Saint Paul ATARI Computer Enthusiasts

SPACE

An independent computer user group

SEPTEMBER 1985

Next meeting: Friday, September 20th, 7:30 PM
October meeting: Friday, October 11th, 7:30 PM

Bob Floyd . . .	President	487-2627	Jon Nelson . . .	Editor	484-9027
Bruce Haug . . .	Vice-Pres	774-6226	Max Feuer. . . .	Treasurer	483-3895
Frank Haug . . .	Disk Lib.	774-6226	Bruce Haug . . .	Cass Lib.	774-6226
Jim Schulz . . .	Paper/Ed. Lib.	537-5442	Steve Pauley . .	Secretary	560-2917

SPACE/TAIG Bulletin Board . . . 473-2897

AT THIS MEETING:

A BRODERBUND SOFTWARE REPRESENTATIVE

FEATURES IN THIS NEWSLETTER

A REVIEW OF THE ACE C COMPILER

PART III OF CHRIS CRAWFORD'S ASSEMBLY LANGUAGE COURSE

PLUS PRES' BYTES, D.O.M. NEWS, AND AUG. MINUTES.

BYTES FROM THE PRES.
by Bob Floyd

September Meeting - This month's meeting has been delayed 1 week so that a rep from Broderbund software could attend both the SPACE and TAIG meetings. (The TAIG meeting will be 1 week early, on the same weekend as us.) He will be showing software for the 400/800/XL/XE line. I have told him that we will have an ST at the meeting, but I don't know whether he will bring any ST software. There will be a doorprize. The Broderbund rep will also be on the Computer Talk show on the 21st - 9:00 am on WWTC 1280.

October Meeting - We will have Charlie Devine of Continental Marketing, ATARI's manufacturer's rep, at the October meeting. He will be answering our questions and giving an ST demo. I hope to convince him to bring a Casio keyboard for a Midi demonstration. Charlie was at the last TAIG meeting and gave a very good talk and demo. Among the things I found out are:
- a double sided 3.5" drive for the ST will soon be here (perhaps already).
- an 80 column (hardware) device will be available for the 130XE by 9/30.
- the price of the 800XL will probably drop a little more.
- the 10 megabyte ST hard disk will retail for $600 and should arrive by 9/30.
- the DMA port on the ST can have up to 8 devices daisy-chained to it. So, you can have a CD-ROM, hard disk, (extra RAM?), etc. on this versatile port. Incidently, the 1.33 megabyte/second speed of this port enables the ST to do things (such as CD-ROM) that no other computer can do. This includes the AMIGA, Mac & PC-AT. This sort of planning is why I think ATARI will succeed.
- the ST ROM chips will be available by 10/15. It will require 4 chips.
- Asteroids, Missile Command and Star Raiders will be released on the ST.
- the 260STD will retail for $500. This machine has the single-sided drive built in, but has only 256K (I feel foolish saying "only") and does not come with a monitor. It works on your TV, but only in low res 16 color, 40 column mode. However, you can still buy a monitor.
- the CD-ROM machine will retail for $599 and the encyclopedia disk for $60 to $70. The early versions will not have pictures. They are to ship in September.
- the dot matrix printer for the ST is now shipping. The letter quality version will be available soon.

I'm sure this has stimulated some questions from you. Be sure to write your questions down on paper and give them to me at the September meeting. I will forward them to Charlie so that he will be able to have the answers available at the October meeting.

Meeting Room Saved - Our terrific meeting room has been saved thanks to Phyllis and Gordy at Wizard's Work. It appeared that we were not going to be allowed to use the room, now that the room reservations changed to a different person at the "U". Gordy pulled some strings over at the University that allowed us to keep the present room. Thanks a lot, Gordy!

