

SLCC Journal

San Leandro Computer Club '85 Special Edition

\$3.00

**Legend of
Stargunner**

Master Pencil

**A View From
Japan**

RamTalker

**Cartoon
Machine**

Flitterbug

& More!



Blackjack in Action!

3E SOFTWARE & SYSTEMS

ATARI...POWER WITHOUT THE PRICE...ATARI...POWER WITHOUT THE PRICE

	<u>3E Price</u>	<u>(S & H)</u>
Atari 130XE Computer.....	\$ 139.95	(10.00)
Atari 1050 Disk Drive.....	179.00	(10.00)
Atari 1030 Modem with software.....	74.95	(5.00)
Tech Sketch Light Pen with MicroIllustrator..	44.95	(3.00)
Legend 1080 Dot-Matrix Printer.....	249.00	(10.00)
CompuServe Starter Kit.....	29.95	(3.00)
Batteries Included Homepak.....	37.95	(3.00)
Broderbund Print Shop.....	34.95	(3.00)
Broderbund Print Shop Graphics Library.....	24.95	(3.00)
Epyx/Lucasfilm Ball Blazer.....	29.95	(3.00)
Epyx/Lucasfilm Rescue at Fractalus.....	29.95	(3.00)
MicroProse F-15 Strike Eagle.....	27.95	(3.00)
subLogic Flight Simulator II.....	39.95	(3.00)
Tronix S.A.M.....	49.95	(3.00)
Adventure International QuestProbe w/ Hulk...	24.95	(3.00)
First Star Spy vs. Spy.....	27.95	(3.00)
O.S.S. Action!.....	69.00	(3.00)
O.S.S. Action! Toolkit.....	24.95	(3.00)
O.S.S. BASIC XE (for Atari 130XE.avail.July.)	69.00	(3.00)
O.S.S. BASIC XL.....	49.00	(3.00)
O.S.S. BASIC XL Toolkit.....	24.95	(3.00)
O.S.S. Mac/65.....	69.00	(3.00)
O.S.S. Mac/65 Toolkit.....	24.95	(3.00)
O.S.S. DOS XL.....	24.95	(3.00)

3E Software & Systems
P.O. Box 178
931 A Street
Hayward, CA 94541

Phone (415) 537-3637

Open Tuesday thru Saturday 10:30 A.M. to 6:00 P.M.

Located on "A" Street between Mission Blvd. and Main Street
Next to Payrite Drug Store

Parking in Rear - Free Meters on Saturdays

We accept MasterCard, Visa, and American Express Credit Cards

Mail & Phone Orders Accepted. Please include Shipping & Handling
Charges (S & H) on shipments. CA residents add 6.5% Sales Tax.

Table of Contents

PROFILES AND REVIEWS:

4	BETTER BASIC	- Bill Wilkinson
7	ANTIC'S 520ST	- Jack Powell
8	COMPUSERVE'S GEM CONFERENCE	- Compuserve
12	DOS 2.5	- Atari Corp.
28	A CRY FOR HELP	- Makoto Nagata
37	ATR-8000 PROGRAM LIBRARY	- Bill George
38	LOIS IN ATARILAND	- Lois Hansen
41	RECURSION IN LOGO	- Lois Hansen
46	TISBY'S TOPS	- Tom Tisby
46	SARGE'S SELECTIONS	- Sgt. Slaughter

UTILITIES:

9	FORMATTER	- Kenneth J. Pietrucha
10	MASTER PENCIL	- Joe Eash
17	SYSTEM KEY TO THE KEY SYSTEM	- Mike Sawley
19	TOOL KIT DEMO	- Jim Warren
19	XLATOR	- Bill Eash
25	DISK MAP	- Michael Curry
27	DIRECT SCREEN WRITING	- Frank Daniel
29	DOS 2.0S MODIFICATION	- Makoto Nagata
30	PLAYER-MISSILE DRIVER	- Makoto Nagata
32	UTILITY PACKAGE	- Makoto Nagata
39	DIGITAL ALARM CLOCK	- Steve Kunze
40	ATARIWRITER UNDERGROUND	- Frank Pazel
44	CARTOON MACHINE	- Cliff Schenkhuisen and Mark Perez
45	SUPER SCREEN DUMP	- Ted Burger and Paul Gifford
47	TEXT SCREEN DUMP	- Tom Reichard

SOUND:

16	USING 16 BIT SOUND	- Jerry White
16	SOUND IDEAS	- Lee Minard

GAMES:

6	FLITTERBUG	- Jim Warren
26	BLACKJACK	- Frank Daniel
42	THE LEGEND OF STARGUNNER	- Alex Leavens

HARDWARE:

20	RAMTALKER	- Randy Holmes
22	810 DRIVE MODIFICATION	- Banford Wong
24	1050 DRIVE MODIFICATION	- Bill Fletcher

The San Leandro Computer Club for Atari Microcomputers is an independent, non-profit organization and users' group with no connection to Atari Corp. Membership fees are \$20 per year. Membership includes access to the computer library, subscription to the Journal, and classes when held. Permission to reprint articles in any non-commercial publication is permitted without written authorization, provided proper credit is given to the San Leandro Computer Club and the author. Opinions expressed are those of the author and do not necessarily represent the views of the S.L.C.C.

S.L.C.C. OFFICERS

PRESIDENT Bob Barton 352-8118
VICE-PRES. Jim Hood 534-2197
TREASURER Lois Hansen 482-2222
SECRETARY Dan Chun 471-9286

SPECIAL EDITION JOURNAL STAFF

EDITORS Ron Seymour 537-3183
Tom Bennett 276-4466

GRAPHICS
EDITOR Jim Hood

PRODUCTION Karen Bennett, Ron
ASSISTANTS Devine, Tom Tisby,
Nate Hood

CONTRIBUTORS Antic Magazine,
Atari Corp., Ted Burger, Eric Clausen,
Michael Curry, Frank Daniel, Bill Eash,
Joe Eash, Bill Fletcher, Bill George, Paul
Gifford, Lois Hansen, Randy Holmes,
Steve Kunze, Alex Leavens, Lee Minard,
Makoto Nagata, Frank Pazel, Mark Per-
ez, Jack Powell, Tom Reichard, Mike
Sawley, Cliff Schenkhuizen, Sgt.
Slaughter, Tom Tisby, Jim Warren,
Jerry White, Bill Wilkinson, Banford
Wong.

TYPISTS Pat Baratta, Karen Bennett,
Schell Dietzman, Stewart Dimon,
Janice Eaton, Pieter Galiston, Victor
Johnson, Mike Sawley, Dick Scott.

CORRESPONDENCE ADDRESS & NEWSLETTER EXCHANGE

SAN LEANDRO COMPUTER CLUB
P. O. BOX 1525
SAN LEANDRO, CA 94577-0152

JOURNAL ADVERTISING RATES

FULL PAGE: \$40.00
HALF PAGE: \$20.00
QUARTER PAGE: \$10.00
BUSINESS CARD \$ 5.00

Special Edition disks (2 disks, double sided) with magazine are available by mail order for \$15.00 (postage and handling included) by sending your request to the above correspondence address. Supplies limited.

Editors' Notes

by Ron Seymour and Tom Bennett

fa-nat-ic adj. (L. *fanaticus*, of a temple, hence enthusiastic, inspired) unreasonably enthusiastic; overly zealous; also fa-nat-i-cal - n. a person whose extreme zeal, piety, etc. goes beyond what is reasonable; zealot.

We figure we must fall into this definition. It did not seem "overly zealous" at the time. As a matter of fact, that Friday evening, December 21, around 11:45, we found ourselves in a jam. We had finished layout for the January issue, only to realize that we were at 15 pages of copy, not the 16 required to get the newsletter out.

"How about an ad for a special edition Journal", Ron joked. "Why not, we have been sort of kicking this around for a while, let's give it a try", Tom responded. We did not think that anyone would really respond anyway. So we could fill the page, get the January issue rolling with the ad being an easy way out.

Curse that night!

With the religious fervor of meeting announced deadlines (much like our friends at Atari), here it is, the June Special Edition in July. (Note it is now the Special Edition, not "June" Special Edition.)

We were overwhelmed. The enthusiasm in response to our page filler was outstanding. From the delivery of "Master Pencil" in its earlier form by the Eashes, we knew things were going to roll. We had a number of typists come forward and really help make this fly. Jim Hood was "volunteered" to do the graphics and many contributed hours to this special effort.

The biggest single delay was the copy editing, copy editing and more copy editing. Then we had to repeat the whole process once again after typeset. We know we missed a number of things even after all our efforts. So, thank you for your understanding of all of our delays.

Inside we hope you will find the "meatiest" Journal yet. Games, utilities, profiles, reviews, insights and hardware are all covered here in a way that we have not been able to present to you in the past. Hopefully this will be the start of a new concept in user group publications. We intend to mail our disk text files to select groups across the nation to begin a newsletter file exchange that will both get our club out to others and in turn allow us to print other groups' materials here in the Journal. If this exchange concept works, look for top quality Journals in the future.

Thanks go to Jim Hood, Karen Bennett, Tom Tisby, Ron Devine, Bill Wilkinson, Atari Corp., all of our contributors and typists. And very special thanks go to the spouses and families of all involved with the Journal for their patience and understanding. We can not do without your continuing support.

Enjoy.

Better BASIC

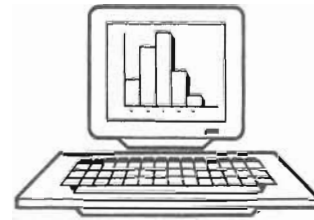
by Bill Wilkinson, OSS

Special to the San Leandro Computer Club Journal

Hopefully, by now you have all noticed the OSS ads for BASIC XE. And if you have, you have read that BASIC XE is faster than BASIC XL and, of course, much faster than Atari BASIC. Since I was the turkey/hero (your choice—the vote's still being counted) who wrote the specifications for the original Atari BASIC, and since BASIC A+ BASIC XL, and BASIC XE are all modifications and extensions of good old Atari BASIC, I thought you might be interested in finding out how we at OSS can continue to eke more performance out of an existing language. In other words, stand by for some secrets.

First of all, I should like to point out that all the OSS versions of BASIC really are extensions of Atari BASIC. Even in our newest BASIC XE, there are still a few hunks of code which are unchanged since the original 1978 vintage. But other major routines have been rewritten from the ground up. And, almost inevitably, rewriting a piece of code to make it faster involves making it bigger. So Atari BASIC takes up 10K Bytes (if you include the floating point routines), BASIC XL uses 18K Bytes, and now BASIC XE gobbles fully 28K Bytes! Of course, not all those thousands of bytes we now use are there just to speed up existing code. We do tend to like to add features, also. For example, **PRINT USING** and **SORT** each use about 1,000 bytes. (The real trick in BASIC XE was devising a method of packing 28K Bytes of code into your computer in such a way that the language is compatible with Atari BASIC even to the point of giving you the same working memory space for your programs! The result: BASIC XE won't work in Atari 400 and 800 computers, though it does fine in XL and XE machines. Sorry about that.)

Okay. So now you know that you have to use more room for BASIC if you want more speed. But just what did OSS spend all that room on? Actually, in the pro-



cess of trying to leverage BASIC for all the speed we could reasonably get, we worked on three major portions: (1) execute expression, (2) program transfers, and (3) floating point math. We are going to talk about all those in reverse order.

FASTER MATH

BASIC XL had already been given a major overhaul on parts 1 and 2, but item 3 is uniquely BASIC XE. To be fair, a floating point math speed-up has been done before: Newell Industries has been selling the FastChip, a replacement for Atari's Floating Point ROM, for years now. But when Charles Marslatt (sorry if I spelled that wrong) wrote the code for that chip, he was constrained to put all his improvements into the space we used originally. In view of this, his changes were almost spectacular. With BASIC XE, we are and were under no such restraints. Where the entire math pack of Atari BASIC uses only about 2,500 bytes, BASIC XE uses over 4,000 bytes for Add/Subtract/Multiply/Divide alone!

I don't have space here to describe everything which was done, but I will give one example: In the routine to convert a floating point number to an integer, Atari BASIC multiplies the integer it has built so far by 10, picks up the next digit of the floating point number, and adds that digit to the budding integer. That multiply by 10 is slow. The process of picking up digits one at a time is slow. BASIC XE, on the other hand, has a 160 byte translation table which takes pairs of digits (i.e., bytes) from the floating point number and converts them (in two machine instructions!) into their binary equivalent. And that's just for the units position of the floating point number! Why is this speed up important? Because it is used over and over again in BASIC. Every place which calls for an integer number calls this subroutine. Examples include the line number of a GOTO or COSUB; the address and data of a POKE; all the numbers in a SOUND, SET-COLOR, PLOT, DRAWTO, OPEN, and many more! Changing this one routine alone can speed up many Atari BASIC programs by a factor of 10% to 20%!

GETTING THERE FASTER

One feature of Atari BASIC of which I am especially proud is the fact that you can STOP a running program, examine variables, SAVE it to disk, change the Program, and *still* CONTINUE right where you left off! Almost no other microcomputer BASIC can do this. For example, all Microsoft BASICs (yes, even on the PC) will *not* allow you to change your program and continue.

The freedom you enjoy with Atari BASIC is not without its price, though. Consider the following: When a STOP occurs, we are in a subroutine called from a GOSUB. Now you change the program (such that the line which contained the GOSUB moves in memory) and CONTINUE. What happens when RETURN wants to do its job? If GOSUB caused BASIC to "remember" the address of its line (as does Microsoft BASIC), we're in trouble! So instead Atari BASIC and its kin remember the line number where the GOSUB is. Neat, right? Yes, but then when we RETURN we have to search for that line number. S-s-l-l-o-o-w! And guess what! FOR/NEXT loops work the same way.

The story doesn't stop there. One thing Atari BASIC and Microsoft BASICs have in common is the way they execute GOTO and GOSUB: After they've figured out which line you want to transfer to (and that can take a while...remember the floating point to integer conversion discussion above?), they *always* start at the beginning of your program and search forward until they find your requested line. More s-s-l-l-o-o-o-w.

So what do BASIC XL and BASIC XE do differently? Well, if you do nothing, everything works exactly as in Atari BASIC (albeit a bit faster because of other factors). This is on purpose, for compatibility. However, when you use the FAST command, these BASICs actually do a pre-compile of your program. All line numbers referenced in the program are looked up and converted to absolute addresses instead. Then when you use **GOTO 100**, BASIC doesn't search for line 100 because it already knows where it is. Fast!

Of course, we didn't stop there. BASIC XL and XE in FAST mode "remember" the absolute addresses of GOSUBs and FORs (and WHILEs and CALLS and...), so the line number search is not needed when a RETURN or NEXT (or ENDWHILE or EXIT) is encountered. Is there a penalty for this speed. Of course: You can **NOT** stop a Program, which is in FAST mode and then CONTINUE. Period. Still, we don't think that's too bad, since you can still debug your programs in the good old Atari BASIC style and only move to FAST mode for the final touches.

EXPRESS EXPRESSIONS

Although the new math routines involve the largest changes (in terms of sheer code space), the most complex changes (because of the effects on existing code) were those which Steve Lawrow made to one of the "master" routines of BASIC: Execute Expression. This routine is used by almost every statement and

function. Any time the syntax of the BASIC language allows an expression of any kind, the routine named Execute Expression is called. As a simple example, consider a statement of the form

PRINT A + 3

The expression in that statement is $A + 3$. With Atari BASIC, the execution of it goes something like this:

1. Fetch the value of A to FRO
2. Push FRO onto the argument stack
3. Fetch the value of the constant 3 to FRO
4. Push FRO onto the argument stack
5. Pull the top of the argument stack into FRO
6. Pull the next value on the stack into FR1
7. Add FRO plus FR1, the result is in FRO
8. Push the result onto the stack again
9. Pull the top of argument stack into FRO
10. Convert and print the value in FRO

(In the above, "FRO" and "FR1" are the two primary floating point "registers"—actually six bytes each in zero page.) Got all that? Now watch what BASIC XL and XE do:

1. Fetch the value in A into FRO
2. Move FRO into FR1
3. Fetch the value of the constant 3 into FRO
4. Add FRO plus FR1, the result is in FRO
5. Convert and print the value in FRO

A *lot* shorter and a *lot* faster. We estimate that, for simple expressions (e.g., no multiply or divide), this new method saves 30% to 40% of the time used by execute expression.

Now, those of you new to computing may say, "Well of course. Why would you ever do it any other way?" The answer to that question is complex: first., because you have to provide for expressions such as $SQR(X*X + Y*Y)*SIN(X/2)^2$ equally as well as provide for simple things like $A + 3$. Trying to analyze the kind of expression being presented is not a trivial task. Second, along the same lines, even after you figure out *how* to make execute expression faster, you have to find room to implement the needed code. The additional space simply isn't there in the Atari BASIC cartridge. (Could Atari BASIC have been made better in the space available? Yes, but not as much so as BASIC XL and XE have done.) Anyway, this kind of optimization is done as a matter of course by compilers (yes, including ACTION!), but it is seldom done as comprehensively as we have done in our expanded BASICs.

And that's about it. Oh, there are many other more subtle differences and improvements which we made as we built our advanced BASICs. And let's not forget the amount of work which went into getting BASIC XE to understand bank-switched cartridges, memory under the OS ROMs, memory under the cartridge, and (last but definitely not least) 64K of program space and over 30K of string/array space. And still be able to run with cartridge alone, without using expanded memory, in a mode compatible (except for speed) with Atari BASIC. That was (and is, even as I write this) a job! We hope you will agree with us that the effort was worth it.

Flitterbug

by Jim Warren

Is it possible to explain ACTION! to someone completely unfamiliar with it and at the same time show a more experienced ACTION! hacker how to use the routines in FLITTERBUG? I do not know, but I am going to try by talking about the source code of Flitterbug. If you will PLAY with the program for a while you will have an easier time with my explanations.

Now take a look at the FLITTERBUG code. Despite the seemingly huge variety of statements there are only three major "structures" to this code: COMMENTS, VARIABLE DECLARATIONS (i.e. BYTE BUG_EGGS—meaning, set aside one memory location and do to it what I say to do to BUG_EGGS, and PROCEDURES (subroutines containing the actual business of the program parcelled out into reusable, task specific, modules).

The top lines of FLITTERBUG are comments, followed by MODULE, meaning that the following variables can be used by any procedure anywhere in the program. The three INT variables (INT indicates +/- numbers getting two memory locations each) are used by several procedures and have their values set in PROC JOYSHTIK() so that every time PROC JOYSHTIK() is called (run, gosubed, JSR ed, executed, etc.) those variables are updated reflecting the condition of the joystick. Next there are fourteen CARD variables (two memory locations, all positive numbers), some of which have been given initial values.

Next there are twenty-seven BYTE variables (one memory location— all positive) some of which have been assigned values and some of which have been assigned a memory location (i.e. CONSOLE = 53279). This is convenient since, now, I can say simply, IF CONSOLE = 6 THEN DING() FI, meaning if the START button is pressed execute the DING routine. Notice the backwards IF indicating the end of that particular set of IF testing. Next in the list of FLITTERBUG code is a BYTE ARRAY called HARDLINE. It is a list of single memory locations used to store a word, sentence, or list of numbers giving the shape of

a player missile or redefined character. In this case HARDLINE is used to hold and print the different levels of game difficulty (EASY, HARD, FAT CHANCE, DOCUMENTATION). It is manipulated in PROC TITLE().

Next in the code is DEFINE POP = "[68 68]". DEFINE is a mere typing convenience offered by the compiler. It has no intrinsic programming value. It simply means, in this case, wherever you see the word POP in the code, substitute the stuff between the quotations. In this case the stuff between the quotes are two machine language instructions [PLA PLA] which will remove an address from the stack so that we can RETURN (if and when one is issued!) to the procedure that called PREVIOUS to the last.

Next in the FLITTERBUG code is, finally, a real procedure declaration; PROC RE MAIN = ERROR(). However, it does not say here what RE MAIN does, it just says that it is found at the same place that the ERROR procedure is found and that it does not accept any data for manipulation. Why would I tell the compiler such a rotten lie? Because later I will set ERROR equal to the REAL address of PROC MAIN() and call for RE MAIN() which will get us to MAIN in a very round about way. This overcomes the inability of the compiler to handle forward references.

Next in the code we come to a procedure that actually does something. It Pulls an address off the stack and jumps to the ERROR procedure which now you know could be ANYWHERE in the program.

FLITTERBUG, then, consists of a long list of procedures, which you and I can remove and put into other programs having nothing to do with FLITTERBUGS. So, even if you do not like the game FLITTERBUG, any routines within the game that you do like are yours for a very small effort involving almost no typing.

One way to reuse parts is to CTRL/SHIFT DELETE the procedure that you want, then do a CTRL/SHIFT

2 to get into a separate window, then do a CTRL/SHIFT P (for put) to put the procedure there. Then do a CTRL/SHIFT W (for write) to save that procedure to disk for later inclusion (with INCLUDE "D:FILENAME") into a different program.

When you take these routines from me and turn them into your own masterwork, I hope you will publish at least some of it in the public domain so that we can improve our efforts with some of YOUR knowledge and effort. Send the code (and PLENTY of explanation) to me. I will see that it gets distributed widely and that YOU get the proper credit.

Jim Warren
San Leandro Computer Club
P.O. BOX 1525
San Leandro, CA 94577-0152

The very first thing (the first procedure executed is always the last one in the list) that FLITTERBUG does is to copy the character set in ROM into RAM so that it can be redrawn. You will notice that PROC RELOCATE_HARSET() calls for the execution of POKE_A_CHAR() just before it calls for the execution of MAIN() as its last instruction. POKE_A_CHAR() is the second to last procedure and is little more than the list of data that comprise the new letters and numbers of the character set. Only the lower case letters were redefined. The upper case letters are used by graphics mode 1 in TITLE() and ENDLE() and work just fine as is. I drew these new letters using the EDIT() procedure in this program (the same procedure appears in the TOOL KIT demo in this special edition). You can draw your own character set or redefine just one or two letters that I did not do quite well enough by using the editing box of this game or the one in the TOOL KIT DEMO. Just leave one line blank at the top and bottom and two "bits" blank on the right. Then write down the data and type it in the appropriate part of the BYTE ARRAY in POKE_A_CHAR(). For a fuller explanation see my TOOL KIT DEMO article in this special edition.

If you want to see why I had to go to the trouble to redefine the character set, write a little document all in uppercase in the first paragraph and all in lower case in the second paragraph and save it to disk with the name "FLITTER.DOC". Then read that document with the FLITTERBUG doc reader (get DOCUMENTATION by pressing OPTION in the title screen then press START). You will see that the ROM character set (upper case letters) just does not appear clear in antic mode 4.

After the character set is relocated and the new data Poked into place the main procedure is called and controls the flow of the program from now on. The main controlling procedure is named, appropriately, PROC MAIN() and is little more than a list of procedures indicating the order of execution of the main modules. It in turn, call other procedures as needed to get the job done. I have tampered with the normal procedure calling scheme of ACTION! in a couple of procedures with the "RE ACTIONARY" methods of forward referencing and recursion. For the most part, you will see procedures calling for procedures which are "above" themselves in the list. They can not call procedures "below" themselves in the list. This is what I mean by forward referencing. To facilitate my "tampering" I used address variables to refer to the forwardly referenced procedures; and the first thing that PROC MAIN() does is assign the right address to those variables (i.e. ADR_MAIN = MAIN).

Then TITLE() is called where the player sets the MINIMUM_EGG, TIME, BUG_BIRTH_RATE, and variables by pushing OPTION. When START is pressed in TITLE() a RETURN is issued and execution returns to MAIN() where SETUP() is next in line. After the return from SETUP(), we point ANTIC at our custom characters and jump into an infinite (meaning there are no controlling statements saying when to stop) DO OD loop with five procedures in it which are repeatedly executed.

You can write procedures and have them executed anywhere by calling them by name where you want them executed. For example, if you write PROC BEFORE_THEY_ARE_HATCHED(), and write BEFORE_THEY_ARE_HATCHED() right after COUNT_EGGS() in PROC MAIN(), then you will have your eggs counted just before they are hatched! I would encourage you to experiment with FLITTERBUG in just this way but, unfortunately, the source code is too big to be in memory at the same time as the object code. You will have to break it up into sections, working on one part in memory with the rest "INCLUDED" from the disk. I suggest that your files be single procedures that might also be useful for inclusion into other programs. Likely procedures within FLITTERBUG for such an "excision" are JOYSHTIK(), ANTIC4(), DOCREADER(), EDIT()&LOOKUP(), POKE_A_CHAR(), and RELOCATE_CHARSET(). These are all potentially reusable "as is" in other programs. Sorry the program is not smaller for you to play around with but, as usual, I got carried away!

Antic's 520ST

**ANTIC PUBLISHING INC.,
COPYRIGHT 1985.
REPRINTED BY PERMISSION.
by JACK POWELL
ANTIC TECHNICAL EDITOR**

Our Atari 520ST just arrived in the Antic offices. This is the \$5,500 development package, and it includes the computer, two 3½" disk drives, one medium-resolution (640 X 200) RGB analog monitor and one mouse.

The software in the package is the "C" compiler, machine language assembler and debugger by Digital Research, the Mince screen editor by Mark of the Unicorn, Kermit (a modem protocol program for file transfer), CP/M-68 and, of course, GEM which is in ROM in the machine.

On back order, but expected soon, is a high-resolution (640 X 400) monochrome monitor and a ten-megabyte hard disk.

Along with all this came 1000 pages of documentation and since it will take some time to digest all of this, we thought you might like a first impression of this new, high-level Atari computer.

The development 520ST is a preliminary model and there will be some changes between now and the time you see it in the stores, but all parts of this machine were factory made in the same manufacturing plants as the final product will be. The only real difference, besides the price, is that these machines were hand assembled.

The first thing you notice when taking the ST out of its box, is that it is very light. Although somewhat larger in size than the 800XL, it feels lighter. This may be because the shielding has not yet been added.

It looks exactly like the ST on our May cover but there are some details you can not see from the photograph. On the right edge of the machine, to the rear, are two joystick ports identical in appearance to current Atari joystick ports, except they are also used for the mouse.

On the left edge, rear, opposite the

joystick ports, is the cartridge slot. This will accept a 40-pin board, 20 upper and 20 lower.

In back of the computer are various switches and ports, each labelled beneath and with an indicating icon etched in the plastic above. From left to right they are:

- Reset - a small, square button.
- Power - identical to previous Atari power switches.
- Power In - 7-pin, male DIN.
- MIDI Out - 5-pin, female DIN.
- MIDI In - same as above.
- Television - RCA, female.
- Channel - mini-switch, labelled "L" "H".
- Monitor - 13-pin, female DIN.
- Printer - female D-25, IBM-PC/Centronics compatible.
- Modem - male D-25, IBM compatible.
- Floppy Disk - 14-pin, female DIN.
- Hard Disk - female D-19.

Besides the standard keyboard and ten-key pad, are ten function keys, labelled F1 to F10. The isolated cursor section is particularly well designed with the lower three keys representing Left, Down and Right, and the Up arrow centered above them. On either side of the Up key are Insert and C1r/Home. The top two keys in the cluster, which are enlarged, are Help and Undo. The Undo key may become particularly useful.

The drives accept Sony 3½" disks. To boot the machine, first turn on the drives and insert both disks before turning on the computer. A disk must be in a drive for the computer to later access that drive.

When booted, the GEM desktop appears as a light green background with pale blue border and black-outlined icons. In the upper right corner of the screen are two disk icons, one over the other, that look like file cabinet drawers. In the lower right corner of the screen is a trash can.

In the border area, above the upper left section of the green background, are the words, "DESK FILE VIEW OPTIONS." In the middle of the screen is a thin, black arrow-cursor which is moved by the mouse.

We will save details on GEM for later articles. And it is fast! It can redraw an entire screen of icons in the blink of an eye.

This is just a surface description of an exciting new machine. Antic wants to get the information out to you as soon as possible and we plan to share our ST experiences as they happen. Stay tuned for further details.

Compuserve's GEM Conference

*This article contains portions of the transcripts of the GEM online conference held on May 9, 1984. The conference was cohosted by SIG*ATARI and DR SIG. The featured guests were the members of the GEM Development Team.*

(DRI) Hi and welcome to DR SIG and to the GEM CO. On my left is Tim Oren, Toolkit expert, next to him is Gregg Morris, Applications expert, John Grant, VDI Ace, Scott Raney, all around good guy, Dave Mackenzie, GEM Application Support. Others are on their way. Tonight we will welcome the Desktop owners and the soon to be released ATARI ST owners. Also a special welcome to those veterans who have been working with GEM development on the IBM PC.

(MODERATOR) Besides the new ATARI ST what other 68000-based machines will GEM be available for (in the near future)?

(DRI) There are no other 68K machines that we can comment on at this time.

(Ron Winn) I know very little about GEM. As a future ST owner, can you suggest any good books or manuals to study up on GEM?

(DRI) Nothing out as yet. Look for something this Fall from several publishers.

(Steve Brecher) I am a Mac developer. Could you comment on whether Porting from the Mac to ST is likely to be feasible. (I understand "feasible" is undefined, but your thoughts?)

(DRI) Ok, there are a number of possible answers to that. Let me try a few. If you have done your job on the MAC "cleanly", that is, done a proper "non-moded" modular program in a language like C or PASCAL, the port should be relatively fast. But, GEM and

MAC are nowhere near call-for-call compatible, so you need logical separation points (shells) around the machine specific code. To give you some actual feedback, one European developer moved a MAC application in about 3 weeks of intensive effort (that was the time to an alpha release). A number of other developers are taking the opportunity to translate their programs from MAC Pascal into C. Obviously, that takes longer.

(Steve Brecher) Per answer to Ron above, understand I can not go out and buy a book on GEM internals, but is there any way I can learn about such porting issues (in some detail) currently?

(DRI) The best way to get answers to that would be to purchase the GEM Programmer's Toolkit with or without interactive support. The present Toolkit is available from DRI for \$500 with interactive support on this SIG. The retail version will be available soon.

(MODERATOR) This is the IBM Toolkit only?

(DRI) At this time it is for IBM only. The 68K version for GEM DOS will be a little later.

(Steve Brecher) Without support? Does that include a written programmer's guide?

(DRI) That does include written programmer's reference manuals and sample programs.

(Steve Brecher) Re GEM/ST do I talk to Atari?

(DRI) Yes.

(Ron Winn) With an ST out of the box, what will GEM do for me? What else will I need to get to use GEM in my programs?

(DRI) 1) The ST, as shipped, will include the equivalent of our PC Desktop. There will also be some other programs available from Atari, particularly Logo, and, of course, our own GEM stuff will be ported. 2) The programming inter-

face will be very similar to PC GEM, so someone could get started now on the PC. Atari will release their own toolkit but I do not have a date on that and we will be doing a "generic" 68K Toolkit. (That is, it will not be Atari specific).

(Ron Winn) Can you get to GEM thru Basic or LOGO?

(DRI) First for GEM LOGO. This will include some VDI type primitive such as draw circle, fill, and so on. However, the first release will not have a binding into the AES, that is, it will not let a user program open windows and such. As to Basic, the question is still open and under discussion. How about it, do you think people really want to drive windows from an interpreted language?

(MODERATOR) Absolutely! Especially a useable BASIC.

(Ron Winn) I do not know for sure. Some want to use whatever they have Paid for.

(DRI) More about BASIC... What are people looking for, MBASIC compatibility? CBASIC? GWBASIC?? You tell us.

(MODERATOR) That's tough to answer for an ATARI owner. Except for a few owners of MS BASIC we have been a captive audience of ATARI BASIC.

(DRI) Pardon my ignorance, but what type of BASIC was on the former ATARI machines? Was it a Microsoft effort?

(MODERATOR) ATARI 8k BASIC was NOT a MS BASIC. It was developed by OSS and is similar to a BASIC used by the old NORTHSTAR systems.

(DRI) Hum, well maybe we should find a manual for it and take a look. Whatever we do, it will probably have a strong BASICA flavor, since we are starting on the PC. But we can be careful not to introduce gross incompatibilities from current ATARI BASIC.

(Ron Winn) Even ATARI basic is a start. There is a whole generation of BASIC programmers. They would like to start there where they feel at ease, then move them up to modern languages.

(Steve Brecher) Is there any intermediary between \$500 and, e.g., Infoworld superficial descriptions wherein I might obtain info on what GEM is and is not, from a programmer's view?

(DRI) We are strongly considering a Retail Programmer's Toolkit which would be released late in the summer. If this happens the GPS support that you get with the current Toolkit will be unbundled. Instead we will offer on-line DR SIG subtopic like Borland does for

Turbo PASCAL. Someone could then "buy up" into GPS if they start a serious project.

(Steve Brecher) But right now, without necessarily requiring detailed info as for coding, but rather a programmer's conceptual overview? Anything available?

(DRI) Not really, you are going to have to be a little patient. We are kind of maxed out right now with shipping the PC and 68K Toolkit. There is some catching up to be done on the documentation side. We are strongly encouraging in-house people to write articles on GEM PC and GEM ST. Keep an eye on places like Dr. Dobb's and Antic. I am now told that there will soon be an article in Analog.

(Ron Winn) You mentioned the 1st GEM LOGO. Does that suggest that early ST buyers will end up putting in new ROMs later on?

(Steve Brecher) The first ST buyers won't get ROMs, I hear. They will be all in RAM.

(DRI) No, just that they bought source rights and could go to a later release if they chose. (I kind of doubt it, but that is strictly my opinion.) BTW, Steve the business about RAM was reported in the press but we can not officially comment.

(Steve Brecher) WSJ today says all of OS ("near" 256KB) will be in RAM for 1st 8 weeks of production. (Maybe all to Europe.)

(DRI) Well, the WSJ usually has good sources, but we can not comment, call Jack.

(Steve Brecher) Ron, in case you did not hear, ST availability in the U.S. is now July.

(MODERATOR) Please clarify a point of possible confusion. The programming commands that will allow a user to, for example, "open a window" from BASIC are a function of that version of BASIC he is running. Whether such operations will be possible on an ST does not depend on upgrading to a new version of GEM (changing ROMs). It is simply a matter of buying a new version of BASIC. Correct?

(DRI) Um, I assume we are talking about a hypothetical BASIC. Generally what you say is true but there are ways to work around requiring a whole new interpreter when GEM changes. As an example, if anyone out there has ever used CALL or the "& trap" on Applesoft, there are ways to jump directly to assembly routines from BASIC. If we took that route, then only those routines

would change if the underlying GEM changed. HOWEVER, we will make every effort to be sure that future versions are fully backward compatible.

(MODERATOR) I guess I was trying to state a slightly different issue. Many novice programmers might think they have to wait and not buy one of the first few STs simply because they only know BASIC and the existing BASICs may not have a way to "open a window" for example. Someday, somebody probably WILL sell a version of BASIC which has these commands just like some BASICs support automatic line numbering, others do not. The ability to talk to GEM depends on the programming language.

(MODERATOR) He can buy an ST now and WILL NOT have to buy new GEM ROMs just to program in BASIC. Do you follow my point?

(DRI) Ok, that is right, because GEM is not language dependent. However, we could not (with a straight face) recommend that anyone pay \$4500 for a ST right now to run BASIC. The environment on the STs from ATARI is only suited for a programmer who is an expert in C and the 68K.

(MODERATOR) "at this time".

(Ron Winn) \$4500? You mean the development STs? Is it worth paying \$1500?

(DRI) Yes, the development STs. \$1500 would be more like it. All we meant to say was that it is difficult these days to market a program written in BASIC, and that would be the only way to justify paying the developer's price.

(Bill Bolton) I have not seen the release GEM yet. It is still sitting in customs (cleared today they tell me). But with the Toolkit version there was a definite limit on the number of things that could be installed on the Desktop. Has that been "fixed" in the release version?

(DRI) The limit remains. There is a quick way to delete previous programs. We will post the transcript tomorrow and you can read it there. Again, note that just because you can not get a special icon installed for a program does not mean you can not run it! Remember, .COM and .EXEs will default to DOS mode, and .APPs to GEM mode.

(Ron Winn) I hope you will pursue the "buy up" policy you talked about so that the many machine buyers can grow with their machine without getting cost shell shocked.

(DRI) I happen to agree. (Ha ha, we noticed.) Look for further news on this in a few weeks.

