ATARI COMPUTER ENTHUASISTS

OF COLUMBUS
NEWSLETTER
JULY 8, 1985 ISSUE 21

Published by
Atari Computer Enthuasists of Columbus, Ohio

for ACE of Columbus membership. Dues are on an annual basis and entitle the members to all club benefits (Newsletter, Disk or Tape of the month, group discounts, etc.). Monthly meetings, at St. Francis De Sales High School, 4212 Karl Road, Columbus, Ohio are open to nonmembers. Meeting dates are the second Monday of the month at 7:30 pm. Disk of the Month distribution from 7:00 to 7:30.

PRESIDENT
Bill Eckert
6632 Lisamarie Road
Columbus, Ohio 43229
614-891-9785

VICE PRESIDENT
Joe Blue
6360 Sunderland Dr.
Columbus, Ohio 43229
614-436-7339

MEMBERSHIP CHAIRMAN
Tim Adcock
7544 Satterfield Road
Worthington, Ohio 43885
614-764-9492

TREASURER
Mike Compton
1342 Gumwood Dr.
Columbus, Ohio 43229
614-885-3757

DISK LIBRARIAN
Charles Lusco
4624 Channing Ter. Apt. C
Columbus, Ohio 43232
614-863-4016

CASSETTE LIBRARIAN
Roger Stultz
2162 Eden Ave.
Columbus, Ohio 43224
614-471-5573

NEWSLETTER EDITOR
Norman Knapp
1222 Norton Avenue
Columbus, Ohio 43212
614-291-2849

ADDRESS ALL MAIL TO
ACE of Columbus
P.O. Box 849
Worthington, Ohio 43085

SECRETARY
Kathy Fellows
1719 Shaton Ct.
Worthington, Ohio 43085
614-889-4763

CASSETTE LIBRARIAN
Don Bowlin
230 Orchard Lane
Columbus, Ohio 43214
614-262-6945

PROGRAM CHAIRMAN
Don Noble
614-890-4333

THE ATARI ASSEMBLER EDITOR
(Part 2)
by Charles Brown

In the first article I tried to introduce you to machine language programming. In this article I will try to give you a little more information on it.

In order to program in machine language, you have to know something about the different numbering systems. In computer programming there are 3 numbering formats. These are decimal, hexadecimal and binary. In case you don't know decimal is a base 10 system. Hexadecimal is a base 16 system. Binary is a base 2 system. I have included a short chart comparing these three systems:

DECIMAL	HEXADECIMAL	BINARY
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
20	14	10100
30	1E	11110
50	32	110010
75	4B	1001011
100	64	11001011
150	96	10010110
200	CB	11001000
225	E1	11100001
255	FF	11111111

The knowledge of these 3 numbering systems is very useful. Without them, it would be very hard to understand what is happening.

When you start entering a source program in the editor mode, the first line of code tells the computer where to store the object code in location 600 (hexadecimal). This is converted to 1536 in decimal form. I bet you have heard of this location before. One of the most important features of maching language is the accumulator; a special register that is used very frequently in a program. To see how this is used I will show you how it is used. Let's say that we want to put the value 64 into memory location 710 decimal. This would change your screen color to red. In Basic you wuld simply use Poke 710,64. In machine language you would first load the accumulator with number 64. The command

would look like this: LDA #64. Then you would store the contents of the accumulator into memory location 710 (the screen color location): STA 710. In this example you could perform tis task in Basic with one command. In machine language it would take two commands. Anytime you want to change the value of something, you almost always use the accumulator.

In this article I have tried to show you a little more about machine language programming: understanding the numbering systems, how to tell the computer where to store your program and the use of the accumulator. These 3 things will help you understand what machine language is all about.

==========

THE EDITOR'S COLUMN
by Norman Knapp

My column will be rather short this month, but there are a couple of items I would like to bring to your attention. In September ACE of Columbus will be electing officers for the following year. Please seriously consider running for an office. You do not have to be an "expert Atarian" to do an adequate job. ACE of Columbus needs offiers who have the vision to set up and run programs to satisfy the needs of the membership.

The current officers are working on a constitution which will facilitate transfer of leadership. This document better define the operation of ACE of Columbus and the part each officer has in its operation. The constitution is also required to satisfy legal requirements for the status of ACE of Columbus as a nonprofit organization.

Don't forget that the next meeting of the SIGs is July 25 at the State Savings Bank in Worthington. Please read the announcement on the last page of the newsletter.

SOFTWARE REVIEW: MICRO LEAGUE BASEBALL

Reviewed by Charles Brown

There is a pretty good game on the market for our Atari; Micro League Baseball put out by the Micro League Sports Association. I have to say it is definitely a different game from the others. I actually hate baseball, but I enjoy playing this game.

It is not like joystick controlled game that rely on quick reflexes. This game doesn't even use a joystick. It is played with the keyboard. When you load the game you are given 3 options. One is to start a new game. Another is to continue a previously saved game. The last option is to see a demonstration of the game. If you choose the new game option you will be shown a choice of different teams. These are actual teams that have played major league baseball. Some of these teams are supposed to be among the best in baseball. They include the 75 Reds, the 73 Athletics, and the 27 Yankees. For those who watch baseball I am sure you will recognize these teams. You choose first the visiting team. Then you decide if you or the computer will manage it. Then you choose the home team. Once again you decide if you or computer will manage it. This game can be played against another person or the computer. I think you can even have the computer compete against itself. Once you have made your choices then you will be given an option to use a designated hitter or not.

After the disk drive runs for a little while, you will be shown the players. Then you will be shown the visiting team's pitching staff. This includes the starter and the bench. You can leave the starter as is or choose any pitcher on the bench. It will show you a complete chart of the pitchers' stats to help you decide. Once you have made your choice you should exit by hitting the escape key. Then you will do the same for the home team pitcher. Once again you hit the escape key to leave this mode.

After you are finished with the pitchers, you will be shown the visiting team's lineups and their stats. This includes the starters and the players on the bench. You can leave the lineup as is or change it. If you decide to change it you have several options. You can replace the starters with players on the bench. You can also change the batting order for the starting lineup. You can even change the fielding postions for the starting players. When you are studying the palyers you have 3 different screens of the stats to help you decide. When you hit the escape key, you will be able to do the same for the home team. When you hit the escape key for the 4th time you can then start the game.

The defensive team has many options to choose from. You may choose from 4 different pitches. You have several different defensive options to use such as pitch out, corners in, or bring the whole infield in to cut the run off at the plate.

The offensive team also has some options to choose from. When no one is on base, the batter has 2 choices. He can either swing away or make a surprise bunt. If there is someone on base then the batter has more options to use. He can either hit with aggressive base running, hit with safe running, have a runner

steal, or do a hit and run.

From what I understand when both the offensive and defensive options are chosen, then the computer will use the pitcher's stats and the batter's stats under that particular situation and make a decision as to what will happen. You will actually see the ball being hit, the players catch the ball, and the runners run the bases. You can even see the players slide into the bases. What is so amazing is that the computer knows who the players are. Say for instance if you fly out to center field the comuter will tell you who caught the ball. The computer also knows who is running the bases. For example, let's say you are the 75 Reds. The first 2 players get on base and the 3rd man up hits a home run. Then as you see the players run the bases, the computer will tell you that Rose has scored. Then it will tell you that Morgan scored followed by Bench. With this game hou can even bring in a relief pitcher if your starter gets in trouble. The offensive manager can bring in other players too. He can use a pinch hitter or a pinch runner.

In trying to be honest I do see 2 draw backs to this game. The first is that if 2 people are playing they both have to use the keyboard. This way one can see what the other is going to do. For example if the defensive manager sees the offensive mamager hit 3 for a steal. Then most likely the defensive manager will hit 5 for a pitchout. Unless he has an awfully poor catcher or a world class sprinter on base. The offensive manager wouldn't have a chance. The other flaw is when you play against the computer. You have a certain batter up against a certain pitcher. Then you hit a key to make your offensive choice. Then all of sudden the computer decides to bring in a different pitcher. You are then stuck with the choice you had make for the former pitcher. I think that if the computer decides to change pitchers, then you should be given a chance to make your offensive choices again.

I also feel there is a pretty good future for this game. They already have other data disks for this game. For example, they have a disk out for the complete 84 season. It contains the players and the stats for all 26 teams from the last year. There are also other disks available. These will give a wide choice of other teams to choose from, both past and present. There is also a general manager's disk available. With this disk you can trade players back and forth between teams. I even think that you can create your own team. This will make it a really interesting game. You could almost call this a game generator type game, a type that I really enjoy.

Even though I find the game of baseball boring, I really enjoy playing this one. I strongly suggest that you should look into this one.

Machine Language: What Does It All Mean?

By Dr. Warren G. Lieuallen

In previous discussions, we have examined both how to load and store a machine language program, and a simplistic view of how the hexadecimal codes produce any meaningful results. This article will now deal with the specific commands available in the assembly language, and how they work to comprise a complete, working program.

To make machine language programming a little easier for human beings, it was decided that the numerical codes which represent the actual commands would be given an associated "mnemonic" word, or label, describing the command. A mnemonic is simply something which makes remembering easier, due to associations which may be made between the mnemonic itself and the item being remembered. In other words, the command which is represented by the number 76 (or "4C" in hexadecimal) is also indicated by the mnemonic label "JMP", which stands for "jump". This JMP command is very similar to the GOTO command in BASIC, which is essentially a "jump" in the program's execution. So, by remembering "JMP", you can remember how this command functions (just like remembering "GOTO" in BASIC.).

All the mnemonic labels have been reduced to three letters in length for the sake of consistency. Therefore, it's not too difficult to create a program which will accept the three-letter mnemonics, and convert them into the appropriate numerical codes, thereby saving us the trouble of looking all of them up in a table and performing the conversion ourselves. This is exactly what the various assembler/editor programs do (of course, they also do a lot more, but that's another story, or at least another article!). In this way, all we need to remember is the mnemonic labels, and not concern oursleves with the actual numerical, hexadecimal codes.

Previous newsletter articles by Charles Brown have discussed that everything your Atari does is the result of changing certain memory locations, either with PEEK's and POKE'S, or just with other BASIC commands, which alter the memory locations for you (whether you realize it or not!). Machine language is no different; everything happens because of changes in the memory locations. The only difference is that in machine language, you do have to be aware of these changes, because it is your machine code which will produce these changes. BASIC's extra "helping hand" is no longer available, and much of the housekeeping functions must be dealt with by you directly. In actuality, this is not as bad as it sounds.

The simple screen-fill routine we discussed last month

will serve as a good example. This routine was made up of thirty-four numbers, which correspond to the thirty-four commands in this program. Here is the source code for that program (The "source code" is the set of mnemonics, understood by your assembler, and hopefully by you as well; the "object code" is the set of numbers generated by the source code, and saved as a binary load file.):

```
 Byte 1= PLA         - get # of arguments
         CMP   #$1   - is it 1?
         BNE   (#-2) - if not, kill program (infinite loop)
         PLA         - get MSB of argument (not needed)
         PLA         - get LSB of argument (character code)
         TAX         - store it in X
         LDA   $58   - store contents of $58 in A and...
         STA   $CC   - store A in location $CC
         LDA   $59   - store contents of $59 in A and...
         STA   $CD   - store A in location $CD
         TXA         - retrieve the argument into A
         LDY   #0    - let Y=0
  LOOP-> STA   $CC,Y- store A at location ($CC+Y)
         INC   $CC   - increment location $CC
         BNE   (#-6) - if $CC<>0 then go back to loop
         INC   $CD   - increment $CD
         LDX   $CD   - store location $CD into X
         CPX   #$A0  - compare X with $A0 (160)
         BNE   (#-14)- if X<>160 then go back to loop
 Byte 34=RTS         - return to BASIC
```

Depending on your background, this listing may or may not make any sense! What is shows, though, is that this program works exactly like its BASIC counterpart--it places the internal code for an asterisk in the memory locations corresponding to the graphics zero screen display. It does this by first checking to see that only one number was recieved (this number [the argument] is the internal code for an asterisk, which you specified in the USR call from BASIC, eg. X=USR(1536,10).). It then transfers this value to the X register (a special memory location used to hold values needed in the program, just as variables do in BASIC), and then gets and stores the pointer to the beginning of screen memory (this is location $58 in hexadecimal). It then retrieves the character code, and begins to place it in the memory locations. The rest of the program simply counts how many locations have been filled, increments the screen pointer and stops when one screen-full is done! A slightly more detailed discussion can be found in "Atari BASIC Faster and Better", by Carl M. Evans (published by IJG Enterprises).

So, although you may not think so just yet, that's all there is to programming in machine language. While it certainly seems harder than other programming languages at first, with a little practice and familiarity, you'll find that it really is just another language in which programs may be written. Best of luck!

Book Review: "Presenting the ATARI ST", by L. English and J. Walkowiak, Abacus Software, 1984, $15.

Reviewed by Norman Knapp

The introduction of a new computer has usually been accompanied by the introduction of new books and magazines. To my knowledge, this is the first book about the Atari ST computers. Since there is a lag time of several months in publishing books, some of the material in this book may not be current. The status of the GEM (Grapics Environment Manager) operating system is uncertain as of this date. Is GEM going to be in RAM or ROM or is it even going to be released with the first STs? This book assumes that GEM will be in ROM and illustrates the friendliness of the ST. More about this later.

With introduction of the ST, Atari has broken continuity with past models by basing operation of the ST around the 68000 microprocessor. There are two reasons for going to the 68000 processor: greater speed and direct access to larger amounts of memory. In the first chapter the authors present a brief comparison between the 8 bit (6502, Z-80, 8088,8086) and 16 bit microprocessors. The pin layout, register set, operands, and instruction set for the 68000 processor are presented as well as its advanced features. The capabilities of the 68000 are that it is suitable for microcomputer systems with lots of memory, supports separation between operating system and user programs, supports multi-user applications, and implements modern high level languages.

Perhaps the most publicized feature of the ST has been it architecture. I'll just briefly mention it main features: 192 K system ROM, 128 K ROM expansion cartridge, and 512 K RAM for the 512ST. The 128 K ROM can be used for either operating system expansion or for user programs. In addition, 32 K of address range is reserved for peripherals. Of the 512 K RAM, 32 K is required for screen RAM. The peripheral interfaces and devices ST supports are:

1. Two 3.5 in disk drives.
2. DMA interface for a 10 MB hard disk.
3. Centronics parallel printer interface.
4. RS 232 interface.
5. MIDI interface for control of external music synthesizers.
6. Two joystick ports, one for a mouse.
7. Video connections for B/W and RGB color monitors.
8. Intelligent keyboard.

Just a few words about the Direct Memory Access (DMA) interface. This is what I consider to be the most important feature of the ST. Data is transferred (bypassing the 68000 processor) between mass storage and the computer at 1.33 MB per second, the same speed at which the 68000 processor reads data from memory. In effect, at transfer rates this high, the entire hard disk memory can be considered to be part of RAM (virtual memory).

The chapter on the ST operating system was a little puzzling

since it started out with a discussion of CP/M (Control Program for Microcomputers) until it was pointed out that two sections of CP/M had rewritten for the 68000 microprocessor. This new version of CP/M is called TOS (Tamiel Operating System) by the authors and is stored in ROM; the user loses no RAM to TOS. If the new ST ower is familiar with the DOS commands of DOS-XL and OS/A+, he will find the transition to TOS fairly easy.

Many Atari users will miss the familar DOS 2 menu and its ease of use. However, the ST has built GEM into the user interface. GEM takes the CP/M functions such as formatting a disk, looking at the directory, copying files, erasing files, system status, etc. and replaces them symbols of commonly used office items: desktop, manila folders, blank paper, pens, erasers, trash cans, calculator clock, copy maching, etc. These symbols are manipulated by a technique called windowing using a track ball device called a mouse. The objects referred to above are called icons. Moving the mouse on a smooth surface moves an arrow on the screen to the desired icon where it is activated by clicking the mouse button. The the icon is then displayed in reverse. If for example the arrow points to a disk drive icon and the mouse if clicked, then when the arrow points to FILE and the mouse button is held down, a pull down menu appears. After the arrown is moved to the OPEN choice and the button is released, the pull down menu disappears, another window appears displaying the contents of disk as icons. Moving the arrow to VIEW and holding the mouse button down makes another pulldown menu appear. Moving the arrow to SHOW AS TEXT and releasing the button makes the menu disappear permitting the file to be displayed as text. After the user is finished with a window, it closed by pointing the arrow to the box in the upper left corner of the window and clicking the mouse. The window disappears. The windows can be scrolled to display more icons than can fit on a single screen. The window size can also be changed. Windows can overlay one another. This is just one example of how GEM operates. The authors illustrate several other uses as well as more advanced features of GEM. On the early STs, GEM will be in RAM and stored on disk. Later versions of GEM will be on a ROM chip thus freeing more RAM for user applications.

For the more advanced user, those who wish to create their own programs, the ST is also an exciting machine. TOS has built into it a simple line oriented editor for creating source files for assemblers and compilers, an assembler for conversion of assembly language source programs into machine code and DDT (dynamic debugging tool) for testing machine language programs. I'll have mention here that the editor, assembler, and DDT were described as CP/M commands. Let's hope that these commands are still in TOS.

The higher levels languages which are being released along with the ST are BASIC and Logo. The authors do not elaborate on the implementation of Basic for the ST, but do their best to convince you that Logo should be seriously considered as a versatile programming tool for the ST. You don't need to buy this book to investigate this for yourself. Just look at the back issues of our newsletter containing the Atari demopaks which deal

with Logo.

Is this book really an "in-depth look" at a sensational new computer. I do not think that this book discusses the features of the ST "in depth" because I could not use it as a reference book. This book is more in the style of an incomplete "overview" of a sensational new computer. The print font used throughout this book is large and there is quite a bit of space between lines to stretch the text over just 196 pages. There are also quite a few one and two sentence paragraphs. In addition, the time limit of three months the authors set for themselves between receiving their ST and the delivery date of the manuscript indicates to me that much material may have been left out. There is also the possibility that some material may have been deleted during the translation from German to English. There is written on the spine "Vol. 1" which implies that a second volume may follow; this is not indicated in the text.

I think that it is good that an overview has been written for the ST. The neophyte user such as myself needs one to view the forest before plunging into the bushes and trees. Let's hope that the manuals that will come with the ST and its accompany software fill in the gap.

==========

PRINT SHOP Tip
by Norman Knapp

There is in PRINT SHOP an editor for creating your own graphics symbols. This editor is easier to use than the explanation given in the manual. The cursor may be moved about the screen using a joystick rather than the cursor control keys. The user may toggle between the Drawing and Erasing modes by pressing the D and E keys. The important point to remember is that the current mode is activated only while the firing button of the joystick is pressed; this procedure makes it much easier to do rough drawing and editing. For more precise cursor control, the user may revert back to keyboard cursor control, activating the current editor mode when required by pressing the firing button.

The Atari logo on the cover of this issue was created using this technique.

ANNOUNCEMENT!

The Special Interest Groups (SIG's) of the Atari Computer Enthusiasts of Columbus have been quite popular since their formation several months ago. Currently groups dealing with programming languages, telecommunications, and adventure gaming are in full swing. Another part of the SIG meetings which have been very popular have been demonstrations given on the use of AtariWriter (by Kathy Fellows) and on printer usage and utilities (by Dr. Warren Lieuallen). These informal, back and forth discussions have helped many people understand how to use their Atari to its fullest potential. Because these demos have been so well-received, and so much fun, I'd like to offer the following list of other possibilities for some newly-forming branches, or demonstrations to the Special Interest Groups:

A. An introduction to machine and assembly language

B. A FORTH tutorial and demonstration

✓C. A graphics utilities demonstration, including simple animation and player-missle graphics

✓D. Various approaches to music on the Atari systems

✓E. Basic spreadsheet operation and uses

✓F. Intermediate to advanced disk analysis utilities, and repairing "mangled" files

G. Writing your own BASIC and machine language adventure games

And More!

If even slight interest is shown in any one or more of these topics, Warren would be more than happy to begin scheduling the sessions, either in conjunction with the regular ACEC meetings, SIG meetings, or as a separate ACEC-sponsered "mini-course".

Please contact Warren during the "intermission", or after the meeting tonight if you would be interested in any of these groups, or if you have some other area in which you would like to participate in a demonstration and/or discussion.