Election Process Begins - We will be electing a nominating committee of 2 members at this meeting. It will be their duty to nominate candidates for the club offices to be filled at the November meeting. They will report on their nominations at the October meeting, with elections planned for November. We will be electing a new president, VP, secretary and treasurer. In addition, we need a new newsletter editor. This is a volunteer post/appointment. Frank Haug, Jim Schultz (I finally spelled his name right), and I will continue in our posts as Disk Librarian, Paper Librarian and Analog Disk Librarian. I would like to add that if you are interested in being a club officer, please leave your name and phone number with me or one of the nominating committee members at the meeting. I will go over the election particulars at the meeting.

Mindtools Authorized - Mindtools, the new joint venture of Steve Pauley and Todd Burkey, is now an authorized ATARI dealer. Congratulations!

Mapping the ATARI, revised - This book has finally arrived and is in the club library. It is essentially the same book with the same pages, except there are new appendices to cover the XL and

XE and errata. I would buy it only if you don't already own the original version. The book is great, but not that much better than the original.

Dave Duberman - Dave is ATARI's user group coordinator. After seeing that SPACE was still not listed in the most recent ATARI Explorer, I sent Dave a copy of my previous 2 attempts to register the club with ATARI. He personally called me to apologize only 2 days after I mailed the letter. He was really a nice guy.

PILOT Ref Guide - Anyone that bought the $3.00 PILOT language from "Computer Mail Order" did not get the manual. I have 2 copies if anyone out there needs them. Call me before the meeting if you want me to bring one for you.

1024ST ? - I noticed on Compuserve the other day that someone has already figured out how to upgrade a 520ST to 1024K RAM. It is an upgrade for hardware hackers only at this point, but I think it foreshadows things to come.

D.O.M. News
by: Frank Haug

Well last month I had to get up on the ole 'soap box' because of the lack of SPACE member submissions to the DOM. Appearently no one read it, or if they did, it wisked in one ear and ... We are in dire need of member-written programs. (Some of you do write programs, don't you?) If you donate a program to the DOM library, and we use it, you will receive the DOM it is used on FREE!

Well this month, besides the DOM, we will be selling the A.C.E. 'C' language disk. There is no DOS on the disk because of lack of free sectors. One of the programs written in C source has a bad sector in it but we hope to get a fix by the meeting. This is not a tutorial on C. It will not teach you how to program in C. It does recommend a 'helpful book' on learning C, and explains some differences between this and the Deep Blue C. It will be sold for $4.00 at the meeting.

Now on with the programs on the September 1985 D.O.M.

1. KINGKONG.BAS - You must avoid jets and barrels as you run, jump, and climb open windows to get to the top. As soon as you get there, you race through Kong and reappear at the bottom. Then you do it all over again.
2. RACE - Use paddle to steer car while avoiding sides and other cars.
3. TAKE5 - An interesting game of skill for one player against the computer.
4. QUEST - A dungeons & dragons type game with graphic display and player saving features.
5. LINEGRAF.COM - Makes graphs and allows you to print them and add data to them.
6. PICTRIX.BAS - Basic source of PICTRIX.COM, I put it out in case anybody wants to see how the program does what it does, and/or modify it. (Note in line 5100 you might have to change the char code depending on your printer, this is for chars. per inch. For the Gemini 10X you replace CHR$(24) with CHR$(16). On any other printer if the spacing isn't right replace the 24 with th proper c.p.i. number.) The print option should work with most Epson clones.
7. PICTRIX.COM - This is a compiled picture utility program that allows you to load/save Koala Pad or Micro-Painter picture files, allowing you to convert one to the other. It also lets you make a hard copy of the file and, of course, puts it on the screen.

The ACE C compiler
by Curtis S. Gibson

ACE C was written by Ralph E. Walden of the Eugene, Oregon ATARI group (ACE), and is a descendant of the DEEP BLUE C compiler from APX by John Palevich. Both ACE C and DEEP BLUE C are descendants of a public domain C compiler by Ron Cain. The original C was written for inhouse use at AT&T Bell Laboratories by Dennis Ritchie. ACE C is semi-public domain, ACE is letting SPACE sell it to the members of SPACE to cover our production expenses. See Frank Haug at the September SPACE meeting for a copy.

