APX ATARI® PROGRAM EXCHANGE

VOLKER MULTHOPP    MARCH 1982

# DSEMBLER

<u>CASSETTE</u> (APX-10065)
REQUIRES: 24K RAM

<u>DISKETTE</u> (APX-20065)
REQUIRES: 24K RAM

User-Written Software for ATARI Home Computers

VOLKER MULTHOPP   MARCH 1982

# DSEMBLER

<u>CASSETTE</u> (APX-10065)
REQUIRES: 24K RAM

<u>DISKETTE</u> (APX-20065)
REQUIRES: 24K RAM

# DSEMBLER

by

Volker Multhopp

## TRADEMARKS OF ATARI

*********************************************************************************

Distributed by

The ATARI Program Exchange
P. O. Box 427
155 Moffett Park Drive, B-1
Sunnyvale, CA 94086

To request an APX Software Catalog, write to the address above, or call toll-free:

800/538-1862 (outside California)
800/672-1850 (within California)

Or call our Sales number, 408/745-5535.

*********************************************************************************

## CONTENTS

INTRODUCTION ___ 1

- Overview ___ 1
- Required accessories ___ 1
- Optional accessories ___ 1
- Contacting the author ___ 1

GETTING STARTED ___ 2

- Loading DSEMBLER into computer memory ___ 2
- The first display screen--the menu ___ 3

USING DSEMBLER ___ 4

- Introduction ___ 4
    - Entering data and responding to prompts ___ 4
    - Numerical input ___ 4
    - BREAKs and ERRORs ___ 4
    - DSEMBLER's tasks ___ 4
- M: Menu ___ 4
- C: Convert number ___ 5
- D: Disassemble ___ 5
- L: Label regenerator ___ 6
- T: Table of labels ___ 6
- S and R: Store and Recall labels ___ 6
- K: Kill label ___ 7
- P: Memory page displayer ___ 7
- H: Hexpacker ___ 7
- J: Jump to a machine language subroutine (JSR) ___ 8
- O: Operations ___ 8
- F: Find sequence ___ 9
- U: User section ___ 10

TIPS FOR USERS ___ 11

- Working in direct mode ___ 11
- Examining USR functions contained in strings ___ 11
- Merging programs ___ 12
- How to renumber DSEMBLER ___ 12
- Sundry notes about DSEMBLER ___ 13
- Strings used in DSEMBLER ___ 13
    - General purpose strings ___ 13
    - Data strings ___ 13
    - Strings holding USR functions ___ 13

# INTRODUCTION

## OVERVIEW

DSEMBLER is a deluxe tool for the study of 6502 machine language code. Through a series of menu options, you can perform a wide variety of tasks. You can disassemble any of the valid 6502 machine code in memory and display or print the results in a five-field format: (1) the location of the disassembled instruction; (2) the hex value of each byte of the instruction; (3) if you labeled the location while debugging the code, the label name (i.e., not necessarily the label you assigned as you wrote the code); (4) the op-code for the instruction; and (5) any associated operand using standard 6502 format (if you labeled an operand address while debugging, the label name will appear instead of that address).

Using DSEMBLER, you can also perform the following tasks. (1) You can convert integers between 0 and 65535 to hexadecimal and vice versa. (2) You can regenerate labels (a memory address and associated name), rename existing labels, and remove an existing label. DESEMBLER can create as many as 254, two-to-eight character labels. (3) You can display or print a list of label values and their names. (4) You can store a file of labels to diskette or cassette and recall them. (5) You can load your machine code quickly into memory. (6) You can display "pages" of memory for quick scanning. (7) You can jump to a machine language subroutine with the 6502, X, Y, and A registers set as you wish. (8) You can search any part of memory for any specified byte sequence, using wild cards for byte values if you wish. (9) You can write your own operation as an option on DSEMBLER's menu.

## REQUIRED ACCESSORIES

For cassette version
- 24K RAM
- ATARI 410 Program Recorder

For diskette version
- 24K RAM
- ATARI 810 Disk Drive

ATARI BASIC Language Cartridge

## OPTIONAL ACCESSORIES

ATARI printer or equivalent printer

## CONTACTING THE AUTHOR

Users wishing to contact the author about DSEMBLER may call him at:

301/675-4473

## GETTING STARTED

LOADING DSEMBLER INTO COMPUTER MEMORY

1. Insert the ATARI BASIC Language Cartridge in the (Left Cartridge) slot of your computer.

2. If you'll be using a printer, turn it and the ATARI 850 Interface Module, if applicable, on.

3. <u>If you have the cassette version of DSEMBLER:</u>

   a. Turn on your TV set.

   b. Turn on your computer and connect your program recorder to the computer and to a wall outlet.

   c. Slide the DSEMBLER cassette into the program recorder's cassette holder and press REWIND on the recorder until the tape rewinds completely. Then press PLAY.

   d. Type CLOAD on your computer and then press the RETURN key two times. The tape will load into computer memory.

   e. After the tape finishes loading, the word READY will display on your TV screen. Type RUN and press the RETURN key. The first display screen will appear on your TV screen.

