ATARI
COMPUTER ENTHUASISTS
of COLUMBUS, OHIO
NEWSLETTER

July 9, 1984

THE EDITOR'S COLUMN
THE FUTURE OF ATARI
A SIMPLE SCREEN DUMP
PLAYER-MISSLE GRAPHICS

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, in the basement of State Savings, 6895 N. High Street, Worthington, Ohio are open to nonmembers.

Upcoming meeting dates at 7:30 pm are

DATE

July 9
August 13
September 10
October 8
November 12
December 4

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

VICE PRESIDENT
Donald Noble
7536 Northfield Court
Reynoldsburg, Ohio 43868
614-866-4587

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
Sheldon Wesson
444 North Pearl Street
Granville, Ohio 43823
614-587-2716

CASSETTE LIBRARIAN
Larry Fletcher
12388 Oak Drive
Orient, Ohio 43146
614-877-9530

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

THE EDITOR'S COLUMN

This is my third issue of ACE of COL's newsletter. However, this is the first opportunity I have taken to speak to you in this manner. It has been our policy that our publication would print articles written by the membership. When I took on the editorship in the middle of the year, it was with the understanding that I would edit rather than write the newsletter.

By necessity, we've had to back down from these high standards. This has been become necessary since there have been fewer submissions of articles than in our first year of publication; even then many of the articles were written by the staff. Yes, there was a staff of one or two people then in addition to the editor and a proofreader. The job formerly done by 3 or 4 people is now being done by just one person with some assistance.

The other way in which our standards are being diluted is that we are reprinting material from other sources; DEMOPAC's published by Atari. This doesn't bother me too much as long as the material is useful and of good quality.

Circulation of our publication is only to members; if you want to write articles just to sharpen your writing skills and are nervous about a wide distribution of our publicatio, you have nothing to worry about. On the other hand, if you would like to submit articles to national publications as well to ACE of Columbus, I don't think that ACE of Columbus should retain all rights for subsequent publication of our articles. I have not published in my issues a retainer of rights of subsequent publication and will not. If the author can sell his articles, that's great since writing takes a lot of effort.

Now that I've touched upon the subjects of quality, quantity, and publication rights, I would like to throw out a few ideas for articles.

1. New hardware: R-VERTER (modem adapter); MCPEN (light pen); ATR 8000 (CP/M, MS-DOS); SixPack; AMDEK hard disk drive;

2. New software: ACTION; SUPER-TEXT; SPARE CHANGE; DECISIONS; AGENT USA;

3. Book reviews.

4. Foreign language instruction.

5. Personal opinion on special topics concerning computers.

6. How you are using your computer: education, relaxation, spread sheets, genealogy, communications, gambling, investments, writing, professional purposes, art, etc.

7. What would you like to able to see computers used for?

The listing should give you some topics that you may write an article about.

In this issue we have a personal opinion article by Charles Brown. Read it carefully. Let's have an open discussion about what ACE of Columbus could be doing in the next year.

Your editor has submitted an article about how a program was written. I am not trying to make programmers out of everybody, but I am trying to show in a small way what is involved in writing a program.

We also have DEMOPAC #5 published by Atari.

NFK

THE FUTURE OF ATARI

I often ask myself lately how Atari is doing, go around to many of the computer stores in Columbus, and try to find Atari products. Instead, I find a lot of items from other computer systems, but not from Atari.

I have to ask myself if there is anything that we as Atari users group can do? I know that we can't move mountains, but we can try to accomplish. something.

Maybe we could check out other system users groups to see what they are doing. I have, seen where a Commodore users group has special days at Software City on 161. They have a meeting and open demonstration on the Commodore computers and accessories. The public is invited to come in, ask questions and see various demonstrations on that line of equipment. This way the public can see the computer work and actually see what they can do. They also can see the various software and accessories in action. This way the people will better understand what they are looking at and will more likely be more willing to buy it.

I think this is something we could do even if it was just as a public service gesture. Not only would the general public be helped, but our own club members could come and have their questions answered. It would also let the public know about our club, such as who we are, what we do and where do we meet. This would get more people interested in our group and allow it to grow.

I think this is just one of the many options that we have to consider. I am sure we can come up with more ways to expand tha interest in Atari computers and accessories. If we get more interest in Atari computers, then I am sure that more products will be produced and that means more things we can do with our wonderful computers. Who knows what the future could be like.

by Charles W. Brown

A SIMPLE SCREEN DUMP

by Norman Knapp

While I was recently writing a program, I wanted to be able to send to my printer text displayed on the screen. A quick look through my books and magazines did not produce the program which I had seen previously. Since I needed the program and could remember the main ideas involved, I decide to write another program from scratch. The amount of time involved to produce the first working version was comparable to the time I spent looking through my library for the published program (which I found while writing this article). Computer programmers call this utility a screen dump, a subroutine which reads information from a displayed screen and sends it to a printer. I will present here a Basic screen dump which can be appended to most Basic programs. The text of the main program is modified in only three places, thus permitting easy appendage of the screen dump to a Basic program.