Disk Formatting



by Kenneth J. Pietrucha - JACG

To me, the most sensible approach to formatting has always been to format a new box of disks at one time. I made this decision after I tried to save a program to an unformatted disk. Now I do not even take the cellophane off the box until I am ready to format them all.

Formatting an entire box of disks using DOS involves many key strokes. Fortunately, the Atari has an XIO command which allows you to format directly from BASIC. Actually, the whole formatting program can be written as a one or two liner.

5 REM FORMATTING PROGRAM

10 XIO 256, #1,0,0,"D:"

That is it! Type RUN, hit RETURN and your disk will be formatted. This may be one of the shortest formatting programs ever but you do not have to keep it that way. Now is your chance to write your own formatting program. You can have the screen change color during the formatting process or you can add a counting step, line N = N + 1, to tell you how many disks have been formatted. You can use the special function keys to start the formatting process instead of typing RUN each time.

There is very little additional information on the XIO command. In Ion Poole's book, **YOUR ATARI COMPUTER**, only a brief mention is made of this XIO function. I do not know the specifics, only that it works. As a matter of fact, every BASIC formatting program I have ever seen uses this command.

Here is my version of a formatting program: which I keep on my utility disk. It keeps track of the number of disks formatted. If I start with a new box of ten disks, I better have the number 10 on the screen when I am finished.

I think you will like formatting with this approach. It is definitely easier than using DOS.



By Joe Eash

Do you recall a child's drawing tool called "Etch-a-sketch". With Master Pencil, you can now do this electronically. Ah, but this program has developed into more than just moving the cursor around and making lines.

This program can make perfectly straight lines at any angle when you give it the "from" and "to" points and ask the program to "fill" in the line for you. You can draw pictures with a joystick and (unlike Etch-a-sketch) selectively erase portions you wish to change. You may also copy or move portions of the picture around the screen. There is an Alphabet mode for adding words and numbers to your drawings.

Once you have created a "master-piece" you can save it and load it later to show to your friends.

You can print your pictures if you have PrintWiz (or any other special printer utility that prints graphics). If you are using PrintWiz, the pictures usually look their best under "Triple Width" and "Double Height" options (part of PrintWiz's special commands).

If you have the disk that accompanies this magazine you will find the source program in BASIC "MPEN-

CIL.BAS", the ACTION! source code "MENCIL.ACT", and the machine language version "MPENCIL.BIN" which is really the ACTION! source code that has been compiled by the "RUN TIME KIT" (c) 1984 Action Computer services. The one you will usually use is "MPENCIL.BIN". You may use the BASIC or ACTION! versions to modify *Master Pencil* to fit your needs or to get ideas. NOTE: both versions make use of all the memory available on my 48K computer, so modifications may cause an "Out of Memory" error. Action users must do a compile from disk.

Please feel free to share this program with your friends, but please keep all credits in the program.

Now, down to business. I have done my best to make this program very easy to use so that you'll only have to read this once.

When the program is run the screen will turn black and a small cursor will be blinking on the screen. This cursor can be moved with the joystick in any of 8 different directions. This "magic" cursor will not erase any dots it crosses while being moved. To plot a dot, press the fire button. You can change the color of the dot by pressing the space bar. There are only 2 colors, the foreground and the background. When the words "COLOR ON" is in the window at the bottom of the screen, you will be drawing dots on the screen. When the words "COLOR OFF" are in the window, you

will be erasing dots. Your joystick is your "Master Pencil"! So, with your pencil in hand and your paper on the screen, doodle for a while and get use to "COLOR ON" and "COLOR OFF" (press the space bar to switch them).

Now that you are used to using this much of the program, let's go on. The only two keys you need to remember while you are drawing are: the space bar, and the escape key [ESC]. Press the ESC key and a menu will appear. At the bottom of the menu screen is an up-to-date display on a few important things. Here is an explanation of them:

HSPEED - This is the horizontal speed of the cursor. This will be explained later under "H".

VSPEED - This is the vertical speed of the cursor. This will also be explained later in detail.

X POS - This is the horizontal position of the "Magic" cursor on the screen.

Y POS - This is the vertical position of the "Magic" cursor on the screen.

4-WAY MIRROR - This will be explained later under "E".

GET DOT mode - Explained under "G".

The other items on this menu perform very special features and have taken a long time to perfect, so I'm sure you will appreciate the things that they do. Here they are, explained in detail:

A) ALPHABET. This feature lets you put letters, numbers, lower case, graphics characters and punctuation anywhere on the screen you want. Press "A" and you will be asked the height and then the width of the letters to be drawn. For now, press 1 for the height and 1 for the width. One is the smallest size of a letter and is the same size as the text mode letters. Next you will be asked to position the cursor in the upper-left corner of the area where the first character is to be printed and press the fire button. For now, put the cursor towards the left side of the screen and press the fire button. Now type anything you want, or press return to move cursor somewhere else or [ESC] to return to the menu.

B) MOVE/DUP. This is for moving a piece of your picture to another place on the screen. If you respond "Move" then the area to be transferred will be erased, otherwise, "duplicate" will not erase that area. Follow the rest of the prompts and you will be on your way to complete the process. Note: since BASIC doesn't have as much memory available this function is slightly different.

C) CLEAR SCREEN. This will clear your drawing screen and reset horizontal and vertical speeds to 1.

D) DIRECTORY. This displays ALL the files in DRIVE 1.

E) 4-WAY MIRROR. When this is turned on, anywhere you are on the screen when you plot a dot it will be mirrored in the 3 other quarters of the screen.

F) FILL (FOR GET DOTS). See "GET dots".

G) GET DOTS (100 MAXIMUM). When using this

mode, the computer stores the next 100 dots plotted on the screen. For example: press "G" in the menu, "L" to return to picture, then randomly plot about 20 dots fairly well spread out. Now press [ESC] for the menu, then press "F", and PRESTO! All the dots will be connected in the exact sequence that you plotted them. Now, try to make a star.

H) HORIZONTAL SPEED. This value determines how many dots the cursor will "skip" when moving horizontally across the screen. To change the value, press "H" while in the menu and enter a number from 1 to 10. Do not forget to press [RETURN].

I) VERTICAL SPEED. This is the same as "H" except this controls how many dots the cursor will skip when moving vertically. Press "I" and enter a number from 1 to 10. Again, don't forget to press [RETURN].

J) SAVE. Enter a name for the picture (maximum of 8 characters, with an optional 3 character extension) and press [RETURN]. Your picture will now be saved to disk under the filename you gave it. A long, thin, white cursor will show you where the cursor is while saving.

K) LOAD. Enter a name for the picture and press [RETURN]. Your picture will be loaded from disk onto the screen for you to view, edit and save again. Note: the BASIC version loads and saves much slower (about 2 minutes, 30 seconds) than the machine language version* (about 10 seconds).

L) EXIT. This returns you to your picture.

TIPS: If you are familiar with artifacting in graphics mode 8 (which is what this program uses) here is a neat trick. For red artifacting, position the cursor in a column where the color is red. Then press [ESC] for menu and set the horizontal speed to 2 and the vertical speed to 1. Do the same for blue, except position the cursor in a blue column. If you are not sure about the color of a column, press [ESC] for the menu, then look at the value for the "X POS". If the value is an even number, then you are on a blue column. If the number is odd, then you are on a red column. Plotting two dots side-by-side will cause the color to be white, not red or blue.

Use the "GET" and "FILL" modes as much as possible to save time. These modes are very helpful in drawing geometrical shapes as well as your own creative designs.

If you wish to clear the window of instructional text to do a screen print, press [ESC] for the menu, then press "L" to return to picture. Now you have a clean picture to print.

* the machine language version is the ACTION! version compiled by: The RUN TIME KIT, (c) 1984 Action Computer Services.

ATARI®

DISK OPERATING SYSTEM 2.5

(Editor's note: The following is being reprinted by the SLCC JOURNAL with permission of Atari Corp.)

Every effort has been made to ensure the accuracy of the product documentation in this manual. However, because we are constantly improving and updating our computer software and hardware, Atari Corp. is unable to guarantee the accuracy of printed material after the date of publication and disclaims liability for changes, errors and omissions.

ATARI, ATARI BASIC, AtariWriter, 1050, 810, 130XE, 65XE and 800XL are trademarks or registered trademarks of Atari Corp.

No reproduction of this document or any portion of its contents is allowed without the specific written permission of Atari Corp., Sunnyvale, CA 94086.

Copyright 1985 Atari Corp. Sunnyvale, CA 94086

INTRODUCING ATARI DOS 2.5

In an ongoing effort to provide the highest quality of products for use with your ATARI Computer, the new ATARI Corp. is supplying you with the enclosed DOS 2.5 Master Diskette. Its advantages over ATARI DOS 3 include ease and convenience of use (most utilities are contained within a single file and need not be loaded from disk) and compatibility with DOS 2.0S. DOS 2.5 also allows you to use the full capacity of your ATARI 1050 Disk Drive and to access the full RAM potential of the ATARI 130 XE.

This short manual provides you with instructions for getting started with DOS 2.5. For complete information on DOS 2.5, including detailed discussions on the menu items, compatibility with DOS 3 and 2.0S, the RamDisk and the 2.5 Utilities, you may consider obtaining the new *ATARI DOS 2.5 Manual*. Available from ATARI Customer Relations, P.O. Box 61657, Sunnyvale, CA 94088. Cost: \$10 plus \$2.50 for shipping and handling. California residents add 6.5% tax.

Please write *ATARI DOS 2.5 Manual* on the outside of your envelope when you order the book.

Getting Started With DOS 2.5

DOS 2.5 allows you to format diskettes and store information in either single or enhanced density. With enhanced density you can record about 50 percent more data on each diskette than you can with DOS 2.0S. Enhanced-density storage is only possible if you have an ATARI 1050 Disk Drive; the 810 Disk Drive is not capable of formatting or managing data stored in enhanced density. You need a 1050 Disk Drive to begin working with DOS 2.5 because your DOS 2.5 Master Diskette is recorded in enhanced density. If you often use an 810 Disk Drive to access your files, you may want to format all your diskettes in single density.

DOS 2.5 works with any cartridge-based program that runs on your Atari Computer and uses DOS—even programs that predate DOS 2.5, including the AtariWriter word processor and ATARI BASIC. With such programs you can always use DOS 2.5 instead of DOS 2.0S to prepare data diskettes and manage files.

Many diskette-based programs designed for use with the earlier DOS 2.0S can also be used with DOS 2.5. However, you may have to continue to use DOS 2.0S with certain protected diskette programs (see your program user's manual if you are unsure whether a program is protected.)

THE DOS MENU

Load DOS into your ATARI Computer using the same procedures you use for either DOS 3 or DOS 2.0S. (If you have an ATARI 130XE, 65XE, or 800XL with built-in BASIC, type DOS and press [RETURN] to go from BASIC to DOS). The DOS Menu on your TV or monitor screen presents a list of the DOS 2.5 options.

The prompt below the menu invites you to make a selection. You choose the function you want to use by pressing the letter corresponding to your selection

and pressing [RETURN].

DOS then asks you for the information it needs to proceed.

Summary of DOS 2.5 Menu Options

If you have used DOS 2.0S, you will be familiar with most options. Note the change in Option J and the new Option P.

If you have only used DOS 3, read this section for an introduction to DOS functions.

A. DISK DIRECTORY allows you to call up a complete or selective list of the files on a diskette, showing the filenames, extenders (if any), the number of sectors allocated to each file and the number of free sectors still available on the diskette.

B. RUN CARTRIDGE (can ONLY be used with built-in BASIC or with a cartridge installed in the computer). This option allows you to return control of your system to built-in BASIC or to the cartridge inserted in the cartridge slot.

C. COPY FILE For use when you have two or more disk drives and you want to copy files from one diskette to another. Also use this option to copy a file on the same diskette, assigning a different name to the copy.

D. DELETE FILES lets you erase a file from a diskette, increasing the available space on a diskette.

E. RENAME FILE Use when you want to change the name of a file.

F. LOCK FILE can be used to prevent you from changing, renaming, or accidentally erasing a file. You will still be able to read the file, but will not be able to write to it. An asterisk is placed in front of the filename in the directory to indicate that the file is locked.

G. UNLOCK FILE This removes the asterisk in front of the filename and allows you to make changes to the file, rename it, or delete it.

H. WRITE DOS FILES lets you add the DOS files (DOS.SYS and DUP.SYS) on your Master Diskette or System Diskette to a diskette in any disk drive.

I. FORMAT DISK Used to format a blank diskette, which is necessary before you can record any information on it. Be sure you do not have any files you want to keep on a diskette before formatting it. This option will format a diskette in enhanced density provided you are using a 1050 Disk Drive; otherwise, it will format in single density.

J. DUPLICATE DISK Use when you want to create an exact duplicate of a diskette. This option will automatically format the destination disk.

K. BINARY SAVE saves the contents of specified memory locations on a diskette.

L. BINARY LOAD lets you retrieve an object file from diskette

M. RUN AT ADDRESS Use to enter the hexadecimal starting address of an object program after it has been loaded into RAM with BINARY LOAD.

N. CREATE MEM.SAV reserves space on a diskette for the program in RAM to be stored while the DUP.SYS file is being used. For some applications like

programming, it is a good idea to create a MEM.SAV file on each new diskette you intend to use as a System Diskette. As you become more familiar with DOS, you may find there are cases where a MEM.SAV file serves no useful function. The inconvenience of waiting for MEM.SAV to load into memory may warrant deleting it from the disk.

O. DUPLICATE FILE copies a file from one diskette to another, even if you have only a single disk drive.

P. FORMAT SINGLE formats a diskette in single density using a 1050 Disk Drive.

DOS 2.5 AND THE ATARI 130XE RAMDISK

The ATARI 130XE Computer is equipped with 131,072 bytes—128K—of Random Access Memory (RAM), twice the maximum 64K available with earlier model ATARI Computers. The additional 64K RAM can be useful for many purposes: fast exchange of screen images for animation, additional storage for large data bases and so forth.

You can also use the extra RAM of the 130XE as a very fast "virtual" disk drive. Set up as a "RamDisk" (recognized by DOS 2.5 as Drive 8 in your system) it can accommodate up to the equivalent of 499 sectors on a diskette. That is about half what you can store on a diskette formatted in enhanced density.

The "storage" capacity offered by the RamDisk is volatile memory. Information stored in it will be lost when you turn off your computer system. So before turning off your system, be sure that any data currently in the RamDisk that you want to save permanently is recorded on an actual diskette.

The RamDisk can be a very convenient tool. It allows you to switch almost instantaneously between BASIC (or any other programming language) and DOS, and back again. Use it to work with files "stored" on Drive 8; a technique that might prove especially useful when you are transferring large amounts of data between two programs that are chained together (that is, when one program RUNs the other).

To Activate the RamDisk

Your DOS 2.5 Master Diskette contains a file called RAMDISK.COM that automatically sets up the extra 64K RAM of the 130XE as a RamDisk.

When you boot your 130XE system with a DOS 2.5 Master or System Diskette containing RAMDISK.COM, DOS will:

- Display a message that it is initializing the RamDisk;

- Set up your computer's extra 64K of memory to act very much as a disk drive, telling DOS to regard it as Drive 8; and

- Copy the DOS file DUP.SYS and establish MEM.SAV on the RamDisk, and use the versions of these files on the RamDisk rather than those on your Master Diskette.

If you wish to expand the usable capacity of your RamDisk, you may recover the memory used by DUP.SYS and MEM.SAV by:

- Changing the contents of location 5439 (\$153F) to ATASCII 1, for example, POKE 5439,ASC("1"); and
- Deleting the files DUP.SYS and MEM.SAV from the "diskette" in Drive 8, that is, the RamDisk. Use option D., DELETE FILE(S), on the DOS Menu and enter D8:* in response to the DELETE FILESPEC prompt.

Note: Booting a disk which does not contain DUP.SYS will cause RAMDISK.COM to initialize the RamDisk, but DUP.SYS and MEM.SAV will not be moved to the RamDisk.

Using DOS With the RamDisk

Because of the size of the RamDisk, you may not use DOS Menu option J., DUPLICATE DISK, to copy either a single-density or enhanced-density diskette to the RamDisk. Instead, you must copy individual files, taking care that they do not exceed in size the capacity of the RamDisk. You can ask DOS to duplicate the contents of the RamDisk on an actual diskette.

From then on, however, that diskette will be capable under DOS of accessing only 499 sectors worth of data, though you can always duplicate its contents back to the RamDisk.

If You Do Not Want to Use the RamDisk

If you do not want to use the ATARI 130XE RamDisk, you can either delete or rename the RAMDISK.COM file on your DOS 2.5 Master or System Diskette. You may then use the extra RAM for other purposes.

If you have applications for which you do not wish to use the RamDisk, it is recommended that you leave the RAMDISK.COM file intact on your DOS 2.5 Master Diskette. You might wish to make one working copy of DOS (System Diskette) that contains RAMDISK.COM, and one that does not. Or you can simply rename the RAMDISK.COM file on your System Diskette, then rename it back to RAMDISK.COM when you wish to use it.

THE DOS 2.5 DISK UTILITIES

Your DOS 2.5 Master Diskette contains three new utility programs in addition to the standard disk utilities handled by the DUP.SYS file—those available from the DOS Menu. The programs, each of which appears on the disk directory with a .COM extender, function as follows:

COPY32.COM allows you to copy files from diskettes formatted and written to from ATARI DOS 3 to DOS 2.5 diskettes, converting the files in the process from DOS 3 to DOS 2.5.

DISKFIX.COM allows you to correct some problems that may occur with files on DOS 2.5 and 2.0S diskettes. Under certain conditions, you can also use this utility to recover deleted files.

SETUP.COM allows you to change certain DOS parameters. You can also use it to create an AUTO-RUN.SYS file that will automatically load and run a

BASIC program when you boot your system.

Note: RAMDISK.COM is not a disk utility. It is used only to set up the RamDisk on a 130XE Computer.

Selecting and Loading a Utility

All three utilities are binary files that are loaded and run using option L., BINARY LOAD, from the DOS 2.5 Menu. For example, to begin using the COPY32.COM program, with the DOS 2.5 Menu on your screen, you would type L and press [RETURN], then type COPY32.COM as the name of the file to load, and press [RETURN] again.

Specific instructions for using the COPY32.COM follow. There are also brief instructions for DISKFIX.COM and SETUP.COM. For more detailed instructions for the latter two utilities, consult the ATARI DOS 2.5 Manual (see the Getting Started section of this manual for ordering instructions).

COPY32.COM

Using this utility is much like using the COPY FILE function on the DOS Menu. After you load the COPY32.COM program, you are prompted to specify which drive will hold your DOS 3 (source) disk and which drive will hold your DOS 2.5 (destination) disk. If you have only one drive, type 1 in response to both prompts. In this case, you will have to swap your DOS 3 and DOS 2.5 diskettes during the copying process. If you have more than one disk drive, you may select one to hold your DOS 3 diskette and another to hold your DOS 2.5 diskette.

At this point, if you have only one drive, the utility prompts you to insert your DOS 3 disk in Drive 1. For safety, place a write-protect tab on your DOS 3 disk so that you will not erase valuable data if you make an error while swapping diskettes.

If you specified two different drives, the utility prompts you to insert both your DOS 3 and DOS 2.5 disks.

After you insert the diskette or diskettes, press [START]. The COPY32.COM program reads the directory of the DOS 3 diskette and displays the files it contains, sixteen at a time, by number. Press [RETURN] to see the next sixteen files. When all the files on the diskette have been listed, you have the options to restart, return to DOS or view the files again.

To convert a file, enter the number of the file you wish to convert. The utility prompts you to confirm your choice by pressing [START].

When you press [START], the program begins the conversion process by reading the specified file from the DOS 3 diskette. After COPY32.COM reads the entire file (or as much data as it can accommodate in its memory buffer), it asks you to swap disks if you specified the same drive for your DOS 3 and DOS 2.5 disks. With very large files, you may have to swap diskettes several times. If you are using two drives, the program copies and converts the file in a single operation.

After the file has been copied and converted, press [START] to return to the listing of files on your DOS 3 diskette, from which you may choose another file to convert.

If an error occurs during the copy process, COPY32.COM displays an error number and prompts you press [START] to restart, or [SELECT] to return to the DOS 2.5 menu.

Note: Unless you have two disk drives, you will be unable to convert files of more than 124,700 bytes (300 bytes less than the maximum file length possible under DOS 2.5).

DISKFIX.COM

This program begins by showing you the current drive number and a menu with these five options:

1. Change Drive #
2. Unerase File
3. Verify Disk
4. Rename File by #
5. Quit to DOS

Type the number of the function you wish to use but do not press [RETURN] after typing your choice. After activating an option, follow the prompts.

SETUP.COM

This program begins by showing you a menu with

these four options:

1. Change current drive number
2. Change system configuration
3. Set up an AUTORUN for Boot
0. Quit - Return to DOS

Menu selections 1 and 0 are used for "housekeeping" purposes. The two main functions of this utility are menu selections 2 and 3. Press the number key that corresponds to the function you wish to use, then follow the prompts.

Customer Support

Atari Corp. welcomes any questions you might have about your Atari Computer product.

Write to:

Atari Customer Relations
P.O. Box 61657
Sunnyvale, CA 94088

Please write the subject of your letter on the outside of the envelope.

We suggest that you contact your local Atari User Group. They are outstanding sources of information on how to get the most out of your Atari Computer. To receive a list of the user groups in your area, send a self-addressed stamped envelope to:

Atari User Group List
P.O. Box 61657
Sunnyvale, CA 94088

HANDICARDS™

Quick Reference Instructions for Atari Programs

- Organized commands
- Durable plastic (11" x 4 1/4")
- Easy to read
- Use on or off computer



Now available for:
ATARIWRITER®
Beginning BASIC



ACTUAL SIZE: 11" x 4 1/4"

As an introductory offer, User Group members may have the benefit of reduced prices through quantity group purchases. The regular price of HANDICARDS is \$8.95 each postpaid.

Ten (10) or more HANDICARDS may be purchased at \$5.95 each, shipping and handling included (to one address).

HANDIDISKS - multiple programs and files at the unbelievable price of \$5.95 per disk (plus \$2.00 shipping and handling per order).
Available Disks: PICTURES I, PICTURES II (SPECIFY KOALA OR MICROPainter), AMS MUSIC I, AMS MUSIC II, PRINTER UTILITIES, GENERAL UTILITIES, MODEM PROGRAMS, MENU PROGRAMS, SAMPLER DISK.

Only \$8.95 ea. postpaid

(Both for only \$15.95 ppd.)

NY residents add sales tax

HANDI PUBLISHING INC.

©Trademark of Atari Corp.
P.O. Box 453, Ardsley, NY 10502



Copyright © 1985 Antic Publishing.
WUN bulletin reprinted by permission.

Using 16-Bit Sound

by JERRY WHITE

(Copyright 1985 by Antic Publishing, Inc.)

(Editor's note: This submission by Jerry White was donated by Antic Magazine for the SLCC "Special Edition" Journal.)

Beef up your music from Atari BASIC with this short program by Antic Contributing Editor Jerry White. Learn how to program 16-bit dual-voice sound that gives you a well-tuned 7 octaves—instead of the thinner-sounding 4 octaves you'd ordinarily get. Works on all Atari computers of any memory size, with disk or cassette.

If you have experimented with SOUND commands in Atari BASIC, you probably noticed that some of the higher notes seem a bit flat. You may have also found that your lowest note is the B generated by SOUND 0,255,10,8.

Using SOUND commands with a distortion value of 10 for clear sound, you have a range of just over four octaves. If you'd like to fine-tune your music and extend that range to seven octaves, this tutorial will tell you how.

The SOUND 16 program will demonstrate what is called 16-bit sound. It is based on using two combined voices to create one sound. Only two sounds can be produced at once but the frequency of each pitch will be more accurate and much deeper bass notes can be generated.

The BASIC program uses an assembler subroutine to turn 16-bit sounds on and off. A commented source code listing has been provided for assembler hackers. The assembler routine also appears in the BASIC program as the DATA statements starting at line 20010.

Sound Ideas

by Lee Minard
STARFLEET Users Group, Denver

You have already heard me complain about the lack of realism in the sounds you get from the instructional books (and the ATARI can do wonders with sound) so here is another of my attempts to get more out of my 800XL!

This time I am working on the sound of a train. Why you ask? Well I did it for the challenge. And I learned a lot from it. I hope you do too.

This is an attempt to build the sound of a steam locomotive pulling out from a station. It gets close. I was never satisfied with the whistle sound. I would love to hear how you improved it.

The FOR/NEXT loop at line 130 reads this data and stores our subroutine in the string S16\$.

Using BASIC's USR function, the desired frequency and volume for one or two voices can be passed to the subroutine, as shown in the demo program. Note that the SOUND and POKE commands found in line 150 must be executed before your first USR call.

The BASIC program reads frequency data into an array called FREQ. This array stores 12 frequencies for each of seven octaves. Octave one contains the highest note frequencies, while octave seven contains the lowest bass notes.

Each octave begins with C as its lowest note (pitch 12), and ends with B as its highest note (pitch 1). Middle C (SOUND 0,121,10,8) is frequency 3414 or FREQ(4,12) in our array. The next highest note, C#, is FREQ(4,11). The next lowest note, B, is FREQ(5,1).

The program uses a countdown timer to clock delays. When you POKE a number from 1 to 255 into location 540, it will be decremented every 1/60th of a second. 1/60th of a second is called a "jiffy." Thus, if you set the variable WAIT=60, then go to the subroutine beginning at line 480, you will return in one second.

Octave, pitch and note will be displayed on the screen as the program cycles through all frequencies. Next, the double 16-bit sound option will be used to demonstrate the use of consecutive octaves. Finally, a short tune is played just before the program ends.

Seeing and hearing the demonstration program as it runs and studying the program listing should help you understand the use of 16-bit sound. With any luck, you and your Atari will soon be making beautiful music together.

System Key to the Key System

THE KEY SYSTEM - AN ATARI BBS OPERATED BY THE SAN LEANDRO COMPUTER CLUB

415 352-5528 - ALWAYS OPEN
QUICK REFERENCE CARD

By Mike Sawley

GENERAL SYSTEM COMMANDS: USE THESE COMMANDS AT THE MAIN *GO PROMPT.

[A] ASCII/ATASCII TOGGLE

Used to switch between these two translation modes. If you change the translation mode at your end only, then the BBS will no longer understand your commands! Use this toggle!

[D] DOWNLOAD A FILE

You must give the file name exactly as it is shown in the files listing. Colons (;) and periods (.) are not allowed. If you misspell the file name, the system will tell you "File not found."

[E] ENTER A MESSAGE

Drops you into the message enter routine of the currently active message base. See further commands in the message section.

[F] FILES LISTING

Presents a list of the available files for downloading. You will be asked to supply a category from a menu. You may skip this menu by appending the category code to the [F] command. [F G] will find all the games. [F U] will find all the utilities.

[G] GOOD BYE

The proper way to log off. Confirmation is made to be sure you really want to leave.

[K] KILL MESSAGE

Drops you into the delete message

routine of the currently active message base. You need to supply a message number to kill. The message must have been posted by you or addressed to you for you to delete it.

[L] LEAVE MESSAGE FOR SYSOP

Lets you enter a private message to the Sysop. The system will accept messages of 15 lines and 80 columns maximum. Since these messages go to the system printer, no one but the Sysop will see it. Also see message editor commands.

[M] ELECTRONIC MAIL

Activates the Electronic Mail Base. This is the only place where private messages are kept. You may only read messages addressed to you and/or posted by you. Since the system scans the base sequentially it may take some time if your message is at or near the end of the queue. When you first enter this section you will be told if you have any messages waiting.

[P] PROFILE OF YOUR PASSWORD

Lets you see how you have been using the system. In addition, you may change the following contents of your file:

- 1) You may enter a new password (must be 4 characters!).
- 2) You may enter a new phone number.
- 3) You may change your system parameters.
- 4) Toggle system clock on/off.

[R] ENTER THE CURRENTLY ACTIVE MESSAGE BASE

Will tell you what message base is active and ask if you want to search for messages addressed to you. If you have messages waiting, you will be told which messages to read. You are then advised how many messages are in the base and the low and high message numbers. You are then given the mes-

sage base Select: prompt. See message base commands below.

[U] UPLOAD A FILE TO THE SYSTEM

You must supply a file name. As with downloading, colons and periods are not allowed. You have to tell the system how long the file is in single density sectors. You must catalog the file from menus that will be displayed. You should always use XMODEM for uploading. If you use non-XMODEM, begin your transfer IMMEDIATELY, since a pause of 10 seconds tells the system to save out the file or abort if there has been no data sent.

[Y] YELL FOR SYSOP

If paging is off you will be told that the Sysop is not around. If paging is on a message will be displayed telling you that the Sysop is being paged and you may continue to use the system. The Sysop has the ability to break in at any point.

[Z] MESSAGE ZONE SELECTION

There are several message zones (bases) available on the system. You will be given a list and asked to choose one of them by number.

[*] DATA BASE AND HELP FILES

There are a number of interesting text files to be found here, including a short help file. You should choose the file to read by its number from the menu. You will be given the option to use XMODEM when reading the file, but since they are currently all text files, this is not really necessary unless you want to save one of them to your disk.

[?] COMMAND LIST

Will display a list of available commands.

MESSAGE BASE COMMANDS

USE THESE COMMANDS AT THE MESSAGE BASE SELECT: PROMPT

[R] READ MESSAGES COMPLETE

Lets you read the complete message.

[B] BRIEF MESSAGES

Reads only the header of the message (to, from, date, title).

[T] TITLE OF MESSAGES

Reads only the title and date of the message. Lets you mark messages for future reading (See [M]).

[M] READ MARKED MESSAGES

Reads the messages you marked using the [T] option. This must be the next command after [T] or the list you generated with [T] will be lost. The maximum number of messages that can be marked is 16.

[D] DELETE A MESSAGE

Lets you remove a message from the system. The message must have been addressed from you or posted by you to be deleted.

[S] SEND A MESSAGE

Lets you enter a message into the system. Similar to the [E] command but from the message base instead of the main prompt. See message editor commands below.

[C] CONTINUOUS TOGGLE

The default is to read a message, display a command string at the end of each message. If you would like to do away with this command string, turn on continuous display. There will be a short pause at the end of each message. If you press a key during this pause, the command string will be displayed. If you do not press a key, the next message in your queue will be displayed.

[Q] QUIT

Exits the message base and takes you to the main *Go prompt. If you have posted or deleted messages, there will be a short delay while the system writes out a new index.

[Z] MESSAGE ZONE SELECTION

Lets you change message zones (bases). Choose the one you want to go to by number from the list.

NOTE: When requesting the [R], [B] and [T] commands, you will be asked for message numbers. There are several ways to tell the system which messages to display:

- [+] ALL messages, lowest to highest.
- [-] ALL messages, highest to lowest.
- [3,5,9,2] gets these message numbers.
- [40-50] gets all messages from 40 to 50 inclusive.
- [20,22,70-65,16] gets combination.

MESSAGE EDITOR COMMANDS**USE THESE COMMANDS FROM WITHIN THE MESSAGE EDITOR**

(These commands must be entered as the first characters of a new line and must include the [/] so the system can tell a command from a character that is part of the message.)

[/L] LIST MESSAGE

Lets what you have entered. This command works two ways. If entered as shown, it will list the entire message. If a number is appended to the command ([/L5]), the system will list the next 5 lines starting with the line that you are on.

[/T] GO TO TOP OF MESSAGE

Takes you to the top line (line 1) of your message.

[/B] GO TO BOTTOM OF MESSAGE

Takes you to the bottom line of your message so you can add to the message.

[/N] NEXT LINE

Takes you to the next line in the message so you can edit it.

[/U] MOVE UP

Moves you up a line.

[/G--] GO TO LINE NUMBER ##

Lets you go to line number -- for editing.

[/D] DELETES THE CURRENT LINE

Removes the line that you are on from

the message. All lower lines will be moved up to fill in the space.

[/I] INSERT A LINE

Anything after the second slash will be inserted as a new line in your message. All lower line will be moved down. Please note that the second slash is required!

[/C/string1/string2] CORRECTION

Finds the FIRST occurrence of [string1] and replaces it with [string2]. Note that all three slashes are required.

[/S] SAVE

When you are satisfied with your message, this will save (or post) it.

[/A] ABORT THE MESSAGE

Lets you get out of the message enter routine without saving the message.

[/? or /H] HELP

Lets a brief summary of the message editor commands.

NOTE: The message editor is line oriented. You must be on a line to edit it. Also please note the symbol printed after the line number. [A]) means that you are creating this line. [A :] means you are editing this line.

OTHER COMMANDS**[CTRL S] PAUSE DATA TRANSMISSION**

Used to tell the system to stop sending you data for a while so you can read it. The system will automatically start up again after about 3 minutes.

[CTRL G] RESUME DATA TRANSMISSION

Used after a CTRL S was sent to tell the system to start sending data again.

[CTRL C] CANCEL COMMAND

Will generally cancel any command that you have entered.

[CTRL N] READ NEXT MESSAGE

Lets you skip on to the next message in your queue without having to wait for the command string at the end of the message.

[CTRL X] CANCEL XMODEM TRANSFER

Will cancel an XMODEM transfer before any data is sent. To cancel after data transmission has begun, refer to your terminal program documentation.

TERMINAL PARAMETERS

The Key System is an Atari BBS but any computer type may log on. If you are not an Atari, or log on in ASCII translation with an Atari, you must specify your terminal parameters. These may be changed with the [P] command. The terminal parameters are:

- 1) Line feeds Y/N
- 2) Clear Screen code (or Home value).

The system is asking for the decimal value of the ASCII code that will clear your screen and place the cursor at the top left of the screen.

- 3) Line Length. Default is 40 columns. This is used to determine how long a line may be type in before the system generates a RETURN during message entry. If you specify 40 or less the system will enter a return after that many characters. Any message you enter will be a maximum of 30 lines of that many characters. If you specify 41 to 80 columns the system will enter a return after that many characters again, but your message may be a maximum of 15 lines of that many characters. A message to the Sysop will always be 15 lines of 80 columns, the format of the printer.

NOTES ON XMODEM TRANSFER

AMODEM, SMARTERM, ETMODEM, TERM 1030 and HOMETERM all support XMODEM file transfer. It is the only way to reliably get tokenized or object files. XMODEM is a way of sending blocks of data so that there is a checksum made at each end. If the check-

sums match the computers go on to the next block. If there is an error the current block is resent. There will be a retry up to about 9 times before the transfer is aborted. The file is also saved out to disk automatically at the end of the transfer. The BBS and most terminal programs work differently when it comes to setting up for XMODEM transfer. The BBS needs a [D] or [U] command to tell it that a file transfer is to take place. The BBS will ask, "Are you using XMODEM (Y/N)." Answer Yes or No as appropriate. On the other hand, most terminal programs have FOUR different commands for file transfer. In most versions of AMODEM there are: [C] Capture incoming data. [U] Upload data without XMODEM. [R] Receive data with XMODEM. [S] SEND data with XMODEM. As you can imagine, it is easy to get all these commands confused so be careful! Use the same protocol at both ends for a successful transfer.

NOTES ON COMMAND STACKING

The BBS allows you to bypass most of the sub-prompts and sub-menus with command stacking. The places you can do this are too numerous to list, however a few examples should get you going.

[Z 2] - Takes you to message base number 2 and starts reading messages in the reverse direction.

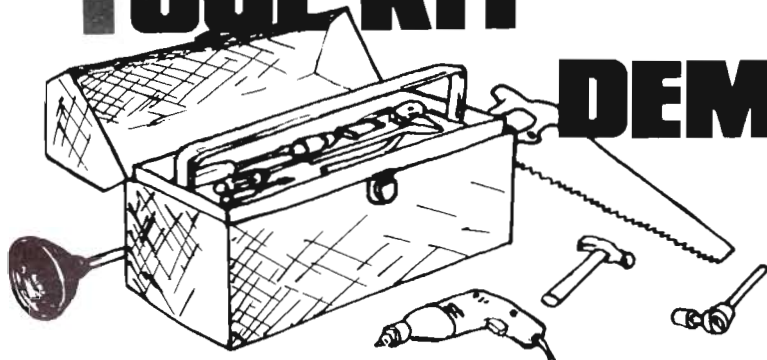
[R 80-100] Begins reading messages in the current message base starting with message 80 and ending with message 100.

