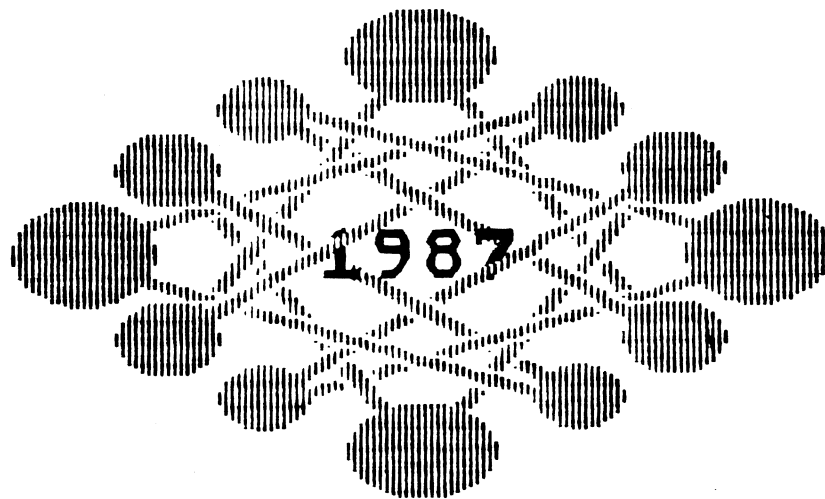


FUJ FACTS

Newsletter of the
Atari Computer Enthusiasts of Columbus

Volume 5, No 1

January, 1987



LOCKUP.PIC
from DOM #23C2

INSIDE THIS ISSUE:

The Editor's Column
December SIG Notes
ACEC Library Listing
Page Zero Pointer's
Decision in the Desert
Basic DOS Usage
MARGAME Construction Set
Beginner's DOS Tutorial

Millionaire
1050/XM301 Interfacing
Micro League Baseball
Assembly Lang. Tutorial
Eliza and SAM
Mapping the Atari
Fine Tuning DOS
ST Supplements

This newsletter is written and published monthly by the Atari Computer Enthusiasts of Columbus, Ohio (ACEC). ACEC is an independent, non-profit organization interested in exchanging information about any and all Atari Home Computer Systems.

Meetings for eight-bit users are held on the second Monday of each month at 7:15 p.m., at DeSales High School (on Karl Road, just south of Morse Rd.), and are open to the public.

Dues are \$12.00 per year, and entitle members to all club benefits (Newsletter, Disk of the Month, Publications Library, SIG meetings, group discounts at selected area merchants, etc.).

The ACEC Newsletter welcomes contributions of articles, reviews, editorials and any other material relating to the Atari computers, or compatible hardware devices and software packages.

PRESIDENT:

Charles Lusco
4624 Channing Terrace, #C
Columbus, OH 43232
863-4016

XL/XE VICE-PRESIDENT:

Dave Beck
1194 Country Club Road
Columbus, OH 43227
863-8600

XL/XE DISK LIBRARIAN:

Paul Felfrey
2199-D Cortside Drive
Columbus, OH 43232
866-1454

PUBLICATIONS LIBRARIAN:

Bill Morgens
4943 Batsford Drive
Columbus, OH 43232
861-2431

MEMBERSHIP CHAIRMAN:

Mike Compton
1342 Gumwood Drive
Columbus, OH 43229
885-3757

ACEC is not affiliated with the Atari Corporation; "ATARI" and the "Fuji" symbol are registered trademarks of the Atari Corporation. All other trademarks, copyrights and service marks are registered with their respective owners.

The statements expressed in this newsletter are solely the opinions of the authors, and do not necessarily reflect those of ACEC, its officers or its members. Material contained in this newsletter may be reprinted provided credit is given to both ACEC and the author(s).

The cover of this month's newsletter was printed with a Star SG-10 dot matrix printer, using XLent Software's TypeSetter 130. The newsletter itself was also printed with a Star SG-10 printer in elite pitch, using Batteries Included's PaperClip version 1.2K on a 256K RAM modified Atari 800 XL.

NEWSLETTER EDITOR:

Warren Lieuallen
1652 Hess Boulevard
Columbus, OH 43212
488-3977

ST VICE-PRESIDENT:

Larry Mendel
211 Brookhaven Drive
Gahanna, OH 43230

ST DISK LIBRARIAN:

Scott Wade
938 South Weyant Avenue
Columbus, OH 43227
239-7285

TREASURER:

Dave Feeney
2665 Blue Rock Boulevard
Grove City, OH 43123
871-0524

SECRETARY:

Don Bowlin
230 Orchard Lane
Columbus, OH 43214
262-6945



The Editor's Column

ACEC Newsletter Name Suggestions
From the Members (this means you!)

Here, as promised, is the complete list of new names I received for our newsletter, presented in no particular order:

<u>From:</u>	<u>Name:</u>
Gene Strojny	The ACE Line
Doug Luce	Electronique
"The Emperor"	The Knowledge Base
Andy Weaver	Atari Computer
Scott Schumaker	Enthusiasts Chronicle
Mike Carney	ACE in the Hole
Jim Malaby	Heart of the ACE's
	ACE Base
	ACE News
	ACE HotLine
	ACE Byte
	ACE Sector

And finally, a list I received in the mail anonymously (although it accompanied an article which I assume was written by Kenny Shrif):

FUJI FACTS; Upper c.A.C.E.; 8 Bits; Bits, Bytes and Bauds; Folded, Spindled and Mutilated; Analog Anagenesis; Beware, ACE Bytes!; Computerese; Computerize; Hard Wire; The Assembly; This Month; The Monthly Assembly; Loose Bits Link Chips; The ACE of Spays; Warren's Warblings; Doctor's Digital Delirium; Lieuallen's Loquacious Laments; News and Views; The News; The Columbus Regional Atari Publication; DOS Dross; I/O; Hip Chips Quips; and The New and Improved Newsletter of the Atari Computer Enthusiasts of Columbus

Needless to say, I was overwhelmed! The quality, and the quantity, took me quite by surprise. However, as you can

see from the cover, the selection has been made. This selection was made very informally, and is therefore subject to change at my slightest whim! My sincere thanks to all who offered suggestions, my deepest apologies to those whose suggestions I did not select, and eternal raspberries to those who didn't help out at all (and you know who you are)! If anyone would like to claim ownership of the winning entry, I will see to it that your name is spread all over the next issue, and that your "prize" is delivered to you.

As you will soon see from reading this issue, another name which is very appropriate, but was not suggested is "The Charles W. Brown Newsletter". This month we have a total of FIVE articles from Charles. And as you know, we haven't published a newsletter without at least one article from Charles in the recorded history of mankind (I think it's in our Constitution, or something)! What's the story with the rest of you? You all have computers; you all have plenty to talk about at the meetings (and often too much!); you all have word processors, or can get one for free on our Disks of the Month. Why aren't I getting five articles from all of our almost 130 members each month? It's a mystery to me!

Perhaps I exaggerate a bit (my editorial license is showing!). However, I would like to once again (surely no one is surprised!) emphasize that I am always in need of material for YOUR newsletter. I am the editor, not the author. Put me to work -- give me something to edit!

To all of those who did contribute this month, I thank you. I had the pleasant chore of trimming material to fit, rather than trying to think up some more stuff! 'Nuff said!

Finally, as I mentioned once before several months ago, I would very much like to put together an archive of all the back-issues of The ACEC Newsletter

(no matter what its name!). I personally have all of 1985 and 1986. From 1984, I need January through May, and September (if there was an issue that month). I also need the first three issues, published in 1983. I'm hoping some of our long-time members can help me out here. Please contact either myself, or Bill Morgens

I'm also hoping to have a transcript of the history of our group (presented by Bill Eckert at the XL/XE meeting a few months ago). While it probably won't become a best-seller, it is interesting to trace our "roots".


If anyone would be interested in applying for the position of Advertising Manager, the job is yours. For the past few months, I have supposed to have been soliciting paid advertisements from local retailers. We have a special package from Batteries Included which makes this especially attractive. However, in all honesty, I have yet to do much with all this, and am beginning to seriously doubt that I ever will. So, if anyone would like to help the club out a great deal, please let me know. The job will require very little time and effort (a few phone calls), and should produce some very nice results (money, basically!). Please, consider it.