The C language, and this means anybody's C, is like an odd mix of assembler and higher level structured languages, like Pascal. It has a strong structured design like Pascal. It has a set of simple operations, or commands, like assembler. The set of operations is not small, but they are not impossible to master in a short time.

From these simple operations more complex commands, called functions can be built. Functions are just like procedures in Pascal or subroutines in BASIC, they perform a task. A function consists of one or many function calls, and the function code. A function call is the function name and arguments, and is placed in the main program code and executed like an expression in BASIC. The arguments are the variable name(s) if any, and constant(s) if any, used by the function. An example: plmove(n,x[n],y[n],shape[n]). The code of the function is not listed in the main program, it is listed and compiled before the main program. The code of a function consists of the function name, arguments if any, argument declarations if any, an opening delimiter character(s), other declarations if any, executable expressions (these may be functions or simple operations), and a closing delimiter character(s). If descriptive function names are used, the main listing reads like a psuedo-code algorithm.

Examples of tasks for a function would be; input from keyboard, write to screen, printer output, read from disk, write to disk, performing a math function, controlling a loop, sorting a list, testing a condition, format a record, searching a list, increment-decrement a counter or value, move a value to a new location in memory, ect. You can build up a library of functions that you can use in other programs, and save writing the code again and again.

C uses two types of statements, simple and compound. A simple statement is a single line of code, like a line of BASIC. The compound statement is made of an opening delimiter character(s) braces on the main frame C or $(with ACE C, a block of two or more simple or compound statements, and the closing delimiter character(s) braces or $) with ACE C. The compound statement is like a line of BASIC commands separated by semicolons, they will all be executed (unless you jump over some of them). This compound statement gives C the power to be written in a structured way.

C has a control structure that is not available in most BASICs, the WHILE loop. The while loop is some what like the for loop, but it depends on a conditional expression rather than a counter to control the exit from the loop. Example; while(total(= 100) $(statement $) , this while loop would execute the next statement until the total became larger than 100.

ACE C and DEEP BLUE C are P-code compilers. When the program is compiled it is converted into instructions (called p-code) for a small machine language program called a run-time interpreter. This p-code is not 6502 machine language and is not as fast as an assembly language program, but it is faster than BASIC.

When you get your copy of ACE C make a listing of all the .TXT files, you can't learn C from them, but you can learn to use the ACE C. To use the ACE C first make a work disk. Put on the work disk the files: DOS.SYS, DUP.SYS, ACEC.COM, LINK.COM, FASTC.COM, CFORMAT.COM, ENGINE.OBJ, and ENGLOAD.OBJ. Work from the work disk only, if you make an error the programs can wipe out the disk (it can happen folks!). You will also need a word processer or text editor. The BASIC cartridge or Assembler Editor cartridge will not work as the text editor.

The first step is to design your program, think first-act later. Second, key it in with the text editor, saving it under the name you choose with the ".C" extension: NAME.C. Third, enter the DOS, type L (for Load File), press the RETURN key, type in ACEC.COM and press the RETURN key. The Compiler will be loaded and ask for the name of the file to be compiled. The compiler takes your file called the source code and translates it into p-code putting it under the filename with the extension ".CCC", this is called relocatable code, the program is not yet ready to run. The source code file is not changed, if you need to add to or change your program you will have to use the text editor on the ".C" file and recompile it. The compiler can take a few seconds to many minutes to compile a program, typically about a minute.

The forth step is to create a new file with the ".LNK" extension. This file is a list of files the linker will put together to make the executable code; the relocatable code of your program, the relocatable code of the function library if there is one, and the last file must always be the run-time interpreter ENGINE.OBJ or ENGLOAD.OBJ. Fifth step, load th file LINK.COM with a DOS option L like the compiler, and ent the file name. The linker will run for about a minute, creati the executable code file with a ".COM" extension. Finally to run the program do a DOS option L of the executable code file: FNAME.COM. The executable code can be moved to other disks and run from DOS option L.