[Z 4 E] Switches to message zone number 4 and drops you into the message editor.

[* 3] Reads Data Base item number 3. As you can see, command stacking is a fast way to get around the system. However, until you get used to the system and know where everything is you could easily get lost, since you are skipping past most of the sub-prompts. Just experiment and see what happens. You cannot hurt the system!

TOOL KIT

DEMO



by Jim Warren

This little demo shows only part of what you can do with four players using the PM and joystick routines of the ACTION! TOOL KIT. The routines allow you to easily manipulate players without knowing much more about them than what they are.

They are four independent vertical stripes that can be shown in three widths, 2 vertical resolutions, any color and can be moved rapidly about without interfering with anything else on the screen. They can be made to appear in front of or behind other screen objects. Their confluence with any other object can be detected at any time. Any section of them can be shown at any time so that they do not have to appear as stripes. They can be animated by changing sections of the stripe shown on the screen. If they are put side by side they can almost cover the screen. In short, what they are is neat!

This little demo was written in ACTION! and uses the player missile and joystick routines on the ACTION! TOOL KIT. There is an editor that allows you to draw on an eight by eight grid and then display your drawings as sections of the player0 and as the control characters displayed on the title screen. You can display various lengths of the player starting at various places on the stripe and you can move the player around with the joystick. It is a simple little demo designed to give you a "feel" for player missile graphics. If you experiment with the program you will quickly grasp the ideas involved.

The demo uses an Antic mode 4 display list so that we can have four colored text in a graphics 0 "environment". I have had to draw a new set of letters and numbers, however, because the standard character set in ROM does not show clearly in Antic mode 4. I drew the

new character set with the editor in this demo. The editor shows a section of player0 and a character in standard and inverse mode just to the left of the editing box. You can redraw 9 sections of player0 and the 9 graphics characters that are displayed in the title screen by pressing OPTION to select the section that you want. You then draw in the box with the joystick. Then press START to make a simultaneous change in the player and the joystick. The editing page will give you a chance to experiment with the four color characters of the IR (instruction register, in this case antic 4) modes of the antic chip.

In the IR antic modes, the characters are not displayed as an eight by eight grid of dots as in graphics 0 where the dots are displayed individually, but as an eight by eight grid of dots where the dots are displayed in pairs. The color of each two bit (no puns please!) pair is determined by how the bits in that pair are set. The gold and blue bars at the top and bottom of the editing box indicate the dots that are displayed together. Think of the box as four columns across and eight lines deep with alternating "gold" and "blue" columns. There is a legend at the bottom of the screen showing how the paired bits are colored. If no bits are set in a column then that pair takes color from register 4. Since register four also sets the background color, that pair of bits will not appear. If both bits are set in a column then that pair of bits is colored by register 2 in standard mode and by register 3 in inverse mode. If the left bit only is set in a column then that pair of bits takes color from register 1. If the right bit only is set then that pair of bits is displayed with the color of register 0. You can change the color of the registers by holding down the ESC, 1, 2, 3, 4, 0 keys. You can toggle the direction of the color change by pressing the up or down arrow so that if you go past an interesting color, just change direction and go back. Up adds to the values displayed and down subtracts.

XL-ator

NEW XL TRANSLATOR/MENU

Bill Eash

Included on the *Special Edition* disk is a special XL Translator. This package was put together by Dan Philipps, an SLCC member for the past year. Dan used the "FIX-XL" (OCT. 84 DOM), the "SPIFFY MENU" (APR. 84 DOM) and an automatic turn off of BASIC.

This was put together as an "AUTORUN.SYS" file (with much rearranging and debugging of code). Together with a standard DOS, these files will allow you to have multiple BINARY files all on the same disk as your translator. You will have to rename "XLATOR" to "AUTORUN.SYS".

This is how you use it:

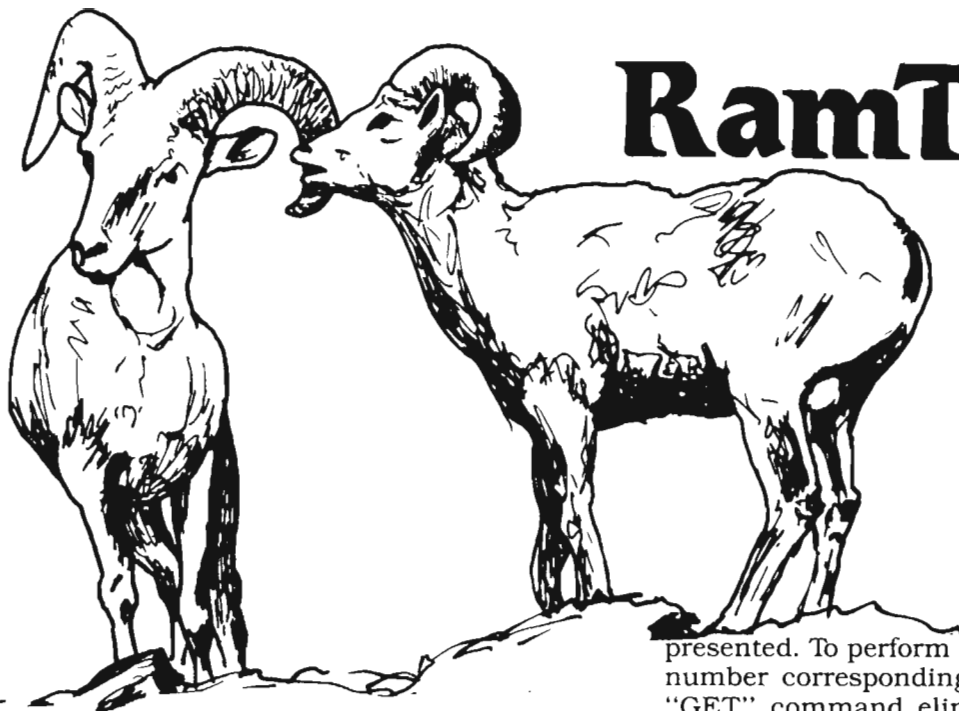
- 1) Boot disk which auto loads the translator
- 2) AUTOMATICALLY removes BASIC
- 3) Pulls up a menu with your programs on it
- 4) Press the number next to the program selection you want

Non XL users can use this also. The computer just ignores the translator load and BASIC turn off. Just remember to pull your BASIC cartridge. You can have faster loads, though, if you just use DOS and the SPIFFY MENU without having to load the translator.

There is an **INSTRUCTION** program on the disk and it is aimed at the beginning user. The instructions were written by my son, Joe, in ACTION(c). Joe has called this package the "XLATOR" for the XL - TRANSLATOR.

Although the instruction program states that this package does not give the option to load a binary boot disk, it will work. When the menu appears remove the XLATOR disk, insert the boot disk and press 1 (as if you selected program 1 on the menu). The disk will now boot. The translator is still in the system and basic is still removed. This was tested using Letter Perfect, Data Perfect, and others. When Filemanager (which requires BASIC) was tested, the message appeared to insert the BASIC Cartridge verifying BASIC was still removed.

RamTalker



by Randy Holmes - S.T.A.T.U.S. Users' Group

(Editor's note: This article was taken from the S.T.A.T.U.S. newsletter, January, 1985. It is Randy's first in a series of articles on digital sound. Others may follow in future issues of the SLCC Journal.)

This is the first in a series of articles on the digital recording of sound with your Atari. As you may recall, an article in the July, 1983 issue of ANTIC magazine presented an article on this same subject. The ANTIC article included a program, which itself was an adaptation of an Apple program from an early issue of Byte magazine.

In this set of articles, I will present several modifications to what was originally an awkwardly coded and hard to follow program. In the course of constantly updating this program, we should all learn something about programming, digital sound and special functions of the Atari. Rather than enumerate the changes that have been made to Ed Stewart's program (the program is now almost unrecognizable), I will discuss the specific points of the program as it is now.

The program requires a special circuit to be built to allow the Atari to be able to read an analog voltage at its paddle port (Port 3. Sorry, XL owners, we'll fix this problem next month: in the meantime, get your circuit built). The schematic is included in this article. In working with this circuit, I have found that the two MegOhm potentiometer that was included in the original circuit schematic may be eliminated with no ill effects on the circuit's operation. Once you have this circuit built (see accompanying construction article), it is possible to read a changing voltage (as might be produced by a microphone), as a resistance value from 0 to 255. This circuit works well in its present form, although we may change it in future articles.

The program, **RAMTALKER**, is a friendly, fast, easy-to-use program. After initialization, a menu is

presented. To perform a desired function, press the number corresponding to that function; a BASIC "GET" command eliminates the need to type a [RETURN]. If the function you picked is not the one you wanted, simply press [RETURN], and the program will take you back to the original menu. After a function is selected, the program will prompt you for more information.

RECORD -will ask for a sample speed. Sample speed is the speed at which the program will read the information coming in at the paddle port. A sample speed of 1 will render the highest quality sound, while a sample speed of 255 will result in nearly unintelligible noise. Once a sample speed is specified, followed by a [RETURN], press the START button to begin recording.

PLAY -will ask for a sample speed. This sample speed will be the speed at which the sound information contained in memory will be played back. A good speed is usually around 55, giving a natural sound to the playback. Of course, you may wish to have your recorded sounds resemble the Chipmunks of Lurch, in which case you would choose a higher or lower sample speed. Again, pressing START after giving a sample speed will then begin playback.

THROUGHPUT -will ask for a recording sample speed, and will allow you to play sounds through the speaker with no time limitations. Press START to begin, and a [SYSTEM RESET] will get you out of this one.

SAVE -will ask you for a file name. Include "D:" or "C:" in your file specifications.

LOAD -will ask you for a file name. Again, use "D:" or "C:" in your specs. The program uses the Atari's Central Input/Output (CIO) routines, which makes saving and loading sound files quite fast, even though sound files are 132 sectors long (Single Density).

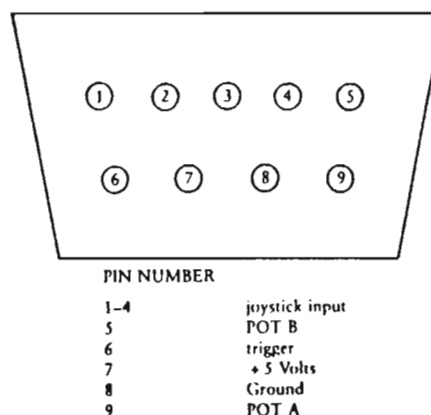
WAVEFORM PLOT -does just what it says. It plots a picture of the sound stored in memory (in locations 16384 to 32767, a full 16K) on a graph of Time against

Frequency. The Time at a sample speed of 1 is a little over 7 seconds, and I have not measured the frequency response of the system. We'll do that in another article. The sound is divided into four separate bands, so that we are able to plot the entire contents of memory with some detail. To me, the waveform plotting routine is an exciting feature of this program. You can say a few words into your Atari, and then have the computer show you what your voice looks like. You can see how different sounds are similar, and where they are different. Looking at a plot of myself saying "file" and "while" gives me quite an appreciation for the difficult task that a speech recognition system has to perform.

These are the basics of RAMTALKER. In the coming months we will modify the program even further. Some things I hope we can do with the program: editing the sounds in memory, improvement of sound quality, special effects (echo and speed effects) and maybe, just maybe, some speech recognition.

I am sure you have your own ideas as to where this program could go. I would be happy to hear your suggestions, criticism and comments.

Fig.1 Pin Configuration of Console End



RamTalker Circuit Construction Notes and Parts List

In place of a microphone, you may want to substitute a simple quarter-inch jack of RCA-type jack to allow you to plug in a guitar, keyboard, or tape player; this will give you higher quality sound than recording from a microphone will. Adjust the volume control on your source to get the best, most distortion-free sound.

PARTS

1 .1 μ F nonpolarized capacitor available at Radio Shacks everywhere.

1 NPN transistor 2N2222, this is a general purpose transistor, almost any NPN transistor will work.

1 100K Ohm Fixed resistor (Brown-Black-Yellow resistor code)

1 DB-9 connector plug, that's a joystick plug to you and me.

You may mount these components on a small circuit board (see Figure No. 2) or simply wire them together without a board; either way will work fine.

Remember not to keep the soldering iron on the transistor too long, or you may damage the component. The same goes for the resistor and the capacitor, but they have a higher tolerance to heat. Also, be sure to observe the Emitter-Base-Collector specifications in the circuit. The back of the transistor Package should have the pins specified in a diagram such like the transistor in Figure 2.

After the circuit is constructed, simply run a two-conductor wire from the specified points on the circuit to the correct pins (7&9) on your port plug. Test the circuit by plugging it into Port 3, and run the RAM-TALKER program. Select the THROUGHPUT option, with a sample speed of 1. Plug in a microphone, guitar, tape player or some other source, and see if any sound comes from the TV/Monitor speaker. If not, go back and check your wiring, making sure all connections are good.

With this circuit up and running, you are ready to begin digital sound recording with your Atari computer!!

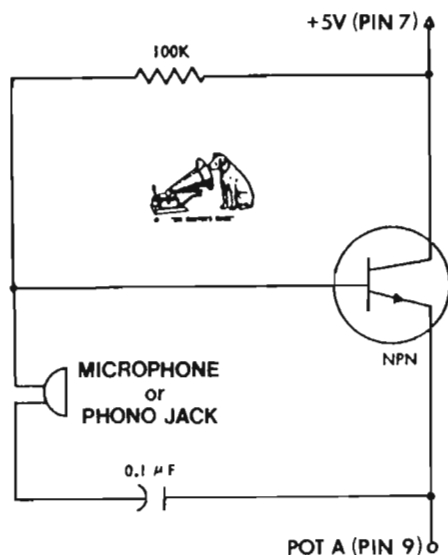


Fig. 2 Voice Input Circuit

DRIVE MODIFICATIONS

810



by Banford Wong

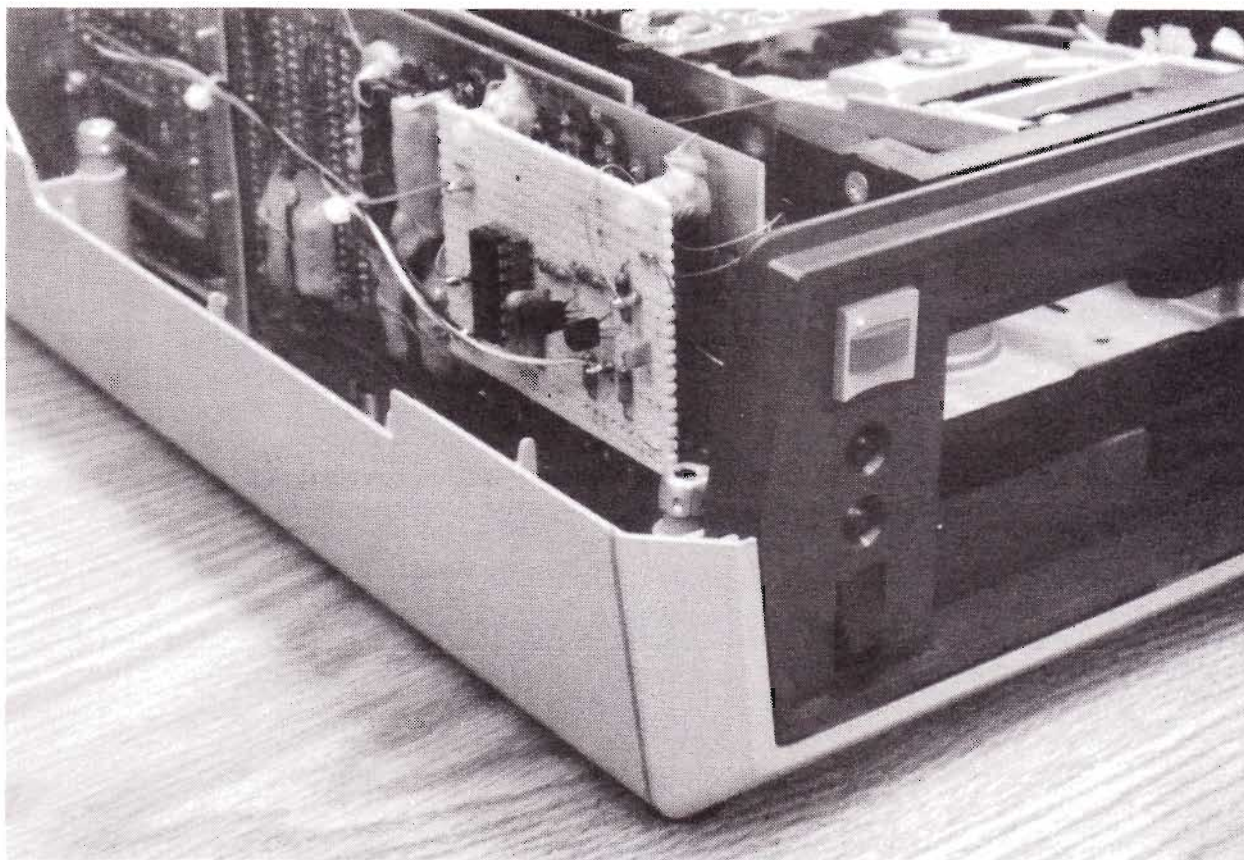
(Editor's note: Do not attempt to do this drive modification unless you have had experience doing soldering and wiring. The SLCC JOURNAL is not responsible for the accuracy of this modification or damage caused by this. There are other versions of this switch available if this does not suit your needs.)

A previously published article in the **SLCC Journal** on an 810 Disk Drive write-protect modification prompted me to come up with this version. It uses a momentary switch, a toggle circuit and an LED for "OVERRIDE" indicator. This idea is more involved but I believe the advantages it provides are worth the effort.

The main advantage of this override modification is that the circuit will always come up in the normal-safe mode whenever the disk drive is turned on. Second, because of the unique momentary switch with a built-in LED and the location on the front bezel on which the switch is mounted, the disk drive looks very nice.

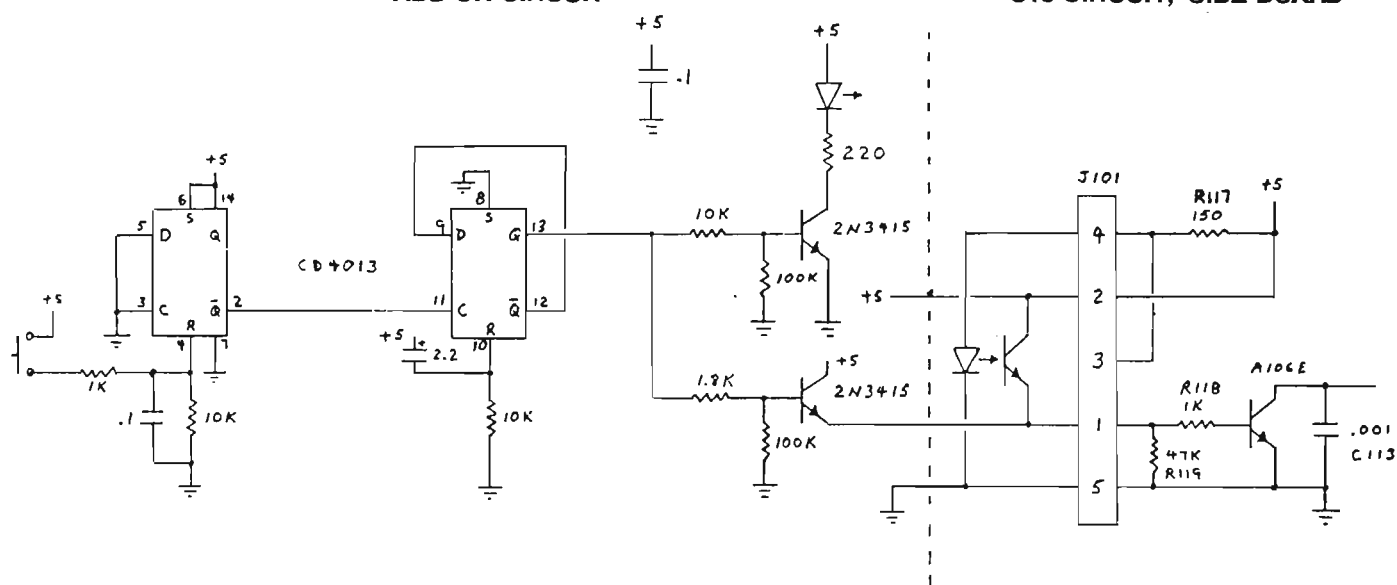
This modification uses a momentary switch with a built-in LED, a CD4013 dual D Flip-Flop, and 2 NPN transistors as output drivers. Section 1 of the D Flip-Flop is used as a switch debouncer. The RC network on pin 4 provides the filtering. Section 2 of the Flip-Flop toggles each time the switch is depressed. The RC network on pin 10 assures that the circuit will come up in the safe mode whenever the disk drive is turned on. The momentary switch is pressed to activate the write-protect override and the LED will indicate its status.

The switch with built-in LED I chose to use is strictly for cosmetic reasons. Any momentary switch and LED would work. The switch/LED is the UNIMEC series made by MEC. The circuit is built on a small perf board and mounted towards the front of the side board using 2 threaded stand-offs. To avoid having to drill holes through the side board I used RTV to secure the spacers to it. The MEC switch is mounted on a square hole where the Atari logo is. Be very careful cutting the square hole. A tight fit is required. The switch binds slightly on the side board. A fiber spacer added between the PC board and the chassis took care of that. The ADD ON circuit is connected to the disk drive via J101 on pins 1,2, and 5. Pin 1 is on the bottom.



ADD ON CIRCUIT

810 CIRCUIT, SIDE BOARD



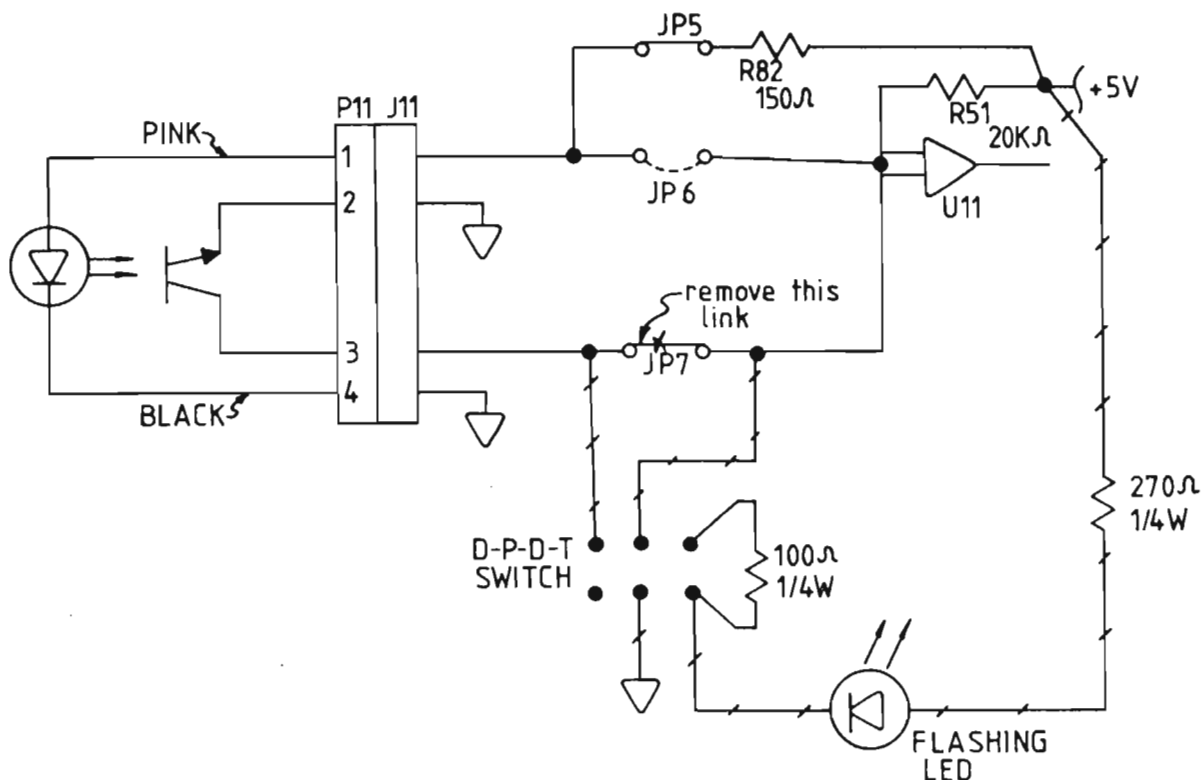
(Editor's note: Do not attempt to do this drive modification unless you have had experience doing soldering and wiring. The SLCC JOURNAL is not responsible for the accuracy of this modification or damage caused by this. There are other versions of this switch available if this does not suit your needs.)

Undo the four screws in the drives casing and lift off the cover. Choose a suitable place to mount the switch and LED before starting work. The LED would naturally be on the front face of the drive but the switch may be mounted anywhere it will not foul the mechanism or touch any of the components.

Connect the poles of the switch as shown in the circuit diagram below, 5 volts can be found at "**TP13**" located near the middle of the board, approximately a quarter of the way from the rear. The earth can be connected to "**TP15**" located near "**TP13**".

The 100 Ohm resistor is there as a safeguard. It is recommended but not essential. The more adventurous of you could get a 3 position, 3 pole switch wired as follows:

POSITION 3 - as shown here.




DIAGRAM

KEY:-

EXISTING CIRCUIT

MODIFICATION


SWITCH



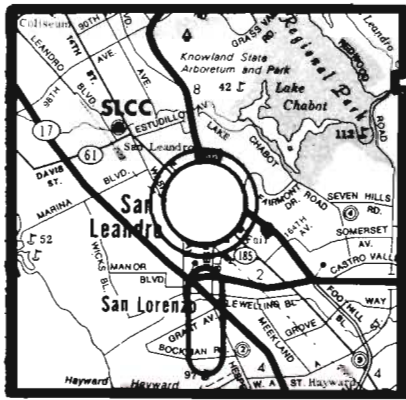
LED



EARTH


RESISTOR

Light
Emitting
Diode polarities.
A K [-ve is short lead]



Copyright 1985 Michael Curry.
All Rights Reserved.

Introduction:

Have you ever wondered how the Atari file manager allocates space? Or where a file resides on a disk? Or even where the free space on the disk is located? DMAP will answer these and other mysteries of the Atari disk structure.

DMAP was designed to display the Atari diskette file structure in a meaningful and (hopefully) entertaining way. DMAP is easy to use. There are no complicated options or commands to remember.

BASIC DMAP INSTRUCTIONS:

BASIC DMAP requires an Atari BASIC Cartridge. Boot your DOS 2.0S disk.

Insert the disk with BASIC DMAP into the drive.

Type: RUN "D:DMAP.BAS"[return]

DMAP will then ask you for the drive number you wish to map.

Enter the drive # and press return.

DMAP will then ask you to insert the disk to map in the selected drive and press return.

After you have done this, DMAP will show a map of used and unused sectors on the screen.

When you are done viewing the map, press any key to rerun the program.

EXTENDED DMAP INSTRUCTIONS:

Boot your DOS 2.0S disk.

If you have a BASIC Cartridge in the computer, type DOS [return].

When the DUP menu appears, remove the disk from the drive and insert the disk containing DMAP.COM.

Type L (for binary load) and press [return].

Now type DMAP.COM and press [return].

After the program loads, it will run automatically.

The screen will now show an introductory message and ask what drive you wish to map.

Enter the drive # containing the disk you wish to map or 0 to return to DOS.

The program will now ask you to insert the disk to be mapped into the selected drive. If you haven't already done so, do it now. Press [return] to proceed.

You will now see a menu asking you to select which function is desired.

DISK MAP

You may now select:

1: Map Entire Diskette

This will show the entire diskette surface, including used and unused sectors.

2: Trace Specific File

This will present you with a menu of files on the diskette and allow you to select which file you want to trace.

After DMAP is finished with either option, it will display "Press any key to continue" in the upper right hand part of the screen. When you are finished looking at the map, press any key and DMAP will return to the starting display and you will be allowed to repeat the above functions.

Atari Disk Structure Explained

When the Atari DOS creates a file on the diskette it must find an unused section of the diskette to place the file. In order to quickly find where the unused sections of the Atari diskette are, the DOS uses what is known as the Volume Table Of Contents (VTOC).

The VTOC is a single sector on the disk that is used to maintain a record of where used and unused sectors are located.

The VTOC sector is organized as follows:

Bytes 0 through 9:

Miscellaneous Information

0: Type Code (0 in DOS 2.0)

1: # of sectors total (low byte)

2: # of sectors total (high byte)

3: # of unused sectors (low byte)

4: # of unused sectors (high byte)

5: Reserved

6-9: Unused

Bytes 10 (\$A) through 99 (\$63):

Sector Usage Map

720 bits, each bit representing a sector in DOS 2.0

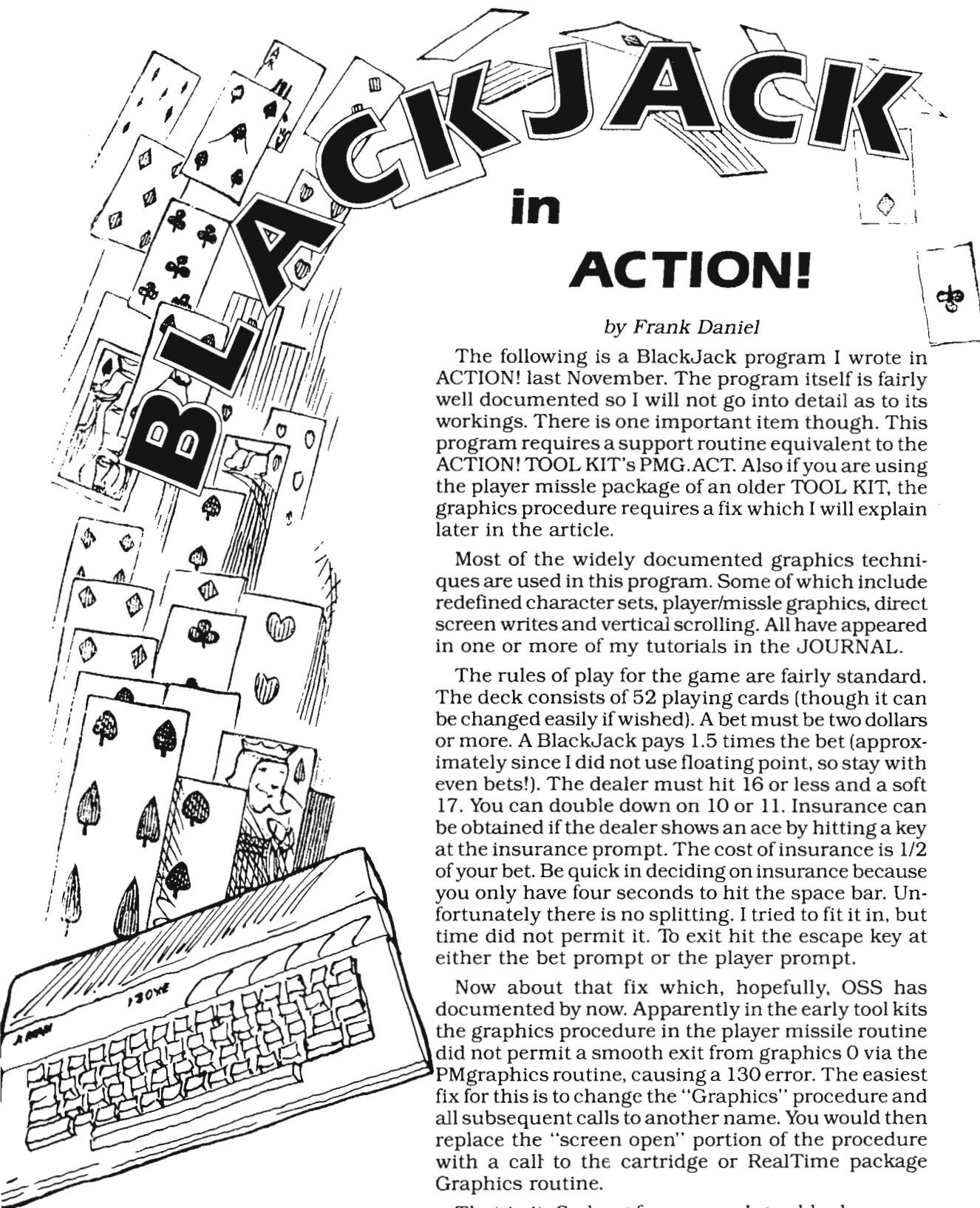
If a bit is 1, the sector is unused.

If a bit is 0, the sector is used.

For a thorough explanation of the Atari DOS 2.0S structure, please buy a copy of "Inside Atari Dos", the most complete and detailed documentation on the subject I have come across.

Thanks to Bill Wilkinson of OSS and Atari for explaining how everything works.

EXTENDED DMAP was written in Deep Blue C.



by Frank Daniel

The following is a BlackJack program I wrote in ACTION! last November. The program itself is fairly well documented so I will not go into detail as to its workings. There is one important item though. This program requires a support routine equivalent to the ACTION! TOOL KIT's PMG.ACT. Also if you are using the player missile package of an older TOOL KIT, the graphics procedure requires a fix which I will explain later in the article.

Most of the widely documented graphics techniques are used in this program. Some of which include redefined character sets, player/missile graphics, direct screen writes and vertical scrolling. All have appeared in one or more of my tutorials in the JOURNAL.

The rules of play for the game are fairly standard. The deck consists of 52 playing cards (though it can be changed easily if wished). A bet must be two dollars or more. A BlackJack pays 1.5 times the bet (approximately since I did not use floating point, so stay with even bets!). The dealer must hit 16 or less and a soft 17. You can double down on 10 or 11. Insurance can be obtained if the dealer shows an ace by hitting a key at the insurance prompt. The cost of insurance is 1/2 of your bet. Be quick in deciding on insurance because you only have four seconds to hit the space bar. Unfortunately there is no splitting. I tried to fit it in, but time did not permit it. To exit hit the escape key at either the bet prompt or the player prompt.

Now about that fix which, hopefully, OSS has documented by now. Apparently in the early tool kits the graphics procedure in the player missile routine did not permit a smooth exit from graphics 0 via the PMgraphics routine, causing a 130 error. The easiest fix for this is to change the "Graphics" procedure and all subsequent calls to another name. You would then replace the "screen open" portion of the procedure with a call to the cartridge or RealTime package Graphics routine.

That is it. So long for now and good-luck.

Direct Screen Writing

Direct Screen Writing

by Frank Daniel

Some years ago I was writing a program that in order to be user friendly required a few menus. Well as you can imagine, this was not too much of a problem. Anybody that has ever written a multi-task utility has used a menu routine at one time or another.

But as the program got larger, the number of menus got larger and most of the menus were getting sub-menus. This WAS getting to be a problem. Not only were these menus taking up a lot of memory, but it was taking longer and longer to get from point A to point B in the program.

Now there are two thing I really hate. One is programs that gobble up too much memory. The other is waiting for the program to finish printing a menu. I faced a real dilemma. It is bad enough having just one of these problems in my programs. But both?? NO WAY!! My self respect could not take it. I had to do something!!

I was now faced with three options if I were to continue the project. These were:

1. Doing a complete rewrite of the command processor into a CPM type system.
2. Developing a hybrid which would be a cross between the menu system and the CPM system.
3. Find a way to change the menus fast.

I had to rule out the first option right off the top. A major rewrite just could not be done in the time available. The second option went very quickly afterwards. Though it would not mean a major rewrite, I just could not bring myself to do it (self respect again... drat it). All that left was changing the menus rapidly.

There are two methods of updating

the screen quickly. One is page flipping and the other is direct screen writing. Page flipping is the fastest method of changing the display known to a ATARI programmer. Just change two bytes in the display list and the whole screen changes. But with all this speed comes a few problems.

One, you have to preset all of your menus. By that I mean you have to make sure that all of the characters in your menu have been offset correctly. This is because ANTIC has its own set of character values which are very different from ASCII or ATASCII. Another problem is arranging the menus in memory. The ANTIC is a bit touchy about which page boundaries get passed when describing the screen's data area. (Hint: NEVER-EVER try to pass a 4K boundary!!)