Warren Lieuallen



**SIG
Notes**

I'm writing this column just to remind you that unless I get a volunteer to write them in the future, this will be the last installment of "SIG Notes". I no longer attend the XL/XE SIG's on a regular basis, and the ST SIG has been elevated to the status of a regular meeting, so....



HEY, WAKE UP!

**YOU WON'T WANT TO
MISS THE NEW AND
IMPROVED
ACEC BBS!**

**268-0405
GIVE US A CALL!**

The ACEC SIG's met on December 18th, at the Grandview Heights Public Library. New SIG's will be forming (ACTION!, Logo [?], etc.), and will be announced at the main XL/XE meetings. All interested parties are invited to attend.

For the sixteen-bit material, watch for the ST Newsletter Supplements, kindly provided by Nassir Amra. I will continue to supplement this Supplement with general information pertaining to the ST as I see fit.

Warren Lieuallen



Librarian's Report

A.C.E.C. PUBLICATIONS LIBRARY

COMPLETE LISTING

January 12, 1987

by: Bill Morgens

Many thanks to those who donated publications at the recent meetings (Oh! My achin' back!).

These publications are available to be checked out by members only at no charge. We request that library items be returned at the meeting following check out so that others may share. This listing will be updated periodically.

The A.C.E.C. Publications Library eagerly solicits your donations of no longer wanted magazines and books. Also wanted are photocopies of Atari related articles of general interest appearing in other magazines such as Computer Shopper.

ANTIC Magazine

1982 - Apr, Jun, Aug, Oct/Nov, Dec/Jan83
 1983 - Feb/Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
 1984 - Jan, Feb, Mar, Apr, Jun, Jul, Aug, Sep, Oct, Nov, Dec
 1985 - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
 1986 - Jun, Apr, May

ANALOG MAGAZINE

1983 - #10 Feb/Mar, #11 Apr/May, #12 Jul/Aug, #13 Sep/Oct, #14 Nov/Dec
 1984 - #15 Jan, #16 Feb, #17 Mar, #18 Apr, #19 Jun, #20 Jul, #21 Aug, #22 Sep, #23 Oct, #24 Nov, #25 Dec
 1985 - #27 Feb, #28 Mar, #29 Apr
 1986 - #45 Aug, #46 Sep, #47 Oct, #49 Dec

ATARI EXPLORER - Feb 1985, April/May 1985

ATARI PROGRAM EXCHANGE - Catalog 1982

DR. DOBBS' JOURNAL - June 1985

CREATIVE COMPUTING - Aug 82, Sep 82, Oct 82, Nov 82, Dec 82, Feb 83, Mar 83, Jun 83, Dec 83, Jan 84, Feb 84, Sep 84, Jan 85, May 85

COMPUTER SHOPPER - Apr 86

BYTE - Sep 81, Feb 84, Jun 84, Oct 84, Feb 86, Jul 86, Aug 86, Sep 86, Oct 86, Nov 86, Dec 86

COMPUTE! MAGAZINE

1983 - Aug, Sep, Oct, Nov, Dec
 1984 - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
 1985 - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
 1986 - Jan, Feb

PERSONAL COMPUTING - Oct 85

BOOKS:

STIMULATING SIMULATIONS - C. W. Engel - Twelve unique programs in BASIC

ATARI BASIC - Bob Albrecht, et al.

INSIDE ATARI BASIC - Bill Carris

ADVANCED ATARI BASIC TUTORIAL - R. A. Peck - A "SAM's" book

DE RE ATARI - A guide to effective programming, by top Atari programmers

ATARI OPERATING SYSTEM SOURCE LISTING - Atari Corp.

MOSTLY BASIC: Applications for your ATARI - Howard Berenbon - Books I & II

KIDS AND THE ATARI - Howard H. Carlson

Programming Techniques - Vol 2,

Simulation - Ed. by B. W. Liffick; Vol 3,
Numbers in Theory & Practice

The Best of Creative Computing - Vols. 1,
2, & 3 - Ed. by David Ahl

Crash Course in Microcomputers - Howard
W. Sams

The Book of Atari Software - 1983 & 1984

Atari Special Additions - A 1982 product
catalog from Atari Corp.

The Creative Atari - Ed. by David Small
et al

Some Common Basic Programs, Atari Edition
- Lon Foole et al

WQZ, The Prodigal Son of Silicon Valley -
Doug Garr - a biography of the developer
of Apple computers

Hands-On Basic for the Atari - Herbert
Peckham

Machine Language for Beginners - Richard
Mansfield

Hackerbook for your Atari computer -
H. C. Wagner

The Atari Assembler - Don & Kurt Inman

VisiCalc for Science & Engineering -
Stanley Trost and Charles Pomernacki

VisiCalc Home & Office Companion - David
M. Castlewitz et al

Controlling Financial Performance -
Dennis P. Curtin, et al

Atari Computer Program Writing Workbook -
Alan North

Atari 400/800 Basic Reference Manual

Atari Programming with 55 programs -
Linda M. Schreiber

We are currently exchanging bulletins
with several other Atari groups around

the country and there are several issues
available for check-out. There are some
quite good articles in these exchanges
and this is an excellent way to keep up
with what Atarians are doing around the
country. We hope to be able soon to
publish excerpts from some of these
bulletins.

EXCHANGE BULLETINS

The listing of bulletins available is
not a month by month compilation. If you
are currently subscribing to an out of
town, Atari oriented bulletin, please
share your old copies with other members
by donating them to the library.

Current Notes - Washington D.C. area

J.A.C.G. - Jersey Atari Computer Group

Mile High Atari Magazine - Denver
Colorado

P.A.C.E. - Pittsburgh Atari Computer
Enthusiasts

Nybbles & Bytes - Phoenix, Ariz.

Milatari - Milwaukee Area Atari Users
Group

S.L.C.C. Journal - San Leandro (Calif.)
Computer Club

Neuron - Austin (Texas) Atari Computer
Enthusiasts

P.A.C.E. World - Peninsula Atari Computer
Enthusiasts of Virginia

N.O.A.U.G. News - New Orleans Atari Users
Group (*Our last issue was returned as
undeliverable, so.... - Ed.*)

The Pokey Press - Atari Computer Club of
the Palm Beaches (Florida)

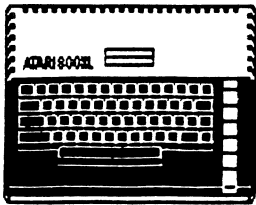
Blackhawk A.C.E. Newsletter - B.A.C.E. of
Waterloo, Iowa

Keeping FACE - Pittsburgh Atari Computer

Enthusiasts

Flagpost - Fort Leavenworth (Kansas)
Atari Group

We seem to be accumulating a rather large collection of duplicate copies of some of the more popular Atari magazines. In no way do I want to discourage donations to the library. I think the best solution will be to offer the duplicates as a give-away to whomever wants them.



Eight-Bit Atari Programming

Machine Language Programming
Part 8 -- Using Page 0 Pointers

by Charles W. Brown

When you hear about memory locations, you sometimes hear about pages of memory (A page of memory is a 256 byte long block of RAM, which starts at an "even" hexadecimal location, such as \$1000 - Ed.). One that I am sure you have heard of is page 6. This is where a lot of small object codes are stored (The reason for this is that this area of memory has been reserved for just this purpose. The 6502 NEVER uses the locations \$0600 to \$0700. - Ed.). If you have seen X=USR(1536) in a BASIC program, then you've seen the use of page 6. Another very important page in memory is page 0. One of the reasons for its importance is the page 0 pointers.

Page 0 pointers are a very important part of programming. First of all, with a page 0 pointer and the Y register, you can easily access all of the Atari's memory. It is also faster. This is why the operating system uses several page 0 pointers for it's own uses. Two very important page 0 pointers are at decimal locations 88 and 89. Location 88 contains the low byte of the address of

the screen memory. Location 89 contains the high byte. This address is where the computer stores the data to show on the screen. You could try a BASIC command like this:

```
A=PEEK(88)+PEEK(89)*256:POKE A,33.
```

This should put an "A" in the upper left hand corner of a graphic 0, 1 or 2 screen (This is because you have just forced the code for the letter A [33] into this position in the screen memory. - Ed.). It will put a colored block in the same place in any other mode.

I have included a machine language program, which was written with the Atari Macro Assembler Editor. By using a page 0 pointer, it will POKE the work ATARI to the screen, using the technique discussed above. In this program, you can move the word around the screen by using a joystick in port 1.