A given program written in C will run faster than the same program written in BASIC. If you use a library of functions and use structured design, you will be able to write code faster with less effort.

Some books on the C language are: <u>The C Programming Language</u> by Brain W. Kernighan and Dennis M. Ritchie, <u>Going from BASIC to C</u> by Robert J. Traister, <u>Learning to Program in C</u> by Thomas Plum, <u>The C Primer</u> by Les Hancock and Morris Krieger, <u>The C Programming Tutor</u> by Leon A. Wortman and Thomos O. Sidebottom, and the one I use <u>The C Programmer's Handbook</u> by Thom Hogan. All of these books are in print and available at larger book stores.

==

<u>August meeting minutes</u>

By Joanne Floyd

The last SPACE meeting was held on August 9, with 90 people in attendance. During the President's report, Bob Floyd discussed the possibility that the club may need to find a new meeting room. He also reminded members that the officer election process begins in September. In the Vice President's report, Bruce Haug asked members to submit programs for the disk of the month, reminding us that the supply of public domain programs is limited. He also discussed his appearance on the Sunday morning computer show (<u>Computer Line</u>) on WWTC.

Phyllis and Gordy from <u>Wizard's Work</u> next reviewed their meeting with the new ATARI rep in the Twin Cities. At this meeting, they were shown several soon-to-be-released ST programs such as a wordprocessor, spreadsheet, and database master. They were particularly impressed by the RGB monitor. Gordy mentioned that ATARI seems to be committing itself to the ST and hardware development, leaving software development to third parties. Todd from <u>Mindtools</u> then demonstrated the 520 ST, pointing out such features as the screen editor, windows, and desktop programs. He believes that the 520 compares very favorably to the MacIntosh in terms of speed, useability, and general functioning.

With regard to new business, Bruce Haug said that the club is planning to purchase a printer ribbon re-inker that will re-ink 95% of commercially available ribbons. Members would be charged a minimal fee for this service. With regard to BBS information, he announced that the BBS password file has been sabotaged so people may need to re-register for passwords.

At the end of the meeting, Bob demonstrated his label-maker program. (See the last newsletter for a description of this program.) He also demonstrated two APX programs available from Antic. Mars Mission II is a cross between the arcade game Scramble and the classic computer game Caverns of Mars. King Tut's Tomb is similar to the arcade game Tutankhamen. Bob considers this second game a very good value at $15 because it includes several different series of mazes, many of which are quite challenging. It also includes a "tomb construction set" which allows you to develop your own mazes.

==

RIBBON RE-INKER

SPACE is considering the purchase of a printer ribbon re-inker. Before we go ahead with this purchase, we need the following information from the membership:

Members Name
Printer Brand
Printer Model
Ribbon # used

Please turn this information in ONLY IF YOU ARE INTERESTED in having your ribbons re-inked for approximately $1.00. Turn in to Bruce Haug V.P. at the Sept. meeting.

I will have samples of printing done with this re-inker at the meeting. It looks good to me, the sample printing was done with a ribbon re-inked 36 times.

REPRINTED BY PERMISSION.

CHRIS CRAWFORD ASSEMBLY LANGUAGE COURSE - LESSON 3: LOGIC

BOOLEAN LOGIC

A great deal of programming involves the use of Boolean logic. This is a standardized system for handling logical manipulations. It's sort of like algebra for logic. You must understand Boolean logic if you are to write assembly language programs, so let's get started.

Where algebra deals with numbers, Boolean logic deals with propositions. A proposition is just a statement such as "Fred eats worms." It can take only two possible values -- True or False. In our programs we seldom bother with broad and glorious propositions such as "Love is the universal language of truth" or "War is the extension of policy by other means". Instead, we normally deal with propositions such as "The joystick trigger has been pressed," or "There is a diskette in the disk drive."