   <u>If you have the diskette version of DSEMBLER:</u>

   a. Turn on your disk drive.

   b. When the BUSY light goes out, open the disk drive door and insert the DSEMBLER diskette with the label in the lower right-hand corner nearest to you. Close the door.

   c. Turn on your computer and TV set.

   d. When the READY prompt displays on your TV screen, type RUN "D:DSEMBLER" and press the RETURN key. The program will load into computer memory and start.

## THE FIRST DISPLAY SCREEN--THE MENU

DSEMBLER's menu appears on the screen after the program loads into computer memory. It looks like this:

```
        *  DSEMBLER : MENU OF TASKS  *
            (c) V.Multhopp  5/1981

C: Convert number  M: MENU
D: Disassemble     O: Operations
F: Find sequence   P: Page displayer
H: Hex packer      R: Recall labels
J: Jump sub (JSR)  S: Store labels
K: Kill label      T: Table of labels
L: Label regenertr U: User section

what now (give a capital letter)?
```

Figure 1  DSEMBLER's Menu

# USING DSEMBLER

## INTRODUCTION

DSEMBLER is a powerful and sophisticated program for studying 6502 machine language. Although the following instructions may seem extensive, using DSEMBLER is simple. The following notes will help you quickly master DSEMBLER's capabilities.

### Entering data and responding to prompts

Whenever DSEMBLER isn't executing a command, it's waiting for you to enter something. You enter all information (data, responses to prompts, etc.) through the keyboard and you press the RETURN key after each entry.

In some cases, you can press the RETURN key without entering information first. We'll refer to this action as a "default". Some tasks (for example, menu options D, P, and T) automatically assign values to defaults for your convenience. Most tasks, however, abort if you default (i.e., press the RETURN key without first entering information), and the program then prompts you for another task.

### Numerical input

DSEMBLER uses only whole numbers between 0 and 65535 ($0 and $FFFF). Using other numbers may result in problems.

DSEMBLER accepts both decimal and hexadecimal numbers. Always precede hexadecimal numbers with a dollar sign.

### BREAKs and ERRORs

If you press the BREAK key or if an untrapped ERROR occurs, do not type RUN. Instead, give the direct mode BASIC command GOTO MENU . Otherwise, important accumulated data will be purged from computer memory.

## DSEMBLER's TASKS

DSEMBLER performs a wide variety of tasks and subtasks. Each task is identified by a capital letter on DSEMBLER's menu (see Figure 1). The tasks are explained below.

## M: MENU

The Menu displays (1) when you RUN the program, (2) commit certain errors, (3) select task M in response to a prompt for task selection, or (4) enter a direct mode GOTO MENU command. Besides listing the tasks DSEMBLER performs, the Menu corrects certain types of chicanery you may have committed while in direct mode.

Beneath the Menu is the prompt "what now (give a capital letter)?". This prompt appears often while you use DSEMBLER. It indicates that the program has done everything expected of it and is waiting for your next task. Select a task by typing its capital letter (e.g., C for the task CONVERT NUMBER) and pressing the RETURN key.

## C: CONVERT NUMBER

Use task C to convert a decimal number to its hexadecimal equivalent and vice-versa. Remember to use only whole numbers between 0 and 65535 ($0 and $FFFF) and to precede a hexadecimal number with a dollar sign.

This task reprompts itself so that you can convert a series of numbers without having to reselect C each time. Or, you may go to another task by pressing an appropriate capital letter.

## D:DISASSEMBLE

Use task D, the heart of DSEMBLER, to disassemble any valid 6502 machine code located in memory. Of course, as with any disassembler, if you start disassembling in a data field, or at the second or third byte of a multibyte instruction, your results will be garbage.

Important note! If you disassemble code contained in a BASIC string, the address of the string in direct mode is not equal to the address in execute mode.