This is just one example of how the page 0 pointers work. This also gives you another example of the use of the X and Y registers (Refer to last month's article for more details. - Ed.).

```
ORG $4000 ;STORE OBJECT
CODE AT 4000 HEX
LIST -M
;1ST WE DEFINE THE VARIABLES SCNPOS
; AND SCNADR AS BEING PAGE 0
;LOCATIONS THESE WILL BE OUR POINTERS
SCNPOS = $CB ;USED TO POINT
WHERE OUR MESSAGE WILL BE
SCNADR = $CD ;USED TO TELL
WHERE TO ERASE
LDA #1
STA CRSINH ;POKE 752,1 (TURN
CURSOR OFF)
PUTREC CLEAR,#1,#0,#0 ;THIS
MACRO CALL WILL CLEAR THE SCREEN
;1ST WE WILL TAKE THE HIGH BYTE OF
;SAVMSC WHICH CONTAINS THE HIGH BYTE
;OF THE ADDRESS OF SCREEN MEMORY AND
;STORE IT INTO THE HIGH BYTE OF OUR
;PAGE 0 POINTERS (SCNPOS+SCNADR)
LDA SAVMSC+1 ;LOC 89 DEC
STA SCNPOS+1 ;HIGH BYTE OF
```

```

PAGE 0 POINTER
    STA SCNADR+1 ;SAME FOR OTHER
POINTER
    CLC ;CLEAR THE CARRY
BIT FOR ADDITION
;SAME THING BUT WITH THE LOW BYTE
    LDA SAVMSC ;LOC 88 DEC
    ADC #60 ;WE WILL START AT
COLUMN 20 OF THE 2ND ROW
    STA SCNPOS ;LOW BYTE OF OUR
PAGE 0 POINTER
    STA SCNADR ;SAME FOR OTHER
POINTER
    LDY #0 ;SET THE Y
REGISTER TO 0
    JMP STORE ;DISPLAY VAR MESS
ON SCREEN
LOOP: LDA STICK0 ;PEEK(632)
(JOYSTICK 0)
    CMP #14 ;IS THE STICK
PUSHED UP?
    BEQ UP ;GO TO LABEL UP
IF VALUES ARE EQUAL
    CMP #13 ;IS THE STICK
PUSHED DOWN?
    BEQ DOWN ;GO TO LABEL DOWN
IF VALUES ARE EQUAL
    JMP LOOP ;GO TO LABEL LOOP
(WAIT FOR PROPER STICK POSITION)
UP: CPY #245 ;SEE IF WE HAVE
GONE TO FAR TO THE RIGHT
    BNE CONT ;GO TO LABEL CONT
IF VALUES DON'T EQUAL
    JMP LOOP ;GO TO LABEL LOOP
IF Y=245
CONT: JSR ERASE ;ERASE ANY OLD
DATA ON SCREEN
    LDY HOLDY ;RESTORE THE Y
REGISTER
    INY ;INCREMENT THE Y
REGISTER
    JMP STORE ;GO TO LABEL
STORE
DOWN: CPY #5 ;SEE IF WE HAVE
GONE TOO FAR TO THE LEFT
    BNE CONT1 ;GO TO LABEL
CONT1 IF VALUES DON'T EQUAL
    JMP LOOP ;GO TO LABEL LOOP
IF Y=5
CONT1: JSR ERASE ;ERASE OLD DATA
ON SCREEN
    LDY HOLDY ;RESTORE Y
REGISTER
    DEY ;DECREMENT THE Y

```

```

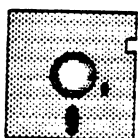
REGISTER
STORE: LDX #0 ;SET THE X
REGISTER TO 0
    STY HOLDY ;SAVE THE VALUE
IN Y REGISTER IN VARIABLE HOLDY
SLOOP: LDA MESS,X ;GET A CHARACTER
FROM THE VARIABLE MESS
    SEC ;SET THE CARRY
FLAG FOR SUBTRACTION
    SBC #$20 ;SUBTRACT 20 HEX
FOR FOKING TO THE SCREEN
;NOW WE WILL STORE IT INTO THE
;LOCATION POINTED TO BY OUR PAGE 0
;POINTER SCNPOS WITH THE OFFSET OF Y
;IN BASIC IT WOULD LOOK LIKE THIS
;POKE PEEK(88)+PEEK(89)*256+Y
; ,MESS+X
    STA (SCNPOS),Y ;STORE IT IN
ADDRESS POINTED TO BY SCNPOS
    CPX #4 ;HAVE WE FINISHED
YET?
    BEQ DONE ;GO TO LABEL DONE
IF VALUES ARE EQUAL
    INX ;INCREMENT X
REGISTER (GET NEXT CHARACTER)
    INY ;INCREMENT Y
REGISTER (GO TO NEXT SCREEN POSITION)
    JMP SLOOP ;GO TO LABEL
SLOOP
DONE: JMP LOOP ;GO TO LABEL LOOP
ERASE: LDY #0 ;CLEAR ENTIRE
AREA OF Y REGISTER ON SCREEN
;THIS SUBROUTINE WILL CLEAR THE AREA
;ON THE SCREEN SO THAT NOTHING IS
;LEFT OVER
ELOOP LDA #0 ;USED AS A SPACE
    STA (SCNADR),Y ;STORE IT INTO
ADDRESS POINTED TO BY SCNADR
    INY ;INCREMENT Y
REGISTER (GO TO NEXT SCREEN POSITION)
    BNE ELOOP ;GO TO LABEL
ELOOP IF Y<>0
    RTS ;RETURN FROM
SUBROUTINE
MESS: DB 'ATARI'
CLEAR: DB #7D
HOLDY DS 1

```

You may have noticed that I had to save the Y register in HOLDY. You have to use the Y register when you use a page 0 pointer. As I was using two different pointers (SCNPOS and SCNADR), I had to use the Y register twice. So, I had to

save it to a variable so that I could remember where it left off. This shows one disadvantage of the 6502 chip: it only has the accumulator and the X and Y registers to use. In any particular program, you usually have to use them many times. So, you have to save their values somewhere else in order to avoid confusion.

I hope this article has given you a little more insight into machine language programming. Again, I don't claim to be an expert. If you do have any questions, feel free to ask me. If I don't know the answer, I will try and find it for you.



Software Review

Decision in the Desert
Microprose Software

review by Mike Carney

I confess to a love for strategic games, in particular anything using tanks. I remember as a kid watching T.V.'s "Rat Patrol" and "Combat", and reading "Sgt. Rock" comics. If any of this strikes a chord in you, then Microprose has given us an object for our affection.

The game opens with an auditorium scene in which the General explains the difficult situation that the Allies face in North Africa (this can be skipped by pressing START, but is worth a look the first time or two). The player is then asked for the operational "Day Code", which can be found in the manual (without the manual, only one scenario may be played). The access code is one way in which software authors try to combat piracy. After proper authorization, a menu is presented containing the game scenarios and options. You may choose to play either side (Allies or Axis), full or limited intelligence, level of

advantage (gives added strength to either side), use military icons, and keyboard or joystick input. All major orders may be issued via the joystick.

There are five major scenarios to select from, depicting pivotal times in the battle for North Africa, and variants of each. Many of the variants provide a "what if" type of game (i.e. What if Rommel had sufficient men and equipment when he began his push to retake Tobruk?).

The battle field is drawn and the cursor is an open square showing one "hex" (approximately three square miles). Moving the hex over a unit and pressing the fire button once brings up the units status in the text area at the top of the screen. Pressing the fire button twice allows one to issue orders to the unit under the hex. Play moves in what Microprose calls "accelerated real time", that is, the clock moves forward 15 min. every few seconds (the pace can be sped up or slowed down at your discretion).

Units respond to orders given according to their status (a mobile unit will move sooner than one that is dug in, etc.). Other variables that affect movement and combat are terrain, experience and fatigue. Each evening the status of the two armies is displayed and units are resupplied (every armchair general knows to keep a secure supply line).

An option is provided for saving the current game; in fact up to 20 games may be saved. When the game's timetable has run it's course, a victor is declared and the margin of victory based on "victory points" given for objectives taken and armies defeated.