The worst problem with page flipping though was the amount of memory it would use. The "GRAPHICS 0" screen mode uses 960 bytes for its Load Memory Scan (LMS) or data area. Page flipping would require that a number of blocks this size be set aside for the exclusive use of the menu driver. Add to this the dead areas between the menus needed to prevent page boundary problems and you can easily see that the amount of usable program memory is quickly diminished.

That leaves direct screen writes. Doing direct writes also has its problems. The first, like page-flipping, is that most of the text requires offsetting. This is not a big problem. There are many ways to rectify this. You can precalculate the offsets or write a short program that does it for you. Another method is to include an offset routine in the program. This is not very efficient for a menu driver but is very necessary when the text is varied or unknown. I actually use this method in the preceding demo.

Another problem with direct writes is parameter passing. How do you tell the routine where the text is, how long it is and where on the screen to put it. The

solution to this are also varied. The first that comes to mind is reserving a place in memory for the parameters. When dealing with BASIC though, it is easier to use the stack.

This brings me to the BASIC demo that follows. The demo will permit a BASIC user to write directly to the screen.

To call the routine you use the BASIC command

A = USR(CDE,X,Y,ADR(A\$),LEN(A\$))

or the alternate

A = (CDE,ADR(A\$),LEN(A\$)).

CDE is the address of the machine code string. X is the screen column and Y is the row position where you want to start the display. A\$ is the string to be displayed. If you do not pass a X and Y parameter, the routine assumes that you want to use the present cursor position as default.

The routine will display all characters with one exception. The EOL character (155) is used as a line delimiter. This for multi-line displays without the need of counting the characters. To go to the next line, simply insert an EOL character at the appropriate location in the string. One warning, the routine assumes that your starting column position is the left margin and will start from there.

While we are on the subject of warnings, let me caution you about a few items.

First, with the exception of screen position, the routine DOES NOT do any error checking. You can pass strings larger than the screen. This normally is not a problem, but forlorn is the person who does this with a relocated display list and no backup!

If you do make an error in the screen position, the value 141 (cursor out of range) will be passed back.

Do not let the machine code call a subroutine. BASIC gets a little confused. The program does not crash, but does not work right.

Along with the BASIC demo I have included the assembly listing of the routine.

A View From Japan

A Cry For Help



By MAKOTO NAGATA

2117 HASTE ST. #310
BERKELEY, CA 94704

*U.S. representative.
Fuji Atari Computer Users Group, Japan
Member, San Leandro Computer Club*

(Editor's note: The following set of superb articles are by new SLCC member Makoto Nagata. We have decided to devote a whole section to his articles. We hope to hear more from Makoto!)

Hello to all of the members of the S.L.C.C. I really thank you for giving me a place to introduce our club, Fuji Atari Computer Users Group.

One very unfortunate situation has made it extremely difficult to maintain our club; there has been no dealer of Atari computers in Japan and no support from Atari itself.

Our club was founded by Mr. Iori Fujita (who is now in Peru) and myself in 1982. Shortly before that time, several computer stores in Japan individually imported Atari computers and sold them. There were rumors that Atari had a plan to enter the Japanese market and it is likely that those dealers wanted to take them in advance.

Of course we bought Atari because it was the best we had ever seen. But Atari did not come and one day we got letters from those computer stores saying, "Sorry, we are not going to support you any more. Here is a list of those to whom we sold Atari's. It is your own choice to keep your Atari or throw it away and buy an Apple (!)". I sent letters to everyone on that list, and our club was founded.

Although our club was started by Japanese who bought Ataris at those stores, there was another group of people out there weeping at their Atari in Japan; those Americans or Japanese who brought their Atari from the U.S.

Mr. Iori Fujita was one of those people. He and those

Americans, including Mr. Joe Langdon and Mr. Bob Rutherford, greatly helped our club by individually importing magazines, software and hardware from the U.S. I really thank those members for their devoted activities.

Some of our members were so devoted that they almost ruined their usual activities, because we had to do everything by ourselves.

Early on, our main activity was to ask Atari to sell their computers in Japan but we soon gave up because they would not listen to us. Atari has never supported us although our club has been an official one. We had to find another way to survive by ourselves.

There was an Atari Coin-Op Division office in Tokyo before the selling of Atari last year. When we asked them, they very kindly did their best privately to support us although they were not selling Atari computers.

They bought hardware and software for us through their private connections with the Atari Home Computer Division and they gave us some technical manuals. They even offered us a room in their office for our monthly meeting. We helped them in return by giving information and advice about the Japanese market. It might have been the happiest period of our club. (I cannot thank Mr. Hight, the president of that former office, enough for his help.)

Last year the new Atari company closed that Tokyo office and opened a new one (Mr. Hight and other people were fired). When we asked the new office to continue to support us, they refused, saying, "We will never sell Atari computers in Japan." I believe that a trusted company should support, or at least try to support, all the users of their products, wherever those users are. This was just what Mr. Hight tried to do.

Now we are forsaken again (forever?). We lost our last connection. We are holding our club meetings at a small cafe in Tokyo. I am really anxious about the future of our club.

Despite our tough situation I believe our club has been very active, as might be seen in some of my recent uploads to your club BBS and in the accompanying articles. Most of our members are as creative and enthusiastic as you S.L.C.C. members. We developed lots of software and hardware by ourselves, including printer cables and 64K RAM boards. I really hope to be a bridge between our two clubs, although our club seems to be fading away now.

I know that there are many almost-forsaken Atari users around the world. Mr. Iori Fujita recently reported to me about those in Peru. But there seems to be no other country like Japan, where there are many Atari enthusiasts and no dealers. Once we were even interviewed by a Japanese computer magazine because the editors were so interested in Atari.

If some of you are going to Japan with your Atari, please do not forget to call our president Mr. Masayuki Hata at (03) 653-1258 (Tokyo). You will not be alone in Japan cursing the poor support of the Atari company.

(Some more information about our club can be found in *Antic magazine*, international edition.)

DOS 2.0S Modification

Makoto Nagata

Fuji Atari Computer Users. Group, Japan

This program modifies ATARI DOS 2.0S so that it can use non-resident handlers (such as R:) more safely.

1. How to modify your DOS 2.0S

Using ATARI BASIC, boot up a disk containing DOS 2.0S and RUN "D:DOSMOD.UTL". When the "READY" prompt appears, type "DOS [RETURN]" and write DOS files to your disk, using the H option. (Cf. DOS 2.0S Reference Manual pp.35-36)

2. Theory of operation

This modification allows the DOS to handle another autorun file named "DEV.SYS". This file is like the usual "AUTORUN.SYS" file, but the process of loading and initializing it has the following differences:

(1) BOOT-UP: The DOS first looks for a file named "DEV.SYS" in drive #1, and if it exists, that file is loaded into RAM and initialized just like an "AUTORUN.SYS" file. Then the DOS looks for an "AUTORUN.SYS" file as usual. (If there is no DEV.SYS file in drive #1, then it works just like the original DOS 2.0S.)

(2) SYSTEM-RESET: After re-initializing various registers and tables, the DOS examines the value at \$1853 (called LDFLAG), and if its value is non-zero, it JSRs to the address contained in \$1854,\$1855 (called INIVEC; LSB first), then returns to the cartridge. If the value at \$1853 is zero, it re-loads the "DEV.SYS" file and initializes it before going back to the cartridge.

(3) RETURNING TO THE CARTRIDGE FROM DUP.SYS: Before DUP.SYS returns to the cartridge, using option B, it re-loads the "DEV.SYS" file in drive #1 and initializes it.

3. How to write a non-resident device handler

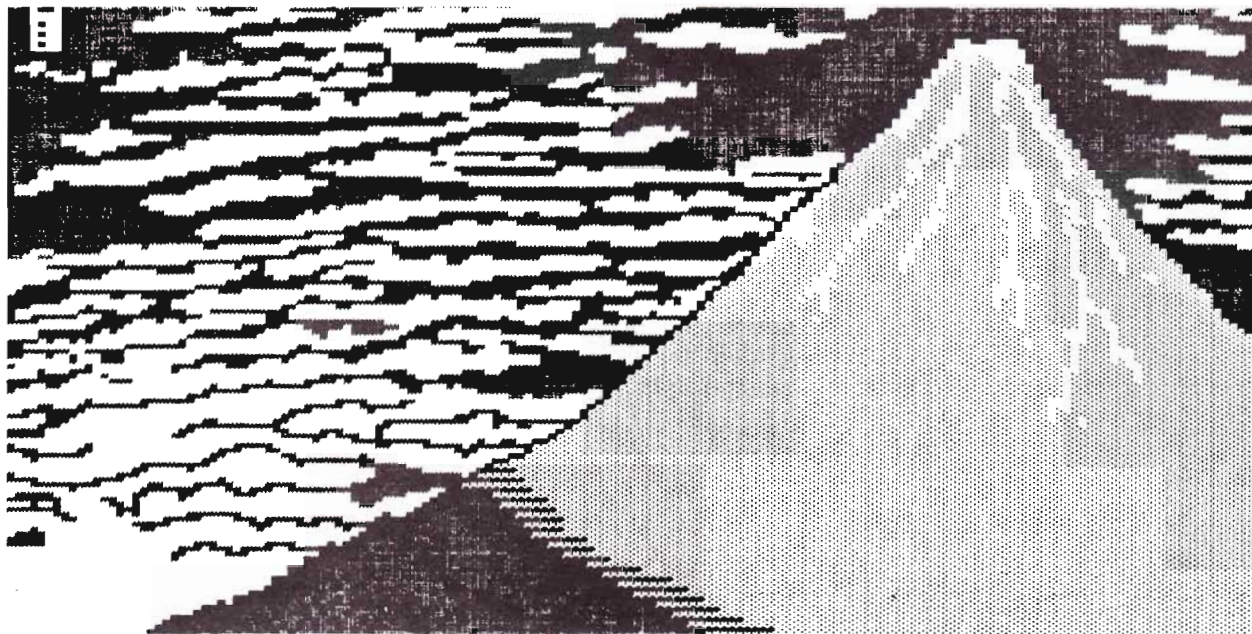
Due to a bug in the OS, the HATABS is re-written at system-reset. So usually you can not press the RESET button while you are using a non-resident handler (such as R:). As explained above, using this modified DOS, your non-resident handler works just like resident handlers if you add an initialization routine which registers its name to the HATABS and sets values of registers \$1853 to \$1855 appropriately. Usually, \$1853 should contain some non-zero value, and \$1854, \$1855 should contain the initialization address of the handler.

REMARKS:

1. So far no incompatibility has been found with the original DOS 2.0S. If you find any, please let the author know.

2. The modification is done by using some redundant text regions in the original DOS 2.0S so some DUP.SYS messages are shortened, but still easily recognizable (such as: "NEED MEM.SAV TO LOAD..." is changed to "NEED MEM.SAV").

3. Before going back to the cartridge from DUP.SYS, be sure that the disk used to boot-up is in drive #1. If the disk in drive #1 contains no (or another) "DEV.SYS" file, the handler in RAM might be deleted.



Player-Missile Driver

VER.1.0 (7/28/82)

BY MAKOTO NAGATA
FUJI ATARI COMPUTER USERS GROUP, JAPAN

1. Introduction

This program was developed for beginning members of Fuji Atari Computer Users' Group, Japan, to show what can be done with Atari's unique Player-Missile graphics.

The program is designed to let beginners experience the fascinating world of PM graphics using readable commands, with no PEEKs and POKEs, as safely as possible. It allows the user to hit the [BREAK] or [SYSTEM RESET] keys at any time, or change the background graphics mode, without affecting the PM graphics, etc.

This article does not contain any explanation of PM graphics. If you are not familiar with it, consult any of the many articles about it in various Atari-related magazines.

2. Device "G:" - OPENing and CLOSEing

This program is actually a device named "G:". It analyzes a record (ending with an EOL) sent to it and goes to work. The first thing you do to use this device is OPEN it using the following command:

OPEN#channel,resolution,0,"G:

where:

channel = IOCB number
resolution = 0 for double line resolution
 = 1 for single line resolution

For example, if you want to use double line (vertical-

ly coarser) resolution, you should type OPEN #1,0,0,"G:". You cannot change the resolution until you CLOSE the device using a CLOSE #channel command.

The OPENing process does the following:

- 1) resets various registers;
- 2) calculates the necessary amount of memory for PM graphics of the desired resolution and relocates the BASIC workspace upward accordingly;
- 3) changes the vertical blank interrupt routine so that pressing the BREAK key won't destroy the PM graphics.

The CLOSEing process is just the reverse of the OPENing process; i.e., relocating the BASIC workspace downward to the original address, resetting the VBI, etc. (All the PM graphics are erased.)

Now, once you open the "G:" device, you can send various commands to it. The commands are explained in the next section.

REMARK: To see how the BASIC workspace is relocated, PEEK the content of memory locations 128 and 129 before and after the OPENing.

3. Commands

In all of the following, PM# is a number from 0 to 7, which represents a Player-Missile number as follows:

0-3: Player 0 to Player 3

4-7: Missile 0 to Missile 3

(fifth Player option is not supported.)

1) LOADING THE SHAPE DATA INTO THE PLAYER/MISSILE REGION

?#channel;"OBJ",PM#,address,length

where:

PM# = Player/Missile number

address = starting address of the (binary) shape data

length = number of data bytes to be sent.

For example, if you want Player 0 to be of the following shape:

10000001 (decimal 129)

01000010 (decimal 66)

00100100 (decimal 36)

11111111 (decimal 255)

00011000 (decimal 24)

00011000 (decimal 24)

first define a string (for example, D\$) of length 6, using a DIM D\$(6), and write the data into D\$ using a program as follows:

```
100 FOR I=1 to 6:READ X:D$(I,I)=CHR$(X):
```

```
NEXT I
```

```
110 DATA 129,66,36,255,24,24
```

Then (after OPEN #1,0,0,"G:") type:

?#1;"OBJ",O,ADR(D\$),6

WARNING 1: You may not be able to see the shape of a Player or Missile until you have set up its position and color, as explained below.

WARNING 2: For Missiles (PM= 4-7), only the least 2 bits of the data is loaded into the Missile region.

2) CHANGING POSITION OF A PLAYER OR MISSILE

Current horizontal and vertical position data are always stored in internal registers, pointing to the upper-left corner of the Player/Missile shape.

To change the position of a Player or Missile, use:

?#channel;"POS",PM#,x,y

where:

x = x-coordinate (0-255)

y = y-coordinate (0-127 for double line resolution, and 0-255 for single line resolution).

These coordinates correspond to the value to be stored in HPOSP# and the vertical offset.

WARNING: If x or y is too small or too large, then that Player or Missile may be positioned beyond the screen edges, so it may not be seen.

3) CHANGING COLORS

To change the color of a Player, type:

?#channel;"COLOR",PM#,color,luminance

This is just like the SETCOLOR command in BASIC.

4) Changing the size of a Player

To change the size of a Player, type:

?#channel;"SIZE",PM#,size

where "size" is either 1, 2, or 4. (1 is the smallest, and 4 is the largest; a 4 means that each pixel is 4 times wider horizontally than the smallest.)

WARNING: This "SIZE" command does not affect the vertical length.

5) PRIORITY CONTROL

To change the priority, type:

?#channel;"PRIOR",data

where "data" is the value to be POKEd into the PRIOR register. (Cf. Atari Hardware Manual).

6) DMA CONTROL

To let all the Players or the Missiles disappear or re-appear without changing their position or color data, type:

?#channel;"DMA",data

where the value for "data" is:

0: to let all the Players and Missiles disappear

1: to let all the Missiles appear and all the Players disappear

2: to let all the Players appear and all the Missiles disappear

3: to let all the Players and Missiles appear

7) CLEAR THE SHAPE OF A PLAYER

To clear the shape of a Player, type:

?#channel;"CLR",PM#

8) LOCATE A PLAYER OR MISSILE

The vertical and horizontal positions are kept in internal registers. To get these data, type:

?#channel;"LOC PM#,var1,var2"

where "var1" and "var2" are the names of BASIC arithmetic variables. The horizontal position data is stored into the variable named "var1", and the vertical position data is stored into the variable "var2". For example, after the command ?#1;"LOC O,XP,YP", the horizontal and vertical position data are in variables XP and YP.

WARNING: The device cannot generate data area for new variables. So these variables must be used at least once somewhere in the program as usual variables. (To make sure, use nonsense commands like X=X etc.)

9) READ COLLISION REGISTERS

You can read collision register values using a com-

mand like:

?#channel;"HIT PM#,var1,var2"

where "var1", "var2" are arithmetic variable names as above. The Player-Playfield collision data is returned to the variable named "var1" and the Player-Player collision data is returned to the variable named "var2". (Cf. Atari Hardware Manual).

10) MULTI-COLORED PLAYER OPTION

To toggle the multi-colored Player option, type:

?#channel;"MULT",data

where "data" is 1 to enable multi-colored Player option, and "data" is 0 to disable it. (Cf. Atari Hardware Manual).

11) RESET VERTICAL POSITION REGISTER

The "OBJ" command sends the shape data to the memory region pointed to by the current value in the internal vertical position register so that you can change the shape where ever the current position is; i.e., the new shape data overwrites the old shape data.

Sometimes it is necessary to change the data in the vertical position register temporarily. To do this, type:

?#channel;"RESV",PM#,newdata

(The current value in the vertical position register can be read using LOC command.)

4. REMARKS

1) The program's command-analysis routine can read the values of BASIC variables directly from memory. So if you are not using arithmetic expressions, you may write that variable name inside the quotation marks; i.e., the command

?#1;"POS",O,X,Y

is equivalent to the following:

?#1;"POS O,X,Y"

(where "X" and "Y" are variable names). But you can't write ?#1;"POS O,X*2,Y" instead of ?#1;"POS",O,X*2,Y.

2) This program: is located at \$1CFC-\$239F. The actual PMBASE value is stored at \$1D96. No memory locations usually kept for user application are used by this program, including \$CC-\$D6, and \$0600-\$06FF.

Utility Package

Makoto Nagata

*Fuji Atari Computer Users' Group
Japan*

This package consists of general purpose relocatable machine-language subroutines for BASIC users.

This file is "LIST"ed, so use "ENTER" command to merge the utility package.

1. General Description

This package consists of the following:

1. Hex-decimal conversions
2. Binary-decimal conversions
3. String Search
4. Block Move
5. Disk Access
6. CIO Access

7. 1 byte Search/Replace

8. Basic Relocator*

(The Basic relocater may not work on newer systems.)

The program resides at lines greater than 30000 and there is a single initialization point at 30000. To initialize all the functions above, type "GOSUB 30000" (Actually, the initialization process consists of defining strings and storing machine-language programs in them. So the initialization must be executed once and only once to avoid a re-dimensioning error.)

2. REMARKS

1. This package originally consisted of 10 separately LISTed files. If you want to save memory you can decompose the package as follows:

Suppose that you want to separate SEARCH function.

1. First, ENTER the package
2. Delete all the lines between 30001 and 32766 not relating to SEARCH\$. (Do not delete line 30000 and 32767.)

3. Type: LIST "D:filename",30000,32767

After separating all the functions in the package this way you can merge as many functions as you like without losing the initialization point.

2. The "*" in the description below represents any arithmetic variable. The content of that variable may be altered after the machine language execution.

DECIMAL-HEXADECIMAL CONVERSIONS (WORDDEC and DECWORD)

These programs are decimal-hexadecimal conversion programs. The WORDDEC Program converts a hexadecimal number into a decimal number and the DECWORD program converts a decimal number into a hexadecimal number.

1. FORMAT

Before using these subroutines, you should define a full string of length 4 (for example by the command DIM WORD\$(4):WORD\$(4)=" ") for hexadecimal numbers. Then after the initialization GOSUB 30000, type as follows:

- (1) Hex to decimal conversion: WORDDEC\$

dec = USR(ADR(WORDDEC\$),addr)

where: addr : the address of the string containing the hexadecimal number (0000-FFFF)
dec : corresponding decimal number (0-65535)

- (2) Decimal to Hex conversion: DECWORD\$

*** = USR(ADR(DECWORD\$),dec,addr)**

where: dec : the decimal number (0-65535)
addr : the address of the string for the corresponding hexadecimal number (0000-FFFF)

REMARK: To use the WORDDEC subroutine, the hexadecimal number must exactly be of 4 digits.

2. SAMPLE PROGRAM

After ENTERing the package type in and run the following program. This is a simple hexadecimal listing program.

```
10 DIM WORD$(4),WORD2$(4):WORD$=" "
20 GOSUB 30000
30 ? "ENTER STARTING ADDRESS IN HEX ";:INPUT WORD2$
40 WORD$="0000": WORD$(5-LEN[WORD2$])=WORD2$
50 STA=USR(ADR(WORDDEC$),ADR(WORD$))
60 I=STA
70 X=USR(ADR(DECWORD$),ADR(WORD$),I):?WORD$;" ";
```

```
80 D=PEEK(I)
90 X=USR(ADR(DECWORD$),ADR(WORD$),D):?WORD$(3)
100 I=I+1:GOTO 70
```

BINARY-DECIMAL CONVERSIONS (BINDEC AND DECBIN)

These programs are binary-decimal conversion programs. The BINDEC program converts a binary number to a decimal number and the DECBIN program converts a decimal number to a binary number.

1. FORMAT

Before using these subroutines you should define a full string of length 8 for binary numbers. Then after the initialization GOSUB 30000 type as follows:

- (1) Binary to Decimal: BINDEC\$

dec = USR(ADR(BINDEC\$),addr)

where: addr : address of the string containing the binary number
dec : corresponding decimal number

- (2) Decimal to Binary: DECBIN\$

*** = USR(ADR(DECBIN\$),dec,addr)**

where: dec : decimal number
addr : address of the string containing the corresponding binary number

REMARK: To use the BINDEC program the binary number should exactly be of 8 digits.

2. SAMPLE PROGRAM

After ENTERing the package type in and run the following program. This is a simple binary to decimal conversion program.

```
10 DIM B$(8),B2$(8):B$(8)=" "
20 GOSUB 30000
30 ?"BINARY NUMBER";:INPUT B2$
40 B$="00000000":B$(9-LEN[B2$])=B2$
50 D=USR(ADR(BINDEC$),ADR(B$))
60 ?"BINARY ";B$;" IS ";D;" IN DECIMAL."
70 GOTO 30
```

3. MODIFYING THE DECBIN PROGRAM

The machine language part of the DECBIN program contains a "0" character and a "1" character. These characters determine the form of the returned binary number. If the character "0" is replaced by "." and the character "1" is replaced by "*", then the binary representation corresponding to decimal 13 (usually "00001101") will be changed to "...***".

STRING SEARCH (SEARCH\$)

This program searches for a smaller string in a larger one.

1. FORMAT

Suppose that we want to search a string A\$ in a string B\$. Then (after the initialization GOSUB 30000) type as follows:

X =USR(ADR(SEARCH\$),ADR(A\$),LEN(A\$),ADR(B\$),LEN(B\$))

X is the starting position of the first occurrence of A\$ in B\$; i.e., B\$(X,X+LEN(A\$)-1) is just equal to A\$. If A\$ is not found in B\$ (or the length of B\$ is smaller than that of A\$), then 0 is returned to X.

2. SAMPLE PROGRAM

After ENTERing the package, type in and run the following program. The Program replaces all the "APPLE" in the entered string with "ATARI".

```
10 DIM A$(5),B$(128):A$="APPLE"
20 GOSUB 30000
30 INPUT B$
40 X=USR(ADR(SEARCH$),ADR(A$),5,ADR(B$),LEN(B$))
50 IF X=0 THEN 80
60 B$(X,X+4)="ATARI"
70 GOTO 40
80 ?B$
90 GOTO 30
```

BLOCK MOVE (BMOVES)

This program moves memory block.

1. FORMAT

*** =USR(ADR(BMOVES),addr1,length,addr2)**

where addr1 : starting address of the origin block
length : length of the origin block
addr2 : starting address of the destination block

REMARK: The origin and destination blocks may overlap.

2. SAMPLE PROGRAM (Cf. BASIC RELOCATER)

WARNING: This program may not work on newer systems.

After ENTERing the package, type in and run the following program. This defines a custom character font where the "space" character is replaced by the "underline" character.

```
10 GOSUB 30000
20 BLH=PEEK(129):MLH=PEEK(744)
30 CB=[INT(BLH/4)+1]*4:IF BLH>MLH+7 THEN 60
40 NBL=CB+4
50 X=USR(ADR(RELOC$),NBL*256)
60 RCB=57344
70 X=USR(ADR(BMOVE$),RCB,1024,CB*256)
80 POKE CB*256+7,255
90 POKE 756,CB
100 ?"CHARACTER BASE IS CHANGED TO ";CB
110 END
```

REMARKS:

lines 10-40 calculates new BASLO and character base
line 50 relocates BASIC workspace if necessary

line 60 RCB is the address of ROM character font
line 70 copies the ROM character font
line 80 modifies the character of the "space"
line 90 changes the CHBAS value

DISK CONTROLLER (DSKCTRL\$)

This subroutine enables you to access directly to the disk sectors (without DOS).

1. FORMAT

Before using this subroutine, you should define a sector buffer. This is a 128-byte RAM region and the content of this buffer is transferred to the specified sector ("write" operation) and vice versa ("read" operation). The safest way to define such a sector buffer is to define a full string of length 128 using commands like DIM B\$(128):B\$(128)=" ".

Then type as follows (after the initialization GOSUB 30000):

status =USR(ADR(DSKCTRL\$),rwf, bufadr,sector)

where : rwf : =4 for read operation
 :=8 for write operation
 bufadr : the address of the sector buffer
 sector : specified sector number (1 - 720)
 status : status of operation (= 1 if complete)

(For various values returned to the variable "status", see the ERROR CODE TABLE in ATARI BASIC REFERENCE MANUAL.)

2. SAMPLE PROGRAM

After ENTERing the package, type in and run the following program. This program displays the content of the specified sector in ATASCII form.

```
10 DIM B$(128):B$(128)=" "
20 GOSUB 30000
30 ? "SECTOR NUMBER";:INPUT S
40 X=USR(ADR(DSKCTRL$),4,ADR(B$),S)
50 IF X<>1 THEN ? "ERROR ";X;" AT ";S:GOTO 30
60 POKE 766,1
70 ? B$
80 POKE 766,0
90 GOTO 30
```

CIO CONTROLLER (CIOCTRL\$)

This subroutine enables you to access directly to the ATARI Central I/O system. (Cf. Atari Operating System Users' Manual).

1. FORMAT

status =USR(ADR(CIOCTRL\$),iocb,icom,icba,icbl,icax1,icax2)

(icax1, icax2 are optional)

where: iocb :IOCB number (0 - 7)
 icom :CIO command code
 icba :CIO buffer address
 icbl :CIO buffer length

```
icax1 :ICAXI
icax2 :ICAX2
status :ICSTA (= 1 if the operation is
         complete)
```

2. SAMPLE PROGRAMS

There are many applications of this subroutine. Here are some of them:

(1) READING BINARY DATA FROM A DOS FILE

```
OPEN #1,4,0,"D:filename"
X=USR(ADR(CIOCTRL$),1,7,BUFADR,BUFLEN)
CLOSE #1
```

The program above reads the binary data of length BUFLEN from a DOS file, and stores it in the RAM region starting at BUFADR.

(2) WRITING BINARY DATA TO A DOS FILE

```
OPEN #1,8,0,"D:filename"
X=USR(ADR(CIOCTRL$),1,11,BUFADR,BUFLEN)
CLOSE #1
```

The program above writes the binary data in the RAM region starting at BUFADR of length BUFLEN to a DOS file.

(3) READING BINARY DATA FROM CASSETTE (FASTER THAN USING GET)

```
OPEN #1,4,128,"C:
X=USR(ADR(CIOCTRL$),1,7,BUFADR,BUFLEN)
CLOSE #1
```

The program above reads binary data of length BUFLEN from cassette and stores it in the RAM region starting at BUFADR. This is faster than using GET command.

(4) WRITING BINARY DATA TO CASSETTE (FASTER THAN USING PUT)

```
OPEN #1,8,128,"C:
X#USR(ADR(CIOCTRL$),1,11,BUFADR,BUFLEN)
CLOSE #1
```

The program above writes the binary data in the RAM region starting at BUFADR of length BUFLEN to cassette. This is faster than using PUT command.

(5) DIRECT INPUT FROM THE EDITOR

After ENTERing "D:CIOCTRL.LST", type in and run the following program.

```
10 DIM A$(128):A$(128)=" ":GOSUB 30000
20 X=USR(ADR(CIOCTRL$),0,5,ADR(A$),128)
30 IF X<>1 THEN STOP
40 L=PEEK(840)
50 IF L=1 THEN 20
60 ?A$(1,L-1)
70 GOTO 20
```

This enables you to input a string from Editor without using BASIC's INPUT command. The "?" prompt does not appear.

SEARCH AND REPLACE (SEAREP\$)

This program is a general search-and-replace program: of 1 byte data. Since this program allows you to use data masks, you can use it as a "fill data" program.

1. FORMAT

There are three formats.

(1) * = USR(ADR(SEAREP\$),addr,length,datal, data2)

In this format, a machine language program corresponding to the following BASIC program is executed:

```
10 FOR I=addr TO addr+length-1
20 IF PEEK(I)=data1 then POKE I,data2
30 NEXT I
```

(2) * = USR(ADR(SEAREP\$),addr,length,datal, data2,mask1,mask2)

In this format, the search and replace operation is masked by the "mask" variables. If we denote the binary AND of two numbers by BAND[a,b], the binary OR of two numbers by BOR[a,b], and the binary EOR of two numbers by BEOR[a,b], then the operation is expressed in a BASIC-like form as follows:

```
10 FOR I=addr to addr+length-1
20 IF BAND[PEEK(I),mask1]<>data1 THEN 50
30 X=PEEK(I):Y=BEOR[mask2,255]:Z=BAND[X,Y]
40 W=BAND[data2,mask2]:POKE I,BOR[Z,W]
50 NEXT I
```

(If you do not know about binary AND, OR or EOR operations, see any book about microprocessors.)

(3) * = USR(ADR(SEAREP\$),addr,length,datal, data2,mask)

This format is the shortened form for * = USR(ADR(SEAREP\$),addr,length,datal,data2, mask,mask). See (2).

2. SAMPLE PROGRAMS

After ENTERing the package, type in and run the following programs:

(1) This replaces all the "A" in the entered string with "a":

```
10 DIM A$(128)
20 GOSUB 30000
30 INPUT A$
40
X=USR(ADR(SEAREP$),ADR(A$),LEN(A$),ASC("A"),ASC("a"))
50 ?A$:GOTO 30
```

(2) This inverts all the normal characters in the entered string:

```

10 DIM A$(128)
20 GOSUB 30000
30 INPUT A$
40 X=USR(ADR(SEAREP$),ADR(A$),LEN(A$),0,128,128)
50 ? A$:GOTO 30

```

(3) (FILL DATA)

This program fills the whole screen with "*":

```

10 GRAPHICS 0
20 MEM=PEEK(88)+PEEK(89)*256
30 X=USR(ADR(SEAREP$),MEM,40*24,0,ASC("*"),0,255)

```

BASIC RELOCATER (RELOC\$)

This program relocates the BASIC workspace.

1. BASIC WORKSPACE

The BASIC workspace is the RAM region which contains all the necessary information for BASIC to run; i.e., program statements, names of variables, etc.

2. POINTERS

There is a pointer which contains the lower boundary of the BASIC workspace:

BASLO \$80,\$81 [128,129] (LSB,MSB)

On the other hand, there is a pointer which contains the upper boundary of the RAM region used by the Operating System:

MEMLO \$02E7,\$02E8 [743,744]

This program relocates the BASIC workspace to the region starting at the specified address. This changes the BASLO pointer value to the specified one, but the MEMLO pointer remains unchanged. The BASLO pointer and the MEMLO pointer usually coincide.

```

+---+*****+-----
$0000 workspace

```

MEMLO
BASLO

```

+---+-----+*****+---
$0000 workspace

```

MEMLO
BASLO

The RAM region between MEMLO and BASLO is then user-free, and you can store your custom character font or non-relocatable machine language programs there.

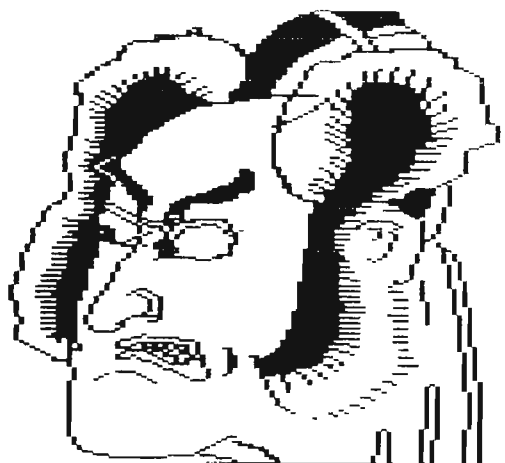
3. FORMAT

After ENTERing the package and the initialization GOSUB 30000, type:

X = USR(ADR(RELOC\$),newbaslo)

where: newbaslo : new value for the BASLO pointer

市川
図十郎



4. REMARKS:

- (1) This program does not interfere with the BASIC program execution so you can write a BASIC program which runs relocating itself.
- (2) You can relocate the BASIC workspace downward but the BASLO pointer value must always be equal or greater than the MEMLO pointer value.
- (3) Once the BASIC workspace is relocated, it is not re-relocated by SYSTEM RESET so the user-free space between MEMLO and BASLO remains safe until power is turned off.
- (4) This program may not work on newer systems. If this program does not work on your system, please let the author know by posting a message on the Key System BBS.



ATR-8000 Program Library



ATR - 8000 SIG

by Bill George

Our ATR-8000 Special Interest Group has been meeting for about 8 months now. We have been concentrating mainly on the ATR's CP/M capabilities. We have issued seven CP/M public domain floppy-of-the-months. The source of our CP/M programs is 60 megabytes of public domain programs on 300, 250k eight inch floppies. I estimate that there are roughly 10,000 programs on those disks. The cream of these programs are placed in our floppy-of-the-month.

Recently several of our SIG members have obtained and installed Co-Power 88 boards in their ATR-8000s. So, our SIG is exploring 16 bit MS-Dos and CP/M 86 operating systems. We have been collecting public domain software that will run on MS-Dos and will soon issue a public domain MS-Dos floppy. We will do likewise with CP/M 86 programs. For this special edition I am presenting you with a condensed catalog listing of all programs from our first seven CP/M floppies. You will be interested to know that we have chosen the Kaypro single sided, double density format for our floppy. This format was chosen because almost any 5" CP/M computer can read the Kaypro format and because it holds a bit more data than the other formats (191k).

The price of these floppies is \$9 each. Mail your checks and which floppy number you want, to the San Leandro Computer Club, CP/M Floppy of the Month, PO Box 1525, San Leandro, California, 94577-0152. Allow about a month and a half for delivery.

