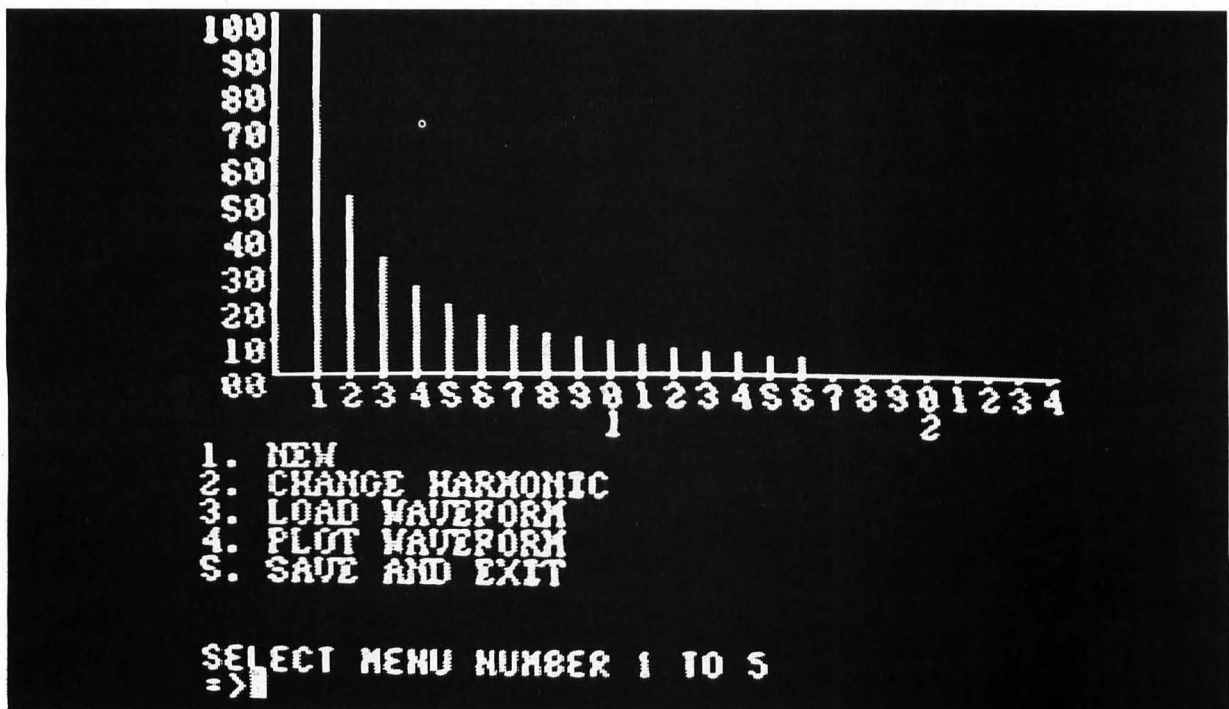
ment it. *MS*'s editor also allows entry of certain dynamic accents in the program (sforzando, for example) that its player simply cannot handle at the present time. Accents which can be entered easily and directly through *MS*'s sound commands menu (that its player can handle) include staccato, tenuto, and percussive. *Alf II*'s method of providing these require changing the envelope (see below).

Neither *MS* nor *Alf II* provides for an automatic accent at the measure bars. If you want it, you have to increase the volume for that note and then decrease it for the text (*Alf II*) or call the sound commands menu, enter an accent, and exit the menu back to the main editor menu (*MS*). There must be a simpler way.

Alf II markets several preprogrammed disks for \$14.95, each with a dozen or more tunes, while *MS* offers none. When I asked a Mountain Hardware representative I met at a trade show, why his firm doesn't have preprogrammed music disks, he replied that they have many programs to handle in comparison to *Alf II*'s one main interest. However he did indicate that if *MS* users will send Mountain Hardware a disk with a tune they've entered, the firm will return a disk to them with

Figure 7

The *MusicSystem*'s Instrument Definer program allows the user to select the relative strength of up to 24 overtones to simulate those used by actual instruments. Below is an exponentially declining overtone series whose weighted frequencies are added to produce the saw tooth wave of Figure 8.



several tunes entered by other customers. *Alf II*, unlike *MS*, not only doesn't provide preprogrammed instrument simulations, but it won't even suggest any settings for its parameters beyond the default ones. I called Alf Products, Inc. in Denver and was told I would just have to experiment. I asked if the default settings were chosen because they imitate a piano, and was told they were chosen not for what they may simulate, but chosen because they sounded good.