Microprose has turned out an excellent product. The graphics and sound are first rate (though I must confess I had some trouble remembering which unit was which -- this is not the fault of the program). Options and

variations abound, providing the gamer with great replay value. The "victory point" system encourages replay to see if a better score, via a different strategy can be obtained. The manual is excellent, providing extensive explanation of all phases of play, as well as noting which sections are necessary if one wishes to get to the game quickly. The manual also provides some history and recommended supplemental reading. While most people have a universal dislike of instruction manuals, this one is worth the time to read.

If you enjoy strategic board games, you should see what a computer can do for them. For a new challenge using "brain power", not just being a "joystick jockey", try Decision in the Desert. Then if the kids ask what you're doing, as Patton said, "You won't have to tell 'em you're shoveling s**t in Louisiana." This game rates an A+.



Foreign Facts From Far Away

Reprinted from CURRENT NOTES, Nov. 1986.

BASIC COMPUTING FOR BEGINNERS
by Ron Peters

Let's talk about DOS and how it's the "secretary" for your disk drive. And, for our purposes, we'll confine our remarks to Atari DOS, since that's the one most of us use.

First of all, DOS stands for Disk Operating System, and is merely a series of programs that allows you to use a disk drive for storing and retrieving information with your computer. Do you need DOS to use a computer? No, not if you plan to store your data on a cassette, or not store it at all.

Since a computer has a volatile memory, that is, it only "remembers" for

as long as the power is on (don't we all!), you have to have some place to store all your letters, mailing lists, art pictures, etc. DOS is the interface between your computer (CPU) and the disk drive, and acts as a traffic cop and a secretary.

The disk drive is like a record player, with the record (floppy disk) rotating at about 288 rpm (Atari 1050 disk drive). The disk drive, like a record player has a "tone arm" that moves across the record and picks up the information -- however, our special record player arm can "play" as well as "record" your information on any part of the record as it spins by the arm (officially known as the read/write head). The disk has a table of contents section that tells the "arm" where to go on the disk to get the data you've selected. The computer disk is housed in a special jacket so you won't get your greasy paw prints all over the magnetic surface (the disk spins within the jacket) and the "tone arm" reads the information through a slotted "window" at the lower part of the jacket.

The disk drive can "play" only one disk at a time, so the disks must be kept in a separate, safe location (disk file of some type). Remember to keep them away from TV, microwaves, telephones, bathtub, pizza eaters, etc.

Each disk, after formatting, is divided into sections (sectors), and each sector will hold about 128 pieces (bytes) of information. If you are using Atari DOS 2.0, each disk will hold about 90,000 bytes of information (a byte is usually one character. Other DOS's will format (prepare) a disk so that the information is more tightly packed in and thus hold more files, programs and data (Atari DOS 2.5 will allow you to store about 130,000 characters on a disk).

At your request, DOS can perform the following for you:

Give you the names of the files and programs on the disk including the amount of space each one takes up (in sectors), and the total amount of space left on the disk (DISK DIRECTORY).

It can also make copies of any file and either place it on the same disk under a different name (DOS can't handle two files with the same name on the same disk) -- (COPY FILE), or place it on another disk under any name (DUPLICATE FILE). Also, a point often overlooked, the COPY FILE command can send the file to disk, printer, or screen.

Without a lot of extra effort, DOS can copy the whole contents of the disk and place it on another formatted disk of your choice. (DUPLICATE DISK)

DOS can find any binary program on the disk, carry it to your computer, load it and make it run (BINARY LOAD).

If you want any file sealed (or unsealed) so the contents cannot be altered, DOS can easily accomplish this (LOCK FILE) and (UNLOCK FILE).

And, if you're really nice, DOS will make a copy of the itself (DOS.SYS and DUP.SYS) and place them on another disk for you (WRITE DOS FILES). Without these files on your disk, you will get a "boot error" message when you start up the computer (with some exceptions, that might be the subject of another article).

DOS can also clean (format) a whole new disk, or reclean a previously used disk, making it ready to accept your programs and data (FORMAT DISK).

What's more, DOS can put a new label on a file (RENAME FILE), or even find a file and throw it in the trash (DELETE FILE).

There are other functions that DOS will do for you, like LOAD AT ADDRESS, SAVE AT ADDRESS, etc., but these are for the high techies. And, the Atari computer will let you use up to eight

different disk drives so you won't have to be swapping disks all the time.

Oh, I almost forgot. If you don't need DOS for a while, it can turn the control over to a cartridge or a BASIC program (RUN CARTRIDGE), and go back to waiting in the wings. When you're ready, it will also SAVE or LOAD your BASIC programs.

That DOS is pretty handy to have around, isn't it? If my secretary were that efficient, I'd have to pay her more money. However, she still has trouble jumping on and off a platform spinning at 288 rpm.

Next time... let's talk about how a word processor is like your own typewriter and printer, with DOS standing by with scissors, paste, a wastebasket, and a Xerox machine.

Glossary:

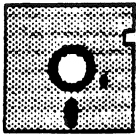
DOS: Some programs (DOS.SYS & DUP.SYS) that allows you to have total control over your disk drive.

FORMAT: A system of preparing a disk so that DOS can store files and programs on it. (If you FORMAT a disk with data on it, kiss that data goodbye forever! *(Deleting you can recover from, Formatting you can't! - Ed.)*)

BINARY PROGRAM: A computer program that has undergone a "translation" into binary format (using only numbers 0 and 1), since the computer has only two fingers and this is how it counts.

BYTE: Refers to a specific unit of space in a computer memory, normally equal to one character (letter, number, etc.) A byte is made up of 8 bits, each bit being a binary number 0 or 1 (like a switch that is either "on" or "off").

BIT: A small "nibble," less than a byte. It takes 8 bits to make a byte, but only one bit to ruin a disk.



Software Review

WARGAME CONSTRUCTION SET by S S I
by Charles W. Brown

In this article I will review a new game called Wargame Construction Set. It is put out by SSI. Not only is this a new game, it is a game generator. Some people may ask what is a game generator? Well, it is a program that gives you a game to play. This same program has the ability to create other games to play. Then, you can play your own custom made games, and even save your custom games off to disk for future use. I have always enjoyed game generators. They can unleash the creator hidden in all of us. This way you can play a game idea that is yours instead of somebody else's. Some examples of other game generators are: Pinball Construction Set; the Arcade Machine; and Lode Runner.

The game comes with two disks -- a program disk and a scenario disk. Both disks are double sided. On one side of the program disk is the game; on the other side is the editor for making games. Both sides of the scenario disk contain data for the provided battles.

Let's look at the game itself. It is for one or two players. Once you have determined which of these you are, you will be prompted to insert your data disk. Upon doing that, you will choose which battle to fight. This can come from the scenario disk provided, or from your own data disk. Once the data for that battle is loaded in, the game will start. I will explain a one player game. A two player game is a little different.

The first part of the battle is the observation phase. In this phase you will be shown a map of the battle site. Using a joystick you can move the cursor

over the different units. When the cursor passes over a unit, information about that unit will be displayed. This will help you to plan your attack or defense.

The second part of the battle is the friendly fire phase. In this phase you will fire at the enemy (*that doesn't sound very "friendly" to me!* - Ed.). You first place the cursor over the unit that will do the firing and push the fire button. Then you take aim by putting the cursor over the enemy unit you are shooting at. Upon pushing the fire button, the results are displayed. You can either hit, miss or eliminate the unit. Your units can only fire once during each turn. Other messages will tell you if you are out of range or there is no line of sight.

Next comes the friendly movement phase. In this phase, you can move your units around to better position. Each unit is given movement points. You can move your units around until it runs out of these points. In moving a unit, you can move it onto a square occupied by an enemy if you like. This is called an assault. During an assault one of four things can happen. Either you or the enemy can be forced back a square, or you or the enemy can be eliminated.

The next four phases are enemy fire, friendly fire, enemy movement and enemy fire. The enemy movement and fire phases are the same as the friendly ones.

In the victory phase you get the results of your previous battle. You will be told how many units each side has left.

The last phase gives you a chance to save the game off to disk for later play. You can either save the game off and quit, or continue on for another round of fighting.

Now we will look at the editor. It has eight options that you may choose

from. The first is disk access. Here you can either save off or load in a game.

The next part is the draw map routine. This is where you create your battlefield. You will be shown a pre-drawn map. You can either erase the map and start from scratch, or simply edit the map. When you are making the map you have different icons to use. You first choose the icon that you want, and then you move it around with the joystick until you find the square where you want it. Pushing the fire button places the desired icon there.