ATR-8000 CP/M Floppy of the Month Disks 1 - 7

Nov 84 - May 85

Program Name	Disk No.	Size in K
ALGOLM .HLP	07	13K
ALLCHARS.PIC	07	4K
AMAZE .BAS	04	4K
AMAZE .INT	04	3K
ANYCODE .ASM	06	3K
ANYCODE .DOC	06	20K
ASM .HLP	03	4K
ASM2 .HLP	03	4K
BIOPRINT.BAS	04	7K
BIOPRINT.INT	04	6K
C .HLP	03	17K
C8411DOC.	02	3K
KCBASIC2.HLP	07	22K

CDIR .COM	06	8K	MASM .HLP	07	8K
CDIR .DOC	06	2K	MEMLINK.COM	02	1K
CDIR/BD .C	06	8K	MLOAD23.OBJ	06	3K
COMBINE .ASM	07	7K	NOTE .COM	02	20K
COMBINE .COM	07	1K	NOTE .DOC	02	2K
COMPARE .ASM	07	4K	NSWP205 .COM	01	11K
COMPARE .COM	07	2K	NSWP205 .DC1	01	2K
CPM .HLP	03	31K	NSWP205 .DOC	01	28K
CPM2 .HLP	03	37K	NSWP207 .COM	02	12K
CPMADR .C	06	3K	NSWP207 .DOC	02	32K
CPMADR .COM	06	6K	NULU .COM	03	14K
D .COM	03	3K	NULU .DOC	03	40K
D .HLP	03	5K	NULU .DOC	06	57K
DISPLAY .COM	03	3K	NULU11 .COM	06	15K
DISPLAY .HLP	03	3K	NULU11 .NOT	06	1K
DU-V86 .COM	07	8K	NULU11F1.ASM	06	2K
DU-V86 .DOC	07	13K	NULUTERM.ASM	06	3K
EBASIC .COM	04	12K	OTHELLO .BAS	05	16K
EBASIC .HLP	04	12K	OTHELLO .DOC	05	5K
EBASPRG .HLP	04	3K	OTHELLO .INT	05	11K
ELSE .ASM	02	6K	PAUSE .COM	06	5K
ELSE .COM	02	1K	PAUSE/BD.C	06	2K
ENDIF .ASM	02	1K	PAUSWAIT.ASM	05	8K
ENDIF .COM	02	1K	PAUSWAIT.COM	05	1K
ERUN .COM	04	12K	POET .BAS	04	4K
FINDBAD .COM	02	2K	POET .INT	04	2K
FINDBAD .DOC	02	3K	PRINT .COM	05	5K
FINDBD54.ASM	02	32K	PRINT .DOC	05	3K
FINDV40 .ASM	05	15K	QUICKKEY.COM	03	2K
FINDV40 .COM	05	2K	QUICKKEY.HLP	03	2K
FINDV40 .DOC	05	3K	QUOTES .PRN	06	13K
FMAP-PD .COM	05	2K	READ .ME	04	1K
FMAP-PD .DOC	05	2K	RESOURCE.COM	01	10K
FMAP-PD .HLP	05	2K	RESOURCE.DOC	01	26K
GOTO .ASM	02	6K	REZ80 .COM	01	7K
GOTO .COM	02	1K	REZ80 .DOC	01	1K
HELP .COM	03	2K	SECTRAN .C	06	1K
HELP .COM	05	5K	SECTRAN .COM	06	4K
HELP .COM	07	5K	SHOW .ASM	04	10K
HELP .HLP	03	7K	SHOW .COM	04	1K
HELP .HLP	04	2K	STORY .BAS	04	7K
HELP .HLP	05	2K	STORY .INT	04	5K
HELP .HLP	07	1K	SUPERSUB.COM	01	2K
HELP-PLN.COM	05	5K	SUPERSUB.DOC	01	4K
HELP-TRS.COM	05	5K	SUPERZAP.COM	01	6K
HELP-V20.ASM	04	39K	SUPERZAP.DOC	01	11K
HELP-V20.COM	04	4K	SURVEY3 .ASM	06	13K
HELP-V20.HLP	04	24K	SURVEY3 .COM	06	1K
HELP-V20.HLP	07	24K	SYNONYM .COM	02	2K
HELP-V21.ASM	05	40K	SYNONYM .DOC	02	1K
HELP-V21.COM	05	5K	SYNONYM3.ASM	05	12K
HELP-V22.COM	07	5K	SYNONYM3.COM	05	2K
IF .ASM	02	11K	SYNONYM3.DOC	05	7K
IF .COM	02	2K	TED .COM	02	17K
IF .DOC	02	9K	TED .DOC	02	23K
LANDER .BAS	04	5K	TRANSLAT.COM	01	3K
LANDER .INT	04	3K	TRANSLAT.DOC	01	1K
LISTT .COM	03	2K	UNERA11 .ASM	05	6K
LISTT .HLP	03	1K	UNERA11 .COM	05	2K
LOAN .BAS	04	3K	WUMPUS .BAS	05	9K
LOAN .INT	04	2K	WUMPUS .INT	05	6K
LRUN .COM	03	2K	YANC .COM	01	32K
LRUN .HLP	03	1K	Z80 .LIB	01	6K
MAC .HLP	07	8K	Z80ASM .COM	01	9K



by Lois Hansen

You will not mistake Atari Corp. for Atari Inc. were you to visit them. There are no hanging plants, thick carpets, fountains, expensive furniture or free coffee and donuts. What there is in the reception area are packing crates, slightly shabby furnishings, a front door that does not close properly and a bulletin board skewered haphazardly with pink while-you-were-outs. People hurry about without the collegiate good-vibing characteristic of old Atari and many other Silicon Valley campus companies. Atari Corp is rock bottom business.

Inside and upstairs it looks like they are preparing for a garage sale. All manner of computers and peripherals, from DEC's to IBM PCs to MacIntosh (and you can bet there are Commodores) are strewn around; some in service, others piled on the floor and on top of each other. Inside the cubicles no one seems paranoid about who the hell I am. Perhaps they have users in every day to try out the goods.

I sit down in a crowded cubicle kindly vacated for me by Richard, a software tester. We are still making smalltalk when an exterminator walks by, pumper in hand. After my comment to the effect that "you guys leave no rock unturned", Richard escapes to exterminate his bugs and I get down to serious testing of Dr. Logo on the ST. There is scarcely room on the table to operate the mouse so I use a notepad.

Richard and my host, John Feagans, left me to my own devices for hours. I cruised the computer, not reading any of the documentation left for my enjoyment. I had a fine time, and just when I was thinking I really should get going back to Oakland, John returned to hear my comments.

What is the Jackintosh like to use? You have to get used to the mouse and windows. It is easy to overshoot the mouse but just as easy to get rid of what you did not want. I did not get into any trouble that I could not get out of. Logo did not come right up, since I was using a disk version (it will end up resident in the ST). I had to figure out how to get it off the disk but it was very obvious.

What is the Logo like? Is it like the Atari Logo we know and love, from LCSI, nearly identical to Apple and IBM? No, it is not. It is Dr. Logo, if you ever saw that, written by Gary Kildall himself and released in August of 1983 to run under CP/M on the IBM PC. Most of the commands are the same as LCSI Logo. The main difference is that it comes up with a screen split sideways between graphics and text and it has a trace feature that allows you to watch the commands at the same time your drawing is executing. On the Jack you can cause all three screens to be visible at once, due to windowing. In the Edit Mode, the cursor control commands are Control F, B, N, etc. instead of using the arrow keys but Atari might change that. The use of color in Logo on the ST, despite the alleged 512 colors, is more like Apple and IBM than Atari 800. You

can not flash your background through even 127 colors and luminances, except perhaps, by including palette-change commands in your program. In hi-res you get no color, in med. res. you get 4 at any one time (although you can change which 4), and in low res. you get 16 at a time. You get similar numbers of colors in the pen (only one pen, although you can vary the brush width) and, of course, there is only the one turtle, a triangle. The list processing is similar to IBM, which is a great improvement over cartridge Atari Logo. For adult use, this is definitely the Logo to have on Atari.

Is this a better deal than Atari Logo on the 800 XL or XE? I hate to hedge but all questions like this require questions back: how much money have you got and what else do you want to do with the computer? For people who have not wanted to spend \$350 on a "kid's" computer and drive that they felt the adults would not use, here is a way to really compromise. You have to spend \$800, but you get the latest thing in "adult" software and hardware as well as Logo for your kids in a version that you will enjoy too. This Logo is not identical, but close enough, to that used in schools.

Are you going to be annoyed to have a ST computer you can not program? I mean, this computer does not have BASIC in it yet! Earlier machines are going to come with some applications and Logo but no BASIC. You have spent years, getting good at BASIC, Atari BASIC at that, and now they give you this poor man's paint program and wish you good luck. Why would an adult want to learn Logo?

Why did you want to learn BASIC? Because you wanted to gain access to this new thing; to learn a new way to organize your thoughts. In BASIC, the inner workings of the computer dictate that you must organize your thoughts in a linear fashion. In order to avoid chaos in BASIC you have to be very careful with your GOTOs and GOSUBs and remember what all your strings refer to. Logo came along on machines with larger memory allowing you to define small procedures with abandon and organizing them later into one or more superprocedures. Many people feel that this is more the way people, as opposed to machines, think. Or, shall we say, the way right-brained people think. Not that you must think non-linearly in Logo but you can. There are no line numbers in Logo and you do not have to reduce ideas to numbers so you can compare them mathematically. You go through Logo "words" and "lists", separating them out with "butfirsts" or, as one adult Logo student put it, "butt first". It is a totally different way to think. It is one of the most engaging occupations you will ever have on a computer (if you like to think about thinking). Elsewhere in this issue you can find a Logo Program of this sort that runs with Atari Logo. Try it and pick up a book about Logo such as Harold Abelson's *Apple Logo*, or David Thornburg's *Discovering Apple Logo*. It is not just a kids' language at all; it is a language for adults who like psychiatry, literature, movies or other stimuli that cause them to examine their own or others' thought processes.

Digital Alarm Clock

by STEVE KUNZE

Age 14

Adelaide Atari Club, Australia

To change the time loop the computer uses for each second you must change the variable "time" in line 4 of the program according to this formula:

A = Actual elapsed seconds

B = Elapsed seconds on computer

TIME = TIME/A × B

For the purposes of setting "TIME" correctly, I thoroughly recommend to leave the program running for at least 10-15 minutes so that your result will be an exact average of time.

Also note that when you enter your data for the clock, the program will not start working until you press START! This is a safety precaution. You have to take this into consideration when entering your data and give yourself about 30 seconds if you want the alarm, and 10 seconds if you do not.

Have fun . . .



AtariWriter

by Frank Pazel - JAGG

THE FORM LETTER

The good folks at the former Atari Inc. gave us what is probably the best word processor for any PC, bar none. It is easy to learn, handles just about any kind of request, is loaded with features and is absolutely dirt cheap. With the advent of the APX Printer Driver almost any printer works with it and the use of Atspell is a godsend to those less lexical. It is, in short, superb.

Now, these very same departed programmers had a few tricks up their collective sleeves when they put this ROM together. However, they were not entirely honest in reporting all the surprises packaged inside. Or perhaps they did tell all with some points lost between technical writer and manufacturing. In any event I am going to try to report what I have gathered from various sources and discovered through experimentation about the often demeaned Atariwriter.

Hidden inside the cartridge lies the latent ability to create form letters automatically with a mail merge, block copy text from one file to another and unleash a resident modem handler. First I will discuss the mail merge.

MAIL MERGE

There have been a couple of articles written on this feature in other newsletters but following their instructions led to frustration and no results. It turns out that a key point was always left out that I stumbled on almost by accident. Here is my report on how to get your Atariwriter to function as a bonafide business type form letter producer.

A form letter, for definition purposes, is a document which will contain personalized information. The bulk of the letter is the same for each addressee. You get these things in the mail every week telling you how you might just have won 10 billion dollars. Suppose you have the need to produce such a letter. Perhaps your club needs to send out a mailing which would look nice personalized or might attract more attention if the receiver's name appears inside the text. Using normal Atariwriter functions write the letter. However, wherever you want to personalize the text hold down the OPTION key and the insert key at the same time. An inverse ESCape character will be printed on the screen in that position. Later on, when you ask Atariwriter to PRINT your document the program and printer would normally halt and allow you to type in the missing information. This procedure is detailed on page 39 of the Atariwriter manual. As the manual says, "... leave blanks in a text file... and fill them in each time you print the file." This is exactly what we want to avoid. We want

to create a file which will automatically merge with our letter file and insert the missing information for us.

When you are sure your letter is exactly like you want, SAVE it. Make accurate notes of how many blank items you need to fill in and what the information needs to be. For example, the first three ESCape characters might represent name, street address, and town and state. Create a new file in the following manner.

1. Enter the Editor and delete the entire format line. Yes. You should now be looking at a blank blue page which used to have inverse letters with numbers after them. This is the absolutely crucial step in making this process work.

2. Using your notes about the empty blanks in your letter type in the missing information with a RETURN at the end of each piece of information. Use no blank line and continue typing in your repeating series of data. You are creating a "sort of" data base for the letter. Hopefully, you will design it so it can be used for other things. The addresses, for example, can be used to make up labels later on.

3. SAVE this file. I use the name MERGE but you use your favorite. Count the number of records in this file. A record is all of the information you need to print one letter. It might be something like name, street, and city-state. The number of records will equal the number of different documents you are going to print. You have ten different names with addresses in your MERGE file; you are going to print ten different personalized letters.

4. LOAD your letter into Atariwriter. Turn on your printer. Position your paper. Begin the PRINT series. At the prompt "PRINT WHOLE DOCUMENT?" answer Y.

5. At the prompt "NUMBER OF COPIES" type in the number should equal the number of records in your MERGE file. The maximum is 99.

6. At the prompt "MAKE ENTRY, PRESS RETURN", hold down the Control Key and press V (CTRLV) and you should hear the one key click. Now type in the specifications of your data file, e.g., D:MERGE. Make sure your file disk is in your active drive.

7. As soon as you press RETURN the printer should come to life and begin churning out your form letters. If you have specified right hand justification (J1) each letter will be printed with the personalized information properly justified. It's near magic!

Underground

PHONE LINES AND BLOCK MOVES

The Atariwriter ROM has, in addition to the ability to do a form of mail merge, the mechanism to transfer files via a modem.

In order to use this hidden modem handler you must boot up a copy of the original DOS 2.0 Master Diskette which came with your disk drive. Most people are unaware that stuck away on it is the RS232 information for handling modem operations. If you are using OSS software it is a file called "RS232.COM". Both communicators must be running through a 850 Interface Module. Using OPTION E rename it "AUTORUN.SYS" and you are in business. Once both ends of the telephone connections have contacted each other files are SAVED or LOADED from "R:filename". Try it and save some transfer time.

The final little trick that Atariwriter will do for you is a variation on its Duplicating Text feature. Rather than using the copy function to copy within a file you can use it just as well to copy from file to file. Use the Duplicating Text sequence described on page 37 of the instruction manual. This amounts to marking the beginning and ending of the text block you want to move with a CTRL-X. At this point, however, press ESC and return to the menu. Use SELECT to Create a new file or Load a file, depending on how you want to use your extracted block of text. If you load a file, enter the editor, position the cursor where you want to enter the saved block of text and press OPTION D. The saved file has been residing in the copy file buffer and can be used again and again. This is especially handy if you are preparing a report which uses a special format that must be repeated. To repeat copy just place the cursor where you want to replicate the saved block of text and press OPTION D. No need to remark and save it each time. If you save a new block of text with CTRL-X, that new text will, of course, replace the previously saved block.

The Atariwriter is truly a fine piece of software. Each day I wonder how I could get through my workweek without it. If you discover some new or undocumented features please send them along so we can publish them for the good of the order.

Thanks for some of the source material for this article goes to Clyde Pritchard of the Portland Atari Club and an article in the ACE of Syracuse newsletter.

Recursion in Atari Logo

By Lois Hansen

The main feature that convinces you that Atari Logo is not just for children is recursion. Unlike other languages which somewhat resemble logical thought, or some other kind of sequential thinking, Logo (and Lisp, from which Logo derives) forces you to do business through recursion.

What is it? Not any easy question, as they say. Let me tell you a story, combined from three in David Touretzky's excellent Lisp book: A poor fool had to find out how many slices of bread were in the loaf a nasty dragon had, so he could win the hand of the local princess. He approached the dragon fearfully and asked, "Please, sir, tell me how many slices of bread you have!" The dragon roared fearsomely and said, "I will not tell you how many I have, but I will give you the first slice." With that, the dragon fell asleep, feeling guilty that he was so mean, even for a dragon. He got to wondering how many slices were in his loaf. He dreamed of another dragon with the loaf minus one slice. That dragon dreamed of another dragon holding the loaf minus one more slice and that dragon dreamed of another dragon holding an ever shorter loaf, who dreamed of yet another dragon . . . None of them could figure out how many slices they had until the dragon whose loaf was so small, it did not exist. He had the empty loaf and he was ecstatic. He woke up the dragon before him and told that dragon that his loaf had one slice. That dragon woke up the one before him and told him he had one plus one slice. All of the dragons were happy as they were awakened in turn with the good news because they could now figure out how many slices they had and the prince could go back with his riddle solved and marry the princess.

Now this may seem like the long way around in terms of problem solving. And it does make Logo slow. But after you get used to it (it's taken me about two years) it kind of grows on you and seems reasonable. This is the method artificial intelligence uses to process words instead of numbers. You search by knocking the first word off a list and trying to match it. If there is no match, you go back and try the butfirst first (second), and on and on until there's nothing left (the empty list). Who is to say it is not the way our brains work?

Many feel recursion approaches the way brains work, except doing many lists of words at the same time would be closer to reality.

Let me give you a brief example of recursion in graphics:

```
To Manysquares :list.of.sizes
If :list.of.sizes = [] [stop]
Square first :list.of.sizes
Manysquares butfirst :list.of.sizes
End
```

```
To Square :size
Repeat 4 [Fd :size Rt 90]
End
```

Type both of these procedures in as you see them, taking care to observe all spaces and punctuation. I'd better define some terms. Dots (:) means the word following is an input and requires you, in this case, to supply a list of sizes of boxes you want. A list is more than one word surrounded by square brackets. [] means the empty list. To run this program you simply type Manysquares [20 30 40 50] and press return. Manysquares will call Square, which makes a square with sides of 20, and goes on to the next line, which "manysquares" the butfirst, or second number in list.of.sizes, and so on. When you get to the empty list, the program stops.

I hope this example was simple enough to give you an idea of how recursion works. You can use recursion on all kinds of sorting problems, from alphabetizing to making phone lists. Other classic problems are poetry generators and "doctor" programs (although the latter is painful in Atari Logo, which lacks property lists). The important part to remember is that the product is not the point; the process is. If you want a certain application to do a job for you, it can probably be written more easily and it will certainly run more quickly in Basic. But in using Basic you are following the computer's rules of thought and operations. If you have learned these rules well, they probably seem second nature. But they are nothing like the way the human mind processes information. If you want to use your computer to simulate human thought, try thinking of stating a problem using recursion and then see if it works in Logo.

The Legend

of Stargunner

by Alex Leavens

Your footsteps echo quietly down the hallway and you notice, as you always do, the worn green linoleum beneath your feet. The linoleum of the Planetary Defense Command. You allow yourself a smile at that because you are the first person from Earth—man or woman—to attain the rank of Stargunner.

You turn the corner, push open the door. The first of the operators notices you, makes the customary salute with its signaling tentacle and adds a gesture of greeting. "Good evening, Gunner."

...Stargunner. A person charged with guarding Yarthae, the hub of the galactic empire, the 25th Century Constantinople. If Yarthae were to be attacked, destroyed, the empire would be shattered. You are there to see that that does not happen. Ever. Once, more than 200 years ago, the outlaw Sphyzigi attempted such an attack. That attack was beaten back—barely...

You nod to the controller as you make your way to the landing bay shuttle. "Evening, Quoxatcl. Clean board?"

The signaling tentacle gives a half-shrug. "Is it ever anything else?"

"No", you reply, as you climb into the shuttle pod. With a whoosh! of compressed air you are shot out to your ship. You climb into the cockpit, Powering on the engines, sensors and guns, even as the hatch

lowers itself snugly over you. The fleet of Stargunner ships around you do the same. You press a series of buttons, see the uninterrupted banks of green lights telling you everything is functioning as it should.

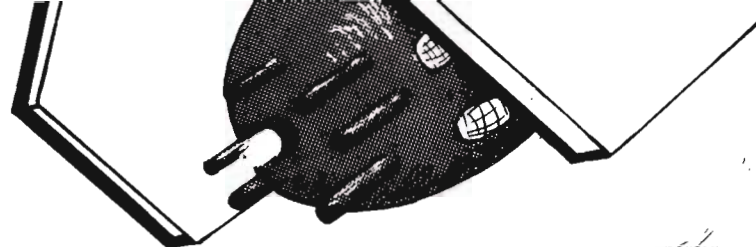
...No one was ever sure if the Sphyzigi were destroyed, or merely driven off. But the Galactic council decided to set up the fantastic (and fantastically costly) defenses anyway. No one was willing to take any chances... Yet 200 years is a long time—more than long enough for the threat of destruction and devastation to fade from memory, to become merely an oft-repeated tale, albeit a deadly and terrifying one...

You guide your ship to the takeoff shaft and the launch sequence takes over. You feel the mass accelerator push you back into the cushions as the magnetic rings flash by you, faster and faster until—space! Stars twinkling around you, the hills of Yarthae below, rolling peacefully under your ship. You handle the controls lightly, surely, the ship responding to your every touch. You press the fire button once, to test the P-laser guns. Immediately the night is shattered by a brilliant green pulse of pure energy, streaking away from your ship with a thunderclap, fading in the distance. You glance at the shield indicators—all 5, green. Then you hear the voice of control in your ears.

"How's your status, Gunner?"

"Looks good. Everything's functioning perfectly. Another smooth night, gang." Control responds with a chuckle. You flip a switch, and a radar display is projected in front of you. Glowing eerily, it seems to hang in midair, just inside the cockpit window. A rapidly moving light breaks the screen's stillness. "Wups. I got something here. See it, control?"

"Yeah, Gunner, we're on it. Probably just some kid joysticking around. We'll get his frequency and tell him to get out of here. Give him an escort, would you?"



"An overgrown nursemaid, that's what I am. Yeah, I'll get him. You better tell him to watch it, too. This high up is restricted air space.

"Roger, Gunner, we copy." Your ship streaks toward an interception point and you feel the hairs on the back of your neck start to rise. Just routine stuff, you think. Nothing to worry about. Control crackles in your ears.

"Gunner, craft is not answering any signals, repeat, craft is not answering any signals. This is not a drill, repeat, this is not a drill!" Instantly you reach over and flick your defense screens on, your attack computer from standby to armed.

"Control, this is Gunner 171-42," you say, your voice becoming precise and formal. "Request E.T.A. and code reference check on incoming craft."

"Roger, Gunner. Incoming craft has no bounce back frequency. Stand by." Suddenly, the blip on your radar screen splits into dozens of objects, each corkscrewing crazily off. "Gunner, this is control! They've Mirved! Emergency, Plan Red D! Gunner, they're coming into—"

A tremendous blast lights half the sky. Your ship rolls wildly, and you fight for control. "Control, this is Gunner 171-42! Control, what the hell happened! Control!" But there is no answer. Quickly you check the other frequencies, trying to raise your fellow Stargunners. Nothing. And then you realize: On the entire radar screen there is only one Stargunner ship. Yours.

And now you know what that craft is. A Sphyzigi ship. They have come back. And all that lies between them and total conquest of Yarthae are you and your ship. You, and you alone, will repel the invasion force. You MUST.

Playing STARGUNNER

Introduction:

Stargunner is your basic "Shoot everything that moves", stripped down, "let's fry a few neurons", raw adrenalin video game. It has no other raison d'être and the player should impart none to it, except, perhaps, the total impecuniousness of the author at the time he wrote it. To play, just plug a joystick (we recommend a sturdy one) into port 1, start the game from DOS (or however it comes packaged in the special edition), and have at it. Here are some notes to help you get more enjoyment from the game.

- 1: Movement is left, right, up and down. Button press fires the missile in whichever direction you're facing at the time. You must release the button, and re-press ("Hmm. Let's zee zose repressions, my dear.") it to fire again.
- 2: There are 6 levels to the game, and you can start at any one of them. By pressing SELECT in the attract screen you will cycle through the levels. Press START to start at the level currently displayed. Press the joystick button to start at the level that you last completed.
- 3: You can turn the music in the attract screen off by pressing OPTION.
- 4: There are 3 waves in every level. You must destroy 10 attackers, 1 at a time, 20 attackers, 2 at a time, and 30 attackers, 3 at a time. Occasionally, you'll only have to destroy a few aliens in one of the waves—kind of a gift for overtaxed thumbs. At the end of every wave, you're awarded a bonus. At the lower levels they don't add up to much, but they can be quite significant at the higher levels.
- 5: Bonus ship every 2000 points.

CARTOON MACHINE

By Cliff Schenkhuizen and Mark Perez

(Editor's note: Cliff and Mark are currently attending school at Moreau High as sophomores. They have been working together for a year now but this is their first published program. Mark is the creative artist and Cliff is the dedicated programmer of this truly unique "Dynamic Duo.")

Lately a new trend in features of Electronic Bulletin Boards has been catching on. BBS's around the country are now including, in addition to message bases and free software, on-line "cartoon" sequences of characters as an added attraction to the board. These cartoons are ATARI's control graphics characters animated with the resident screen editor functions available.

Few programs are available for Atari users that take advantage of the ability to show short cartoons over the phone lines. The ones that are available are usually just "bare-bones" text editors, that is, they only offer the entering of text and then the saving of it. Using these programs proved to be disappointing for us as well as others. They lack many helpful features and are somewhat awkward to use. It is this that led to the development of **CARTOON MACHINE**.

USING THE PROGRAM

The **CARTOON MACHINE** is set up in such a way that the user can easily access any part of the program with a minimum amount of keystrokes. The three console keys (OPTION, SELECT, and START) are the only keys needed. At the main menu the SELECT key moves the "cursor" (a little ball) down to each of the available options. To choose that particular option, one needs only to press the START key. The OPTION key serves to terminate all functions such as viewing a cartoon. OPTION will also abort any keyboard input and return the user to the main menu.

The first option, **VIEW CARTOON**, does exactly as it should: view the cartoon in memory. If no text has been entered into memory, a "NO FILE IN MEMORY" message will result. While viewing, however, OPTION will abort the function and return the user back to the menu. The program is set up in such a way that the cartoon is shown at a speed simulating a 300 baud

modem. Hitting SELECT while viewing acts as a toggle for 300/1200 baud viewing thus allowing the 1200 baud user to take advantage of the program too.

Saving and loading files have been simplified for the user. Both are nearly identical in operation. A directory of drive 1 can be obtained by pressing RETURN at the "Filename?" prompt. An append function has also been added. Typing "/A" after the filename will append the file to an existing cartoon in memory if LOADING or will append the cartoon in memory to a disk file if SAVEing.

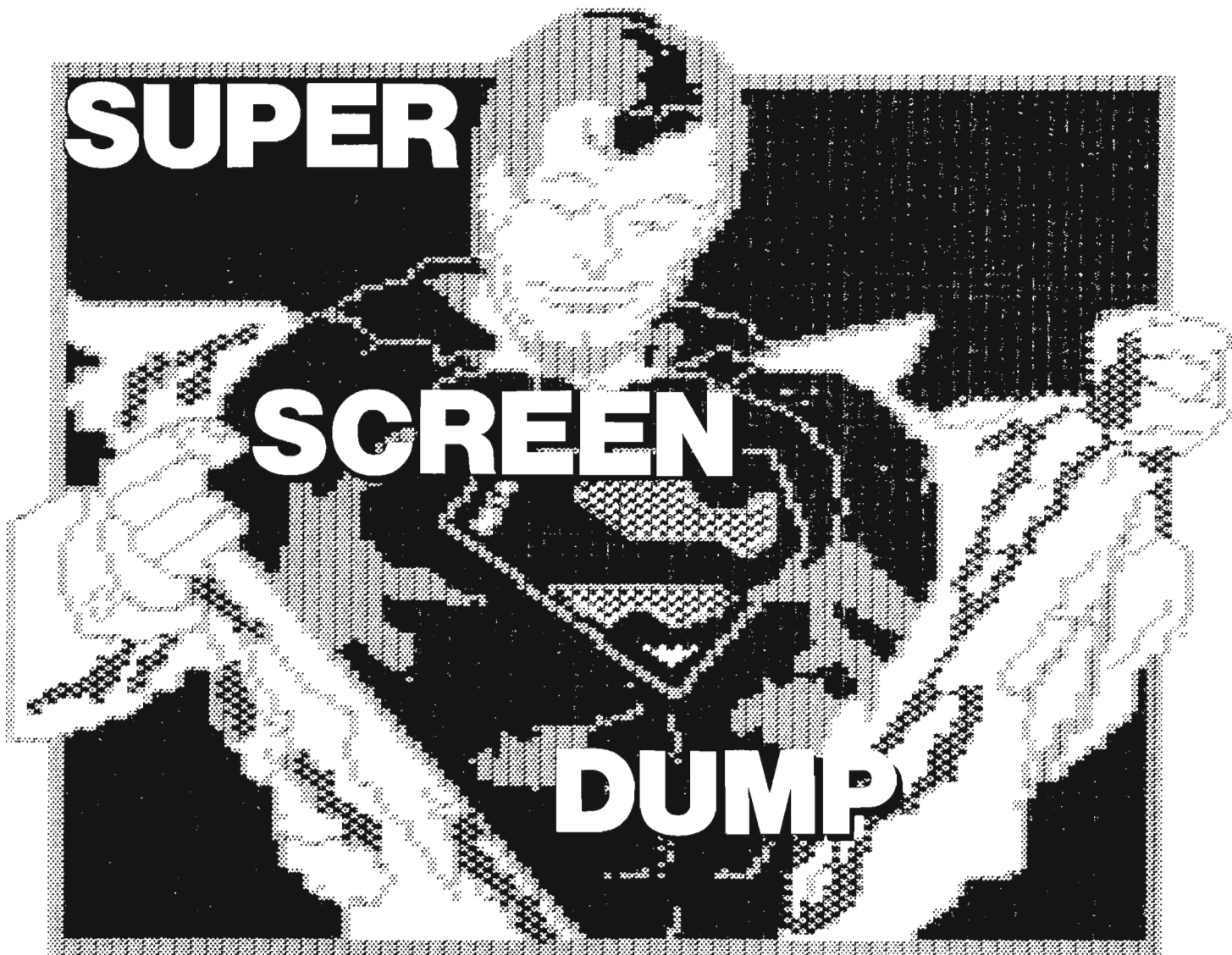
The next option, **ZERO MEMORY**, clears the program's 15000 byte buffer.

After the **ZERO MEMORY** function comes the **CREATE CARTOON** option. It allows the user to start a cartoon from scratch. Every keystroke is recorded except, of course, console keys and the BREAK key. Since no one is a perfect typist, we have added a helpful feature to the editing process. The START key now functions as a delete key. Pressing it will erase the last keystroke from the cartoon so that typographical errors will not show up on the finished copy. When the buffer contains only 100 free bytes, the computer will signal this by changing the color of the border to red. The user can either finish the cartoon and terminate cartoon entry with the OPTION key or he can save it and append the end to it with another file.

The last function is **APPEND CARTOON**. Not to be confused with disk appends, this allows the user to continue a cartoon in memory. For example, a file can be loaded and the user could add more text to it with the **APPEND** option.

Last on the menu is **HELP MENU**. This is not a function. By selecting the **HELP MENU** option the user is presented with a screen of helpful hints. Included on this screen are quick explanations of the console keys, explanations of the menu options and also tips for the novice user.

We hope that you find this program to be entertaining and useful. We would be glad to hear from any of you cartoonists out there who have designed or orchestrated their own personal cartoon. Also, if you have found anything that could make this a better program, please drop us a line in care of the newsletter. By doing this, we can make improvements to **CARTOON MACHINE**. Have fun and keep us informed!!!



by Ted Burger and Paul Gifford

This program is a result of becoming frustrated while printing pictures from the Koala Pad system. The program is written in Atari Basic with many machine language subroutines. Some of the ML routines are my own and some of them came from other authors. At the moment I can not remember the sources. If you see some of your code in here, thanks!

The program supports several printers: Epson, Gemini, NEC, and Prowriter. Modifying it for a different printer is very simple. You add the printer name to the list around line 100 and then add the first letter to the check in the same area. The lines between 800 and 900 are where the printer is set up. Add another "IF" statement with the proper graphics control characters to format your Printer.

OPERATION OF THE PROGRAM

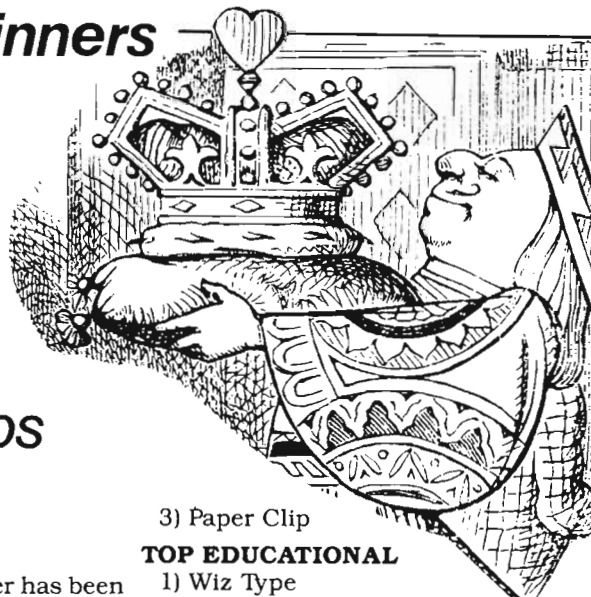
The program will ask you for the "picture disk". Koala format picture files must end with the ".PIC" extender and Micropainter format picture files must end with the ".MIC" extender. The screen will display a list of the files on the disk, up 38 files. You make your selection from the list. The program will then load the selected picture from the disk. The picture will appear in color at first and then switch to black and white. The bottom of the screen will switch to text to prompt you. Below the text block is another line of graphics. This shows the shade of gray assigned to each of the four color registers. To change the gray shade of one of the color registers, select which register, 1-4. The Prompt will change and ask which shade you want (black = 1 and white = 5). The numbers 2,3,4 will give you progressively lighter shades of gray. This will allow you to set up the picture to look right in black and white.

NOTE: If you set color 3 and 4 to the same shade, the text will not be visible but things will still work.

When you get the picture set up the way you want it, a tap of the "P" key will send it to the printer. The other choices you have here are: "M"enu will take you back to the list of the picture files and "V"iew will temporarily show you the entire graphics screen just in case something important is hidden by the text block.

This is a case of "what you see is what you get". The gray scales from the screen will be duplicated on the printer.

Have fun! (Ted Burger @ 71376,1263 and Paul Gifford.)



Tisby's Tops

by Tom Tisby

The Atari Home Computer has been around for a long time. Because of this many software programs have been written. And throughout the thousands that are produced, just a few really stand out. I am not talking about the great Pacman or Missile Command we have played so much that our cartridges have worn out. What I am talking about are the games that made Atari famous and unique; the programs that take our computers to the limit. So without further ado, may I Present for your inspection **THE GAMES AND PROGRAMS** in their respected order from a group of credible computer enthusiasts like you and me.

AND NOW...ON WITH THE AWARDS...

TOP 11 GAMES

- 1) Star Raiders
- 2) Alley Cat
- 3) Seven Cities of Gold
- 4) Whistlers Brother
- 5) Ghostbusters
- 6) Rescue on Fractalus
- 7) Ball Blazer
- 8) Archon
- 9) Agent U.S.A.
- 10) Microleague Baseball
- 11) Bruce Lee

TOP 11 ADVENTURES

- 1) Ali Baba & 40 Thieves
- 2) Return of Heracles
- 3) Zork, I, II, III
- 4) 7 Cities of Gold
- 5) Suspended
- 6) The Count
- 7) Mystery Fun House
- 8) Ultima III
- 9) Planetfall
- 10) Mask of the Sun
- 11) Transylvania

TOP WORD PROCESSORS

- 1) Atari Writer
- 2) Letter Perfect

3) Paper Clip

TOP EDUCATIONAL

- 1) Wiz Type
- 2) Mickey & the Great Outdoors
- 3) Agent U.S.A.

MOST INNOVATIVE GAMES

- 1) Ball Blazer
- 2) Rescue on Fractalus
- 3) Microleague Baseball
- 4) Star Raiders
- 5) Spy Vs. Spy
- 6) Pinball Construction Set
- 7) AE
- 8) Bruce Lee
- 9) Way-Out
- 10) Flight Simulator II
- 11) M.U.L.E.

TOP 11 UTILITIES

- 1) Printshop
- 2) Printwiz
- 3) Disk Wizard II
- 4) Lister Plus
- 5) Picture Plus
- 6) Colorprint
- 7) Diskkeeper
- 8) Humpty Dump
- 9) S.A.M.
- 10) Megafont
- 11) Syn-File

TOP BUSINESS

- 1) Syn-Calc
- 2) B-Graph
- 3) Visi-Calc

TOP BASIC COMPILERS

- 1) ABC
- 2) Datasoft
- 3) MMG

And so there you have it! The best of the best. If you do not have these programs, I suggest that you go and run (not walk!) to your nearest computer store and buy these great programs. You are missing a lot if you do not! And for all those who do own some or all, then all I can say to you is congratulations!!! You have made some excellent purchases. Now aren't you glad you did not buy that Pacman clone instead???