Task D first prompts you for an address at which to start disassembling. If you default, the program assumes the location immediately following the last disassembly. It then prompts you for the number of 6502 machine language instructions to disassemble. If you default, the program does 16 instructions. (You may also select another task in response to either prompt by pressing the appropriate capital letter.)

After you answer the two prompts, a header appears and the disasssembly begins. The listing is formatted in five fields, as follows:

Field 1--contains the hexadecimal address of the instruction being disassembled.

First 2--contains the hexadecimal value of each byte of the instruction.

Note. Neither field uses dollar signs to denote hex values.

Field 3--the label name, if the instruction address has been labeled.

Field 4--contains the correct op-code for the instruction.

Field 5--contains the associated (if any) operand using standard 6502 format. If the operand refers to a labeled address, the label name will appear instead of the address value. Immediadiate operands never show label names.

Task D reprompts itself. Or, you may go to another task by pressing an appropriate capital letter. If you have a printer and want a hard copy disassembly, see task O.

## L: LABEL REGENERATOR

By "label", we mean a memory address and a name associated with that address. Task L first prompts you for the address to be labeled, and then for the label name. You can use this task to rename an existing label. Just give the old address and the new name.

The following rules apply to label regeneration:

1. You can create a maximum of 254 different labels.

2. Do not label address $00.

3. Label names must be between two and eight characters.

4. Although the program doesn't restirct you as to the characters you may use in label names, you should limit yourself to characters your printer can produce.

Task L reprompts itself. Or, you may go to another task by pressing an appropriate capital letter.

## T: TABLE OF LABELS

Use task T to produce a listing of label values and their associated names. The task prompts you for an initial specification. You need enter only as many characters as are contained in all the label values you want to list. If, for example, you respond with "C", all labels with names beginning with C display (and no others). If you respond with "COL", all labels with names starting with COL display. If you default, the program lists all labels.

You cannot select task T without getting some kind of listing, even though the list may contain zero entries. The program lists 40 labels to the screen at one time; to continue the listing, press the RETURN key (default). Or, you may go to another task by pressing an appropriate capital letter. To obtain a hard copy of the Table of Labels, see task O.

## S and R: STORE and RECALL LABELS

Use task S to store a file of labels on cassette or diskette and task R to load a file of labels from diskette or cassette into computer memory. Task R erases all labels currently in RAM.

It's good practice to create a file of standard labels with which you're familiar and to recall the list at the start of each session. You can then produce a specialized file of labels for each section of machine code you're studying.

### For CASSETTE USERS

Respond to the prompt with a "C". No filenames are used with the cassette. Then follow the regular cassette handling procedures with which you are familiar.

For DISKETTE USERS

Respond to the prompt with "D:filename", where "filename" is the name you assign the file. Always use the extension ".LBL" to identify a Label file. Specify an alternate drive by preceding the device code with the drive number (e.g., D2 for drive two). A Label file occupies 23 single-density disk sectors. These two tasks are somewhat slow; have patience.

## K: KILL LABEL

Use this task to remove a current label. Respond to the prompt with the address of the label you want purged. You needn't use this task to rename a label; select task L (Label Regenerator) directly.

If you want to remove a label named, for example, "GEORGE", and have forgotten the address, first select task T (Table of Labels) and enter GEORGE in response to the prompt. Then select task K and enter GEORGE's address.

Task K does not reprompt itself. You may go to another task by pressing an appropriate capital letter.

## P: MEMORY PAGE DISPLAYER

Memory is divided into 256 "pages" of 256 bytes. The page number is the same as the high byte of an address. Use task P to scan a page (or two) of memory quickly. A primary use of task P is to scan memory for ATASCII characters. To use task P, give the number of the page you want displayed in response to the prompt.

The extensive character set of the ATARI 400/800 Home Computer assigns a unique character to (almost) every byte value. The one exception is CHR$(155), whose control function cannot be escaped. For this character, DSEMBLER prints the escape character and gives a peep.

Task P reprompts itself. Defaulting displays the next page. Or, you may select another task by pressing an appropriate capital letter.

## H: HEXPACKER

DSEMBLER does not contain an assembler, but with this hex loader you can quickly load RAM with machine code. Most assembly listings give the hex value of each byte.

Task H first prompts you for the address at which to begin the loading process. Then it prompts you for each byte. IMPORTANT NOTE: Enter each byte as a one- or two-digit hexadecimal with NO DOLLAR SIGN! A dollar sign will precede the prompt. This is the exception to the rule that DSEMBLER is indifferent to the hex or decimal mode of a number, and it was done to save you keystrokes.