You have many icons to choose from. They are: roads, river, woods, bridges, buildings and mines. Within each of these types you have different ones to choose from. In other words, you have different road icons to show the road the way you want it. There are horizontal, vertical, curves, and even an intersection.

In the next phase you edit the friendly units. In this, option you create your own custom made units. You can decide on many different factors for each unit. The 1st is the type of units. It can be either an infantry, tank, engineer, mortar, gun, truck, helicopter, or a boat. Then you decide on its offensive or defensive strength, how many movement points it has, and also the range of fire it has. There are other options you have for each unit. The next feature is used to edit enemy units; it does the same thing as for the friendly units.

In the deploy unit phase you determine the starting positions for both sides.

There are other phases such as choose color. Then there is the print map mode. This is supposed to print your battle map on the computer. I don't think it does a very good job of it, though. You can even choose the scale your map is in. I don't understand what this option is for.

As you can see, this program has a lot of features. It also has some drawbacks. For one thing, the game does not realize that all of the enemy forces are gone. It seems to want to continue the game. I also think that it should have other options for different types of battles, such as earlier wars when there were no tanks or helicopters.

I hope you have a little better understanding of this new game, and how it works.



New Comers' News

Beginners' DOS Tutorial
Excerpted from an article by
Roger Downey

in the August 1986 Nybbles & Bytes

Buyers of the Atari 1050 Disk Drive should check the disk supplied with the drive. If the disk has DOS 2.0 or DOS 2.5 written on it, simply refer to the instruction manual, or any experienced ACEC member. If, however, the disk has DOS 3 on it, follow these directions to the letter:

1. Carefully remove the disk from its jacket.
2. Throw it as far away from you as you can.
3. Go over to it where it lays and stomp on it repeatedly (re-formatting is too good for this ill-conceived beast).
4. Run, do not walk, to you nearest ACEC meeting or local computer store, and request a copy of DOS 2.5.

Seriously, there are several reasons for avoiding DOS 3 like the plague. One is that it is not compatible with DOS 2 (which has become standard), or anything else for that matter, without first going through a lengthy file conversion process. Another is that its formatting system penalizes users who save a lot of small files by taking up large amounts of space for them (the smallest unit of storage is the equivalent of 8 normal DOS 2 sectors). The list goes on and on.

A Antic On-Line

ANTIC Publishing, Inc.
copyright 1987. Reprinted by
permission from Antic Online.

Millionaire, by Blue Chip Software
reviewed by John McCormick

(Requires BASIC \$19.95, 48K disk)

I bought 300 shares of IBM and 10 call options on Exxon. My charts showed that General Motors looked good in view of the earnings projections in news reports, so I bought 200 shares of GM for my portfolio. Two weeks later I had lost \$50,000 and needed to sell all my shares of Dow Chemical just to pay the bills. Some say money is the root of all evil -- others claim it's the lack of money. I don't necessarily support either opinion, but I do know that the winning, losing, earning and spending of money fascinates me -- much more than shooting down aliens or getting lost in mazes.

Millionaire is as much a stock market simulation as it is a game, combining real-life situations with exciting action. Millionaire, and other Blue Chip Software games are so realistic that stock brokerage firms use them to train new brokers! Millionaire can either create a new game for you or restart one that you saved in progress. It also keeps track of the top scores for seven players. I have been an active investor for years, so my interest in the game is understandable, but my wife, who has no interest in finance, still found Millionaire a most fascinating game.

You start with \$10,000. Your goal is to increase this amount to \$1 million through shrewd investments in 15 stocks covering computers, oil and gas, retail, auto and heavy industries (steel, chemical, machinery). You also have 14 weeks of market information for all 15 stocks, along with graphs for each industry group, and you have 77 weeks to

meet your \$1,000,000 goal.

As a Novice investor you can only buy stocks for cash (much like the real world), but as your profits grow and your net worth increases, you move up through the ranks of Investor, Speculator, Professional and Broker. As your rank increases so does your range of investment tools. Soon you can use margin purchases, or put and call options. Eventually you can even borrow up to 80 percent of the value of your assets and leverage your investments. This increases the rate at which you gain (or, more likely, lose) money.

Each week you automatically see graphs of all your investments, price and price movement information on all stocks, and news reports. You can also view graphs of every stock, or follow the activities of entire industry groups. Purchases are easily made, with the software handling all needed calculations, telling you how many shares you can buy with available cash, and even deducting commissions and interest.

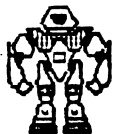
This might sound complicated to those unfamiliar with the stock market, or it might sound too simple to an investor. Believe me, neither is the case. I had great trouble tearing myself away from Millionaire once I started, and I rarely quit playing before finishing a full 77 turns. My wife said she learned more about how I invest money from playing one whole session of Millionaire than from reading the Wall Street Journal on and off for 15 years.

My only complaint is that Millionaire is slow at times -- though my wife found that the program was always ready for her next move long before she'd made her decisions. Millionaire is completely menu-driven, often returning you to the main menu after completing an action when you might prefer finishing another transaction before leaving a particular area. This is really minor, though, because you can always leave the main

menu again at any time.

The only tip I can offer in playing this simulation is not to pay too much attention to the news reports. Not that they aren't honest -- it's just that, as in the real world, you can't always tell what will happen to a stock from the news about the company.

The only real improvement I can think of is if Blue Chip would send me a check when I win big at Millionaire. But then they'd probably want me to pay them when I lose. Oh, well, I guess using play money is probably the safest. So far all I have really lost is a fair amount of sleep.



Hardware Modification

1050 / XM301 Fix
reprinted from the Mile High
Atari Magazine, Nov. '86

Many users are having booting problems when using the Atari XM301 modems with their Atari 1050 Disk Drives. Thanks to the Status Users of Virginia Beach, VA, we have an easy hardware fix. Please note that this fix is NOT an official Atari modification, and that your warranty will become null and void once you perform it. Also, it should not be performed by the non-technical, non-soldering types. *(Finally, it has not been tested by me, and I have no idea whether it truly works or not - Ed.)*

The parts needed are: three 470 ohm resistors; a small amount of solder; a soldering iron (15 to 25 watts); a wire cutter; a small Phillips screwdriver; and three small lengths of shrink tubing.

Open the XM301's case and remove the PC board assembly.

Locate the area where the cable is

attached to the board. The wires will be labelled with numbers. Some modems will also have these contacts coated with a sealant. Remove the sealant carefully with your fingers, a small knife or the screwdriver.

Now the tricky part! De-solder (if you don't know what this means, don't do it!) the lines labelled 3, 9 and 13, and pull the wires through the board. Remember which wire goes into what hole! Might be a good idea to label them with a small piece of tape.

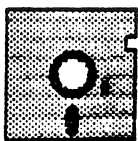
Install each of the 470 ohm resistors into the open holes on the PC board. Push the resistors all the way through, leaving them standing straight up.

Trim off the excess wire sticking through the PC board, and carefully solder the resistors in place.

Place the heat shrink tubing on the ends of the wires, and then solder them to the resistors in their corresponding holes. Trim off any excess wire, and heat the tubing carefully so it shrinks to fit.

Put the PC board back into the case, close it up, and congratulate yourself!

Now, when you boot up the system, everything should work the way it was supposed to in the first place!



Software Review

MICRO LEAGUE BASEBALL
STAT COMPILER AND BOX SCORE DISK
by Charles W. Brown

In this article I will write about a new disk for the Micro League Baseball game. This disk is the Stat Compiler and Box Score disk. This is a new utility disk, and can be used with any of the disks from the Micro League series.

This disk offers three different features. They are: a stat compiler; a box score creator; and a program to produce hard copy of any data from your micro league disks.

The first feature I mentioned is the box score generator. To use this feature, you must boot the disk. Then, you choose the Play Game function. You put your game disk in the drive, and proceed to pick your teams and then play your game as usual. After the game has ended, a "box score" is displayed. This will show you the actual statistics from the game, and has a lot of useful data. Some examples are Errors, Doubles, Double Plays, Home Runs, RBI's, etc. These data can really help you see how your players are doing.