The Graphics 0 screen

Textual and numeric information in most Atari Basic programs is displayed on what Atari calls the Graphics 0 screen. This screen displays characters in a rectangular field of 40 columns and 24 rows; Atari Basic uses only 38 columns, but the programmer can use 40 columns by changing the left margin of the screen (POKE 82,0). This rectangular field can be imagined as a set of rectangles each of which contains a character.

The set of characters universally used (all makes of computers and printers) is called the ASCII set of character codes; each character is assigned a unique numeric code; any number from 0 to 127. The alphameric portion of this set of 128 characters is made up of the upper/lower case letters and the numbers 0-9. Punctuation, math, and business characters are the space, !, ", #, $, %, &, ', @, (,), <, >, _, -, |, =, \, +, ^, *, ?, /, .,], [, and comma. There is also a group of 'control' character codes accessed by the simultaneously pressing the ctrl key and an alpha key which produce graphics characters (heart, triangle, circle, rectangle, etc.) when printed to the screen. When directed to another peripheral, usually a printer, these nonprinting codes will control some aspect of the mode of operation of that device. My program will not dump to the printer the control characters.

The Atari computers use an expanded version of the ASCII set of codes; a 256 character set of codes (0-255) which is referred to as the ATASCII codes. The additional characters (128-255) are identical to ASCII characters except that they are displayed in the inverse video mode. My EPSON printer prints inverse video mode characters (except control characters)as italics characters. Some printers will print control code characters and inverse video mode characters. The EPSON MX-80 with the Graphtrax + option requires the use of machine language utility in the Atari to print these characters in an EPSON dot graphics mode.

A discussion of the Atari graphics modes is not complete without discussing the 'cursor', the solid rectangle which moves around the screen during execution of a program, either stopping occasionally when the program requires input by the user or coming to a final rest when the program is finished. Even though the cursor may be blinking, be only partially visible, or even invisible, its function is still the same: its location on the screen indicates where a character is to be placed (printed to the screen) or where we want the program to find out what is on the screen at a specified location (read the screen). 'Reading' a screen is used for user friendly data input and screen dumps.

POSITION XCORD,YCORD

The POSITION statement defines where the cursor is to be placed on the screen. It must be followed by two numeric constants or variables separated by a comma. The first number indicates the column and the second number indicates the row of the position on the screen occupied by the cursor. These numbers are similar to the Cartesian coordinate system except that the Y coordinate varies from 0 for the top row of the screen to 23 for the bottom row. The column numbers range from 0 on the left to 39 on the right side of the screen. The GRAPHICS 0 screen has 24 rows and 40 columns.

GET #3,CODE

The GET statement, when used with the POSITION statement, obtains the ATASCII code of the character at the place on the screen defined by the current X and Y coordinates.

Now we've gotten to the point where we can tie things together. Instead of printing out the entire screen, let's just consider how a single line is printed. To be more specific, let's print out the top line of the screen. The following program does this task. The text following this program contains a line by line discussion of the program.

```
10 DIM LINE$(40)
20 CLOSE #3:OPEN #3,4,0,"S:"
30 GRAPHICS 0
40 REM *** Run here the program which
prints to the GR. 0 screen ***
50 XCORD=0:YCORD=0
60 LINE$=""
70 POSITION XCORD,YCORD
80 GET #3,CODE
90 LINE$(LEN(LINE$)+1)=CHR$(CODE)
100 XCORD=XCORD+1
110 IF XCORD<=39 THEN 70
120 LPRINT LINE$
130 CLOSE #3
140 STOP
```

Line 10 contains a DIMension statement which must be used to

define the size of a string variable: LINE$. A string variable, which may contain any ATASCII character, has a "$" at the end of its name. A numeric variable has assigned to it an integer, real, or floating number: XCORD and YCORD, examples of integer numbers, are first used in statement 50.