To end the packing process, press the RETURN key (i.e., a default) in response to the prompt for another byte.

## J: JUMP TO A MACHINE LANGUAGE SUBROUTINE (JSR)

Use this task to jump to a real machine language subroutine with the 6502 X, Y, and A registers set as you wish. Task J prompts you for each register in turn and then for the address to be JSRed to. You may also preset register P; see task O (Operations). After the appropriate RTS is processed, control passes back to DSEMBLER and the program displays the exit values of the X, Y, A, and P registers.

Note: Unless you know what you're doing, using task J will probably cause DSEMBLER to crash. You'll have to turn off the computer, turn it on again, and start everything from scratch. This is no task for faint hearts and fickle folks. You may select another task at any time by entering an appropriate capital letter.

## O: OPERATIONS

Task O is a submenu of five items. It looks as follows:

```
 Operations:
   1 count labels
   2 printer on/off
   3 force set P- on/off
   4 table of all DOS files
   5 table of *.LBL files
which number?
```

Figure 2  Operations Submenu

Respond to the prompt with the appropriate number. Operations 2 and 3 are toggle switches. Calling them once turns them on; calling them again turns them off. You can do the following operations with Task O.

1: Count Labels
This subtask displays a count of the current number of labels in RAM.

2: Printer on/off
This subtask affects only tasks D and T. Every disassembly (task D) automatically prints after you select this subtask. For task T, the label table also displays on your TV screen and prints. The screen display doesn't stop after every 40 labels; it displays continuously until the end.

You may communicate directly to the printer after selecting the subtask by pressing the BREAK key and then typing the direct mode command PRINT#4;-. After completing this activity, you can return to DSEMBLER with a direct mode GOTO MENU statement. No labels will be lost.

3: Force set P- on/off
This subtask has nothing to do with printer. It affects task J only. After toggling

on subtask 3, everytime you select task J, you must preset register P (the processor status register). Doing so significantly increases the likelihood of crashing the system. However, task J will ignore any attempt to set P=32($20).

4: and 5: Table of (diskette) files--for DISKETTE USERS only
These subtasks list all diskette files, or only those with the extension ".LBL". These subtasks prompt you for a drive number. The default is drive one.

F: FIND SEQUENCE

Task F uses another submenu. Use task F to search any part of memory for any specified sequence of bytes. You may also designate any byte value to be a "wild card". The program ignores a wild card during the search except as a place holder. You may use the wild card repeatedly in your search specification. The initial appearance of the submenu is as follows:

```
Find:
1: Start  addr- $00
2: Finish addr- $FFFF
3: Wild card?- NO
4: spec. bytes and go find.

Which subtask?
```

Figure 3  Find Sequence Submenu

The values next to subtasks 1, 2, and 3 change as you use the subtasks.

1: Start addr- $00
Select 1 to change the address at which your search is to begin.

2: Fnish addr- $FFFF
Select 2 to change the address at which your search is to end.

3: Wild card?
Use 3 to set or change the value of the wild card. To cancel the wild card, give a value greater than $FF (255).

4: spec. bytes and go find
Use 4 to enter your search specification. Each byte is indivdually prompted. You may enter from two to fourteen bytes. Pressing the RETURN key (a default) initiates the search process. Each address matching the input specification displays on the screen.

You may select another task at any time by pressing an appropriate capital letter. The values you enter in subtasks 1, 2, and 3 remain in effect until you change them. As always, you may use either decimal or hexadecimal values.

An example can help explain how to use this task. Suppose we want to find all places in the ATARI BASIC Language Cartridge where a Load A from zero-page instruction is followed by a BNE. The BASIC cartridge runs from $A000 to $BFFF. An LDA Z-page is coded as $A5. A BNE is $D0. The unspecified operand of the LDA instruction will have to be wild-carded. We do the following:

```
F <RETURN>      --> call task F

1 <RETURN>

$A000 RETURN     --> set the start

2 <RETURN>

$BFFF <RETURN> --> set the end

3 <RETURN>

$FF <RETURN>   --> set a wild card

F <RETURN>      --> just checking!

4 <RETURN>      --> start specifying

$A5 <RETURN>   --> the LDA

$FF <RETURN>   --> wild-card the operand

$D0 <RETURN>   --> the BNE

<RETURN>        --> start search
```

The listing that follows depends on your BASIC cartridge. Of course, some of the addresses might represent data fields and not machine code. Check using the disassembler.