The second feature is the Statistics compiler. This is a very powerful feature. In order to use this feature, though, you have to have the desired team on the box score/stat compiler disk. If you have used the desired team in a game and then choose the compile option, the computer will simply take the data from the just-played game. It will compile it with the data on that team's file on the disk. This way you can update your team's data, based on the actual game situations. This way your stats will be more realistic. This will help you

decide how your players are doing. Is your player really that good, or are the stats misleading?

The third feature is the print option. This feature will give you printed copies of your game results. This can either be the box scores of a just-completed game, or the compiled stats of a team. This feature will work with the data of teams on any of the micro league disks.

This program is a powerful addition to the Micro League series. It allows you to get more enjoyment out of your games. I have said before that I don't particularly like the game of baseball. However, this utility makes me want to play the game, because it adds so much more realism.

There is one major disadvantage to this utility -- it requires a full 64K to run. It checks your computer out when you boot up the disk. If you don't have 64K, the program will stop and warn you. If this happens, you are just out of luck. It is too bad that the program had to be so big. That means that the people with 400's and 800's can't use it. Also, I think that some of the features that are on this disk should have been included on the game disk to begin with, especially the print option.

I hope that I have given you a glimpse of this program and what it can do.



In-Depth Tutorial

ANTIC Publishing Inc.
copyright 1985.
Reprinted by permission

CHRIS CRAWFORD'S
ASSEMBLY LANGUAGE COURSE - Lesson 4 of 8

LESSON FOUR: BRANCHING

One of the most important ideas in computing is the concept of conditional execution. This is the ability of the program to execute different routines depending on conditions at the time of execution.

The significance of this capability is best realized by considering how programs would operate in its absence. A program without conditional execution would not be able to change its program flow in response to conditions.

In other words, it would always execute exactly the same code in exactly the same order. Every run of the program would follow exactly the same sequence and perform exactly the same operations. Not very interesting, right?

To get a grip on conditional execution, we need to look at it in its simplest expression. The simplest type of conditional execution is binary in nature. We have a chunk of code; the 6502 will either execute it or it will not execute it. The decision is made on the basis of a boolean value; a true value will tell us to execute the chunk, while a false value will tell the 6502 not to execute the chunk.

The basic mechanism for doing this is through an instruction that performs a transfer of control. This involves

nothing more than altering the program counter. You may recall that the program counter is a register in the 6502 that points to the address of the currently executed instruction.

When that instruction has been executed, the program counter is increased by the length of the instruction (1, 2, or 3 bytes, depending on the instruction). It now points to the next instruction. This little system allows the 6502 to step through a program in sequence.

But there are also instructions that will alter the value of the program counter, allowing the 6502 to jump to another area of memory and another part of the program. The simplest of these is the JMP instruction. It takes the form JMP LABEL.

This loads the value of the LABEL into the program counter. Its effect is to make the 6502 jump to the address of LABEL and continue execution from there. It is directly analogous to a GOTO instruction in BASIC.

For conditional execution we need something more. We need the 6502 to have capability to make a binary decision based on a binary value. The solution used by the 6502 involves flags. These are single-bit Boolean values stored together in a single byte of the 6502 called the processor status register (SR).

The status register is eight bits wide but stores only seven flags. These seven flags are labelled N, V, B, D, I, Z, and C. You have already encountered the C (Carry) flag and the D (Decimal) flag. In this chapter, we are concerned only with the N, V, Z, and C flags.

The magic instruction that makes possible conditional execution can take many forms. Its general form is Bfv LABEL. The B stands for "branch". The "f" stands for a flag, and the "v" stands

for the value of the flag, either true or false. However, in this case, we do not use the terminology "true or false".

Instead we use the terms "set" or "clear". "Set" means the same thing as "1" or "true", while "clear" means "0" or "false". The label is the address to which the 6502 should branch if the condition is satisfied. If the condition is not satisfied, then the 6502 will simply skip this branch instruction and go to the following instruction.

For example, suppose that we have the following instruction sequence:

```
        LDA    #0
        BCS    KARELIA
        LDA    #5
KARELIA STA    FISH
```

This will first load the accumulator with a zero. Then the 6502 encounters the BCS ("Branch on Carry Set") instruction. It looks at the Carry flag. If this flag is set then the 6502 will indeed branch to the label KARELIA (For all you geography buffs, Karelia used to be in Finland.). In other words, if the Carry flag is set, the 6502 will skip over the LDA #5 instruction. Thus, a zero will be stored into FISH.

However, if the Carry flag is clear, then the 6502 will not take the branch. It will instead continue executing the next instruction, which will load a 5 into the accumulator. Then it will come to the label KARELIA and store that 5 into FISH. Thus, the value of the Carry flag determines whether a zero or a five is stored into FISH.

The converse of BCS is BCC ("Branch on Carry Clear"). This will cause the 6502 to take the branch if the Carry flag is clear.

There is also a pair of similar instructions for the V-flag. These are BVS and BVC. They will cause the 6502 to branch on the value of the V-flag.

Now the situation gets unnecessarily confusing. The instructions for the Z-flag should be BZS and BZC -- "Branch on Z Set" and "Branch on Z Clear". Unfortunately, the dumb designer of the 6502 thought he would get cute at this point, so instead he called these instructions BEQ and BNE, for "Branch on Equal" and "Branch on Not Equal". He never mentioned what he thought is supposed to be equal to what. We're stuck with it, so make the best of it.

Just remember what these instructions really mean BZS and BZC. If you think in terms of the Z-flag, it will work out just fine. If you try to think in terms of equal or not equal, your attention will be diverted from the real truth of the matter and you may make mistakes. So keep your eye on the ball and think in terms of Z!

The next pair of branch instructions use the N-flag. These are even more insidious than the previous two. They are called BMI and BPL, meaning "Branch on Minus" and "Branch on Plus".

At first glance, these appear to be reasonable substitutions for BNS and BNC. After all, if you load the accumulator with a signed number, and the number is negative, then the N-flag will be set, while if the number is positive, the N-flag will be clear.

Thus, it would seem that BMI is truly equivalent to BNS and BPL is truly equivalent to BNC. This is the source of many a bug in beginner's programs. Consider the following fragment of code:

```
LDA    FISH
SEC
SBC    GOAT
BPL    POSANSR
```

This code is supposed to branch to POSANSR if FISH is greater than GOAT. And indeed, if FISH is greater than GOAT, then when you subtract GOAT from FISH, you will get a positive result, right?

Not necessarily!

Suppose, for example, that the value in FISH is #C1 and the value in GOAT is 1. When the 6502 subtracts GOAT from FISH, it will get a result of #C0. Note that the highest bit of #C0 is set to 1. This is the value that will go into the N-flag. In other words, even though FISH is greater than GOAT, the 6502 will not take the branch, and this code will fail.

The moral of his tale is, don't take those instructions literally. They are misleadingly named. When you see BFL, don't think "Branch on Plus", think "Branch on N Clear". Otherwise, you'll screw up someday.

By the way, the correct branch to use in the above problem is BCS.

Now for a catch with the branch instructions. A JMP instruction is a simple absolute jump -- you specify the target address and it goes there. The designers of the 6502 realized that the vast majority of branch instructions only go a short distance. They therefore decided to implement the branch instruction as a relative branch.

The machine code doesn't specify the target of the branch, it only specifies an offset. In other words, instead of saying, "jump there", it says, "jump so many bytes forward or backward." The allowable range is 126 bytes forward or backward. Thus, you can't branch anywhere you want, only to nearby locations. If you must branch further, reverse the logic of the branch and use the branch to skip over a JMP statement.

(ELIZA is available, among other places, on ACEC Disk of the Month #21A1 - Ed.)



The Doctor Is In!

Eliza and SAM
by Kenny Shrif

Although Eliza has been bounced around for several years now, this was my first encounter with her. She has been modified to run with SAM (*the Software Automated Mouth speech synthesizer for the eight-bit machines - Ed.*) so she talks to you through what has been compared to a typical psychoanalysis session. To use this modification, SAM must first be loaded, and one of the SAM utilities called KNOBS must also be present. The KNOBS routine allows for changes in SAM's speech characteristics and gives Eliza a vaguely female voice (*if this is all that is required of KNOBS, it can also be done with SAM itself - Ed.*).