Just as *Alf II's* music editor is its strong point, *MS* almost completely overshadows its rival when it comes to imitating the actual instruments. Included in the *MS* package is a disk containing software which promises to imitate the following instruments: clarinet, bass, gong (chime or bells), wood block (xylophone), brass (tuba or French horn), piano, organ, cymbals, and clavichord. I tried out the first five with a simple major scale and was reasonably convinced. Each of these sounded much more realistic than an electric organ whose stops are set for the same instruments. However, the drum sounded weak and the cymbals sounded like the muted cymbal set of a jazz drummer, not at all like the grand clash of those used in a symphony orchestra.

I decided that the organ and piano were important enough to justify entering pieces composed strictly for them. The piece I entered on the organ sounded like it was being played on a good pipe organ in a large church. Unfortunately the piano piece I entered, Béla Bartók's *Snow in the Market Place*, didn't turn out as well. Because this piece is written almost throughout for double notes in each hand, it also gave me the opportunity to use *MS's* chord mode, in which all the notes of a chord are entered at the same time and are separated into parts automatically at the time of playback. The chord mode of entry worked just fine, and should be very useful for pieces that consist largely of chords. Regrettably, it sounded just awful.

Suspecting that the chord mode was at fault, I reentered a few measures using separate parts — no improvement. Finally I slowed the tempo down and found the problem — speed. *MS's* piano cannot be played very fast before its electric origins show through. Hoping the problem was just in the piano program, I tried the same fast pieces on the organ and the clavichord. No luck. *MS* apparently cannot play fast passages without sounding like an electric organ.

Despite their advertisements which

usually feature a violin, *MS's* instrument file disk doesn't have a violin program on it, nor does its instruction manual mention strings.

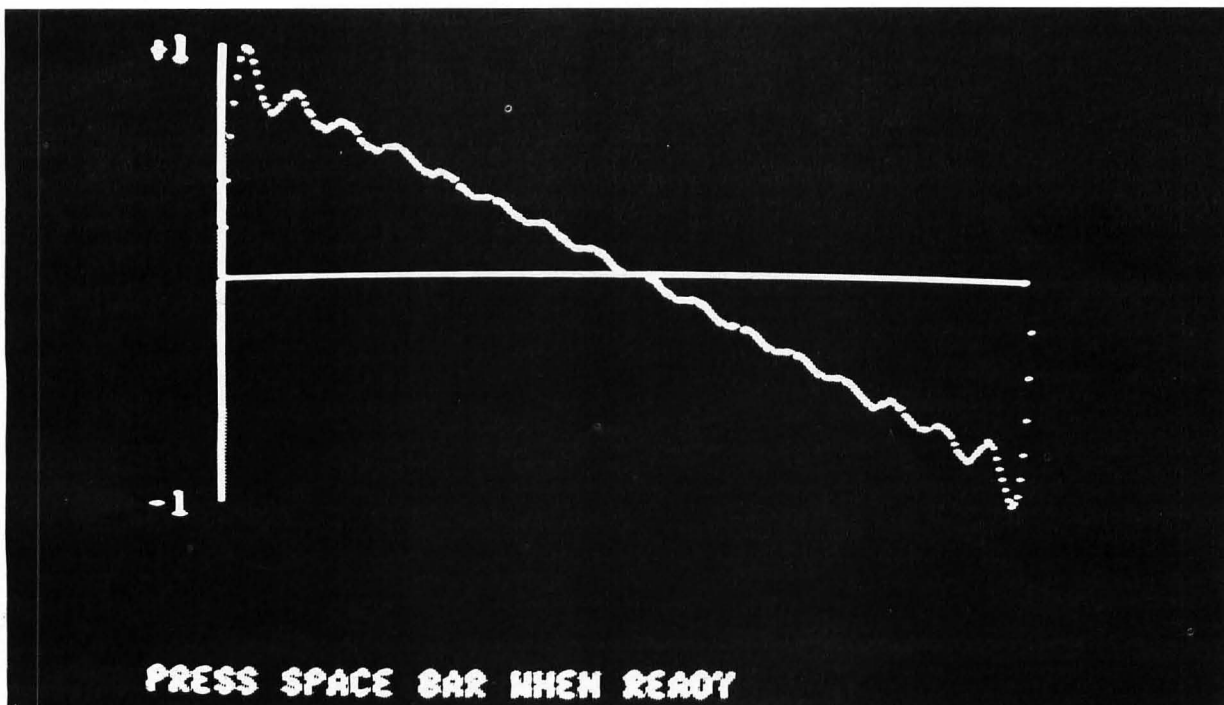
When I asked Mountain Hardware about a program to simulate violins, I was told they used to carry a strings program they thought was great, but too many customers complained that it didn't sound like a violin. I asked for the parameters so I could try the simulation myself. I was refused. When I persisted, I was told it had been designed for an earlier version of *MS* and wouldn't play on the current one. I mumbled something about the violin in those full-page ads and then, realizing I was getting nowhere, gave up.