Sarge's Selections

by Sgt. Slaughter

Many times people will ask me, "Sgt. Slaughter, what is your favorite game?" I sometimes tell them MULE or the Pinball Construction Set or even Bruce Lee. Occasionally they will say, "Aw, Sarge, everybody likes those. You got any favorites that no one knows about?"

Indeed I do. Often, good games get good reviews and do well in sales. Other times, bad games get good reviews and do well in sales (but the buyers often regret it). It is rare that there can be a game or two out there that no one buys, no one has ever heard of or no one wants, but is still a very good game. This is the purpose of this review. I have a few games in my library that I play a lot even though they may not be a Bruce Lee or a MULE. Here is a list, with comments, on what I think are some of the greatest closet classics on the Atari.

RAINBOW WALKER: I happen to think this game is probably the most fun and involving Q*Bert clone on the Atari. The idea is to color grey and black holes in the rainbow without getting kicked off. It is a tough job but somebody has got to do it! The sound, graphics and action are second to none. There is a bonus round that is probably the best simulation of a bucking bronco ride I have ever seen! A definite plus for a game library.

COHEN'S TOWERS: This release is a sort-of Donkey Kong clone with a twist. You are a lowly courier who has to shuffle letters into various mail drops in a high-rise building. I play this game a lot because it is challenging and the graphics, animation and sound are top notch. In fact, with all the times I have Played this game, I have only completed the first building once which just goes to show you how much of a challenge this game has to offer.

WIZTYPE: Of all the typing educational programs that are on the Atari, this virtually unknown program is great! It offers you many different ways to hone typing skills, from games to actual paragraph typing, plus on-going charts of your progress and incredible graphics representing the many characters of the Wizard of Id. Not even the Atari Star Award winner, Typo Attack,

does as good a job of teaching typing skills as this program does. It is fantastic for anybody of any age because it can be friendly enough for a child or serious enough for an adult.

BOULDERDASH: Ok, Ok, I know this did pretty well but I feel that even if it was a best seller, it would not have done this excellent game justice. With all the terrible Dig Dug clones (Atari's Dig Dug being one of them) this one stands high and mighty above the rest. To win at this game you need something more than just fast reflexes (which seems to be a staple in most games today). You need to think ahead and plan. This game requires intelligence to play and I still play it many times when I find myself with some spare time.

THE RETURN OF HERACLES: As far as action adventures go this one is the tops! I have solved it 5 times already and I am going for a 6th! The mixture of players plus a well supported backing of mythic descriptions make this an epic blockbuster. Unlike Ultima 3, the characters are much more developed, much more interesting and less binding. For instance, with the right amount of gold you can help any character become a great hero, unlike Ultima 3, whose glitch makes it almost impossible to get to the stage where you can help your characters gain attributes. If you have never played Heracles or Ali Baba, I strongly suggest that you check it out and see what you are missing.

MIG ALLEY ACE: This is a fantastic flight combat simulation with multiple players and combat mode. When I found out that all of this was written in compiled BASIC, I could not believe my eyes! The speed and action can not be beat, even by the over-burdened and over-detailed Flight Simulator 2 by Sublogic. This is not to say that Flight Simulator 2 is a terrible program, but just that with the given amount of memory and play action Mig Alley Ace is a whole lot more fun. This is in comparison to the War game on Flight Simulator 2, not the actual simulation.

AMS2 (ADVANCED MUSIC SYSTEM 2): Few people realize what a great thing AMS2 is for Atari users and many shrug it off. I personally can not write music so the programming features are lost on me. What I do know is that not only are there hundreds upon hundreds of songs written for it (AMS2 has more user support for it than any other music program for any computer), all the songs are public domain and it is very inexpensive with the price of around \$20 as opposed to \$99 for lesser programs on other systems. If you know how to write music and have a good ear,

I would really strongly suggest that you take a look (and listen) to AMS2. You might very well be surprised.

SPELUNKER: Even when re-released from Broderbund, I do not think this game has ever gotten the respect it deserves. Truly an awesome game in both graphics and planning, I believe this is the game that SHOULD have replaced miner 2049er. It is a lot more related to mining than the latter is. This game can keep you going for a long time and the novelty still has not worn off for me.

Oh, yea, before I go I would like a couple of shots at some definitely over-rated games too. . . .

LODE RUNNER: Innovative but BORING!

POLE POSITION: If an actual car drove like that, this game would be a great argument to take the bus!

DEFENDER: Sure, it was great for about 1/2 an hour. But after you are stuck on the 99th level for about 15 minutes, you sort of get a little bored. . . .

CHOPLIFTER: Good for the times but got quickly outmoded in other games and in its own challenge.

ULTIMA 3: Yea, it is an epic. But all that work for what???

THE QUEST: Well, remember how I did not like it the first time I reviewed it? Well, right now, I like it even less!!! I still have not booted it up after I played it the first time. I did see a positive review for it somewhere, so this is in response to that unwary reviewer (wherever you are!)

Q*BERT: This is a brainless, artless copy of the arcade version without any of the inherent charms or action of the first one. You would think that they would fix the fact that you have to go diagonal to move. . . nope. This one is a real turkey. But because of the arcade popularity, it stayed around. . . .

Anyway, I would just like to leave you off with a little suggestion on how to pick the winners and losers of the gaming world. Try to see how the game/software is. If you can not see it and still have doubts, don't buy it. Another good idea is not to be the "first kid on the block" with something new. Very often the game you get for \$35 today will be in the \$4.95 bargain box tomorrow. Just be aware and be smart.

Until next time.

Goodbye and Good Gaming!

At Ease,
Sgt. Slaughter



Text Screen Dump

by Tom Reichard - JACG

Several micros have a built-in screen dump that allow a screen full of data to be dumped to the printer with a single keypress. The following is a simple BASIC program that essentially accomplishes the same thing.

As a stand alone program one simply RUNs it then types to the screen using all of Atari's great screen editing features. When you want a screen dumped to your printer simply press START. To continue on to another screen just press SHIFT and CLEAR (or CONTROL and CLEAR) then resume.

You need to be careful in using control characters as some of them affect printer actions. However, if you learn their functions for your printer, adding them to your text can give you some additional interesting features. For example a control N at the beginning of any line will cause that line (but not the next one) to be written in expanded type for many printers.

SPECIAL NOTE: In lines 50, 130 AND 190 the "!!!" are supposed to be ESC DOWN followed by ESC UP arrow which cannot be printed using this word processor. Please change them in your program!

PROGRAM NOTES

Line 40 - This routine is not well known. It allows line by line input from the screen. (See also the loop from line 130 to line 180)

LINE 110 - Disallows entry of data to Line 23 of the screen. It avoids problems associated with scrolling. Printouts are thus limited to 23 lines (0-22).

With some additional programming one might, with relative ease, add a screen save routine giving you a simple word processor which you could couple with various BASIC programs as a sub-routine.

[illegible]

48


```

890 REM
900 REM SCREEN TO PRINTER LOAD
910 REM
920 SCR=PEEK(88)+PEEK(89)*256:POKE 764
,255
930 FOR X=0 TO 39:IF PEEK(764)<>255 TH
EN POP:POKE 764,255:GOTO 950
935 PICT=191*40+SCR+X:STR=USR(DUMP,PIC
T,PR1,PR2)
940 IF PR1<>PR3 THEN ? #1;GR$;PR1$;
942 ? #1
946 IF PR2<>PR3 THEN ? #1;GR$;PR2$;
947 ? #1
948 NEXT X
950 ? #1;" ";
960 TRAP 32767
970 CLOSE #1:GRAPHICS 0:GOTO 620
1000 REM
1005 REM LOAD M-PAINT TYPE FILE
1010 REM
1015 CLOSE #1:OPEN #1,4,0,NAMES:GRAPHI
CS 8+16
1017 A=USR(ADR(MLD$))
1018 IF A<>1 THEN ? "K":GOTO 480
1020 GET #1,C1:GET #1,C2:GET #1,C3:GET
#1,C4:CLOSE #1:POKE 712,C1:POKE 708,C
2:POKE 709,C3:POKE 710,C4
1025 GOTO 800
1190 REM
1200 REM THIS ROUTINE CHANGES
1210 REM THE GRAY SCALES
1211 REM
1215 BKG=5:CR2=1:CR1=3:CR0=2
1220 GRAPHICS 8+32:A=USR(ADR(MODE$)):P
OKE 752,1
1221 POKE 712,3,7*(BKG-1):POKE 710,3,7
*(CR2-1):POKE 709,3,7*(CR1-1):POKE 708
,3,7*(CR0-1)
1228 ? "K":FL=1
1230 DL=PEEK(560)+PEEK(561)*256+173
1240 SCR=PEEK(88)+PEEK(89)*256-500
1245 FOR C=0 TO 3:FOR I=500 TO 540+4:P
OKE I+SCR+40,C*85:NEXT I:NEXT C
1250 POKE DL+4,65:POKE DL+3,10
1255 POKE DL+5,PEEK(DL+1):POKE DL+6,PE
EK(DL+2)
1260 POKE DL,78:POKE DL+1,SCR-INT(SCR/
256)*256:POKE DL+2,INT(SCR/256)
1280 CLOSE #3:OPEN #3,4,0,"K":TRAP 12
80
1310 POKE DL,70
1315 ? "K" PRESS PRINT OR MENU OR M
IEM."":? "4" CHOOSE COLOR TO CHANGE.
(1-4)"
1320 ? " 1 2 3
4":
1330 CLOSE #3:OPEN #3,4,0,"K":TRAP 13
30
1340 GET #3,KEY
1344 IF KEY=88 OR KEY=112 THEN ? "K+ P
RESS ANY KEY TO STOP PRINTING":GOTO 14
50
1345 IF KEY=77 OR KEY=109 THEN 620
1346 IF KEY=86 OR KEY=118 THEN GOSUB 1
710:GOTO 1220
1350 ? "K BLACK = 1 2 3 4 5 = WHITE"
1360 ? " CHOOSE NEW SHADE. (1-5)"
1370 ? "PRESENT SHADE FOR COLOR #":KEY
-48:" IS "
1375 POKE DL,65
1380 IF KEY=49 THEN ? BKG:GET #3,KEY:B
KG=KEY-48:POKE 712,3,7*(KEY-49):GOTO 1
310
1390 IF KEY=50 THEN ? CR0:GET #3,KEY:C
R0=KEY-48:POKE 708,3,7*(KEY-49):GOTO 1
310
1400 IF KEY=51 THEN ? CR1:GET #3,KEY:C
R1=KEY-48:POKE 709,3,7*(KEY-49):GOTO 1
310
1410 IF KEY=52 THEN ? CR2:GET #3,KEY:C
R2=KEY-48:POKE 710,3,7*(KEY-49):GOTO 1
310
1440 SOUND 1,100,10,10:FOR I=1 TO 20:N
EXT I:SOUND 1,0,0,0:GOTO 1310
1450 RESTORE 1500+BKG:FOR I=260 TO 263
:READ PAT:POKE I,PAT:NEXT I
1460 RESTORE 1500+CR0:FOR I=264 TO 267
:READ PAT:POKE I,PAT:NEXT I
1470 RESTORE 1500+CR1:FOR I=268 TO 271
:READ PAT:POKE I,PAT:NEXT I
1480 RESTORE 1500+CR2:FOR I=272 TO 275
:READ PAT:POKE I,PAT:NEXT I
1490 RETURN

```

```

1500 REM DATA FOR GRAY SCALES
1501 DATA 15,15,15,15
1502 DATA 10,5,10,5
1503 DATA 10,0,5,0
1504 DATA 2,0,0,0
1505 DATA 0,0,0,0
1550 REM
1555 REM
1560 REM PRINTER ERROR ROUTINE
1570 ? "SOMETHING IS WRONG WITH THE PR
INTER!"
1580 SOUND 1,100,10,12
1590 FOR A=1 TO 20:NEXT A
1600 SOUND 1,0,0,0
1610 FOR A=1 TO 100:NEXT A
1620 GOTO 850
1650 REM
1655 REM
1710 GRAPHICS 8+16+32:A=USR(ADR(MODE$)
)
1720 POKE 712,3,7*(BKG-1):POKE 710,3,7
*(CR2-1):POKE 709,3,7*(CR1-1):POKE 708
,3,7*(CR0-1)
1730 FOR A=1 TO 400:NEXT A:RETURN

```

TRAIN SOUND

```

1 REM
2 REM | SAN LEANDRO COMPUTER CLUB |
3 REM | SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525, San Leandro, CA |
5 REM | 94588-0152 |
6 REM
10 REM TRAIN SOUND
20 REM BY LEE MINARD
30 REM STARFLEET, DENVER
40 GRAPHICS 2
50 POKE 752,1
60 B=10:H=1
70 REM STEAM RELEASE ***
80 SOUND 0,1,8,4
90 FOR DLAY=1 TO 50:NEXT DLAY
100 FOR STEAM=12 TO 6 STEP -0.5
110 SOUND 0,1,8,5 STEAM
120 FOR DLAY=1 TO 50:NEXT DLAY
130 NEXT STEAM
140 REM CHUGS
150 FOR G=15 TO 1 STEP -0.25
160 FOR A=G TO 2 STEP -0.5
170 SOUND 0,8,8,A
180 SOUND 2,8+1,8,A
190 NEXT A
200 REM SPEED UP CHUGS
210 FOR DLAY=B*2 TO 150:NEXT DLAY
220 FOR DLAY=1 TO 25:NEXT DLAY
230 B=B+2
240 REM GRAPHICS FOR "TRAIN" ***
250 H=H+1
260 IF H=4 THEN H=1
270 IF H=2 THEN GOSUB 490
280 IF H=3 THEN GOSUB 510
290 IF H=1 THEN GOSUB 530
300 IF B=90 THEN 340
310 IF B=90 THEN 340
320 NEXT G
330 REM WHISTLE ***
340 FOR X=1 TO 100
350 IF X=12 OR X=14 OR X=65 OR X=69 TH
EN SOUND 1,75,10,5:SOUND 3,50,10,4:SOU
ND 2,52,49,4
360 IF X=13 OR X=16 OR X=68 OR X=70 TH
EN SOUND 1,75,10,1:SOUND 3,50,10,1:SOU
ND 2,52,10,1
370 IF X=19 OR X=73 THEN SOUND 1,0,0,0
:SOUND 3,0,0,0:SOUND 2,0,0,0
380 FOR A=9 TO 1 STEP -0.4
390 SOUND 0,66,8,A
400 NEXT A
410 ? "CHOO ":
420 NEXT X
430 FOR A=9 TO 1 STEP -0.4
440 SOUND 0,66,8,A
450 NEXT A
460 FOR DLAY=1 TO 2:NEXT DLAY
470 ? "CHOO ":
480 GOTO 340
490 ? #6;"TRAIN":
500 RETURN
510 ? #6;"TRAIN":
520 RETURN
530 ? #6;"TRAIN":
540 RETURN

```

MASTER PENCIL

```

1 REM
2 REM | SAN LEANDRO COMPUTER CLUB |
3 REM | SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525, San Leandro, CA |
5 REM | 94580-0152 |
6 REM
10 REM
20 REM | MASTER * PENCIL |
30 REM |
40 REM | by: Joe Eash |
50 REM
60 CLR :DIM GX(100),GY(100),Y$(20),A$(
20),B$(20)
70 H$=1:V$=1:X=160:Y=90:C=1
80 DIM FL(2500)
90 GOTO 2440
100 REM BLINK
110 COLOR 0:PLOT X,Y:FOR T=0 TO 50
120 NEXT T:COLOR 1:PLOT X,Y:RETURN
130 REM FILL ROUTINE
140 COLOR C:PLOT GX(1),GY(1):FOR T=2 T
0 GT-1
150 DRAWTO GX(T),GY(T):NEXT T:RETURN
160 REM JOYSTICK ROUTINE
170 S=STICK(0)
175 COLOR L:PLOT X,Y
180 IF F=1 THEN F=0:GOSUB 130
190 IF S=15 THEN RETURN
200 IF S=7 THEN X=X+H$
210 IF S=11 THEN X=X-H$
220 IF S=14 THEN Y=Y-V$
230 IF S=13 THEN Y=Y+V$
240 IF S=10 THEN Y=Y-V$:X=X-H$
250 IF S=9 THEN Y=Y+V$:X=X-H$
260 IF S=6 THEN Y=Y-V$:X=X+H$
270 IF S=5 THEN Y=Y+V$:X=X+H$
280 IF X<0 THEN X=0
290 IF X>319 THEN X=319
300 IF Y<0 THEN Y=0
310 IF Y>159 THEN Y=159
320 LOCATE X,Y,L:RETURN
330 REM PLOTTER
340 FOR O=0 TO WD
350 PLOT X+O+(CM*WD),Y+(RW*HT)
360 DRAWTO X+O+(CM*WD),Y+(RW*HT)+HT-1
370 NEXT O:RETURN
380 REM DEL
390 X=X-(WD*8)-1
400 FOR RW=0 TO 7:COLOR OF
410 FOR CM=0 TO 7
420 GOSUB 330:NEXT CM:NEXT RW:RETURN
430 REM OTHER
440 FOR RW=0 TO 7
450 NM=PEEK(57344+PL+RW):CM=0
460 COLOR OF:IF NM>127 THEN NM=NM-128:
COLOR AN
470 GOSUB 330:COLOR OF:CM=1
480 IF NM>63 THEN NM=NM-64:COLOR AN
490 GOSUB 330:COLOR OF:CM=2
500 IF NM>31 THEN NM=NM-32:COLOR AN
510 GOSUB 330:COLOR OF:CM=3
520 IF NM>15 THEN NM=NM-16:COLOR AN
530 GOSUB 330:COLOR OF:CM=4
540 IF NM>7 THEN NM=NM-8:COLOR AN
550 GOSUB 330:COLOR OF:CM=5
560 IF NM>3 THEN NM=NM-4:COLOR AN
570 GOSUB 330:COLOR OF:CM=6
580 IF NM>1 THEN NM=NM-2:COLOR AN
590 GOSUB 330:COLOR OF:CM=7
600 IF NM>0 THEN NM=NM-1:COLOR AN
610 GOSUB 330:NEXT RW
620 X=X+(WD*8)+1
630 RETURN
640 REM CHECK
650 S=0:IF WD*8+X>319 THEN S=1
660 IF HT*8+Y>159 THEN S=1
670 RETURN
680 REM DCHEK
690 S=0:IF X-WD*8<0 THEN S=1
700 IF Y+HT*8>159 THEN S=1
710 RETURN
720 REM ALPHA (LETTERS, ETC.)
730 POKE 752,1:CLOSE #1:OPEN #1,4,0,"K
:"
740 PRINT "K+4":
750 ? "ENTER LETTER HEIGHT (1-8) ":
760 GET #1,HT:HT=HT-48:IF HT<1 OR HT>8
THEN 760

```

```

770 PRINT HT: ? : ? "ENTER LETTER WIDTH
(1-8) ":
780 GET #1,WD:WD=WD-48:IF WD<1 OR WD>8
THEN 780
790 PRINT WD: ?
800 GRAPHICS 8+32:POKE 710,0:POKE 752,
1
810 ? "REPOSITION CURSOR TO PLACE ON SC
REEN"
820 ? "WHERE UPPER-LEFT CORNER OF FIRS
T"
830 ? "CHARACTER IS TO GO. AND PRESS T
HE"
840 ? "FIRE BUTTON..."
850 GOSUB 160:GOSUB 100:S=STRIG(0)
860 IF S=1 THEN 850
870 POKE 764,255
880 S=0:GOSUB 640
890 IF S=1 THEN ? : ? "SORRY. NOT ENOUGH
H ROOM.": ? "PRESS ANY KEY TO REPOSITIO
N": ? :GET #1,S:GOTO 720
900 LOCATE X,Y,S:COLOR 1-S:PLOT X,Y
910 PRINT "PRESS RETURN TO REPOSITION
": ? "PRESS ESC TO RETURN TO MENU": ? "P
RESS THE LETTER OF YOUR CHOICE: "
920 GET #1,K:AN=1:OF=0
930 COLOR S:PLOT X,Y
940 IF K=155 THEN 720
950 IF K=27 THEN RETURN
960 IF K>127 THEN K=K-128:AN=0:OF=1
970 IF K<32 THEN PL=(K+64)*8
980 IF K>31 AND K<96 THEN PL=(K-32)*8
990 IF K>95 THEN PL=K*8
1000 IF K=126 THEN GOSUB 680:IF S=0 TH
EN GOSUB 380:GOTO 1020
1010 GOSUB 430
1020 GOTO 880
1030 REM MIRROR1
1040 IF X<161 THEN A=319-X
1050 IF X>160 THEN A=160-(X-160)
1060 IF Y>80 THEN B=80-(Y-80)
1070 IF Y<81 THEN B=159-Y
1080 COLOR C
1090 PLOT X,Y:PLOT X,B:PLOT A,B:PLOT A
,Y
1100 RETURN
1110 GRAPHICS 8+32:POKE 710,0:POKE 752
,1
1120 ? "MOVE OR DUPLICATE (M/D) ?":
1130 CLOSE #1:OPEN #1,4,0,"K:"
1140 GET #1,M:IF M<68 AND M>77 THEN
1140
1150 MOV=0:IF M=77 THEN MOV=1
1155 IF MOV THEN ? "MOVE:"
1158 IF MOV<>1 THEN ? "DUPLICATE:"
1160 ? "POSITION CURSOR TO UPPER LEFT
CORNER"
1170 ? "OF AREA TO BE TRANSFERRED. THE
N PRESS"
1180 ? "THE FIRE BUTTON.":
1190 GOSUB 160:GOSUB 100:IF STRIG(0)=1
THEN 1190
1195 OX=X:OY=Y
1200 ? "O.K.. POSITION THE CURSOR TO
THE"
1210 ? "LOWER RIGHT CORNER OF THE AREA
TO BE"
1220 ? "TRANSFERRED AND PRESS THE FIRE
BUTTON"
1230 GOSUB 160:GOSUB 100:IF X<OX THEN
X=OX
1240 IF Y<OY THEN Y=OY
1250 IF STRIG(0)=1 THEN 1230
1255 MX=X:MY=Y
1260 ? "NOW. POSITION THE CURSOR IN T
HE UPPER"
1270 ? "LEFT CORNER OF THE AREA WHERE
THIS"
1280 ? "IS TO BE TRANSFERRED TO. AND P
RESS"
1290 ? "THE FIRE BUTTON":
1300 GOSUB 160:GOSUB 100:IF STRIG(0)=1
THEN 1300
1310 ? "TRANSFERRING..."
1320 X1=MX-OX:Y1=MY-OY
1330 IF X+X1>319 THEN X1=319-X
1340 IF Y+Y1>159 THEN Y1=159-Y
1350 FOR T=1 TO Y1
1360 FOR R=1 TO X1
1370 LOCATE OX+R-1,OY+T-1,COL
1380 COLOR COL:PLOT X-1+R,Y-1+T
1390 IF MOV THEN COLOR 0:PLOT OX+R-1,O
Y+T-1

```



```

2280 IF W=ASC("A") THEN GOSUB 720
2290 GOTO 1810
2300 REM COLFLIP
2310 POKE 752,1
2320 IF C=1 THEN C=0: ? "KCOLOR IS OFF"
: RETURN
2330 IF C=0 THEN C=1: ? "KCOLOR IS ON"
2340 RETURN
2350 REM DRAW
2360 COLOR C
2370 IF STRIG(0)=1 THEN RETURN
2380 PLOT X,Y:L=C
2390 IF M=1 THEN GOSUB 1030
2400 IF G=1 THEN GX(GT)=X:GY(GT)=Y:GT=
GT+1
2410 IF G=1 THEN ? "GET DOT MODE. DOT
R":GT
2420 IF G=1 AND GT=101 THEN G=0: ? "KNO
MORE DOTS ACCEPTED FOR GET DOT MODE"
2430 RETURN
2440 REM MAIN *** MAIN LOOP ***
2450 IF TP=1 THEN GRAPHICS 8+32:GOTO 2
470
2460 GRAPHICS 8:TP=1
2470 POKE 752,1:POKE 710,0: ? "K
BU: Joe Eash"
2480 SCR=PEEK(88)+256*PEEK(89)
2490 IF PEEK(87)=0 THEN GRAPHICS 8+32:
POKE 710,0:POKE 752,1: ? "K"
2500 GOSUB 160:REM JOYSTICK
2510 GOSUB 100:REM BLINK CURSOR
2520 GOSUB 2350:REM PLOT POINT
2530 Q=PEEK(764):POKE 764,255
2540 IF Q=20 THEN GOSUB 1810
2550 IF Q=33 THEN GOSUB 2300
2560 IF H5=0 THEN H5=1:TP=0:V5=1:GOTO
2440
2570 GOTO 2490
2580 REM BYE - BYE !!

```

UTILITY PACKAGE

```

1 REM
2 REM | SAN LEANDRO COMPUTER CLUB |
3 REM | SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525, San Leandro, CA |
5 REM | 94580-0152 |
6 REM
30000 REM MACHINE SUBS
30001 DIM DECWORDS(55) : DECWORDS$=""
FOR J=1 TO 55
    DECWORDS$(J)=CHR$(ASC("A")+INT(RND(1)*26))
NEXT J
30002 DIM WORDDECS(73) : WORDDECS$=""
FOR K=1 TO 73
    WORDDECS$(K)=CHR$(ASC("A")+INT(RND(1)*26))
NEXT K
30003 DIM DSKCTRLS(48) : DSKCTRLS$=""
FOR L=1 TO 48
    DSKCTRLS$(L)=CHR$(ASC("A")+INT(RND(1)*26))
NEXT L
30004 DIM CIOCTRLS(58) : CIOCTRLS$=""
FOR M=1 TO 58
    CIOCTRLS$(M)=CHR$(ASC("A")+INT(RND(1)*26))
NEXT M
30005 DIM BMOVES(171) : BMOVES$=""
FOR N=1 TO 171
    BMOVES$(N)=CHR$(ASC("A")+INT(RND(1)*26))
NEXT N
30007 DIM SEAREPS(126) : SEAREPS$=""
FOR O=1 TO 126
    SEAREPS$(O)=CHR$(ASC("A")+INT(RND(1)*26))
NEXT O
30008 SEAREPS(64)="XXXXXXXXXXXXXXXXXXXXX"
30010 SEAREPS(64)="XXXXXXXXXXXXXXXXXXXXX"
30011 SEARCH$(174)="XXXXXXXXXXXXXXXXXXXXX"
30012 SEARCH$(117)="XXXXXXXXXXXXXXXXXXXXX"
30013 DIM BINDECS(37) : BINDECS$=""
FOR P=1 TO 37
    BINDECS$(P)=CHR$(ASC("A")+INT(RND(1)*26))
NEXT P
30014 DIM DECBINS(44) : DECBINS$=""
FOR Q=1 TO 44
    DECBINS$(Q)=CHR$(ASC("A")+INT(RND(1)*26))
NEXT Q
30016 BMOVES(58)="XXXXXXXXXXXXXXXXXXXXX"
30017 BMOVES(115)="XXXXXXXXXXXXXXXXXXXXX"
30018 BMOVES(115)="XXXXXXXXXXXXXXXXXXXXX"
30019 BMOVES(115)="XXXXXXXXXXXXXXXXXXXXX"
30020 RETURN

```

DISK MAP

```

1 REM
2 REM | SAN LEANDRO COMPUTER CLUB |
3 REM | SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525, San Leandro, CA |
5 REM | 94580-0152 |
6 REM
1000 GRAPHICS 0:POKE 82,0:SETCOLOR 2,0
/0
1010 REM *****
1020 REM *
1030 REM * UTOC DISPLAY PGM 10/20/84*
1040 REM *
1050 REM *****
1060 REM
1070 DIM USINGS(30),BUFS(128),MLS(4):F
OR I=1 TO 4:READ BYTE:MLS(I)=CHR$(BYTE
):NEXT I:POKE 752,1
1080 DIM CL$(1):CL$=CHR$(125)
1090 DIM BINARY$(8),TEMPARRAY$(64)
1100 DIM TEMP$(255)
1110 DIM FILE$(15)
1120 DIM POWER(7)
1130 POSITION 12,12:?"ONE MOMENT...":
1140 FOR I=0 TO 7
1150 POWER(I)=INT(2^I+0.5)
1160 NEXT I
1170 DDEVIC=768
1180 DUNIT=769
1190 DCOMND=770
1200 DSTATS=771
1210 DBUFLO=772
1220 DAUX1=778
1230 DTIMLO=774
1240 UTOC=360
1250 DATA 104,76,83,220
1260 FOR I=1 TO 128:BUFS(I,I)="":NEXT
I
1270 ? CL$:
1280 POKE 752,0
1290 DL=PEEK(560)+256*PEEK(561)
1300 OFFSET=PEEK(DL+4)+256*PEEK(DL+5)
1310 POSITION 12,1:?"DMAP Version 1.1
"
1320 POSITION 6,3:?"Copyright 1985 Mi
chael Curry"
1330 TRAP 1330:POSITION 7,11:PRINT "En
ter Drive Number to Map":POSITION 12,1
2:?"(1-4 or 0 to Quit) ":INPUT DRIVE
1340 TRAP 0
1350 IF DRIVE=0 THEN GRAPHICS 0:END
1360 IF DRIVE<1 OR DRIVE>4 THEN 1270
1370 POSITION 0,15:?"Insert disk to m
ap and press [RETURN]":INPUT TEMP$
1380 POKE 752,1
1390 POKE DDEVIC,DRIVE+48
1400 POKE DUNIT,DRIVE
1410 POKE DTIMLO,255
1420 POKE DCOMND,82:REM READ
1430 BUFFER=ADR(BUFS)
1440 ADDRESS=DBUFLO:VALUE=BUFFER:GOSUB
2030
1450 SECTOR=UTOC:REM READ UTOC SECTOR
1460 ADDRESS=DAUX1:VALUE=SECTOR:GOSUB
2030
1470 X=USR(ADR(MLS))
1480 ? "K":POSITION 10,0:?"[CURSOR]
[ENTER] [QUIT]"
1490 POKE 752,1:REM CURSOR OFF
1500 BL=ASC(BUFS(4))+256*ASC(BUFS(5))
1510 BU=ASC(BUFS(2))+256*ASC(BUFS(3))-
BL
1520 POSITION 0,1:?"SECTORS: ":POSITIO
N 10,1
1530 WIDE=9
1540 VALUE=BU:GOSUB 2080:?" USING$," I"
:
1550 VALUE=BL:GOSUB 2080:?" USING$," I"
:
1560 VALUE=BU+BL:GOSUB 2080:?" USING$;
1570 POSITION 0,2:?" BYTES:":POSITIO
N 10,2
1580 VALUE=BU*125:GOSUB 2080:?" USING$;
" I":
1590 VALUE=BL*125:GOSUB 2080:?" USING$;
" I":
1600 VALE=(BU+BL)*125:GOSUB 2080:?" USI
NG$;
1610 YP=0:XP=0

```

```

1620 FOR I=0 TO 3:POSITION I*10,4:?" I:
:POSITION I*10,5:FOR J=0 TO 9:?" J::NEX
T J:NEXT I
1630 POSITION 0,0:?"DRIVE: ";DRIVE:
1640 FOR LOOP=10 TO 99
1650 BYTE=ASC(BUFS(LOOP,LOOP))
1660 GOSUB 1810
1670 FOR LOOP1=1 TO 8
1680 YP=YP+1:IF YP>10 THEN YP=1:XP=XP+
1
1690 POKE OFFSET+((YP+5)*40+XP),ASC(BI
NARY$(LOOP1,LOOP1))
1700 NEXT LOOP1
1710 NEXT LOOP
1720 REM POKE 764,255
1730 POSITION 3,3:?"<<< PRESS ANY KEY
TO CONTINUE >>>":
1740 FOR I=1 TO 75:IF PEEK(764)<>255 T
HEN 1790
1750 NEXT I
1760 POSITION 3,3:?"<<< Press Any Key
To Continue >>>":
1770 FOR I=1 TO 100:NEXT I
1780 GOTO 1730
1790 POKE 764,255
1800 GOTO 1270
1810 REM SUBROUTINE TO RETURN
1820 REM 8 CHAR STRING OF BINARY
1830 REM CHARS
1840 REM
1850 BINARY$="":REM 8 SPACES
1860 FOR ITEMP=7 TO 0 STEP -1
1870 TEMP=POWER(ITEMP)
1880 IF (BYTE-TEMP)>=0 THEN BINARY$(8-
ITEMP,8-ITEMP)="C":BYTE=BYTE-TEMP
1890 NEXT ITEMP
1900 RETURN
1910 REM CONVERT 8 CHAR STRING TO
1920 REM TO DECIMAL NUMBER 0-255
1930 REM
1940 BYTE=0
1950 FOR I=7 TO 0 STEP -1
1960 IF BINARY$(8-I,8-I)<>" " THEN BYT
E=BYTE+POWER(I)
1970 NEXT I
1980 PARAM=BYTE
1990 RETURN
2000 REM DPOKE HI AND LO BYTE OF VALUE
2010 REM INTO ADDRESS
2020 REM
2030 POKE ADDRESS+1,INT(VALUE/256)
2040 POKE ADDRESS,VALUE-(PEEK(ADDRESS+
1)*256)
2050 RETURN
2060 REM RIGHT JUSTIFY NUMERIC INTO ST
RING WITH LENGTH OF 10
2070 REM
2080 FOR I=1 TO WIDE
2090 USINGS(I,I)=" "
2100 NEXT I:REM INIT STRING
2110 TEMP$=STR$(VALUE)
2120 USINGS(WIDE-LEN(TEMP$))=TEMP$
2130 RETURN

```

RELOC

```

1 REM
2 REM | SAN LEANDRO COMPUTER CLUB |
3 REM | SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525, San Leandro, CA |
5 REM | 94580-0152 |
6 REM
10 GOSUB 30000
20 LOMEM0=PEEK(128)+PEEK(129)*256:GOSUB
8 100
30 NL=LOMEM0+10
40 Z=USR(ADR(RELOC$),NL):GOSUB 100
50 FOR I=LOMEM0 TO NL-1:POKE I,0:NEXT
I
60 Z=USR(ADR(RELOC$),LOMEM0):GOSUB 100
70 END
100 ? PEEK(128)+PEEK(129)*256:RETURN
30000 REM *****
30005 DIM RELOC$(55):RELOC$="hhhhhhhh
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX"
32767 RETURN