## U: USER SECTION

As is, this task simply lists a section of BASIC code. You may want another task not included in the Menu. To create your own task, select task U, press the BREAK key, delete the indicated line, and write your own task in the indicated area. Then type a direct mode GOTO MENU command and debug. You may then select the new User task like any other Menu item.

If the User task requires a numerical input, then use the following coding to achieve a flexible, DSEMBLER-like input:

```
prompt-: INPUT IN$:GOSUB TRANS
```

If the input is a decimal or hexadecimal number, then the subroutine TRANS will return, the variable DEC will contain the decimal value of the input, and IN$ will contain its hexadecimal representation. Otherwise, the subroutine TRANS will POP, and the input is interpreted as a task selection. In the worst case, the MENU will redisplay.

## TIPS FOR USERS

### WORKING IN DIRECT MODE

DSEMBLER has been designed so that you may work in direct mode and then return to DSEMBLER with all data (such as regenerated labels) intact. You press the BREAK key, do your direct mode operations and return to DSEMBLER with a GOTO MENU statement. Your direct mode activity can be as simple as a PEEK, or as complicated as merging and using another program. In the latter case, be careful not to reinitialize DSEMBLER or to clear variables accidentally with a RUN statement.

Diskette users may actively access and use the DOS MENU if a MEM.SAV file has first been created. In that case, you can use DSEMBLER, then press the BREAK key and call DOS. Do your DOS activity (probably a BINARY LOAD or SAVE) and return to DSEMBLER through DOS option B (Run Cartridge). When the READY prompt appears, type the direct mode GOTO MENU command. The disadvantage of this technique is that calls to and from DOS are very slow.

Cassette users with the Assembler/Editor Cartridge need to go through a special step. With orginal ROM configurations, the Assembler/Editor Cartridge produces cassette object files that BASIC can't load without the assistance of a special loader program. Listed below is such a program especially designed for use with DSEMBLER:

```
1 OPEN #5,4,0,"C":GET #5,K:GET #5,K:TRAP 5

2 GET #5,K:GET #5,J:AD=256*J+K:GET #5,K:GET #5,J:R8=256*J+K

3 GET #5,K:POKE AD,K:AD=AD+1:IF AD<=R8 THEN 3

4 GOTO 2

5 CLOSE #5:GOTO MENU
```

To use this utility, carefully copy the five lines and make a LISTed version. Then, to load a cassette object file while using DESMBLER, press the BREAK key and ENTER the utility program. Then use a direct mode GOTO 1 command to load the object file. After the load is complete, the MENU redisplays.

You may wish to go a step further and CSAVE a version of DSEMBLER which includes this utility. In that case, add the line

```
0 GOTO 5000
```

CSAVE the DSEMBLER program with the six added lines. You can then CLOAD and RUN as normal, and BREAK and GOTO 1 everytime you need an object file load.

### EXAMINING USR FUNCTIONS CONTAINED IN STRINGS

Atari BASIC moves strings around a great deal. This movement causes problems when examining USR functions whose codes are stored in strings. It's therefore necessary to copy the code into fixed RAM locations before disassembling. Page 6 is an excellent place to recopy, if the string is not too long. To copy, for example, MACH$, onto page 6, press the BREAK key and enter the following direct command:

```
        FOR J=1 TO LEN(MACH$):POKE 1535+J,ASC(MACH$(J)):NEXT J
```

## MERGING PROGRAMS

You can merge DSEMBLER with other programs to study USR functions. To merge programs, make a LISTed version of one program, LOAD the other program into RAM, and then ENTER the first program. The following problems may arise, however:

1. Insufficient RAM
2. Too many variable names
3. Conflict between variables
4. Conflict between BASIC lines
5. DSEMBLER will run more slowly if there are lower numbered lines.

Paring the program to be studied down to those lines essential to the USR functions helps alleviate these problems.

DSEMBLER uses a large number (about 75) of variable names. You may inspect these by leafing through memory with task P (Page displayer) until you find the variable name table. The last character of each variable name appears in inverse video. Remember that ATARI BASIC allows only 128 different variable names.

If necessary for merging, you may renumber DSEMBLER.