I very quickly realized that the current version of *Alf II* can never succeed at imitating actual musical instruments. *MS* can, and does, imitate some of them, but the violin may just be too complex.

I'll try to explain in as nontechnical a manner as possible, the differences between *MS* and *Alf II* that allow the former to simulate real instruments and prevent the latter from doing the same. Both *Alf II* and *MS* give the user control over the "envelope" (the pattern of the build-up of a note's volume to its maximum and then its fade), but

continued on next page

Figure 8



continued from previous page

only *MS* permits programming the latter. For a further explanation of how the physics of music relates to programming the envelope, see my article *The Tone Envelope* elsewhere in this issue.

Alf II allows the user control over a four-stage envelope: (1) attack, where the note builds up to full volume, (2) decay, where the volume drops a bit to the level called (3) sustain, in which the volume is constant until it begins to fade away in the (4) release stage. In keeping with *Alf II*'s overall simplicity, the envelope commands are inserted in the music editor along with the notes. For example, the command `DECAY:n` sets the rate of initial fade and can produce a staccato effect. Although each of the envelope commands must be made before the first note is entered (*Alf II* has a series of prompts to help at this point), they can be changed at any point later.

Recognizing the actual envelopes is generally much more complicated than the simple model used by *Alf II*. *MS* allows the user to plot up to 15 points on a graph (with the volume on the vertical axis and time on the horizontal) to get the volume changes of the wavering during the attack stage, and 15 more

points on another graph to get the pitch changes during this stage. Instead of a sudden drop in volume at the end of the attack stage, *MS*'s model goes right into the sustain stage, which is not necessarily constant. In the last stage, which *MS* calls the decay stage (the same as *Alf II*'s release stage), the volume drops rapidly to nothing. In addition to allowing the user to specify the amount of time this will take (as *Alf II* does), *MS* also permits the choice of a linear or a logarithmic rate of decay.

Instead of putting the envelope instructions in its editor, *MS* has a separate program of its own on disk called an Instrument Definer. It allows the user to create a simulated instrument that can be saved independently and called by the player program to play any piece whose notes have been saved in a song file. It's also possible to save a song file and an instrument file together and have almost instant playback.

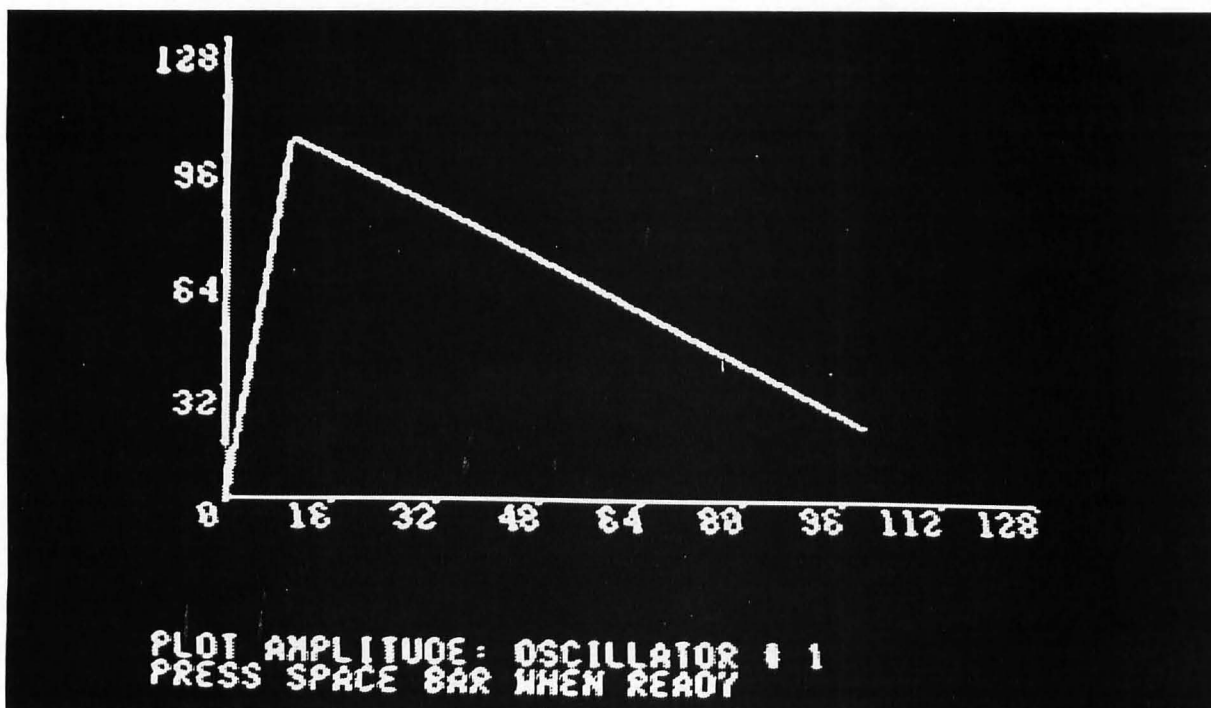
MS's Instrument Definer also gives the user considerable control over an aspect of an instrument's timbre that may be even more important — its pattern of overtones. Using the Instrument Definer, it's possible to specify the volume level for up to 24 overtones