Line 20 contains two commands which we have not discussed yet; OPEN and CLOSE. These commands open and close channels through which the ATARI computer communicates with its peripherals: the screen, printer, keyboard, cassette recorder, disk drive, and RS232 ports. There are eight channels available numbered 0 to 7; channels 1-5 may be used by the programmer without any restrictions. Channels 6 may used by the programmer in the absence of a GRAPhICS statement (#6 sends output to the screen). Channel #7 must be available for the LPRINT command. In line 20, channel #3 is opened up to the screen. Since a channel may communicate with only one peripheral at a time or a peripheral may communicate through only one channel at a time, a CLOSE command is given for a channel before it is OPENed. Otherwise, an error condition may develope which will stop the program.

The second number in the OPEN command, "4", indicates that the computer will accept data from the device designated by the last parameter. "S:" indicates that the input device is the display screen. The third parameter is set to 0 for most peripherals; see p. 376 of "Your ATARI Computer" by Poole, McNiff, and Cook for an extended discussion of the OPEN and CLOSE commands.

Line 30 determines the graphics mode, GRAPHICS 0.

Line 40 just indicates where the application program is placed. The final version of this program will handle this more elegantly than this crude version.

Line 50 defines the X and Y coordinates of the first screen location which is to be read.

Line 60 initializes the LINE$ string; here it has a length of 0 characters.

Line 70 places the cursor at the X and Y screen coordinates determined by the current values of the numeric variables XCORD and YCORD.

Line 80 GETs the decimal code value of the ATASCII character at the current location of the cursor.

In line 90, two new functions are used. The CHR$ function generates the ATASCII character corresponding to ATASCII code following it, represented here by the numeric variable CODE. The LEN function returns the number of characters in the string variable immediately following it. The overall effect of the statement in line 90 is to append an ATASCII character to the string variable LINE$.

Line 100 increases the value of the X coordinate variable, XCORD, by one so that the next position to the right may be read the next time line 70 is processed.

In line 110, the IF ... THEN ... command is used to insure that the program does not try to GET an ASCII code from a position off the screen to the right. As long the value of XCORD is less than or equal to 39, program control is transferred to line 70 which moves the cursor one space to the right. When XCORD=40 control is transferred to line 120.

In line 120, the LPRINT command opens channel #7 to the printer, prints the string LINE$ on the printer, and then closes channel #7.

Now that the line has been sent to the printer, channel #3 to the Screen is closed using the CLOSE command in line 130. The statement is not required, but it is good programming practice, for the reasons given earlier, to close a channel when it no longer being used.

Line 140 STOPs execution of the program. A message is printed to the screen given the line number where the STOP command is encountered.

All of this may seem like a lot of effort to just print out just one line of the screen. With just a little more effort the whole screen can be printed. This can be done by just increasing the value of the Y coordinate variable, YCORD, in a manner similar to the way the X coordinate variable, XCORD, was changed. The following statements can be inserted in the line printing algorithm to permit printing the entire screen.

```
123 YCORD=YCORD+1
126 IF YCORD<=23 THEN 60
```

This is the complete GRAPHICS 0 screen dump. However, there are several features which can be added to make the utility more compact and/or user friendly. The first is that FOR ... NEXT loops can be used to change the variables XCORD and YCORD thus replacing the clumsy construction used in lines 50, 100, 110, 123, and 126. The rewritten program is given below:

```
10 DIM LINE$(40)
20 CLOSE #3:OPEN #3,4,0,"S:"
30 GRAPHICS 0
40 REM *** Run here the program which
prints to the GR. 0 screen ***
60 FOR YCORD=0 TO 23
70 LINE$=""
80 FOR XCORD=0 TO 39
90 POSITION XCORD,YCORD
100 GET #3,CODE
110 LINE$(LEN(LINE$)+1)=CHR$(CODE)
120 NEXT XCORD
130 LPRINT LINE$
140 NEXT YCORD
150 CLOSE #3
160 STOP
```

The latest version of our screen dump dumps an entire screen to the printer, but it does not have some desirable features: 1. A message line which asks you if you want the contents of the screen sent to the printer; 2. An error trapping procedure to tell you to turn the printer on; 3. A change in the screen color to indicate that a screen is being generated which can be printed; 4. A way of stopping the printing process and returning to the main program.

The finished program is listed at the end of this article. If you key in this program, store it on a disk using the LIST command

```
LIST "D:SCRNDMP.BAS
```

instead of the SAVE command. Cassette recorder users must use LIST "C:" rather than the CSAVE command. Line numbering starts at 30000 since most BASIC programs do not have line numbers in this range. This permits merging of this utility with an Atari BASIC or a BASIC XL program. The merging process is:

```
LOAD "D:MAINPRG.BAS"
ENTER "D:SCRNDMP.BAS"
```

Cassette recorder users must first CLOAD the main program and then use the ENTER "C:" command to merge the main program and the screen dump. If your main program processes lines with number ranging 30000 to 30150, use a line renumbering utility on the original SCRNDMP.BAS file before the merging process.