When we use Boolean logic with a computer, we may think in terms of true and false, but the computer is actually working with 1's and 0's. We use the following convention: a 1 corresponds to a Boolean value of "true", while a 0 correspons to a Boolean "false".

Using this system we can represent propositions inside the computer. However, programming requires more than the mere representation of data; we must also be able to manipulate that data. This brings us to the Boolean operators. There are four common Boolean operations necessary for most programming practices:

1) Not

This is the simplest of Boolean operators. It takes a single Boolean value as an input and produces as its output the logical converse of the input. Thus, a true input yields output, while a false input generates a true input.

2) Or

This Boolean operator takes two Boolean values as its input and generates a single Boolean value as its output. The value of the output depends on the values of the inputs according to the following rule: If one input is true OR the other value is true, then the output is true. Otherwise, the output is false.

3) And

This Boolean operator is just like the or-operator, except that it uses a different rule. Its rule is: If one input is true AND the other input is true, then the output is true; otherwise the output is false.

4) Exclusive-Or

This Boolean operator is just like the or-operator, except that its rule is: If one input is true, OR the other input is true, BUT not both are true, then the output is true; otherwise, the output is false.

When we use the 6502 for Boolean operations, you must remember that the operations are eight bits wide. Instead of working with one bit at a time, we use all eight bits of a word in parallel. The bits in a byte are independent and do not affect each other in any way -- at least as far as Boolean operations are concerned.

The 6502 has three instructions for performing Boolean operations. These are AND, EOR, and ORA. The first performs an and-operation. For example, consider the following code:

```
    LDA     FISH
    AND     GOAT
```

This will first Load the accumulator with the value of FISH. It will then And the contents of the accumulator with the contents of GOAT. The result of the and-operation will be left in the accumulator.

The AND-instruction can use an immediate operand if you desire, just as the ADC-instruction can.

The EOR-instruction provides the exclusive-or operator. It works just like the AND-instruction. The OR instruction provides the or-operator in just the same way.

If you wish to obtain the NOT-operation, just use EOR #$FF; this will invert each bit in the accumulator. Because NOT is so easily reproduced with EOR, there is no special NOT instruction in the 6502.

APPLICATIONS OF BOOLEAN LOGIC

If you have any sense at all, you are probably asking, "What good is all this Boolean nonsense? What would I use it for?" Four applications are available:

1) Program Logic

Many times our programs encounter rather complex logical situations. The program must be able to load a file; if the FMS is in place and there is a diskette in the disk drive, and the diskette has the file we are looking for, or the file specification calls for a cassette load, then we will load the program. Many programming problems involve such Boolean operations, Keeping them straight is certainly a headache.

2) Masking Bits

Sometimes we need to isolate particular bits in a byte. For example, in Eastern Front (1941) I used the character value to store the unit type. The color of the unit was encoded in the upper two bits of the byte, the type in the lower six bits.

If I wanted to get only the unit type, I had to mask out the upper two bits. This I did with the following code fragment:

```
    LDA     UNITCODE
    AND     #$3F
```

The AND-instruction eliminated the upper two bits, leaving me with just the unit type. Bit-masking like this is useful in many situations. We use it frequently when we pack bits into a byte to save memory. It is also handy with input handling. If you want to read the joystick port, you frequently mask out the bits in turn to see which is active.

By the way, you mask out bits set to 1 with the AND-instruction. You mask out bits set to 0 with the ORA instruction. The logic is reversed.

3) Setting and Clearing Individual Bits

We also use the AND and ORA instructions to set or clear individual bits within a byte. This is most often useful for handling arrays of flag bits.

4) Folding Bytes Together

This little fragment of code will fold bytes together:

```
LDA     FISH
EOR     GOAT
AND     MASK
EOR     GOAT
STA     ANSWER
```

This is a magical piece of code. See if you can figure out what it does. Experiment with two values of MASK: $0F and $F0.