as a percentage of the fundamental. Thus a flute might be simulated by: fundamental, 100%; first overtone, 15%; second through sixth overtones, 3% to 5%; and nothing thereafter. A French horn might be best imitated by: fundamental, 100%; first overtone, 75%; second, 50%; third, 25%; etc. (These examples are rough and are based on small graphs in *The Harvard Dictionary of Music*.) As you enter these percentages, *MS* displays them on a graph to give a visual idea of what you are doing. Then at any time you wish, you can have the screen display the entire wave form that will be used after these overtones are totalled to make one complex note. If you're trying for a square wave, you can observe the results until you get what you want.

MS also offers a very useful feature in its Instrument Definer which is the ability to hear the note as it's created. Although its music editor cannot call its player, its Definer can. While you're creating an instrument, a simple scale is played through the stereo speakers. Each time you make a change in the instrument you are creating, you can hear the effect of that change immediately on its timbre. If you don't like the C major scale that's offered as a default, you can transpose it to any

Figure 9

The rate at which a note rises to its full volume and fades away to nothing is also an important part of determining an instrument's timbre. This pattern is known as the instrument's envelope, a small portion of which is shown in this figure.



other key. If you don't want a scale, *MS* allows you to load any short tune that you want. The ability to hear without delay what you are doing is extremely useful in creating an instrument, and the control over what you're hearing is a nice touch.

Because *MS* has 16 oscillators (wave generators that send out phonograph-like signals to the stereo), the overtone series could be created by assigning each oscillator to one overtone by using a separate envelope to determine the volume of each. In fact, it's perfectly possible to use *MS* this way, but because all 16 oscillators would be used to produce one note, only one note could be played at a time. Instead, *MS* adds together the fundamental and its overtones — up to 24 with the weights you've selected — to provide one irregular wave that the ear picks up as if all of these tones were being produced simultaneously. This complex wave then determines the frequencies that will sound through the stereo, and the envelope you create determines the build-up and fading away of these frequencies. Unfortunately *MS* has had to compromise here. In practice, the higher overtones decay at a faster rate than the lower ones, and theoretically each should have its own envelope. However some compromises are inevitable, and even without separate envelopes for each overtone, *MS*'s control over these overtones is a tremendous advance over *Alf II*'s use of a simple square wave for everything.

In theory, *MS*'s 16 oscillators could handle 16 parts at once (versus *Alf II*'s nine) and produce 16-part polyphony or truly massive chords. In practice, despite the ability to add overtones to get irregular wave forms, one oscillator doesn't seem to be enough to simulate any but the simplest of instruments. The instrument programs supplied with *MS* use two oscillators each for the organ and clavichord, and three each for the clarinet and French horn. This limits the former two instruments to eight parts and the latter two to five parts; but by themselves, these limitations may not be too serious.

The piano simulation uses three different instrument programs: a low-range for octaves one through three, a mid-range for octaves three through five, and a high-range for octaves four through six. Respectively, these require one, two, and two oscillators for a total of five. Thus if the complete piano keyboard must be available, the maximum chord size is three notes, a serious limitation on an instrument capable of ten notes. Fortunately,

some pieces will stay within the middle three octaves, allowing eight notes to be played at once.

MS has a PRINT command that will allow you to print a score of your opus magnus one part at a time, while *Alf II* does not offer this option. Ironically, I found that I can print *Alf II*'s scores, but not those created by *MS* as the printing option works only with the Apple Silenttype Printer. (I have an earlier version put out under Trendcom label.) This will allow you to print whatever is on the screen as long as it remains in Hi-Res memory. When I type INT, *Alf II* exits its editor and returns to BASIC, where I can use a CALL to print the Hi-Res memory. *MS* won't allow me to exit its system without turning off the computer completely.

And so...after this long analysis of the *Alf II* vs. the Mountain Hardware MusicSystem, the reader is entitled to ask, "Has either synthesizer helped with your original goal — learning new folk tunes so you can play them by ear on your dulcimer?"

The answer should be obvious...Of course not. I don't have time to practice the dulcimer any more. I'm having far too much fun composing music. Perhaps someday you'll hear the Philharmonic perform my *Opus 1*. It won't be a full symphony (no strings), nor can it be a concerto (only a slow piano, so I can't have any allegro movement). But it may very well be the world's finest quartet for pipe organ, celesta, tuba, and xylophone. I'd add a harpsichord, but I've run out of oscillators.