Implementation of this utility if straight forward. Be sure to place a

```
DIM LINE$(40)
```

statement near the beginning of the main Atari BASIC program; the LINE$ string does not need to be dimensioned in a BASIC XL program.

Inspect your BASIC program to locate the first place where information (you want dumped to the printer) is printed to the screen and place a

```
GOSUB 30010
```

statement at that place; the easy way is to substitute this statement for GRAPHICS 0 command. Then place a

```
GOSUB 30030
```

at place just after tha last place where information (you want dumped) is printed to the screen.

The program will work just as before, except that the screen turns red while the screen is being generated. When the screen is finished, a message is printed close to bottom asking if you want the screen dumped to the printer. If the answer is No, the screen turns blue and control returns to the main program.

If you answer Yes, the printing process starts and the screen turns blue when the job is finished. If the printer has not been turned on, a request asking you to turn the printer on flashes across the bottom of the screen until the printer is turned on. Then the message line is restored to its original status and the printing process starts.

The printing process may be stopped at any time by holding down the ESCcape key until the screen color turns back to blue and control returns to the main program.

The final version of this program is somewhat different that the earlier versions in that it is organized into subroutines, small programs which can be processed one or more times during a job. Line 30010 defines the GRAPHICS 0 screen. Lines 30030 to 30130 are the "main" part of the screen dump where screen coordinates are changed and lines are printed. Line 30020 reads a row of the screen. Lines 30072 and 30140 to to 30150 check the condition of the printer and tell you to turn it on if it is off.

This program is all Atari BASIC except for the PEEK statement in line 30100 which checks to see if the ESCape key is pressed down after a line has been sent to the printer.

SUMMARY

In this article I have attempted to show how the idea for a program was found and how the program itself evolved from the crude initial version to a more versatile final version. The versatility and weaknesses of BASIC as a programming language are illustrated. I will say no more about this other the fact that even though the final version of my screen dump is much more useful than earlier versions, it is also more difficult to follow the flow of the program than in the simple, earlier versions.

REFERENCES:

Ion Poole with Martin McNiff and Steven Cook, "YOUR ATARI COMPUTER". Contains a lot of useful information and BASIC programs; often affectionately called the "purple book."

"COMPUTE!'s Second Book of Atari Graphics", pp. 17 to 19, the screen dump I was looking for, but without the "bells and whistles" in my program; I often forget to turn my printer on.

Richard Marmon, 'The Accurate Printer', Micro No. 71 - April/May 1984, pp. 20-28: an AUTORUN.SYS program which permits printing of inverse and control characters on an EPSON printer with the Graphtrax option.

```
30000 REM ****************************
30001 REM SCREEN DUMP UTILITY  GR. 0
30002 REM by Norman Knapp 5JUL1984
30004 REM
30005 REM *** Place a GOSUB 30010 stat
ement at the point in the main program
 where printed screen begins. ***
30006 REM
30010 CLOSE #3:OPEN #3,4,0,"S:":GRAPHI
CS 0:SETCOLOR 2,1,1:RETURN
30020 FOR XCORD=0 TO 39:POSITION XCORD
,YCORD:GET #3,CODE:LINE$(LEN(LINE$)+1)
=CHR$(CODE):NEXT XCORD:RETURN
30024 REM
30025 REM *** Place a GOSUB 30030 stat
ement at the point in the main program
 where printed screen ends. ***
30026 REM
30030 LINE$="":YCORD=22:GOSUB 30020
30040 POSITION 0,22
30050 ? "Print this screen: (Y/N) ";:C
LOSE #1:OPEN #1,4,0,"K:":GET #1,CODE:C
LOSE #1
30060 IF CHR$(CODE)="N" THEN GRAPHICS
0:RETURN
30070 IF CHR$(CODE)<>"Y" THEN 30040
30072 TRAP 30140:LPRINT
30075 POSITION 0,22:? LINE$
30090 FOR YCORD=0 TO 23
30100 LINE$="":GOSUB 30020:LPRINT LINE
$:IF PEEK(764)=28 THEN GRAPHICS 0:RETU
RN
30120 NEXT YCORD
30130 CLOSE #3:TRAP 40000:GRAPHICS 0:R
ETURN
30140 FOR XCORD=0 TO 39:POSITION XCORD
,22:? " ";:NEXT XCORD
30150 POSITION 0,22:? "TURN ON PRINTER
";")";:GOTO 30072
```