## HOW TO RENUMBER DSEMBLER

DSEMBLER occupies BASIC lines 5000 to 9999. You can't shorten this 5000-line span, but you can renumber DSEMBLER to start anywhere from 0 to 27000 by following these directions:

1. RUN the program and press the BREAK key.

2. Enter the folowing lines:

```
32000 INPUT LINE1:K=5000-LINE1:AD=PEEK(136)+PEEK(137)*256

32010 J=PEEK(AD)+N256*PEEK(AD+N1):IF J>=32000 THEN LIST LINE1:END

32020 J=J-K:R4=INT(J/N256):R8=J-N256*R4:POKE AD,R8:POKE AD+N1,
      R4:AD=AD+PEEK(AD+N2):GOTO 32010
```

3. Do a direct mode GOTO 32000 command.

4. A question mark will appear. Give the line number with which DSEMBLER is to start.

5. After a few seconds, a BASIC line with the new line number will appear. The first statement should read "LINE1=5000". Change this number to reflect the new line number.

6. Now do a direct mode RUN command. The menu should display. Test several tasks to make sure everything is ok.

7. Delete the three utility lines and SAVE.

## SUNDRY NOTES ABOUT DSEMBLER

1. Sound channel 3 is used for audible cueing.

2. IOCB#5 is used for communicating to the cassette or diskette. IOCB#4 is used for the printer.

3. The Menu contains a conditional GRAPHICS statement, the idea being to avoid clearing the screen.

4. DSEMBLER uses no DATA statements or tab spacing.

## STRINGS USED IN DSEMBLER

### General purpose strings

IN$ is the primary input/output string. JUNK$ is used for auxilary purposes.

### Data strings

LBL$ contains the current file of regenerated labels. It is divided into 256 fields of 11 bytes each. The first two bytes of each field contain the value of the label, high byte first. The third byte contains the length of the label name. The rest of the bytes hold the given label name.

OPC$ holds all the 6502 opcodes, arranged in alphabetical order.

KEY$ is used in decoding the machine instructions. KEY$ is set up into 256 fields of two bytes each. The first byte is a pointer into OPC$. The second byte determines the proper mode of address.

### Strings holding USR functions

DELA$ generates a time delay for the audible cues. It is used in place of an empty FOR/NEXT loop to make it independent of the number of preceeding BASIC lines.

MAKE$ is used by task L to insert new labels.

KIL$ is used by task K to remove unwanted labels.

CK$ is used to check if a given value has been labeled.

FN$ is used by task F to find the specified sequence of bytes.

PR$ is a somewhat clumsy method of outputting to the printer. It reads the current screen line, converts it to ATASCII, and dumps it into IN$ for output.

JSR$, of course, is used to set up and make the jump to the specified section of code, and then to catch the values of the registers upon the return.

<u>Note</u>. None of these USR functions uses any fixed RAM, except the floating point registers $D4 to $F1. These registers get waltzed over by practically every BASIC statement anyway. JSR$ uses these registers before and after the jump, but nothing is held there for the duration of the subroutine.

# ATARI PROGRAM EXCHANGE

REVIEW FORM

We're interested in your experiences with APX programs and documentation, both favorable and unfavorable. Many software authors are willing and eager to improve their programs if they know what users want. And, of course, we want to know about any bugs that slipped by us, so that the software author can fix them. We also want to know whether our documentation is meeting your needs. You are our best source for suggesting improvements! Please help us by taking a moment to fill in this review sheet. Fold the sheet in thirds and seal it so that the address on the bottom of the back becomes the envelope front. Thank you for helping us!

1. Name and APX number of program ______________________________

2. If you have problems using the program, please describe them here.

______________________________

______________________________

______________________________

3. What do you especially like about this program?

______________________________

______________________________

______________________________

4. What do you think the program's weaknesses are?

______________________________

______________________________

______________________________

5. How can the catalog description be more accurate and/or comprehensive?

______________________________

______________________________

6. On a scale of 1 to 10, 1 being "poor" and 10 being "excellent", please rate the following aspects of this program?

_____ Easy to use
_____ User-oriented (e.g., menus, prompts, clear language)
_____ Enjoyable
_____ Self-instructive
_____ Useful (non-game software)
_____ Imaginative graphics and sound

7. Describe any technical errors you found in the user instructions (please give page numbers).

8. What did you especially like about the user instructions?

9. What revisions or additions would improve these instructions?

10. On a scale of 1 to 10, 1 representing "poor" and 10 representing "excellent", how would you rate the user instructions and why?

11. Other comments about the software or user instructions:

```
 -----
|     |
|STAMP|
|     |
 -----
```

ATARI Program Exchange
Attn: Publications Dept.
P.O. Box 50047
60 E. Plumeria Drive
San Jose, CA 95150

[seal here]