World War II rages across Europe... Castle Wolfenstein is occupied by the army of the Reich and converted into battle-front headquarters. You have been captured and brought to the Castle for interrogation by the dreaded SS. From a hiding place behind the stones of the dungeon a dying cellmate produces a Mauser M-98 pistol fully loaded with ten bullets and gives it to you. Your new mission: Find the Nazi war plans and escape Castle Wolfenstein alive.

Castle Wolfenstein™ is an action adventure game from MUSE demanding fast thinking and quick manual response. Use game paddles, joystick, or your computer keyboard... Castle Wolfenstein™ generates an unlimited variety of castle layouts, each more difficult to escape than the last. For the Apple II and Apple II Plus with 48K. \$29.95

CASTLE WOLFENSTEIN™
by Silas S. Warner

MUSE SOFTWARE™

330 N. CHARLES STREET
BALTIMORE, MD 21201
(301) 659-7212

Apple II is a trademark of Apple Computer Corp.
© MUSE Software Inc.

Call or write for information and the name of your nearest MUSE dealer

THE APPLE SOFTWARE DIRECTORY

THE APPLE II RESOURCE DIRECTORY

from WIDL Video, 5245 W. Diversy, Chicago, IL 60639. Suggested retail prices: *Software Directory Vol. 1/Business*, \$5.95; *Vol. 2/Games*, \$4.95; *Vol. 3/Education*, \$5.95; *Resource Directory*, \$5.95; all volumes in one book, *The Apple II Blue Book*, \$19.95.

Review by Jon Voskuil

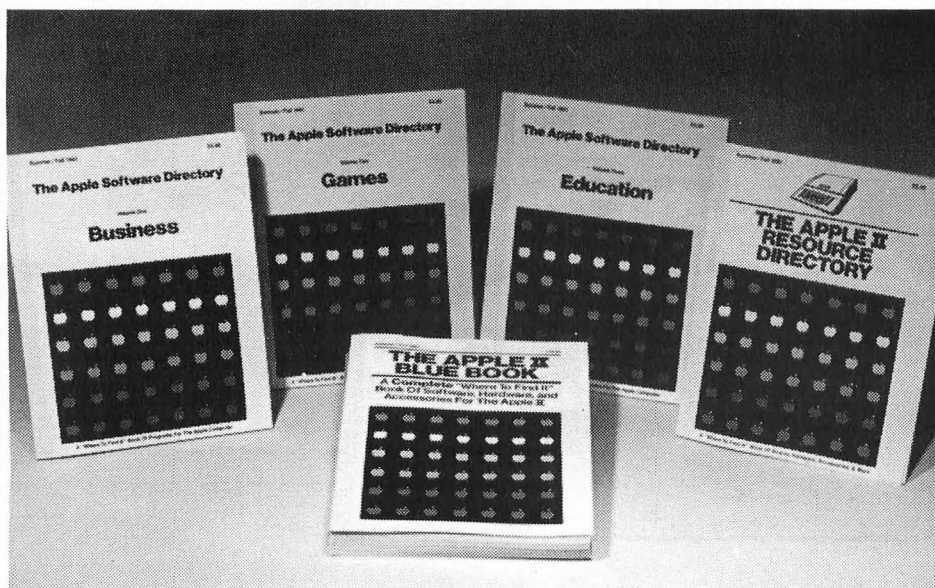
If you want to know what's available for the Apple, these books would be a good place to look. *The Apple Software Directory* comprises three volumes, each specializing in a separate category: business, games, and education. *The Apple II Resource Directory* focuses on plug-in boards, peripherals, and other hardware and accessories. Each volume is paperback, 8½ by 11 inches in size, 80 to 160 pages long, and costs \$4.95 or \$5.95.

Probably very few people could make use of all of these books, but most Apple owners could benefit from owning one or more of them. Some of the material is shared among several volumes. The 19-page index to software suppliers is common to all four; and all but the *Games* volume contain the same 37-page section of utility pro-

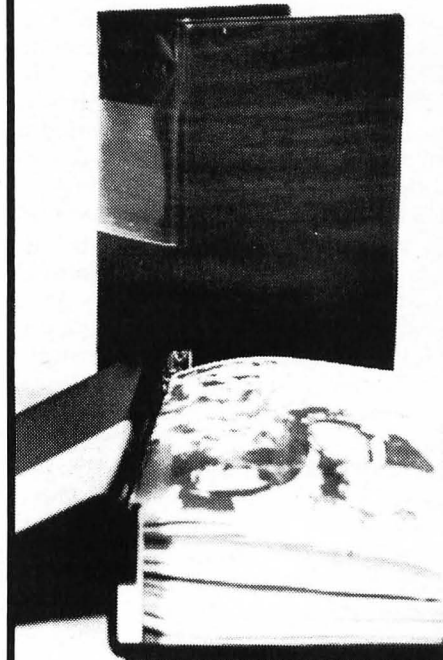
grams and programs for data base management, word processing, and graphics.