The Eliza program is written in BASIC, and due to the continual key word searches and comparisons, is quite slow in responding to your input. Despite all of the searches and comparisons, Eliza does not really have enough intelligence to be more than briefly amusing. Nonetheless, some of her responses are surprisingly accurate and to the point. A typical session might go like this:

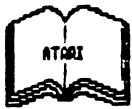
HI, I'M ELIZA. WHAT'S YOUR PROBLEM?
I feel depressed.
TELL ME MORE ABOUT YOUR FEELINGS.
Everything is grey and heavy; I do not want to do anything.
ARE YOU SAYING NO JUST TO BE NEGATIVE?
I hope not. It's just that I do not know a positive way to express negative feelings.
DO YOU OFTEN FEEL?
Yes.
ARE YOU SURE?
Yes.
PLEASE DO NOT REPEAT YOURSELF.

Okay.

SAY, DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?

I guess not. Goodbye.

The program is not new by any means (*it was originally programmed on some of the first mainframes, back in the 60's, I believe - Ed.*), and was one of the first attempts at artificial intelligence. Don't expect too much from Eliza as a shrink, but she can give you some good lessons in decision making logic, especially if you study the BASIC listing. Adding the speech synthesis capability merely increases Eliza's initial appeal.



Book Review

MAPPING THE ATARI
A COMPUTE BOOKS PUBLICATION

by Charles W. Brown

In past articles I always have said that I am no expert on the subject of programming Atari computers (*and I have said that he is too humble! - Ed.*). Someone may ask where they can get more information? A good source is one of the many books on the Atari Computer Systems. Although there are quite a few books out about our computer and it's uses, I think there should be a lot more. I am going to try my hand at reviewing one of them.

In past articles I have mentioned memory locations. After all, that is what our computers are -- just a collection of memory locations. By programing and using our computers, we are just changing the values in different locations. Hopefully, the computer will give us the desired results. If not, we could get in big trouble! If you decide to try your hand at programing, it's a good idea to be familiar with the different locations and what they are

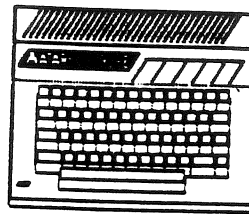
used for.

If you want to know about the different locations and what they are for, then you might try Mapping the Atari by Compute! books. I have found it to be a very useful source of information (*information that can be almost impossible to find elsewhere - Ed.*). Besides telling you about many different memory locations, it also gives you an interesting glimpse of how the computer works. You also see what the different things the computer has to do just to perform a simple task. I have shown you examples before when I wrote the article on printing to the screen.

The preface of the book shows some examples of the different numbering systems. Those, of course, are decimal, hexadecimal, and binary. It also tells about performing a variety of logical functions on numbers. These would be AND, OR, EOR and XOR. Even after all the programing I have done, I still don't understand how those functions work! The preface also explains the difference between cold and warm starts of the computer. It gives a good listing of everything that takes place when you either turn on the computer (cold start) or hit the reset key (warm start).

In the introduction the book explains and compares the different languages that Atari uses. It gives a good example of solving a problem with each of the languages. This way you have a good idea of the different ways to communicate with the Atari.

The main memory map takes up the majority of the book. Almost every location is explained. Each location shows the decimal address, the hexadecimal equivalent and the label. The label is a name given to these locations, and is used by programmers to refer to each one. Usually, the label name will tell what the location is used for.



Eight-Bit Atari Programming

One example would be LMARGN. This is the label for location 82. As the name implies it holds the value for the left margin of the screen. Another example would be location 186,187 called STOPLN. It contains the line number that the program stopped at, due to the break key, an error or the use of the trap statement.

Each memory location usually has a good explanation of what it is used for. In a previous article I mentioned pages of memory. This book tries to explain what each page of memory is used for.

This book also gives some good programming tips for you to try. One example of this is in the section on location 132,133 (called VNTD). This is the pointer to the end of the variable name table, plus one byte. The variable name table is of course where all your variable names are stored. The book gives a utility that you can use in BASIC which will list all the variables that you have used in your program. There are too many other examples to be shown here.

After the main map there are the appendices. They contain some valuable information on the computer. Some of the charts are very useful as well. For those people who don't know how to convert a decimal to hexadecimal number (like me!), there is a chart for you to use. In another article I mentioned the different numbers used for putting a character on the screen. This book has a chart that shows each of the 128 characters and the two codes to access each of them (the ATASCII, and the internal keycode).

The book I reviewed is the earlier version, and was designed for the 400 and 800. I have seen newer versions which show data on the XL series. I am sure this book is not the complete answer for everyone. I am suggesting that you might consider this book, however. It may answer some of your questions, or just give a glimpse into what makes your computer tick.

Fine Tuning DOS

By "The Divemaster"

(reprinted from the Alamo Area Atari Users' Group, Mar. '86)

How to Delete "Twin Files"

Have you ever ended up with two (or more) files on your disk with the same filename? Do you end up cursing and screaming when you try to delete one of them, then find that DOS has deleted BOTH of them? There is a way around that folks:

- 1) Boot up DOS with BASIC in.
- 2) In immediate mode, type POKE 3118,0.
- 3) Type DOS and press return.

Now you'll be able to delete one without losing both files, because POKEing 3118 with a 0 will now only allow you to delete the first file.

Speeding Up DOS 2.0

If your Dos 2.0 seems to write very slowly compared to other DOS's, that's because it has the "built in" write verify turned on. You can easily create a new version of DOS without the write verify on. Here's what to do:

- 1) Boot up DOS with BASIC in.
- 2) In the immediate mode, type POKE 1913,80.
- 3) Type DOS, and press return.
- 4) Now, re-write DOS to your disk by using the Write DOS and DUP function of your DOS menu.

It will write much faster now

ARRGH! Error 164 Again!!

What? You got an error 164 Again?! Well, did you know you can still force DOS to load in the program anyway? Error 164 means you have screwed up data on

your disk. If you have a lot of garbage, you have a big problem. But what if only a little bit of the file got messed up, and you just know you could fix it if you could just get the file to load! Never fear; do the following:

- 1) Boot in DOS with BASIC in.
- 2) In immediate mode, type POKE 4148,234:POKE 4149,234
- 3) Now load your program either from BASIC or DOS.
- 4) You won't get ERROR 164 now, and you can fix that garbled mess. Ain't that peachy!

If You Have More Than Two Drives

Some DOS's (like Smart DOS) are built for systems with multiple drives, but many DOS's (2.0, N-DOS, etc.) assume that you have only two drives, which is a no-no if you just bought a third drive and your DOS won't address it. All you got to do is the following:

- 1) Boot DOS with BASIC in.
- 2) In immediate mode, type 1802,15
- 3) Type DOS and press return.
- 4) Write out the 'new' DOS using the 'H' selection on DOS's menu.
- 5) You now have a DOS version that will look for up to 4 drives.

Caution: Be aware that each drive removes 128 bytes of free memory from program use.

Changing the Wildcard(*) To

If you like to play around with DOS and want to change your wildcard character from (*) to something else (It's up to you as to what that something else is), it's really pretty simple. Dig out your bible (Your Atari Computer, written by Ian Poole) and turn to the ATASCII chart. You'll notice that the ATASCII number for (*) is 42. Now pick out whatever character you'd like instead, and find it's value (example the (+) character has an ATASCII value of

46). To make the change do the following:

- 1) Boot DOS with BASIC in.
- 2) In immediate mode, Type POKE 3783,46 (or substitute what ever character value you want for the 46)
- 3) Type DOS and press return
- 4) Rewrite DOS to the disk using the (H) selection

You now have a 'patched' DOS with your own personalized wild-card character.

Making Lowercase Filenames

Yes fans, you can use lowercase letters for your disk filenames. You have seen it with MYDOS, Right? Here's how to do it with good old ATARI DOS. If you look at the ATASCII codes, you'll see that the value of '0' (zero) is 48, and the value of lowercase "z" is 122. Try this:

- 1) Boot in DOS with BASIC in.
- 2) In immediate mode, type POKE 3818,48:POKE 3822,123
- 3) Type DOS and press return.
- 4) Re-Write DOS to your disk with the 'H' selection.

These POKEs set in the parameters to accept ATASCII values in the filenames starting from 48 and ending in 122 (but the 123 you see in the above POKE is correct). It's risky to go above 123, so stick with that, IT REALLY WORKS!!!

Open Up to 7 Files at Once

Have you ever noticed that you can have no more than three files open at one time with your 'normal' DOS? But ATARI allows you to have seven files, so why not also be able to use all seven at once? It's still a free country isn't it?? Try this:

- 1) Boot in DOS with BASIC in.
- 2) In immediate mode type POKE 1801,7
- 3) Type DOS and press return.