```

DIRECT SCREEN WRITING

```

1 REM
2 REM | SAN LEANDRO COMPUTER CLUB |
3 REM | SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525, San Leandro, CA |
5 REM | 94580-0152 |
6 REM
10 CLR
20 POKE 82,0:POKE 83,40:POKE 752,1:7 C
HRS(125)
30 DIM A$(960),CODE$(195)
40 REM
50 REM CODE LOADING ROUTINE
60 REM
70 POSITION 13,2:?"LOADING CODE":FOR
I=1 TO 195:READ A:CODE$(I)=CHR$(A):5
OUND 1,A,10,4:SOUND 1,A+1,14,2:NEXT I:
CDE=ADR(CODE$)
80 REM
90 REM INITIALIZE TEST STRING
100 REM
110 POSITION 10,4:?"INITIALIZING STRI
NG":FOR J=1 TO 40:FOR I=0 TO 23:K=I*40
+J:AS$(K,K)=CHR$(64+J)
120 SOUND 1,I+20,10,4:SOUND 0,I+21,14,
4:NEXT I:NEXT J:SOUND 1,0,0,0:SOUND 0,
0,0,0
130 REM
140 REM SHOW SPEED DIFFERENCE BETWEEN
A PRINT AND A DIRECT WRITE
150 REM
160 ? CHR$(125):POSITION 12,2:?"FIRST
USE A PRINT":POKE 20,0
170 IF PEEK(20)<100 THEN 170
180 ? AS$:POKE 20,0
190 IF PEEK(20)<50 THEN 190
200 ? :?" NOW THE FAST WA
Y"
210 FOR I=0 TO 1000:NEXT I
220 FOR I=0 TO 241:?"NEXT I:POSITION 0
,20
230 A=USR(CDE,0,0,ADR(AS$),LEN(AS$))
240 FOR I=0 TO 500:NEXT I
250 ? CHR$(125):?" NOW FOR A LIT
TLE FUN WITH":POKE 20,0
260 IF PEEK(20)<100 THEN 260
270 REM
280 REM SHOW POSITIONAL CAPABILITIES
290 REM
300 AS$="":AS$=" FAST RANDOM screen
310 AS$(27,27)=CHR$(30):AS$(1,1)=CHR$(3
1)
310 A=USR(CDE,5,2,ADR(AS$),LEN(AS$)):POK
E 20,0
320 IF PEEK(20)<150 THEN 320
330 FOR I=10 TO 100:X=RND(0)*40:Y=RND(
0)*24:A=USR(CDE,X,Y,ADR(AS$),LEN(AS$))
340 NEXT I
350 FOR I=0 TO 1500:NEXT I:?" CHR$(125)
:POSITION 14,2:?"END OF DEMO":POSITIO
N 2,21:RUN "D:MENU.BAS"
360 REM
370 REM DATA FOR MACHINE CODE ROUTINE
380 REM
390 DATA 169,0,162,5,149,203,202,16,25
1,104,201,2,200,6,164,84,166,85,200,27
,104,104,170,104
400 DATA 104,168,224,41,16,4,192,24,14
4,13,104,104,104,104,169,141,133,212,1
69,0,133,213,96,169
410 DATA 0,136,48,11,24,105,40,133,207
,144,246,230,200,200,242,165,207,24,10
1,88,133,207,165,200
420 DATA 101,89,133,200,138,24,101,207
,133,207,144,2,230,208,104,133,204,104
,133,203,104,133,209,104
430 DATA 170,200,2,198,209,165,207,133
,205,165,208,133,206,160,0,177,203,201
,155,240,49,72,41,127
440 DATA 201,32,16,6,104,24,105,64,144
,11,201,96,16,6,104,56,233,32,176,1,10
4,145,205,200
450 DATA 208,4,230,204,230,206,202,208
,214,198,209,16,210,169,1,133,212,169,
0,133,213,96,202,200,4,198,209
460 DATA 48,240,165,207,24,105,40,133,
207,144,2,230,208,152,56,101,203,133,2
03,144,165,204,200,161

```

DIGITAL ALARM CLOCK

```

1 REM DIGITAL ALARM CLOCK IN 3K
BY STEVE KUNZE age 14
Adelaide Computer Club
2 REM Reprinted in the San Leandro
Computer Club Special Edition
3 REM P. O. BOX 1525
SAN LEANDRO, CA 94577-0152
4 TIME=250
5 GRAPHICS 0:POKE 710,55:POKE 712,176:
POKE 709,15:POKE 752,1:?" :?
6 ? "THIS IS A PROGRAM FOR A DIGITAL-
":?"ALARM CLOCK WHICH HAS A 24-HOUR":?
"COUNTER. I DON'T THINK THAT ANYONE"
7 ? "WOULD LEAVE THEIR COMPUTER ON OVE
R-":?"NIGHT. BUT IT SHOWS YOU JUST WA
HT":?"YOU CAN DO WITH A FEW LINES AND
"
8 ? "HOW EASY IT IS ON AN
9 ? "PRESS START WHEN READY"
9 IF PEEK(53279)<6 THEN 9
10 DIM D$(2),AS$(1),R$(1)
20 OPEN #1,4,0,"K:"
50 GRAPHICS 0:POKE 709,15:POKE 710,0:P
OKE 752,1
60 ? "
70 ?
80 ? "ENTER ALL NECESSARY DATA !"
90 D$=""
100 ? " HOUR ";:INPUT H
110 ? "MINUTE ";:INPUT M
120 ? "SECOND ";:INPUT S
130 ? "VOLUME OF SECOND-BEEP (0-15)";:
INPUT V:TRAP 150
150 ? :?"DO YOU WANT TO SET AN ALARM
(Y/N)?":GET #1,A
160 IF A<>ASC("Y") THEN GOTO 240
180 ? :?"TO TURN THE ALARM OFF YOU MU
ST PRESS":?" THE FIREBUTTON ON JOYSTI
CK #1 ":?" OR PRESS
200 ? :?"ENTER DATA FOR ALARM !"
210 ? " HOUR ";:INPUT H2
220 ? "MINUTE ";:INPUT M2
230 ? "SECOND ";:INPUT S2
240 ? :?"PRESS TO BEGIN !"
250 IF PEEK(53279)<6 THEN 250
300 REM *SETTING UP THE DISPLAY*
320 GRAPHICS 18
325 POKE 712,176
330 POSITION 5,3:?" M H S"
340 POSITION 7,5:?" M ":"
350 POSITION 10,5:?" M ":"
375 POSITION 3,9:?"M ":";H2:":";M
2:":";S2
385 B=176
390 POKE 712,B
400 REM *ACTUAL COUNTER*
405 POSITION 5,5:?" M ":"
410 SOUND 0,50,14,V
415 POSITION 8,5:?" M ":"
420 POSITION 11,5:?" M ":"
550 S=S+1:IF S>59 THEN M=M+1:S=0:SOUND
0,60,6,10
560 IF M>59 THEN H=H+1:M=0:SOUND 0,35,
10,10
570 IF H<=11 THEN D$="AM"
580 IF H>=12 THEN D$="PM"
590 IF H>23 THEN H=0:M=0:S=0:D$="AM"
600 REM *PRINTING THE NUMBERS*
650 POSITION 5,5:?" M ":"
660 POSITION 8,5:?" M ":"
670 POSITION 11,5:?" M ":"
675 FOR P=1 TO 5:NEXT P
680 POSITION 1,5:?" M ":"
690 SOUND 0,0,0,0
700 FOR P=1 TO TIME:NEXT P
750 IF H2=H AND M2=M AND S2=S AND A=AS
C("Y") THEN SOUND 2,30,0,15:SOUND 3,40
,14,15
760 IF STRIG(0)=0 OR PEEK(53279)=6 THE
M SOUND 2,0,0,0:SOUND 3,0,0,0
800 GOTO 400
1000 END :REM RETYPED BY P.J. BARATTA

```

DOS MOD

```

1 REM
2 REM | SAN LEANDRO COMPUTER CLUB |
3 REM | SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525. San Leandro, CA |
5 REM | 94580-0152 |
6 REM
10 GRAPHICS 0: ? "DOS 2.0 MODIFICATION"
: ? : ? "FUJI ATARI COMPUTER USERS' GROU
P": ? "JAPAN"
20 ? : ? : ? "MODIFYING DOS.2.0": ? "WAIT
FOR A MOMENT, PLEASE"
30 RESTORE : TRAP 100
40 READ A,X: POKE A,X: GOTO 40
100 ? : ? "THANK YOU FOR WAITING." : ? : ?
"NOW TYPE 'DOS [RETURN]' "
110 ? "AND SAVE THIS NEW DOS TO YOUR"
120 ? "DISKETTE USING 'H' COMMAND."
130 END
1000 DATA 2174,169
1010 DATA 2175,60
1020 DATA 2176,141
1030 DATA 2177,41
1040 DATA 2178,3
1050 DATA 2179,169
1060 DATA 2180,203
1070 DATA 2181,141
1080 DATA 2182,42
1090 DATA 2183,3
1100 DATA 2184,169
1110 DATA 2185,7
1120 DATA 2186,141
1130 DATA 2187,43
1140 DATA 2189,173
1150 DATA 2190,83
1160 DATA 2191,24
1170 DATA 2192,200
1180 DATA 2193,10
1190 DATA 2194,169
1200 DATA 2195,40
1210 DATA 2196,141
1220 DATA 2197,84
1230 DATA 2198,3
1240 DATA 2199,169
1250 DATA 2200,23
1260 DATA 2201,76
1270 DATA 2202,111
1280 DATA 2203,21
1290 DATA 2204,100
1300 DATA 2205,84
1310 DATA 2206,24
1320 DATA 2207,169
1330 DATA 2208,0
1340 DATA 2209,141
1350 DATA 2210,83
1360 DATA 2211,24
1370 DATA 2212,76
1380 DATA 2213,159
1390 DATA 2214,23
1400 DATA 4867,48
1410 DATA 4868,19
1420 DATA 4869,80
1430 DATA 4871,5
1440 DATA 4872,5
1450 DATA 4873,0
1460 DATA 4874,7
1470 DATA 4875,7
1480 DATA 4879,120
1490 DATA 4950,80
1500 DATA 5009,4
1510 DATA 5014,1
1520 DATA 5015,1
1530 DATA 5017,84
1540 DATA 5025,20
1550 DATA 5033,195
1560 DATA 5041,32
1570 DATA 5044,2
1580 DATA 5046,34
1590 DATA 5047,34
1600 DATA 5049,139
1610 DATA 5050,1
1620 DATA 5105,24
1630 DATA 5108,2
1640 DATA 5110,23
1650 DATA 5111,23
1660 DATA 5113,153
1670 DATA 5185,66
1680 DATA 5186,40
1690 DATA 5201,67
1700 DATA 5202,0

```

```

1710 DATA 5450,8
1720 DATA 5535,192
1730 DATA 5536,224
1740 DATA 5538,225
1750 DATA 5927,155
1760 DATA 5928,68
1770 DATA 5929,49
1780 DATA 5930,50
1790 DATA 5931,68
1800 DATA 5932,67
1810 DATA 5933,86
1820 DATA 5934,46
1830 DATA 5935,83
1840 DATA 5936,89
1850 DATA 5937,83
1860 DATA 5938,155
1870 DATA 5939,32
1880 DATA 5940,146
1890 DATA 5941,8
1900 DATA 5942,100
1910 DATA 5943,250
1920 DATA 5944,191
1930 DATA 5945,23
1940 DATA 6226,155
1950 DATA 6227,0
1960 DATA 6228,11
1970 DATA 6229,29
1980 DATA 6429,76
1990 DATA 6430,51
2000 DATA 6431,23
2010 DATA 6780,2

```

SIMPLE TEXT DUMP

```

1 REM
2 REM | SAN LEANDRO COMPUTER CLUB |
3 REM | SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525. San Leandro, CA |
5 REM | 94580-0152 |
6 REM
10 REM SIMPLE SCREEN DUMP T. REICHARD
20 DIM SCR$(40*25),H$(40)
30 CLOSE #2: OPEN #2,4,0,"K:"
40 CLOSE #3: OPEN #4,13,0,"E:"
50 POKE 82,0: POSITION 0,0: ? "++": REM
M AFTER SCREEN DUMP
60 POKE 764,255
70 IF PEEK(53279)=6 THEN 130: REM CHECK
FOR START KEY
80 IF PEEK(764)=255 THEN 70
90 GET #2,KEY
100 ? CHR$(KEY):
110 IF PEEK(84)=23 THEN 190
120 GOTO 60
130 POKE 752,1: POSITION 0,0: ? "++": RE
M AFTER SCREEN DUMP
140 FOR ROW=0 TO 22
150 H$=""
160 POSITION 0,ROW: INPUT #4:H$
170 LPRINT H$
180 NEXT ROW
190 POKE 752,0: POSITION 0,0: ? "++": RE
M AFTER SCREEN DUMP
200 GOTO 60
500 REM SPECIAL NOTE: IN LINES 50. 13
0 AND 190 THE !! ARE SUPPOSED TO
600 REM BE ESC DOWN FOLLOWED BY ESC UP
ARROW WHICH CANNOT BE PRINTED
700 REM USING THIS WORD PROCESSOR. PL
EASE CHANGE THEM IN YOUR PROGRAM!

```


16-BIT SOUND

```

1 REM
2 REM | SAN LEANDRO COMPUTER CLUB |
3 REM | SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525, San Leandro, CA |
5 REM | 94580-0152 |
6 REM

100 REM GRAPHICS DEMO
110 GRAPHICS 18: ? #6; ? #6; " 16 BIT
SOUND": ? #6; ? #6; " BY JERRY WHITE"
120 DIM S16$(56): REM FOR SUBROUTINE
130 FOR ME=1 TO 56: READ BYTE: S16$(ME, M
E)=CHR$(BYTE): NEXT ME
140 REM FOR GRAPHICS
150 SOUND 0, 0, 0: POKE 53760, 120
160 DIM NS(24), FREQ(7, 12): NS="B ABA GH
G FBF E DHD CHC ": GOTO 220
170 REM FOR SUBROUTINE
180 POSITION 6, 5: ? #6; "OCTAVE="; OCTAVE
:: IF BOTH THEN ? #6; "+" ; OCTAVE+1:
190 POSITION 6, 7: ? #6; "PITCH="; PITCH; "
":
200 SP=PITCH*2-1: POSITION 6, 9: ? #6; "NO
TE="; NS(SP, SP+1): RETURN
210 REM FOR GRAPHICS
220 FOR OCTAVE=7 TO 1 STEP -1: FOR PITC
H=12 TO 1 STEP -1
230 READ FREQ: FREQ(OCTAVE, PITCH)=FREQ:
NEXT PITCH: NEXT OCTAVE
240 BOTH=0: LOWOCT=7: WAIT=10
250 REM FOR SOUND
260 FOR OCTAVE=LOWOCT TO 1 STEP -1: FOR
PITCH=12 TO 1 STEP -1
270 SETCOLOR 4, PITCH, 0: GOSUB 180: VOL=8
: POKE 540, VOL
280 IF NOT BOTH THEN JW=USR(ADR(S16$)
, FREQ(OCTAVE, PITCH), VOL): GOTO 300
290 JW=USR(ADR(S16$), FREQ(OCTAVE, PITCH
), VOL, FREQ(OCTAVE+1, PITCH), VOL)
300 IF NOT VOL THEN 320
310 VOL=PEEK(540): GOTO 280
320 GOSUB 480
330 NEXT PITCH: NEXT OCTAVE
340 IF NOT BOTH THEN BOTH=1: LOWOCT=6:
GOTO 260
350 FOR ME=5 TO 9 STEP 2: POSITION 6, ME
: ? #6; " ": NEXT ME
360 REM FOR GRAPHICS
370 VOL=8: OCTAVE=5: PITCH=7: HOLD=16: WA
IT=8: GOSUB 530
380 PITCH=12: HOLD=4: WAIT=8: GOSUB 530
390 OCTAVE=6: PITCH=1: HOLD=4: WAIT=8: GO
SUB 530
400 OCTAVE=5: PITCH=12: HOLD=4: WAIT=8: GO
SUB 530
410 PITCH=10: HOLD=16: WAIT=8: GOSUB 530
420 PITCH=12: HOLD=16: WAIT=32: GOSUB 530
430 PITCH=8: HOLD=16: WAIT=8: GOSUB 530
440 PITCH=7: HOLD=32: WAIT=8: GOSUB 530
450 GRAPHICS 0: ? : ? "BASIC": ? "IS": EN
D
460 REM FOR GRAPHICS
470 REM FOR WAIT 200MS OF A SECOND
480 POKE 540, WAIT
490 IF PEEK(540) THEN 490
500 RETURN
510 REM FOR SUBROUTINE
520 REM FOR GRAPHICS
530 POKE 540, HOLD: X=USR(ADR(S16$), FREQ
(OCTAVE, PITCH), VOL, FREQ(OCTAVE+1, PITCH
), VOL)
540 IF PEEK(540) THEN 540
550 X=USR(ADR(S16$), FREQ(OCTAVE, PITCH)
, 0, FREQ(OCTAVE+1, PITCH), 0)
560 GOTO 480
20000 REM FOR GRAPHICS
20010 DATA 104, 201, 2, 240, 33, 201, 4, 240,
12, 170, 224, 0, 240, 41
20020 DATA 202, 104, 104, 240, 247, 208, 245
, 104, 141, 2, 210, 104, 141, 0
20030 DATA 210, 104, 104, 41, 15, 9, 160, 141
, 3, 210, 104, 141, 6, 210
20040 DATA 104, 141, 4, 210, 104, 104, 41, 15
, 9, 160, 141, 7, 210, 96
30000 REM FOR GRAPHICS
30010 DATA 27357, 25821, 24372, 23003, 217
12, 20493, 19342, 18256, 17231, 16264, 15351
, 14489

```

```

30020 DATA 13675, 12907, 12182, 11498, 108
52, 10243, 9668, 9125, 8612, 8128, 7626, 7241
30030 DATA 6834, 6450, 6088, 5746, 5423, 51
18, 4830, 4559, 4303, 4061, 3832, 3617
30040 DATA 3414, 3222, 3040, 2869, 2708, 25
55, 2412, 2276, 2140, 2027, 1913, 1805
30050 DATA 1703, 1607, 1517, 1431, 1350, 12
74, 1202, 1134, 1070, 1010, 953, 899
30060 DATA 848, 800, 755, 712, 672, 634, 598
, 564, 532, 501, 473, 446
30070 DATA 421, 397, 374, 353, 332, 313, 295
, 270, 262, 247, 233, 219

```

PLAYER MISSILE DRIVER

```

1 REM
2 REM | SAN LEANDRO COMPUTER CLUB |
3 REM | SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525, San Leandro, CA |
5 REM | 94580-0152 |
6 REM

10 REM FOR GRAPHICS
20 REM FOR GRAPHICS
30 OPEN #1, 0, 0, "G:": OPEN #5, 4, 0, "K:": G
RAPHICS 1+16
40 ? #1; "POS 0, 40, 16": ? #1; "SIZE 0, 2"
50 ? #1; "OBJ 0", ADR("=="): 4
60 ? #1; "COLOR 0, 0, 15": ? #1; "PRIOR 4"
70 X=0: Y=0
80 GET #5, K
90 IF K=ASC("+") THEN Y=Y-1: GOTO 200
100 IF K=ASC("4") THEN Y=Y+1: GOTO 200
110 IF K=ASC("+") THEN X=X+1: GOTO 200
120 IF K=ASC("+") THEN X=X-1: GOTO 200
130 IF K=155 THEN X=0: Y=Y+1: GOTO 200
140 POSITION X, Y: ? #6; CHR$(K):
150 X=X+1: IF X>19 THEN Y=Y+1
200 IF X<0 THEN X=19
210 IF X>19 THEN X=0
220 IF Y<0 THEN Y=23
230 IF Y>23 THEN Y=0
240 ? #1; "POS 0", 40+X*8, 16+Y*8
250 GOTO 80

```

DISK FORMATTER

```

1 REM
2 REM | SAN LEANDRO COMPUTER CLUB |
3 REM | SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525, San Leandro, CA |
5 REM | 94580-0152 |
6 REM

10 REM DISK FORMATTING PROGRAM
11 REM BY
12 REM KENNETH J. PIETRUCHA
13 REM NOVEMBER 29, 1984
20 TOTAL=0: PRINT CHR$(125): POKE 752, 1
25 POKE 709, 14: POKE 710, 24: POKE 712, 20
30 TOTAL=TOTAL+1
35 POSITION 6, 7: PRINT "INSERT DISK TO
BE FORMATTED"
40 POSITION 5, 9: PRINT "PRESS START TO
FORMAT DISK #"; TOTAL
45 IF PEEK(53279)<>6 THEN 45
48 POKE 710, 32
50 FOR X=1 TO 200: NEXT X
52 XIO 254, #1, 0, 0, "D:"
60 POKE 710, 20
65 POSITION 6, 14: PRINT "FORMAT OF DISK
#"; TOTAL; " COMPLETE"
70 POSITION 7, 16: PRINT "PRESS OPTION T
O CONTINUE"
75 IF PEEK(53279)<>3 THEN 75
80 PRINT CHR$(125): GOTO 25

```

```

0100 ;16 BIT SOUND SUBROUTINE
0110 ;REV 9/23/82 BY JERRY WHITE
0120 ;CREATE 1 OR 2 16 BIT SOUNDS
0130 ;FROM BASIC USR CALL
0140 ;STORED IN STRING S16$
0150 ;FOR 1 SOUND
0160 ;JW=USR(ADR(S16$),FREQ,VOL)
0170 ;FOR 2 SOUNDS
0180 ;JW=USR(ADR(S16$),FREQ,VOL,FREQ,VOL)
0190 ;
0200      *=      $600      ;RELOCATABLE
0210      PLA      ;# OF PARAMETERS
0220      CMP      #2      ;IF 2 PARAMETERS
0230      BEQ      TWO      ;SINGLE SOUND ROUTINE
0240      CMP      #4      ;IF 4 PARAMETERS
0250      BEQ      FOUR      ;DOUBLE SOUND ROUTINE
0260 ;
0270 ;AVOID LOCKUP IN CASE OF
0280      TAX      ;INVALID # OF PARAMETERS
0290      OOPS CPX      #0      ;NONE?
0300      BEQ      END      ;STACK CLEAN
0310      DEX      ;SUBTRACT 1
0320      PLA      ;GET PARAMETER
0330      PLA      ;OFF MY STACK
0340      BEQ      OOPS      ;FORCE GOTO OOPS
0350      BNE      OOPS      ;IN EITHER CASE
0360 ;
0370      FOUR PLA      ;HI BYTE
0380      STA      $D202      ;AUDF2
0390      PLA      ;LOW BYTE
0400      STA      $D200      ;AUDF1
0410      PLA      ;VOLUME
0420      PLA      ;NEED ONLY LOW BYTE
0430      AND      #$0F      ;MAKE 0 TO 15
0440      ORA      #$A0      ;DISTORTION 10
0450      STA      $D203      ;AUDC2
0460 ;
0470      TWO PLA      ;HI BYTE
0480      STA      $D206      ;AUDF4
0490      PLA      ;LOW BYTE
0500      STA      $D204      ;AUDF3
0510      PLA      ;VOLUME
0520      PLA      ;NEED ONLY LOW BYTE
0530      AND      #$0F      ;MAKE 0 TO 15
0540      ORA      #$A0      ;DISTORTION 10
0550      STA      $D207      ;AUDC4
0560 ;
0570      END      RTS      ;RETURN TO BASIC

```

FLITTERBUG

```
; JIM WARREN
; SAN LEANDRO COMPUTER CLUB
; P.O. BOX 1525
; SAN LEANDRO CA 94577-0152
; KEY SYSTEM EBS (415) 352-5528
```

```
FOR RUN TIME COMPILE:
SET $495=12 FROM THE MONITOR
USE DOS 2.0
COMPILE EVERYTHING FROM DISK
```

```
MODULE
INT
  VERTICAL_OFFSET,HORIZONTAL_OFFSET,
  TRIGGER_BUTTON
CARD
  TIME=[1500],SCREEN,I,TOTAL_EGGS,
  EATER_BIRTH=[0],BIG_BOY_BIRTH=[0],
  MATE_BIRTH=[0],MATE_BIRTH_RATE=[200],
  BIG_BOY_BIRTH_RATE=[200],
  EATER_BIRTH_RATE=[300],
  ADR_MAIN,ADR_EDIT,ADR_BUG,RAMSET
BYTE
  COLUMNS=[15],ROWS=[23],JOYSTICK,
  BUG_EGGS,AIMA_MATE=[5],
  CONSOLE=53279, OLECHR=93,
  SCREEN_LOK=88,SCREEN_HI=89,
  SCREEN_EGG,HUE=[0],GENERATION=[0],
  POS,POSX,BACKGROUND=[60],
  EATER_COLUMNS,EATER_ROWS,
  BIG_BOY_COLUMNS,BIG_BOY_ROWS,
  OPTION=[0],MINIMUM_EGGS=[25],KEY=764,
  MATE_COLUMNS,MATE_ROWS,GENE=[11],
  RE_CURSE=[0],REQUIRED
  BYTE ARRAY HANDLINE
```

```
DEFINE POP="$68 $68"
```

```
PROC RE_MAIN=ERROR()
```

```
PROC RE_EDIT()
POP
  ERROR()
RETURN
```

```
PROC RE_BUG=ERROR()
```

```
PROC WAIT()
FOR I=0 TO 1000 DO CD
RETURN
```

```
PROC PAUSE(CARD NUM)
CARD I
FOR I=0 TO NUM
DO WAIT() OD
RETURN
```

```
PROC BEEP()
FOR I=1 TO 500
DO
  SOUND(2,255,12,3)
OD
  SNDRST()
RETURN
```

```
PROC DING()
FOR I=0 TO 500
```

```
*)
PRINT("
  position cursor with joystick 0
  draw or erase bit with trigger
") T=1 DONG()
PAUSE(7)
DO
  T=Strig(0) TEST=Stick(0)
  IF T=0 THEN PRINT("*) DING()
  EXIT
FI
IF
  TEST<>15 THEN TEST=1
  EXIT
FI
IF
  CONSOLE=6 OR CONSOLE=3 THEN
  EXIT
FI
  PAUSE(7)
OD
FI
  PAUSE(6)
IF
  CONSOLE=3 THEN PART_COUNT==+1
  IF
    PART_COUNT>5 THEN PART_COUNT=1
```

```
RETURN
FI
  ERROR=ADR_EDIT RE_EDIT()
IF
  CONSOLE=6 THEN
  X=14 LINE0=0 Y=8 S=0
  DO
    S==+1
    FOR I=0 TO 8
    DO
      X==+1 POSITION(X,Y) PRINT("")
      POWER=LOOKUP(X)
      BIT=LOCATE(X,Y)
      IF BIT=170 OR BIT=42
        THEN LINE0==+POWER FI
    OD
```

```
  DEFINE CHARACTER $
  IF PART_COUNT=1 THEN PART=71
  ELSEIF PART_COUNT=2 THEN PART=70
  ELSEIF PART_COUNT=3 THEN PART=96
  ELSEIF PART_COUNT=4 THEN PART=81
  ELSEIF PART_COUNT=5 THEN PART=69
  FI
  POKE(RAMSET+(8*PART)+S-1,LINE0)
```

```
  POSITION(X,Y) PRINT("")
  POSITION(X+1,Y) PRINT("----")
  PRINTBE(LINE0) LINE0=0
  Y==+1 IF Y>15 THEN S=0 EXIT FI
  X=14
  OD
  POSITION(15,8) PRINT("")
```

```
FI
  IF KEY<>255 THEN CHANGE COLORS
  IF KEY=14 THEN ADD=1
  ELSEIF KEY=15 THEN ADD=-1
  FI
  IF
    KEY=50 THEN R0==+ADD
    ELSEIF KEY=31 THEN R1==+ADD
    ELSEIF KEY=30 THEN R2==+ADD
    ELSEIF KEY=26 THEN R3==+ADD
    ELSEIF KEY=24 THEN R4==+ADD
    ELSEIF KEY=28 THEN COLORS()
  FI
  KEY=255
  THEN PRINT REGISTER CONTENTS
  POSITION(0,21) PRINTF("
  R1 %I %I %I %I %I ",
  R4,R0,R1,R2,R3)
  FI
```

```
OD
RETURN
```

```
PROC DOCREADER()
  BYTE LINECOUNT,KEY=764
  BYTE ARRAY FILELINES(256)
  POKE(82,1)
  DO
    ERROR=ADR_MAIN;if there is a disk error
    ;PROC Error will restart the program
    CLOSE(2)
    OPEN(2,"D1:FLITTER.DOC",4,0)
    DO
      GRAPHICS(0)
      SETCOLOR(1,5,8)
      SETCOLOR(2,9,8)
      SETCOLOR(3,13,8)
      SETCOLOR(4,0,0)
    ANTIC4()
    POINT ANTIC AT RAMSET
    POKE(756,RAMSET/256)
    POSITION(1,23)
    PRINT("space bar for text... option for menu")
```

```
  POSITION(1,0)
  FOR LINECOUNT=0 TO 20
  DO
    INPUTSD(2,FILELINES)
    PRINTE(FILELINES)
    IF EOF(2)<>0 THEN EXIT FI
  OD
  KEY=255
  DO
    IF KEY=33 THEN EXIT
    ELSEIF CONSOLE=3 THEN CLOSE(2)
    ERROR=ADR_MAIN RE_MAIN()
  FI
  OD
  IF EOF(2)<>0 THEN EXIT FI
  OD
  OD
  RETURN
```

```
PROC TITLE()
  BYTE S
  CARD FLASH
```

```
POKE(77,0);disable ATTRACT mode
POKE(566,143);disable break key
POKE(567,231);(new OS only)
```

```
GRAPHICS(17);GRAPHICS 1 with no window
SNDRST()
EASY_DEFAULT_VALUES()
```

FLITTERBUG

```
FLASH=4000
;DO
POSITION(0,0)
PUTDE(6)
PUTDE(6)
PRINTDE(6," *****")
PRINTDE(6," * *")
PRINTDE(6," * ----- *")
PRINTDE(6," * FLITTERBUG *")
PRINTDE(6," * ----- *")
PRINTDE(6," * source code *")
PRINTDE(6," * available *")
PRINTDE(6," * *")
PRINTDE(6," *****")
PUTDE(6)
PRINTDE(6," ACTION! COMPILED")
PUTDE(6)
PRINTDE(6," SAN LEANDRO")
PUTDE(6)
PRINTDE(6," COMPUTER CLUB")
PUTDE(6)
PRINTDE(6," JIM WARREN")
SETCOLOR(3,13,6);lower case inverse
;color
SETCOLOR(1,5,6);lower case non-inverse
;color
DO
FOR I=0 TO FLASH DO OD; replace the
;difficulty of levels with a speed
;change for sound and flash
POSITION((21-HARDLINE(0))/2,20)
;center hardline.. HARDLINE(0) is length
;of HARDLINE...21 width of screen...
PRINTDE(6,HARDLINE)
SETCOLOR(2,8,8);upper case inverse
;color
SETCOLOR(0,2,8);upper case non-inverse
;color
SOUND(0,8+3,10,3)
SOUND(1,8+115,10,5)
SOUND(2,8,10,3)
SOUND(3,8+80,3)
S=-2;used to vary sound and color
IF
CONSOLE=6 AND OPTION=3 THEN
SNDST() OPTION=0 DOCDREADER()
ELSEIF CONSOLE=6 THEN
REQUIRED=MINIMUM_EGGS
RETURN
FI
IF CONSOLE=3 THEN
OPTION==+1;counter for sequential
;step thru of options
DO UNTIL CONSOLE<>3 OD;slow down
;the console response
```

```
IF
OPTION=1 THEN
BIG_BOY_BIRTH_RATE=100
EATER_BIRTH_RATE=200
MINIMUM_EGGS=35
HARDLINE="hard"
FLASH=1000
ELSEIF
OPTION=2 THEN
BIG_BOY_BIRTH_RATE=25
EATER_BIRTH_RATE=50
MINIMUM_EGGS=50
HARDLINE="fat ChAr.Ce"
FLASH=0
ELSEIF
OPTION=3 THEN
HARDLINE="documentation"
FLASH=10000
ELSEIF
OPTION=4 THEN
EASY_DEFAULT_VALUES()
FLASH=4000
OPTION=0;reset option sequence
POSITION(0,20)
PRINTDE(6," ")
FI
FI
OD
RETURN
PROC ENDLE()
SNDST() SETCOLOR(2,0,0)
GRAPHICS(17)
POSITION(5,3)
PRINTDE(6,"extinction")
POSITION(4,6)
PRINTDE(6,"GENERATION ")
PRINTCDE(6,GENERATION+1)
PUTDE(6)
PRINTDE(6," EGG TOTAL ")
PRINTCDE(6,TOTAL_EGGS)
PUTDE(6)
PRINTDE(6," REQUIRED ")
PRINTCDE(6,REQUIRED)
PUTDE(6)
PUTDE(6)
PRINTDE(6," OPTION")
PUTDE(6)
PRINTDE(6," change difficulty")
PUTDE(6)
PUTDE(6)
PRINTDE(6," START")
PUTDE(6)
```

```
PRINTDE(6," resume same status")
REQUIRED=MINIMUM_EGGS
DO
IF CONSOLE=6 THEN
HUE=0 TIME=1500 GENERATION=0
ANNA_MATE=5 BACKGROUND=80
SETUP()
EXIT
ELSEIF CONSOLE=3 THEN
ERROR=ADR_MAIN
RE_MAIN()
FI
OD
POKE(756,RAMSET/256)HI BYTE OF RAMSET
RETURN
```

```
PROC WINNER()
SNDST() COLORS(A)
PRINT(")
```

the winner

the end

select")

```
DO
IF
CONSOLE=5 THEN ERROR=ADR_MAIN
RE_MAIN()
FI
OD
PROC COUNT_EGGS()
IF;old age sets in, count eggs & moss
TIME<=0 THEN TIME=0
TOTAL_EGGS=0
FOR I=0 TO 960
```

```
DO
SCREEN_EGG=PEEK(SCREEN+1)
IF
SCREEN_EGG=192 THEN
TOTAL_EGGS==+1
FI
OD
IF;insufficient eggs for survival
;much more than 50 is very hard
;(for me anyway!)
TOTAL_EGGS<MINIMUM_EGGS THEN
ENDLE()
ELSE;if sufficient eggs
;decrease time, go to next generation
GENERATION==+1
IF GENERATION=8 THEN WINNER() FI
EDIT()
TIME=1500-200*GENERATION
IF TIME<500 THEN TIME=500 FI
ANNA_MATE=5 ROWS=23
HUE==+1 BACKGROUND==+1 SETUP()
POKE(756,RAMSET/256)HI BYTE OF RAMSET
FI
FI
RETURN
PROC LOOK_MATE()
; CALL AND TESTS FOR EATER
MATE_BIRTH==+1
IF MATE_BIRTH>MATE_BIRTH_RATE
THEN
IF
COLUMNS=MATE_COLUMNS AND
ROWS=MATE_ROWS THEN MATE_BIRTH=0
MATE_STING()
FI
IF ANNA_MATE=3 THEN
; blank out the old MATE
;position cursor for printing new
;EATER
POSITION(MATE_COLUMNS,MATE_ROWS) PRINT(" ")
POSITION(MATE_COLUMNS,MATE_ROWS-1) PRINT(" ")
IF
MATE_COLUMNS>34 OR MATE_ROWS<2
THEN
MATE_COLUMNS=5
MATE_ROWS=RAND(20)
FI
MATE_COLUMNS==+1 MATE_ROWS==+1
MATE()
FI
MATE()
```


FLITTERBUG

```
IF MATE_BIRTH=0 THEN SNDRST() FI
FI
RETURN
```

```
PROC LOOK_EATER()
```

```
; CALL AND TESTS FOR EATER
EATER_BIRTH==+1
IF EATER_BIRTH>EATER_BIRTH_RATE
THEN
IF
COLUMNS=EATER_COLUMNS AND
ROWS=EATER_ROWS THEN EATER_BIRTH=0
STRING()
FI
IF ANNA_WATE=3 THEN
; blank out the old EATER
; position cursor for printing new
; EATER
POSITION(EATER_COLUMNS,EATER_ROWS)
PRINT(" ")
EATER_COLUMNS==+1
IF
EATER_COLUMNS>34 THEN
EATER_COLUMNS=0
EATER_ROWS=RAND(22)
FI
EATER()
FI
EATER()
IF EATER_BIRTH=0 THEN SNDRST() FI
FI
RETURN
```

```
PROC LOOK_BIG_BOY()
```

```
; CALL AND TESTS FOR BIG_BOY
BIG_BOY_BIRTH==+1
IF BIG_BOY_BIRTH>BIG_BOY_BIRTH_RATE
THEN
IF
COLUMNS=BIG_BOY_COLUMNS AND
ROWS=BIG_BOY_ROWS THEN BIG_BOY_BIRTH=0
STRING()
FI
IF ANNA_WATE=3 THEN
; blank out the old BIG_BOY
; position cursor for printing new
; BIG_BOY
POSITION(BIG_BOY_COLUMNS,BIG_BOY_ROWS)
PRINT(" ")
BIG_BOY_COLUMNS==+1
IF
```

```
BIG_BOY_COLUMNS<1 THEN
BIG_BOY_COLUMNS=34
BIG_BOY_ROWS=RAND(22)
FI
BIG_BOY()
FI
BIG_BOY()
IF BIG_BOY_BIRTH=0 THEN SNDRST() FI
FI
RETURN
```

```
PROC MAIN()
```

```
ADR_MAIN=MAIN
ADR_EDIT=EDIT
ADR_BUG=BUG
SNDRST() TITLE()
SETUP()
POINT ANTIC TO NEW CHAR SET LOCATION
(this will have to be done after
GRAPHICS() calls and...?
POKE(756,RAMSET/256)BI BYTE OF RAMSET
DO; START THE INFINITE GAME LOOP
COUNT_EGGS()
LOOK_BIG_BOY()
LOOK_EATER()
LOOK_MATE()
BUG()
OD;END GAME LOOP
RETURN
PROC POKE_A_CHAR()
LEFT SHIFTED FOR ANTIC4 DISPLAY
BYTE ARRAY
ZERO(9)=
[0 48 204 204 204 204 48 0],
ONE(9)=
[0 48 240 48 48 48 252 0],
TWO(9)=
[0 252 204 12 48 192 252 0],
THREE(9)=
[0 252 12 60 12 12 252 0],
FOUR(9)=
```

```
[0 204 204 204 252 12 12 0],
FIVE(9)=
[0 252 192 252 12 204 252 0],
SIX(9)=
[0 252 204 192 252 204 252 0],
SEVEN(9)=
[0 252 12 48 48 192 192 0],
EIGHT(9)=
[0 252 204 252 204 204 252 0],
NINE(9)=
[0 252 204 252 12 12 12 0],
SKIP : ; < = > ? @
A(9)=
[0 48 252 204 204 252 204 0],
B(9)=
[0 252 204 252 204 204 252 0],
C(9)=
[0 252 204 192 192 204 252 0],
D(9)=
[0 240 204 204 204 204 240 0],
E(9)=
[0 252 204 240 240 204 252 0],
F(9)=
[0 252 192 240 192 192 192 0],
G(9)=
[0 252 192 192 204 204 252 0],
H(9)=
[0 204 204 252 252 204 204 0],
IEYE(9)=
[0 252 48 48 48 48 252 0],
J(9)=
[0 252 48 48 48 48 240 0],
K(9)=
[0 204 240 240 252 204 204 0],
L(9)=
[0 192 192 192 192 192 252 0],
M(9)=
[0 204 252 204 204 204 204 0],
N(9)=
[0 192 252 252 204 204 204 0],
O(9)=
[0 48 204 204 204 204 48 0],
P(9)=
[0 240 204 204 240 192 192 0],
Q(9)=
[0 48 204 204 204 240 60 0],
R(9)=
[0 252 252 240 252 204 204 0],
S(9)=
[0 252 252 192 252 12 252 0],
T(9)=
[0 252 252 48 48 48 48 0],
U(9)=
```

```
[0 204 204 204 204 204 252 0],
V(9)=
[0 204 204 204 204 48 48 0],
W(9)=
[0 204 204 204 204 252 204 0],
X(9)=
[0 204 204 48 48 204 204 0],
Y(9)=
[0 204 204 60 48 240 192 0],
Z(9)=
[0 252 252 12 48 192 252 0]
```

```
FOR I=0 TO 79
```

```
DO
POKE(RAMSET+(8*16)+I,ZERO(I))
OD
```

```
FOR I=0 TO 208
```

```
DO
POKE(RAMSET+(8*97)+I,A(I)) 8*33 FOR UPPERCASE
OD
```

```
RETURN
```

```
PROC RELOCATE_CHARSET()
BYTE RAMTOP=106,RAMSET
```

```
ALLOCATE SAFE RAM
```

```
RAMTOP==5 GRAPHICS(0)
LOCATE FIRST CHARACTER
RAMSET=(RAMTOP+1)*256
COPY CHAR SET
FOR I=0 TO 1023
DO
RAMSET=PEEK(57344+I)
POKE(RAMSET+I,RAMSET)
OD
```

```
POINT ANTIC TO NEW CHAR SET LOCATION
(this will have to be done after
GRAPHICS() calls and...?
POKE(756,RAMSET/256)BI BYTE OF RAMSET
POKE_A_CHAR()
MAIN()
RETURN
```

8

```

$FF $FF $FF $FF $FF $FF
$FF $FF $FF $FF $FF $FF
$FF $FF $FF $FF
],

;The player 'POP-UP' displays

PlrLine="player",      ;Type in ".....player....."
BetLine="bankbet"     ;Type in ".bank.....bet....."
                        ;Note: The periods in the two comments above
                        ;designates the entering of 'CTL',
                        ; [the heart char.]

CARD POINTER  PLAddr   ;General pointer for direct screen changes

BYTE Chrfl=[1]        ;Character reflect status byte [set to off]

TYPE WHO=[BYTE Cnt,    ;Player and Dealer play status
          Ace          ;record description
          INT Amt]

WHO Dealer,           ;Dealer's status record
Player                ;Player's record

-----

PROC HBlnk=*( )

;Horiz. interrupt service routine.

; Routine 'FLIPS' the characters
; upside-down and rightside-up at locations
; on the screen selected by 'SetHBlk()'.

; Has to be straight code block for speed
; and possible additions later.

[ $48      ;Pha
  $AD Chrfl ;Lda Chrfl
  $49 3     ;Eor #3
  $8D Chrfl ;Sta Chrfl
  $A        ;Asl A
  $8D WSync ;Sta WSync
  $8D ChCtl ;Sta ChCtl
  $68       ;Pla
  $40       ;Rti
]
RETURN

-----

PROC SetHBlk(BYTE line)
BYTE POINTER Bp

;SetHBlk sets the 'HORZ.' interrupt in the display list

Bp=DList Bp==+line      ;Compute DL line position
Bp==+4 Bp^==X$80        ;Set a Hoz. Interrupt

RETURN

```

BLACKJACK

RETURN

```
-----
PROC Timer(BYTE Sec)
CARD Time=19,
  D
```

; Just as it is called, do a timed delay

```
D=Sec*60 Time=0
WHILE tick+256*tock<D      ;X Seconds or D jiffies delay
DO
  ;
OD
RETURN
```

```
-----
PROC ClrTable()
BYTE c
BYTE POINTER Bp
```

;This procedure clears the table, the hold card array
;and both the dealer's & player status

```
FOR c=0 TO 3
DO
  PMClear(c)
OD
Bp=SavMsc+21 Zero(Bp,160)
Bp==+240 Zero(Bp,160)
Zero(HoldCrdr,32)
Zero(Dealer,4)
Zero(Player,4)
RETURN
```

```
-----
INT FUNC CrdAmt(BYTE Who,Crd)
WHO POINTER Crdr
```

;This Function updates either the player's
;or the dealer's play status and returns their card count

```
IF Who THEN      ;whose play?
  Crdr=Dealer
ELSE
  Crdr=Player
FI
```

```
IF Crd=1 THEN      ;Test for Aces
  Crdr.Ace==+1
  Crdr.Amt==+11
ELSEIF Crd=0 OR Crd=10 THEN
  Crdr.Amt==+10      ;Got a ten or a face card
ELSE
  Crdr.Amt==+Crd
FI
```

```
IF Crdr.Amt>21 THEN      ;Test for over 21
```

```
IF Crdr.Ace THEN      ;over, got any aces?
  Crdr.Ace==+1      ;yes, deduct usable '1' count
  Crdr.Amt==+10      ;decrease count
ELSE
  Crdr.Amt==+1      ;OPPS.. send BUST
FI
RETURN(Crdr.Amt)
```

```
-----
BYTE FUNC CrdClr(BYTE Value,Crd)
```

;This function determines the card
;color and returns the offsetted value

```
Crd==&1
IF Crd THEN
  Value==+$90      ;Odds are black
ELSE
  Value==+$10      ;Evens are red
FI
```

```
RETURN(Value)
```

```
-----
PROC CrdFlip(BYTE Who)
CARD X,Y
```

; CrdFlip puts the card face display on to
; the screen. The position is determined
; by whose play it is and the number of card
; they presently have.

```
IF Who THEN
  X=(Dealer.Cnt&3)*40+52
  Y=(Dealer.Cnt&$FC)+27
  Who=Dealer.Cnt
ELSE
  X=(Player.Cnt&3)*40+52
  Y=(Player.Cnt&$FC)+79
  Who=Player.Cnt
```

```
FI
Who==&3      ;Select a P/M reg.
```

```
PMCreate(Who,CrdBrdr,22,4,X,Y)
RETURN
```

```
-----
CARD FUNC CardPos(BYTE Who)
```

```
CARD X,Y
CARD ARRAY XPos=[6 11 16 21],
  YPos=[2 3 4]
```

; CardPos calculates the position to display
; the card value and suit. The display address
; is returned to the caller.

BLACKJACK

62

```

-----
PROC Wait()
WHILE Peek(764)=255
DO
    ;wait for a key stroke
OD
Poke(764,255)
RETURN

```

```

-----
PROC Rotate(BYTE speed)
BYTE Colour=$D019,
    VCnt=$D40B,
    c,w,v,nc=[0]
CARD Time=19

```

```

;This routine is ALMOST the color shifting
;color procedure found in the ACTION! manual
;called 'scrollcolors()', it has the addition of
;a timer and the ability to vary the scroll speed.

```

```

Time=0
DO

FOR w=0 TO speed
DO
    c=nc
    DO
        WSync=0
        Colour=c
        c==+1
        UNTIL VCnt&128
    OD
    nc==+1
    UNTIL tick+256*tock)300
OD
RETURN

```

```

-----
PROC Initialize()
CHAR c
BYTE ARRAY Revr=[5 10 21 25]
BYTE POINTER Bp,Bp1
CARD POINTER Cp

```

```

;Line to set Horz. Intrn.

```

```

Graphics(1+16)
BkGrnd=$86 Color(0)=$42 Color(1)=$86
Color(2)=0 Color(3)=$D6

```

```

;set background and
;playfield colors

```

```

Bp=DList Bp==+2
FOR c=0 TO 2
DO
    Bp^=$70
    Bp==+1

```

```

;Position pointer for D/L mod.
;Move D/L down 2 Bytes

```

```

;Twentyfour blank lines

```

```

OD
DList==+2
Bp=DList Bp==+3
Bp^=$47
Bp==+3 Bp^=6
Bp==+1 Bp^=6
Cp=DList Cp==+4

```

```

;Reset Display List location

```

```

;Gr.2 W/ LMS
;Switch back to Gr.1

```

```

Cp^=SavMsc

```

```

Bp==+10 Cp=Bp+1 PLAddr=Cp
Bp^=$67 Cp^=PlrLine+1 ;Gr.2 W/ LMS & Vert. Scroll
Bp==+3 Cp==+3
Bp^=$46 Cp^=SavMsc Cp^==+220 ;Go back to Gr.1
Bp==+15 Cp==+15 Bp^=$41 Cp^=DList ;Close off DList

```

```

VdsLst=HBlnk ;set Horz. Interrupt vector

```

```

;The following routine waits for a vertical blank
;interrupt, then will enable the horizontal
;interrupt service routine.

```

```

;Though it was written as a code block for clarity,
;it could have been written in 'ACTION'.

```

```

[
    $AD $F $D4 ;Wait Lda NMIST
    $29 $40 ; And #$40
    $F0 $F9 ; Beq Wait
    $AD $E $D4 ; Lda NMIEH
    9 $80 ; Ora #$80
    $8D $E $D4 ; Sta NMIEH
]

```

```

OldMemHi=MemHi MemHi==-$D00 ;Compute location for new Char. set
NwChar=MemHi& $FF00

```

```

MoveBlock(NwChar,$E000,$800) ;Move down the character set

```

```

MoveBlock(NwChar+40,$E000+512,8) ;Move specific characters to
MoveBlock(NwChar+48,$E000+640,8) ;to different location
MoveBlock(NwChar+56,$E000+768,8)
MoveBlock(NwChar+64,$E000+984,8)

```

```

ChBase=NwChar RSH 8 ;Reset CHR address
PMGraphics(2) GPrior=$14
FOR c=0 TO 3
DO

```

```

    PMClear(c)
    PMColor(c,0,10)
    SetHBlk(Revr(c))

```

```

OD
FOR c=4 TO 7
DO

```

```

    PMClear(c)
    PMCreate(c,CrdBck,21,4,0,0)
OD

```

```

PrintDE(6," slcc") ;Inverse SLCC
PrintDE(6," BLACKJACK") ;Inverse BLACK std JACK
Rotate(3)

```

```

Close(1) Open(1,"K:",4)

```


BLACKJACK

```

IF Who THEN
  Y=YPos(Dealer.Cnt RSH 2)*20      ;Calc. Dealer Y
  Who=Dealer.Cnt
  Dealer.Cnt==+1
ELSE
  Y=YPos(Player.Cnt RSH 2)*20+220  ;Calc. player Y
  Who=Player.Cnt
  Player.Cnt==+1
FI

Who==&3
X=XPos(Who)                        ;Select a X position
X==+SavMsc+Y                       ;Calc. address offset
RETURN(X)
-----
BYTE FUNC DplCard(BYTE card,Who)
  BYTE i,
        Crd,Value
  BYTE ARRAY SUIT={HEART CLUB DIAMOND SPADE},
        Face={$1A $21}
  BYTE POINTER Bp,Bp1

  ; DplCard brings all the other display support
  ; routines together. This does the P/M calls
  ; and does the actual displaying of the
  ; card value and suit

  Bp=CardPos(Who)                  ;Get display address
  Bp1=Bp+60                        ;find bottom of old card
  CrdFlip(Who)                     ;Put up card face

  FOR i=0 TO 2                      ;Clear bottom of card
  DO
    Bp1^=0
    Bp1==+1
  OD

  Bp1==+18                          ;Find new card bottom

  Value=card MOD 13 i=Value         ;Get card suit
  card==/13                         ;Calc. card value
  IF Value THEN
    IF Value=10 THEN
      Crd=CrdClr(1,card)
      Bp1==+1 Bp1^=Crd Bp1==--2
      Bp^=Crd Bp==+1 Value=0
    ELSEIF Value=1 THEN
      Value=$11
    ELSEIF Value=10 THEN
      Value=Face(Value-11)
    FI
  ELSE
    Value=$1B
  FI

  Crd=CrdClr(Value,card)           ;get color

```

```

Bp^=Crd Bp==+1 Bp^=SUIT(card)      ;DISPLAY TOP OF CARD
Bp1^=SUIT(card) Bp1==+1 Bp1^=Crd   ;DISPLAY BOTTOM OF CARD
RETURN(i)
D:BLKJACK.002
-----
PROC Shuffle()
  BYTE c,i

  ; 'Shuffle' reshuffles the deck ie. clears the
  ; deck array. If a reshuffle occurs during
  ; a game, the cards still in play are removed.
  ; The 'SHUFFLING' prompt is also displayed

  SetBlock(Deck,52,1)              ;Clear deck
  c=0
  Delt=0                            ;Reset count

  WHILE HoldCrd(c)                  ;'PULL' cards still in play
  DO
    i=HoldCrd(c) Deck(i)=0
    Delt==+1
    c==+1
  OD

  RowCrs=0
  ColCrs=5 PrintD(6,"shuffling")    ;Inverse shuffling
  Rotate(7)
  ColCrs=5 PrintD(6," DEALER ")    ;Inverse DEALER
  RETURN

  -----
  BYTE FUNC Deal()
  BYTE c

  ; This function returns a card from the deck.
  ; It also tests to see if a reshuffle is required.
  ; Deal also tests to make sure that a card is not sent
  ; more than once.

  IF (Delt&$3F)>51 THEN              ;Time to shuffle?
    Shuffle()                        ;yes!
    Delt==*$80
  FI

  DO
    c=Rand(52)                      ;'PULL' a card off the deck
    UNTIL Deck(c)                   ;Keep doing 'til find one not sent
  OD

  Deck(c)=0                          ;Set "Pulled" Card
  Delt==+1                          ;Increase played count
  RETURN(c)                          ;Return card #

  -----
  PROC Delay()
  CARD Slow

  ;Wait a constant length of time

```

BLACKJACK

64

```
;(approximately 2 jiffies)
```

```
FOR Slow=0 TO 1500
DO
  |
OD
RETURN
```

```
PROC Results(BYTE Speed,Locate)
```

```
; Display the results of the play (ie.
; win,lost,bust or push) and rotate its color
```

```
Rotate(Speed)
ColCrs=Locate
PrintD(6,"      ")
RETURN
```

```
PROC Busted()
```

```
; Display 'BUST' for either
```

```
ColCrs=8
PrintD(6,"bust")      ;inverse 'bust'
Results(3,8)
```

```
RETURN
```

```
PROC Flip()
```

```
BYTE c
```

```
; 'FLIP OVER' Dealers card
```

```
FOR c=4 TO 7
```

```
DO
  PMMove(c,0,0)      ;Move missiles (back of card) off screen
OD
```

```
RETURN
```

```
INT FUNC GetCrD(BYTE w)
```

```
BYTE c
```

```
INT H
```

```
; GetCrD will get a card from the dealer
; and display it. It will also place the card
; into the 'IN PLAY' array and have the
; player/dealer status updated.
```

```
c=Deal()
HoldCrD(HldCnt)=c
c=DplCard(c,w)
H=CrDAmT(w,c)
HldCnt==+1
```

```
RETURN(H)
```

```
INT FUNC DlrHand()
```

```
BYTE c
```

```
INT Hand
```

```
WHO POINTER dealer
```

```
; This is the dealer logic routine.
; It goes by the following rules:
; If the dealer has 16 or less, or if
; it has a soft 17, it must take a card
```

```
dealer=Dealer
Hand=dealer.Amt
```

```
DO
```

```
Flip()                                ;Turn the hole card over
IF (Hand)0 AND Hand<17)               ;Test for take card or not
  OR
  (Hand=17 AND dealer.Ace)0)
  THEN
    Timer(1)
    Hand=GetCrD(1)                     ;Get a card
    Timer(1)
    IF Hand<0 THEN
      EXIT                             ;Oops dealer busted!!
    FI
  ELSE
    EXIT                               ;Got final hand
  FI
; Wait()
OD
RowCrs=1
RETURN(Hand)                           ;Return Final hand
```

```
INT FUNC PlrHand()
```

```
BYTE c
```

```
INT Hand
```

```
WHO POINTER player
```

```
; This is the player prompt handler
; It waits for a key to be pressed and
; test legitimacy of the response.
```

```
player=Player
Hand=player.Amt
```

```
DO
```

```
;Wait for key
RowCrs=12 ColCrs=0
IF (Hand=10 OR Hand=11) AND player.Cnt=2 THEN
  PrintD(6,"STAND DOUBLEDOWN HIT") ;inverse 'STAND' normal 'DOUBLEDOWN
inverse 'HIT'
ELSE
  PrintD(6,"      HITSTAND")        ;inverse 'HIT' normal '' inverse 'STAND'
FI
```

BLACKJACK

```

DO
  UNTIL CH#255
OD
c=CH CH=255

c==&$3F

IF c=$3E THEN
  EXIT
FI

IF c=$39 THEN
  Hand=GetCrn(0)
  IF Hand<0 THEN
    EXIT
  FI
FI

IF c=$3A AND (Hand=10 OR Hand=11)
  AND player.Cnt=2 THEN
  PlrBet==LSH 1
  Hand=GetCrn(0)
  Timer(1)
  EXIT
FI

IF c=$1C THEN
  Hand=0
  EXIT
FI
OD
ColCrn=0 RowCrn=12
PrintD(6,"")
ColCrn=0 RowCrn=12
RETURN(Hand)

```

```

PROC Scroll(BYTE dir)
  BYTE d

```

```

; 'Scroll' scrolls the two 'POP-UP' prompts

IF dir THEN
  FOR d=0 TO 15
    DO
      VScrol=d
      Delay()
    OD
  ELSE
    FOR d=0 TO 15
      DO
        VScrol=15-d
        Delay()
      OD
    FI
  RETURN

```

```

PROC GetBet()
  BYTE c,i,j
  BYTE ARRAY BnkLn(10)

```

```

; GetBet handles the entire wager process
; From clearing out the old bet, through getting
; and offset the present bank display to
; to getting the input, converting from ASCII
; and test the input.

```

```

DO
  ;Clear & set offsets
  j=1
  Zero(BetLine+7,5) Zero(BetLine+15,5)
  StrI(PlrBank,BnkLn)
  FOR c=1 TO BnkLn(0)
    DO
      BnkLn(c)=+&$60
    OD
  SAssign(BetLine,BnkLn,7,BnkLn(0)+7)

```

```

;Do 'POP-UP's
Scroll(1) PLAddr^=BetLine+1 Scroll(0)

```

```

;Get input and do conversion

```

```

i=0
c=GetD(1)
WHILE c#155
  DO
    IF c=$1B THEN
      PlrBet=-1
      RETURN
    ELSEIF c='0 AND c='9 THEN
      i==+1
      BnkLn(i)=c BnkLn(0)=i
      BetLine(i+15)=c+&$60
    ELSEIF c=127 THEN
      i=-1
      IF i>10 THEN i=0 FI
      BetLine(i+16)=0 BnkLn(0)=i
    FI
    c=GetD(1)
  OD
  Scroll(1) PLAddr^=PlrLine+1 Scroll(0)
  ;Test input
  PlrBet=ValI(BnkLn)
  UNTIL PlrBet>1
OD
RETURN

```

```

BYTE FUNC Insur()
  BYTE i
  CARD Time=19

```

BLACKJACK

99

```
; If the dealer is showing an ace
; better hit a key quick!!

Time=0 i=0
Position(6,11) PrintD(6,"insurance? ")
WHILE tock(1)
    ;Only got 4 seconds
DO
    IF CH#255 THEN
        PrintD(6,"YES") i=1
        Timer(2) EXIT
    FI
OD
Position(6,11) PrintD(6," ")
RETURN(i)
-----
PROC Play()
BYTE c,m,i
INT Hand,DlrHnd,PlrHnd,
    X,Y
WHO POINTER player,
    dealer
player=Player
dealer=Dealer

; Main game logic... all others are ultimately
; called from here.

;Start the game
DO
    ClrTable()
    IF Delt>40 THEN
        Shuffle()
    FI
    GetBet()
    ;player wants to quit!!
    IF PlrBet<0 THEN
        RETURN
    FI
    ;Place cover over dealer hole card
    FOR m=4 TO 7
    DO
        PMMove(m,117+m*5,28)
    OD
    ;Get the first 4 cards
    FOR HldCnt=0 TO 3
    DO
        c=Deal()
        HoldCrD(HldCnt)=c
        c=DplCard(c,HldCnt&1)
        CrdAmt(HldCnt&1,c)
    OD

    ;Test for, and process 'ACE SHOWING'
```

```
i=0
IF (HoldCrD(1) MOD 13)=1 THEN
    i=Insur()
    IF dealer.Amt#21 AND i=1 THEN
        Position(3,11) PrintD(6,"took insurance")
        Timer(3)
        Position(3,11) PrintD(6," ")
        PlrBank==-(PlrBet RSH 1)
    FI
FI

;Go into standard play
DO
    IF dealer.Amt<21 AND player.Amt<21 THEN
        Hand=PlrHand()
        IF Hand=0 THEN
            RETURN
        FI
        IF Hand<0 THEN
            Flip()
            Busted()
            PlrBank==PlrBet
            EXIT
        FI
        Hand=DlrHand()
        IF Hand<0 THEN
            Busted()
            PlrBank==+PlrBet
            EXIT
        FI
    ELSE
        Flip()
    FI

    ;Player got a BLACKJACK
    IF player.Amt=21 AND player.Cnt=2 THEN
        PlrBet==LSH 1
    FI

    RowCrs=12
    ColCrs=7
    ;player wins!
    IF player.Amt>dealer.Amt THEN
        PrintD(6,"winner")
        PlrBank==+PlrBet
    ELSEIF player.Amt<dealer.Amt THEN
        ;Dealer BLACKJACK with insurance
        IF dealer.Cnt=2 AND dealer.Amt=21 AND i=1 THEN
            PrintD(6,"no bet")
            ;inverse 'no bet'
        ELSE
            ;Player loses!
            PrintD(6,"lost ")
            ;inverse 'lost'
            PlrBank==PlrBet
        FI
    FI
```


BLACKJACK

```

FI
ELSE
    ; Tie
    PrintD(6, " push ")          ; inverse 'push'
FI
Results(3,7)
EXIT
OD
Position(3,11) PrintD(6, "      ")
Timer(1)
RETURN
-----
PROC Start()
BYTE b,c,r
BYTE POINTER Bp

Initialize()
Delt=100                                ; Start Game Clean

Position(0,0)
PrintDE(6, " BY FRANK DANIEL ")          ; Inverse FRANK DANIEL
PrintDE(6, " [C] 1984")
Timer(4)
PutD(6,125)
Color(1)=*44
Play()

; End of game reset memory and screen

Graphics(0)
MemHi=OldMemHi
RETURN

```

```

; FILTER
;
; BUG
THE ACTION RE_ACTION RECURSION &
RANDOM PROCEDURE CALLING METHOD
JIM WARREN
SAL LEANDRO COMPUTER CLUB
PURPOSE: to allow any procedure to
call itself or any other procedure
irrespective of it's order in the
>P>1")
IF
; If console=OPTION then set the
; RE_CURSE flag and call yourself
CONSOLE=3 THEN RE_CURSE=1 ONE("1")
ELSEIF
; If console=SELEC>RET>IS P>PRO>2")
IF
CONSOLE=3 THEN ERROR=ADR_ONE
RE_ONE("2") REPEAT TWO("R")
ELSEIF
CONSOLE=5 THEN REPEAT TWO("2")
>3")
IF
CONSOLE=3 THEN ERROR=ADR_ONE
RE_ONE("3") REPEAT THREE("R")
ELSEIF
CONSOLE=5 THEN ERROR=ADR_TWO
RE_TWO("3") REPEAT THREE("R")
ELSEIF
CONSOLE=6 THEN REPEAT
THREE("3")
ELSEIF
KEY=12 THEN ERROR=ADR_DING
RE_DING() RETURN
FI
OD
RETURN;

PROC DING()
CARD I
RE_ERROR
SNDST()
FOR I=0 TO 5000
DO SOUND(0,25,10,8) OD
SNDST()
KEY=255
RETURN;

PROC SETUP()
All of the procedures above this one
except the RE_XXX procedures can be
shifted around in any order. The
RE_XXX procedures must be at the
top of the list and the following
address assignments must come at the
very last (so that they will be the
first statements executed by the
program).
ADR_SYS_ERROR=ERROR
ADR_ONE=ONE
ADR_TWO=TWO
ADR_THREE=THREE
ADR_DING=DING
PRINT("")
POKE(752,1) SETCOLOR(1,0,0)
SETCOLOR(2,13,6)
ONE("S")
GRAPHICS(0)
PRINT("
RETURN; THRU PROC SETUP
TO THE MONITOR")
PRINT("
THE ACTION RE_ACTION RECURSION &
RANDOM PROCEDURE CALLING METHOD
JIM WARREN
SAL LEANDRO COMPUTER CLUB
PURPOSE: to allow any procedure to
call itself or any other procedure "
PRINT("
irrespective of it's order in the
list while preserving the proper
order of RETURNS and with full
parameter passing capability. "
RETURN;

```

RAMTALKER

```

1 REM
2 REM |   SAN LEANDRO COMPUTER CLUB |
3 REM |   SPECIAL EDITION JOURNAL |
4 REM | P.O. Box 1525, San Leandro, CA |
5 REM |   94580-0152 |
6 REM
10 REM RAMTALKER: USER'S MANUAL 11/84
    A.R. Holmes (C) 1985 - Norfolk, VA
    for the STATUS Newsletter
20 GRAPHICS 0:POKE 752,1:?:?:?:? "I
initializing...Please wait."
30 FOR I=0 TO 243:READ Z:POKE 1536+I,Z
:NEXT I
40 GOTO 50
50 DIM Z(255),FNS(13):OPEN #1,4,0,"K:"
60 GRAPHICS 2:SETCOLOR 2,0,0:TRAP 60
70 ? #6;" RAMTALKER":? #6
80 ? #6;" 1 record":? #6;" 2 Playback"
:?:#6;" 3 throughput"
90 ? #6;" 4 save":? #6;" 5 load"
100 ? #6;" 6 waveform graph"
110 TRAP 110:GET #1,ANS:IF ANS>54 OR A
NS<49 THEN 110
120 IF ANS>51 THEN 140
130 TRAP 60:POKE 752,1:?: "What Sample
Speed";:INPUT 55:IF 55>255 THEN 130
140 ON VAL(CHR$(ANS)) GOTO 160,200,240
,270,330,640
150 REM INIT
160 POKE 208,1:POKE 205,0:POKE 206,64:
POKE 207,55:POKE 209,128
170 A=USR(1536):POKE 562,3:POKE 53775,
3
180 GOTO 60
190 REM INIT
200 POKE 207,55:POKE 203,0:POKE 204,64
:POKE 208,0:POKE 206,128
210 A=USR(1536):POKE 562,3:POKE 53775,
3
220 GOTO 60
230 REM INIT
240 POKE 208,2:POKE 205,0:POKE 206,64:
POKE 207,55:POKE 209,128
250 A=USR(1536):GOTO 240
260 REM STATUS SOUND
270 TRAP 270:POKE 752,1:?: "Give file n
ame";:INPUT FNS:IF FNS="" THEN 60
280 IO=4:OPEN #4,8,0,FNS
290 ADDRESS=16384:NUMBER=16383:PROC=11
300 GOSUB 510
310 GOTO 60
320 REM LOAD SOUND
330 TRAP 330:POKE 752,1:?: "Give file n
ame";:INPUT FNS:IF FNS="" THEN 60
340 IO=4:OPEN #4,4,0,FNS
350 ADDRESS=16384:NUMBER=16383:PROC=7
360 GOSUB 510
370 GOTO 60
380 DATA 104,169,0,141,31,208,173,31,2
08,41,1,208,249,160,255,162,255,32,149
,6
390 DATA 136,208,248,169,8,141,31,208,
166,208,224,0,208,3,76,181,6,169,0,141
400 DATA 0,212,141,10,212,141,10,212,1
41,10,212,166,207,32,149,6,173,4,210,1
62
410 DATA 19,142,15,210,162,23,142,10,2
12,142,15,210,142,11,210,174,243,6,224
,0
420 DATA 208,22,41,240,141,242,6,106,1
06,106,106,41,15,9,16,141,1,210,238,24
3
430 DATA 6,76,45,6,106,106,106,106,41,
15,9,16,141,1,210,41,15,13,242,6
440 DATA 206,243,6,160,0,145,205,173,3
1,208,41,1,240,19,130,205,208,163,230,
206
450 DATA 166,206,228,209,208,155,76,15
3,6,202,208,253,96,165,208,201,2,208,1
1,169
460 DATA 0,133,205,169,64,133,206,76,3
7,6,169,64,141,14,212,169,34,141,0,212
470 DATA 96,169,0,141,14,212,141,0,212
,166,207,32,149,6,160,0,177,203,170,10
6
480 DATA 106,106,106,41,15,9,16,141,1,
210,138,41,15,9,16,24,24,24,24,166

```

```

490 DATA 207,32,149,6,141,1,210,230,20
3,208,206,230,204,166,204,228,206,208,
206,76
500 DATA 153,6,0,0
510 REM PRO-READY
520 IO=16*IO
530 IOCB=832+IO:POKE IOCB+2,PROC
540 ADRI=INT(ADDRESS/256)
550 ADRL0=ADDRESS-ADRI*256
560 POKE IOCB+4,ADRL0:POKE IOCB+5,ADRI
570 NUMHI=INT(NUMBER/256)
580 NUMLO=NUMBER-256*NUMHI
590 POKE IOCB+8,NUMLO:POKE IOCB+9,NUMHI
600 I=USR(ADR("hhhLUM"),IO)
610 CLOSE #IO/16
620 RETURN
630 REM QUIT
640 GRAPHICS 8:SETCOLOR 2,0,0:COLOR 1:
POKE 752,1:?: "During plot. press any
key to return to main menu"
650 FOR I=1 TO 600:NEXT I
660 GRAPHICS 24:SETCOLOR 2,0,0:COLOR 1
:A=0:B=0:C=0:D=0:H=0
670 FOR I=1 TO 4095:POKE 764,255
680 A=PEEK(I+16384)
690 B=PEEK(I+20480)
700 C=PEEK(I+24575)
710 D=PEEK(I+28670)
720 H=H+0.07773:PLOT H,(A/6)-5
740 PLOT H,(C/6)+35
750 PLOT H,(C/6)+80
760 PLOT H,(D/6)+120
770 IF PEEK(764)=255 THEN NEXT I
780 TRAP 820:IF PEEK(764)=255 THEN GRA
PHICS 40:?: "TO DUMP TO PRINTER. PRESS
":?: "FOR MENU. PRESS "
790 IF PEEK(764)=10 THEN RUN "D:PRINT.
DMP"
800 IF PEEK(764)=37 THEN 60
810 GOTO 790
820 ? "ARE PRINTER AND INTERFACE ON?":
FOR T=1 TO 500:NEXT T:GOTO 780

```

Second Annual, World's Only Home Computing Fair

PRESS RELEASE (For immediate publication)

Home Computing Centers, Inc.
296 Bay Fair Mall 115 Tanforan Park
San Leandro San Bruno

San Francisco, July 15, 1985

The 2nd annual "Worlds Only Home Computing Fair" was announced today by Home Computing Centers, Inc., a retail specialty chain headquartered in San Bruno, California. When asked what distinguished this fair from the many others that have appeared in recent years, company spokesman, Dan Williams explained, "Well, the name of the fair tells most of the story. We have intentionally set out to create a 'non-techie' exhibition. There are an enormous number of people out there who either own or are thinking about buying a home oriented computer but who aren't in the least interested in hearing about bits, bytes and baud rates. **They are really after a productive home appliance.** They are waiting for someone to show them what you actually **do** with a home computer besides play games and balance your checkbook. That's the thrust of the fair."

"A full spectrum of useful applications will be shown. We'll have a real cross-section of exhibitors ranging from computer makers to software houses, peripheral manufacturers and so on. There will be something for everybody. The new Atari and Commodore computers will be shown. People will get to see demonstrations and state of the art applications that are normally reserved for insiders at trade shows. It's all very exciting."

The fair will be held over the weekend of September 13-14-15 at Tanforan Park Shopping Center in San Bruno, California. Tanforan Park is a major mall located just south of San Francisco. Several dozen manufacturers, distributors and user groups have been scheduled to attend. For more information, interested parties should call Dan Williams at (415) 278-8881 or write to him in care of Home Computing Centers, Inc., 296 Bay Fair Mall, San Leandro, CA 94578. Alternately, you may call Lew Moore at (415) 588-1201 or write to him in care of Home Computing Centers, Inc., 115 Tanforan Park, San Bruno, CA 94066.

September 13TH thru 15TH

TANFORAN PARK SHOPPING CENTER • SAN BRUNO, CA

BASIC XE™ Gives Your Atari 130XE™ All The Performance It Should Have Had In The First Place



In the home computer races, the Atari 130XE stands out as a price leader. But using underpowered Atari BASIC™ on this otherwise fine machine is like racing in the Indy 500 with half your cylinders missing. So don't get left at the starting line with only half an "engine." Change to the performance leader **now!** Buy BASIC XE from OSS, the **only** programming language designed especially for the Atari 130XE.

Just look at what you get for one low sticker price:

BEST MILEAGE: With over 60,000 **more** bytes for your programs, BASIC XE lets you use all the memory you paid for.*

MORE HORSEPOWER: Run Atari BASIC programs 2 to 6 times faster.* Even with its incredible power, BASIC XE is compatible with Atari BASIC.

BETTER HANDLING: With auto line numbering, renumbering, program cross referencing, English error messages, and more.

CLASSIC DESIGN: Show off the sleek **structured** style of your own programs when you use BASIC XE statements like PROCEDURE, IF...ELSE, and WHILE...ENDWHILE.

FREE ACCESSORIES: Get over \$100 worth of Atari BASIC options **FREE** when you buy BASIC XE: complete Player/Missile Graphics support, string arrays, DOS access, SORT commands, readable listings...over 50 extras at no additional charge.

- If you're ready to step up to real performance...YOU need BASIC XE now!
- If you haven't written your first BASIC program...YOU need BASIC XE now!
- If you're already a real pro in BASIC...YOU need BASIC XE now!
- **BASIC XE may well be the best buy any Atari owner ever made.**

*Want to know more? Call or write for free brochure or ask your local dealer.
Atari 130XE™ and Atari BASIC™ are U.S. registered trademarks of Atari Corporation.



Optimized Systems Software, Inc.
12218 Kentwood Avenue, San Jose, California 95129 (408) 448-3099