The books do not evaluate the products which they catalog; the information is that furnished by the suppliers and publishers. All software is listed alphabetically by title, and in some cases grouped by type. The *Education* volume, for example, is divided into sections according to subject area. More extensive indexing would have been beneficial, especially in the *Business* and *Games* volumes.

As one would expect, the descriptions supplied for the games software are generally shorter than for the other types. The *Games* volume (considerably smaller than the others) also seems to be the least comprehensive — probably a consequence of the explosive and diverse nature of the games market. The *Education* and *Business* volumes, however, should be very valuable resources for Apple users with interests in those areas. And the *Resource Directory* includes, in addition to descriptions of a wide variety of hardware products, such extras as listings of Apple user groups and time-sharing networks. 5



Protect Your Investment!

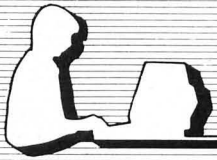


With SoftSide™ Vinyl Binders

Collectors! Protect your *SoftSide* back issues, Volumes I and II, or any publication of your choice, with these durable wood-grain vinyl binders with inside pocket and clear spine sleeve for easy identification. Holds and protects 12 back issues. A regular \$4.95 value. SALE priced at \$3.95*. FREE (while supply lasts) with the purchase of Volume I or II (12 issue collection of *SoftSide*).

SMALL \$3.95
8½ x 11 \$7.95

SoftSide™
6 South Street Milford NH 03055



Hardware Corner

by Edward E. Umlor

In the September issue, I wrote about some of the woes of the first-time drive buyer. That column and this one are basically lead-ins for a series on disk drives. At this time I would like to outline the series for you.

The first column in the series will cover the diskette: how the computer organizes the diskette for data storage, the purposes of the different openings in the diskette cover, the differences between soft-sectored and hard-sectored diskettes, and the differences between floppy diskettes and hard diskettes. The column will cover the magnetic media itself, some of the ways the surface can be flawed, and how the surface can be flawed in manufacturing it.

The second column will cover the disk drives for the 5 1/4 inch floppies: the different models (types) that are available, how they differ from each other, the FORMATTED data storage capacity, and a general discussion on how to pick the type that is best for you.

The third column is going to be on hard disk drives: what they are, how they operate, some of the different ways they are made compatible with present microcomputers, and a general discussion of the direction the technology of hard disks is going.

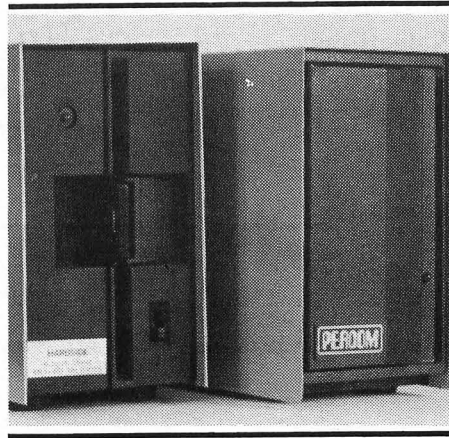
By the time the third column is published, there should be an influx of mail with questions, suggestions and comments. It would be very nice to get a two-way discussion going, as this is the way to get specific information to you, and it gets me doing my homework and extending my horizons as well. Please address your letters to:

Edward E. Umlor
RFD#1, Box 48A
Fitzwilliam, NH 03447

In this column I will cover some of the requirements for satisfactory disk system operation. There are several basic rules that apply to all systems. I will try to cover several of these and why they are necessary to get the best results with your disk drives.

One of the first areas that some people scrimp on is the amount of internal RAM in their machine. The ABSOLUTE MINIMUM amount that I would recommend is 32K. Many DOSs will not do single-drive backups with only 16K of memory. With the cost of

RAM down now, I would recommend that you upgrade your system to 48K or 64K of RAM. Some systems do not have a BASIC interpreter in ROM. These are able to address at least 64K of RAM. When you load BASIC into one of these machines, you will find the amount of RAM left is very close to the same amount as a system with 48K RAM plus BASIC in ROM. The reason for maximizing your RAM memory is to allow the disk system to do backups or copies, and run larger programs with the best TIME efficiency. The



faster your computer can operate, the more work per day it can do for you. Time is money, power, lights, heat, etc.

The number of disk drives in a system will make a vast difference as to the type and complexity of software you can run. In my humble opinion, a single-drive system is good only for learning the very basics of drive use and should whet the appetite for a multiple-drive system. A single-drive system limits you to smaller data storage programs that will access only drive 0 or 1, (whichever is the first drive on your system). You will find it time consuming and irritating doing backups on a single-drive system (insert source, then remove and insert destination, then remove and insert source, until the backup is complete).