4) Re-write DOS by using the 'H' selection.

A word of caution: Each file that you allow open will use a 128 byte data buffer. So, you really shouldn't open more files than you actually need. If, for example, you need to have five files open at once, then the POKE would be POKE 1801,5.

A final word of warning: be careful to identify your "customized" DOS disks, so that they don't get mixed in with anything else. Unless you've told them about your changes, friends that you exchange disks with could be very confused by some of these modifications.



THE ST SUPPLEMENT

Due to a communications problem (someone forgot to call me!), last month's ST Supplement did not appear in the newsletter. To make up for that oversight, I hereby present TWO ST Supplements! - Ed.

by Nassir Amra

Well, I'm back again with the second installment of the ST Newsletter Supplement. Currently, the ST Supplement is part of the regular ACEC monthly newsletter, which is available to ACEC members only. This is another reason for all you dedicated ST users reading this to become a member of ACEC (of course, if you're reading this, you probably already are a member, but.... - Ed.).

My contributions to the ST newsletter will center on Modula-2, GEM, and illustrating various pearls in the public domain. If you want to submit any material about the ST to Fuji Facts (the

ACEC Newsletter) please do so.

MODULA-2

I finally broke down and got a CompuServe account. My main reason was to download any fixes that TDI had posted for their Modula-2 package. The files I downloaded are in the MODUPDAT folder on one of the DOM disks. TDI has managed to fix the bug in IORec with their new release of XBIOS. I haven't tested the other new files, but I presume that they work also.

In my efforts to find out where the bug in IORec was, I ran into a stone wall that had a sign titled 'USER MODE' hanging from a nail. Although you may be the sole owner of your ST, you do not have complete control over your machine. This means that you can not PEEK and POKE with impunity.

THE BELLY OF THE BEAST

The 68000 has two modes, called the user mode and the supervisor mode. When in user mode, the memory addresses 0 to \$800 (\$ = a hex number) and \$FA0000 to \$FFFF00 (the ROM and I/O areas) are off limits unless you like mushroom clouds or bombs on your screen. There is a way to get in a supervisor mode however, where you can do anything.

In Modula-2, SuperExec is a XBIOS procedure that executes a procedure in supervisor mode. Another way is to use a GEMDOS procedure called SUPER to switch to supervisor mode. Unfortunately, TDI did not implement that particular procedure. Instead, I built my own using the CODE procedure which allows machine language to be incorporated within a Modula-2 program.

The code for SUPER as well as for BASIC-like PEEKs and POKEs is in the SUPER folder. The PEEK and POKE procedure can only access memory in units of 2 bytes (viz., words), where the address of the word is on an even

address. Now you can PEEK and POKE to your heart's content by calling SUPERDOS first. SUPERDOS should also be the last procedure called in your program before you exit back to the desktop or weird things will happen.

THE SPACE MONSTER

Despite formatting my disks to more than 800K of storage, I still find the need to buy disks. The basic problem is that I am compulsive when it comes to making backups of original disks that I have. One possible solution to the duplication of material is to use ARC.TTP or EASYSGUEEZE. These programs take files and compress each file and combine the files into one file. This leads to a substantial savings in storage if you are dealing in text (program source code or documentation) as well as give you the ability to transfer the file via modem with one send command. Both programs are available on the DOM disks.

A PLEA

You do not have to write like Hemmingway to contribute to this newsletter. Please try to put together something to submit. We will accept anything about anything.

In this this third issue of the ST Supplement, I will deal exclusively with the Modula-2 programming language.

MODULA-2

One of the nice features of Modula-2 is the ability to divide the code into several modules, compile the modules separately, and then link the modules together later on. I shall illustrate that with one of the programs that I have donated to the public domain library (the programs are in the GEMVDI folder). The programs demonstrate various features of VDI and are Modula-2 translations of C code which appeared in the "Programmer's Guide To GEM" by Balma and Fidler. The

code skeleton for the main module (from \GEMVDI\ARC\VDIRC.MOD) is as follows:

```
MODULE VDIOPEN;
(* ... omitted code ... *)
FROM DRAWINRC IMPORT DRAWRC;
VAR APPL, HANDLE, MAXW, MAXH, KEY, I,
WCHAR, HCHAR, WBOX, HBOX:INTEGER;
WORKIN:GEMVDIbase.VDIWorkInType;
WORKOUT:GEMVDIbase.VDIWorkOutType;
BEGIN
(* ... omitted code that opens an
application and gets the VDI handle ...
*)
FOR I:= 0 TO 10 DO;
WORKIN[I]:=1;
END;
WORKIN[10]:= 2; (* RC *)
VDIControls.OpenVirtualWorkstation(WORKIN
,HANDLE,WORKOUT);
MAXW:=WORKOUT[0]; MAXH:=WORKOUT[1];
DRAWRC(HANDLE,0,0,MAXW,MAXH);
(* this is imported *)
VDIOutputs.GraphicText(HANDLE,MAXW,MAXH,'
PRESS ANY KEY TO CONTINUE');
KEY:=AESEvents.EventKeyboard();
VDIControls.CloseVirtualWorkstation(HANDL
E);
AESApplications.ApplExit;
END VDIOPEN.
```

The above main module imports from module DRAWINRC the procedure DRAWRC. The DRAWINRC module actually consists of two modules. One is the definition module (DRAWINRC.DEF) which follows:

```
DEFINITION MODULE DRAWINRC;
PROCEDURE
DRAWRC(HANDLE:INTEGER;DX,DY,SW,SH:INTEGER
);
END DRAWINRC.
```

Now the above definition module can be compiled and then the main module VDIOPEN (which is in file VDIRC.MOD). The point of all this is you now have a main program that imports a routine DRAWRC from module DRAWINRC, where the main program has been compiled but DRAWRC has not yet been coded. The next step is to code the implementation module:

```
IMPLEMENTATION MODULE DRAWINRC;
```



```

IMPORT VDIControls;
IMPORT GEMVDIbase;
IMPORT AESEvents;
IMPORT VDIAttribs;
IMPORT VDIOutputs;
IMPORT GEMAESbase;
PROCEDURE
DRAWRC (HANDLE: INTEGER; DX, DY, SW, SH: INTEGER
);
BEGIN (* ... here any code can be
placed. There are three files which
demonstrate the use of different VDI
commands: MARKER1, LINETYP, and LINESTY
... *)
END DRAWRC;
BEGIN
END DRAWINRC.

```

There are three files which use the above implementation module skeleton to code for DRAWRC such that there are three different DRAWRC routines. The files are MARKER1.MOD, LINETYP.MOD, and LINESTY.MOD in the \GEMVDI\RC\ folder. Each is compiled separately. To form a running program, the linker is run with the main module (VDIRC.LNK) as input. The linker will stop and ask which file has the DRAWINRC module that the main module uses. One can then choose MARKER1.LNK (this is the compiled version of MARKER1.MOD) and the linker will finish its job and produce a runnable program called VDIRC.PRG.

This file can be renamed to something else and the linker can be called again to link VDIRC.LNK to LINETYP.LNK or LINESTY.LNK. The main goal of all this is that one can create other DRAWRC routines and not have to recompile the main program or any unchanged modules imported to the main program. All that is need is to compile the new DRAWINRC module and then relink the VDIRC main module to the file that contains the new DRAWINRC. The linking is done automatically when you link the main module if the file that contains the DRAWINRC module has the same name as the module name.

I again encourage anyone who is interested in any programming languages,

application programs, and games to write in about their experiences (be they for the ST or the XL/XE series - Ed.).



ATTENTION MERCHANTS:

Wouldn't you like to increase your sales, with a minimum of time, effort and cost?

FUJI FACTS is happy to announce that we are now able to accept advertising in future issues, at rates which are quite reasonable. You will be contacted soon to place your ad.

**ATARJ COMPUTER ENTHUSIASTS
OF COLUMBUS**

UPCOMING MEETINGS:

XL/XE

7:15 pm DeSales High School

February 9

March 9

ST

7:00 pm Doctors' Hospital North

January 22

February 12 ?

ACE of Columbus Newsletter
Warren Lienallen, Editor
1652 Hess Boulevard
Columbus, OH 43212

FIRST CLASS MAIL

To: