DR. WARREN G. LIEUALLEN

ATARI
COMPUTER ENTHUASISTS
of COLUMBUS, OHIO
NEWSLETTER

February 11, 1985

ISSUE 16

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. Upcoming meeting dates at 7:30 pm are

DATE	TOPIC
February 11, 1985	
March 11, 1985	
April day?, 1985	
May 13, 1985	
June 10, 1985	
July 8, 1985	
August 12, 1985	

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
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

THE EDITOR'S COLUMN

At several of the officers' meetings, the topic of short courses on various topic has been disucssed. If you are interested learning more about the Atari system, Bill Eckert will have a sheet you can sign after the meeting. Please indicate subject(s) that interest you and whether you would like to be a teacher or student. The courses would not be on the regular meeting night, but at a place and time yet to be determined.

You'e probably noticed by now that this issue of the newsletter is considerably thinner than last month's, the longest published to date. There are several causes for this situation. The most obvious is that last month we published an index of the programs on all the Disks of the Month and released a very long Atari DEMOPAC. The other reasons are that this month, I received only one article, by Charles Brown, a regular contributor. Unless another DEMOPAC turns up along with 2 or 3 articles, a issue cannot be put out next month. If there is no DEMOPAC, there should be 4 or 5 articles, each at least a page long. We could also use filler material such as tips on software use, programming, and hardware.

There is a great variety of topics to write about: the new hardware Atari will be releasd soon, reviews of software, programming topics (read the enclosed articles), and your personal observations and opinions about personal computers. There are a large number of Atari owners without modems; if you see something on a bulletin board that you like to share, pull it down and send it to the editor. The same holds for magazine articles if copyrights are not a problem.

Eratta

Eratta is a Latin word used to designate printer's or writer's errors in a publication. There are several in the DOM index published last month. The columns on the first and last pages of the DOM index were interchanged. To save the expense of another page, the directory listing for DOM #27 was placed at the bottom of the last page of the DOM index. My apologies for the confusion this may have caused.

Norman Knapp, Editor

ATARI MEMORY AND IT'S USES

When you program your Atari,what you are actually doing is accessing it's memory. The Atari computer is nothing else but a large collection of memory locations. Unless you are doing a lot of simple things you will have to know the uses of many memory locations. You will also have to know what these locations will do under various situations.

For example, I was using Basic XL a while back. I was trying to save a page of text by using the famous MOVE command. I first moved my page to location 20000 (decimal). I used this location as a storage space. Then I went to move the page back to location 40000 (decimal). This is the lowest screen location for the Graphics 0 screen. When I typed in the command I made a big mistake. I forgot and left out a zero. Instead of moving the page to location 40000. I moved it to location 4000. This would be the same thing as pokeing a value to that location but with the move command you can move a lot of bytes at one time. When I tried to save my program to the disk I got a DOS error telling me that my drive won't respond. Naturally I thought that there was something wrong with my drives. Instead what I did was by moving my text to location 4000 instead of 40000. I wrote over the disk operating system that was in memory when the disk was booted. By writing over DOS in memory I scrambled it and as a result the DOS system could not recognize my command. As a result I lost my program and had to start over. Luckily it was a small program and it wasn't anything serious. I could easily restore DOS by turning off the computer and rebooting it. This is just one example of what can go wrong if you miss just one digit. You can easily do some serious damage with the wrong numbers in the wrong place. Luckily for us as far as I know you can not do any permanent damage to the computer by typing on the keyboard. If you do put something in the wrong place and scramble everything. All you have to do is turn off the computer and turn it back on again. This should set everything back to normal again. Unfortunately this method will also force you to lose anything that you had typed in. If you are changing memory locations it may be a good idea to save your program off before you run it. That way if you do scramble something you can simply turn the computer on and off again and load your program back in. This will save you a lot of work retyping your program back in.

If you have tried your hand at programing. You may have heard of the "peek" and "poke" commands. The peek command is like the name implies. You are peeking or looking at something. When you tell the computer to peek a memory location. You are telling it to show you the contents of that location. Or you could say that the computer is showing you what is being stored at that location. It is safe to look or peek any location you want. Peeking a location is harmless to the computer. Even If you type in the wrong number for the location you wanted. The computer would simply show you the contents of the wrong location. You would not damage anything.

On the other hand if you want to poke something you had better know what you are doing. When you poke a location you are putting a value into it. In other words you are changing it. By changing a memory location you are changing the way the computer will work. If you put a value into the wrong location by mistake

you could really mess things up. So remember that peeking or looking at a location in memory is harmless. On the other hand if you poke or change a memory location you could really cause some headaches.

I hope that I have given an insight into how the computer works. If done properly programing can be an enjoyable experience. If you start putting the wrong things into the wrong places it can be a real pain in the neck.

byCHARLES W. BROWN

FINDing

The Basic XL MOVE command discussed by Charles Brown in the accompanying article possesses some of the characteristics of the POKE command. If it is desired to just inspect the contents of a single memory location, the PEEK command permits us to do this. Basic XL has another command which permits inspection of a number of consecutive memory locations, the FIND function.

Before we consider a rather simple use of the FIND function, let's look at its form:

```
Find(Long$,Short$,Strt)
```

This function looks through a long string, Long$, for the presence of a short string, Short$, starting at position in Long$ given by number Strt. If Short$ is present in Long$, then the location of its first character will be assigned to the numberic variable, Lctn as shown in the following example:

```
Lctn=Find(Long$,Short$,Strt)
```

If Strt is 0, the all of Long$ is searched. If Strt is 10, the first 9 characters of Long$ will not be compared with Short$. If Short$ is not in Long$, the value of Lctn is set to 0; this property of Find permits it use in the If ... Then ... statement:

```
150 If Find(Long$,Short$,Strt) Then Goto 200
160 ? "Short$ is not in Long$"
```

The English language translation of the two line of code above is that control is transferred to line 200 if Short$ is in Long$; otherwise line 160 is executed if Long$ does not contain Short$.

Now that we know some of the properties of the Find function, let's use it to check string input for a Basic XL program. In the context of a larger program, we want to enter a date and check the input for keyboard entry of a valid month name:

```
10 Dim Monthyr$(36), Month$(3)
20 Monthyr$ = "JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"
100 Print "Enter Month, 1st 3 letters ";:Input Month$
110 If Len(Month$)<>3 Then Goto 100
120 Lctn =   Find$(Monthyr$,Month$,0)
130 If Lctn=0 Then Goto 100
140 If Int(Lctn+2)/3)<>(Lctn+2)/3) Then Goto 100
```

```
150 Print Month$;" is a valid month name"
160 Stop
```

In line 10, the strings subsequently used are defined and dimensioned. In line 20 the first 3 characters of each month name are stored in the string Monthyr$. Line 100 prints a prompt asking for a month name and then stops execution until the user responds. Lines 110 to 140 evaluate the user's response to see if the following conditions are satisfied:

1. Only 3 characters are to be entered; line 110
2. The string of 3 characters must be in Monthyr$; lines 120 to 130
3. The entry must be a valid abbreviation; line 140

The third condition arises because the Find function looks at every possible string of 3 characters; not just those we are interested in which start at positions 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, and 34.

If you do not have a Basic XL cartridge or Basic A+, you may be wondering now how this is done in Atari Basic. This is accomplished by replacing lines 110 to 140 with the following lines:

```
110 LCTN=0
120 FOR I=1 TO 34, 3
130 IF MONTH$=MONTHYR$(I,I+2) THEN LCTN=I
140 NEXT I
145 IF LCTN=0 THEN GOTO 100
```

All three of the conditions for proper input are satisfied since only the segments of MONTHYR$ looked at are the month names.

The two solutions for string finding given are both equally valid. The advantages for the Basic XL approach are that the coding is more understandable and that the comparison is much more faster than using the Atari Basic solution to this problem. However, speed is essential only when you are trying to FIND a substring in a very long string. In the example that we've looked at, MONTHYR$ is not very long so that the time required to execute the statements in Atari Basic does not require the user to wait for finishing of the test.

If the programmer can write error catching routines for keyboard input, the user can recover from his mistakes and continue without having to start over again and again and again. With entry of strings, the validation process can be tedious to program but worth it in the long run for the user. The validation process can also used to speed up entry of strings. For example, if a library of frequently entered words has been built up it may be possible for the used to enter a long like "magazine" just by keying "ma". If "magic" is also in the library, "maga" or "magi" would have to be keyed in before a valid word could retrieved from the library. If the word is not in the library, then the new word could be added to the library for future entry.

Most Basic approaches to checking keyboard entry that I've encountered have been have for entry of numerical data. A simple test for entry of numeric data would the following short program:

```
10 TRAP 10:PRINT "KEY IN A NUMBER OR LETTER ";:INPUT X
20 PRINT "You keyed in the number ";X:GOTO 10
```

Key it in on your Basic or Basic XL screen, then key RUN <return> and follow instructions. If you do not understand some of commands, please look them up in an the Atari manual which came with your computer. I just did not have time to give explainations of all the Atari Basic commands used in these short programs. One last comment on the programs in this article: the first letter of Basic XL commands, functions, and variables is upper case while the following letters are lower case when listed but may be keyed in either case; all letters in Atari Basic commands, functions, and variables must be keyed in upper case.

My original goal when I started writing this article was just to explain how the Basic XL FIND string function worked. By the nature of the example I chose to illustrate use of the Find function, I encountered a topic which I considered quite important; design of Basic programs so that the user can recover from his mistakes.

by Norman Knapp