To me, the smallest viable disk system has two drives. This will allow the program to reside on the first drive and a data disk with maximum data storage to reside in the second drive. Backups can now be done from the first drive to the second without con-

stantly changing diskettes. Here again, it is maximizing the throughput of the system for a given amount of time. I prefer a triple-drive system for my own use. It allows me to reproduce diskettes in a minimum amount of time and to use programs that utilize interdrive activity.

The actual number and type of drives will be dictated by the system you have purchased, the add-on devices designed for that system, and the actual data storage requirements that you have. For the average home computerist, a dual 35- or 40-track single-density system is adequate. If you are getting into programming for profit, doing your books, or club mail list, then a dual or triple 40-track double-density system is in order. If you are expanding into doing a large mail list or large accounts receivables/payables, you might want to get 80-track double-density drives or go all the way by getting hard disk drives. It really depends upon your requirements and pocketbook.

We have covered the basic hardware needs, now it is time for a shot at the software requirements. There are a lot of different DOSs (Disk Operating Systems) on the market today, and each one is different from all the others. Each DOS is designed to overcome certain problems in communicating to the disk drive, and to enhance the BASIC interpreter for that particular system. It is difficult to write in generalities about capabilities of DOSs, because each operating system is unique unto itself and the system upon which it will function. At the present time, CP/M is about the most universal DOS on the market today. Just as the number and type of disk drives chosen is a very individual thing, the DOS choice is the same. It really depends upon the type of software, system uses, and pocketbook as to which DOS will be best for your needs. I personally use NEWDOS80 zapped to double density for all my programming efforts. For me, it is the easiest to use, and has special features that allow me to build my programs rapidly.

Well I guess that is about all for now. This is OLE GRANITE KNOGGING spinning off for this month. I am looking forward to hearing from all you diskies out there. ☺

Coming next month in

SoftSide

TITAN

Are you prepared to run a mining operation on one of the moons of Saturn?

GAMBLER

Play the lottery, the wheel of fortune, the horse races or roll the dice — all from the screen of your computer.

ATARI DATABASE

The long-awaited complete listing.

Plus:

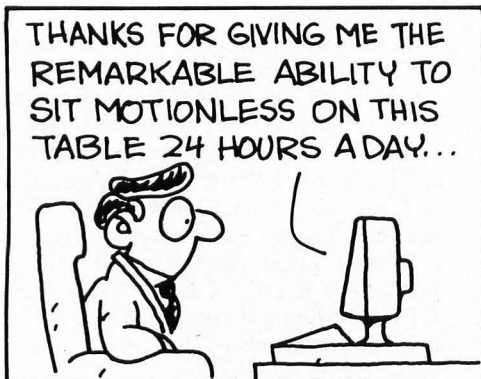
Aircraft Commander, Electronics Assistant, Design Master and much, much, more.