SHIFT AND ROTATE INSTRUCTIONS

The 6502 also has instructions that allow you to shift the bits around inside a byte. The first of these are the shift instructions. One, ASL, shifts a byte to the left; the other, LSR, shifts a byte to the right. Thus, the byte %01101011, when shifted left, becomes %11010110. Each bit is shifted one position to the left. The leftmost bit is rudely pushed right out of the byte and falls away ("Aaaaaaaarrrrrggggg!"). A zero is shifted into the rightmost bit. The LSR instruction does the same thing in the opposite direction.

Note that ASL also doubles the value of the byte, while LSR halves it. Two ASL's multiply by four; three multiply by eight. This makes it easy to do simple multiplication, but be careful with round-off error here. What happens if you try to multiply by 256? What do you get if you halve 3?

A variation on the shift instructions are the rotate instructions. There are two: rotate left (ROL) and rotate right (ROR). These function just like the shift instructions, except that the bit that gets shoved into the bottom is not necessarily a zero; it is the contents of the Carry bit. The bit that gets pushed off the edge of the byte goes into the Carry bit, so it is not lost. Thus, if you rotate either way nine times, you'll be right back where you started.

Rotate instructions are a handy way to get a particular bit into the carry bit where you can work on it. Conversely, once you get your desired bit into the carry bit the way you want it, you can put it back into a byte with some rotate instructions.

INCREMENT AND DECREMENT INSTRUCTIONS

The last instructions I will cover are the increment and decrement instructions. These allow you to add one (increment) or subtract one (decrement) from a memory location. These are not considered to be arithmetic operations so they do not affect the Carry flag, nor are they affected by it.

You cannot increment or decrement the accumulator, only RAM locations.

HELPFUL HINT

Reprinted from the 6/85 issue of HACK the newsletter Atari Anonymous of Rhode Island.

If you want to change something in your BASIC program that has been repeated many times, such as changing every GOTO to a GOSUB, their is an alternative way to just searching through the listing and manually changing every line. As you can imagine, this can be a "Royal Pain" with long programs. What you do is first LIST the program to disk. Then boot your word processor and load the LISTED file into memory. Now you can treat the file like a "text file" and do a "search and replace" to change your commands or whatever you want. When you are done "editing" your BASIC program save it back to disk. Now re-boot BASIC and "ENTER" the file into memory. List it and now you will see the changes you have made.

WANT ADS

It is possible for members to place 'Want Ads' in this newsletter. The ads may be for selling used hardware, used software, tutoring services, or just about anything that has to do with Atari. The rates are as follows:

6 Lines	$1.00	216 Letters
1 Line	.25	36 Letters

The following is a list of advertising rates for vendors or individual members.

Full page	$18.00	7-1/2" X 10"
Half page	10.00	3-5/8" X 10"
Half page	10.00	7-1/2" X 5"
1/4 page	5.50	3-5/8" X 5"

All advertisements must be paid for when they are submitted. Deadline for ad placement is two Mondays before the meeting. To place ad or for more info, call the editor.

B & D HOME COMPUTER SUPPLIES
1697 E. HOYT AVE.
ST. PAUL, MINN. 55106
(612)-774-6226

SSDD FLOPPYS
10 FOR $8.90

BUTTONS $.50 EA
350 TO CHOOSE FROM
OR
ORDER CUSTOM MADE

SPACE BUTTONS
WITH YOUR NAME $.50

St. Paul Atari Computer Enthusiasts
2589 Fisk St.
Roseville, MN 55113

ST PAUL, MN
SEP 17 PM 1985
550
USA

Bob Rhode
630 Ashland Av.
St. Paul, Mn. 55104

Meeting site:
U of M St. Paul campus
Office Classroom Bldg. rooms B-35 & B-36
St. Paul, Minnesota