Advertiser's Index

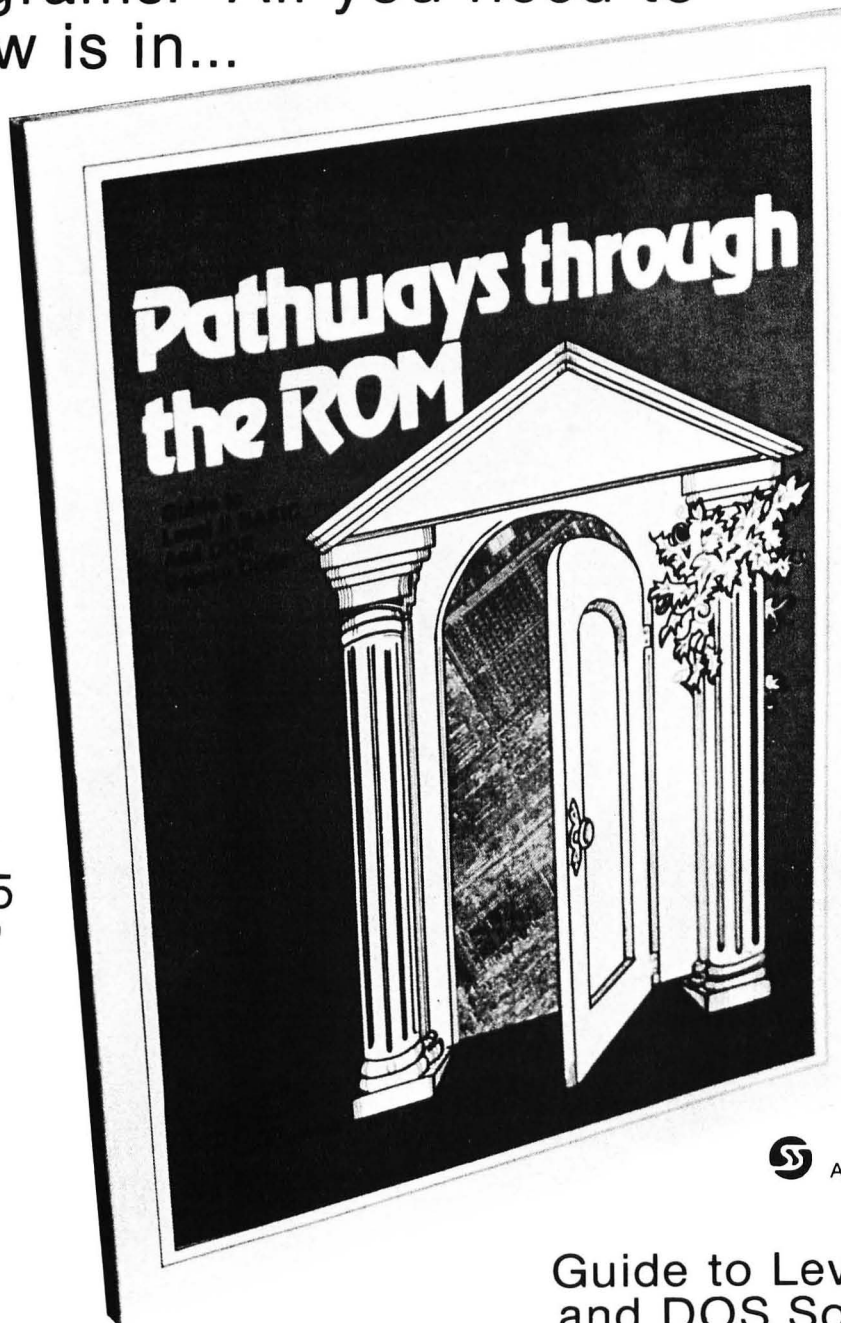
Adventure International.....	Cover IV
Amp Recording.....	72
Atari.....	14
BASF.....	2
Continental Adventures.....	73
Icom.....	59
Micro Ink.....	60
Muse.....	93
National Computer Shows.....	12
Ramware.....	13, 15, 18
SoftSide Selections.....	Cover II, 1, 4, 24, 30, 66, 80, 81, 94, Cover III
Spectrum Computers.....	65
TSE-HARDSIDE.....	22, 26, 28, 54, 80

MACHINE HEAD

BY SPYDER



Unlock the hidden power of your computer for fast and easy programming! Use ROM routines in your BASIC and Assembly Language programs! All you need to know is in...



ALL
ONLY
\$19.95
plus \$1 shipping

INCLUDES:

SUPERMAP

From Fuller Software (\$18.95)

**TRS-80
DISASSEMBLED
HANDBOOK**

by Robert Richardson (\$10.00)

HEX MEM

by John Phillipp
Monitor written in BASIC

**Z-80
DISASSEMBLER**

by George Blank



A SoftSide Publication

**Guide to Level II BASIC
and DOS Source Code**

Description of the contents of the Level II BASIC ROM by memory locations, by function, and in lesson format. Includes several BASIC and Assembly Language programs in listing format to examine and use ROM routines.



STONE of Sisyphus

INCLUDES 2 JAM PACKED DISKS OF DATA BUT
WILL WORK ON YOUR 1 DRIVE MICROCOMPUTER!

AVAILABLE ON DISK ONLY FOR:

TRS-80 MODEL 1 32K	012-0100	\$29.95
TRS-80 MODEL 3 48K	012-0100	\$29.95
Apple 2 Applesoft in ROM	042-0100	\$29.95
ATARI 40K	052-0100	\$34.95

 **Adventure**
INTERNATIONAL

A DIVISION OF SCOTT ADAMS, INC.
BOX 3435, LONGWOOD, FL 32750
(305) 862-6917

ORDER FROM YOUR FAVORITE DEALER
or CALL TOLL FREE (800) 327-7172

SHIPPING & HANDLING ARE EXTRA. PRICES SUBJECT TO CHANGE WITHOUT NOTICE

We want to take you on a journey — a journey into an age undreamed of. When a man's worth was measured by his courage and cunning. With **STONE OF SISYPHUS**, we have re-created the wonderment of that ancient era. The **Maces & Magic Series** allows you to interact with the adventure on an intensely personal level. You create your own character, giving him (or her) the attributes of strength, IQ, constitution, dexterity and charisma. You then arm and prepare your creation for the challenges ahead. Amazingly, your character will evolve and grow as the journey progresses. Prepare yourself — breathe deeply, and step into the enchantment of **STONE OF SISYPHUS** and the **Maces & Magic Series**.

ART COPYRIGHT
1981 RAYMOND BAYLESS