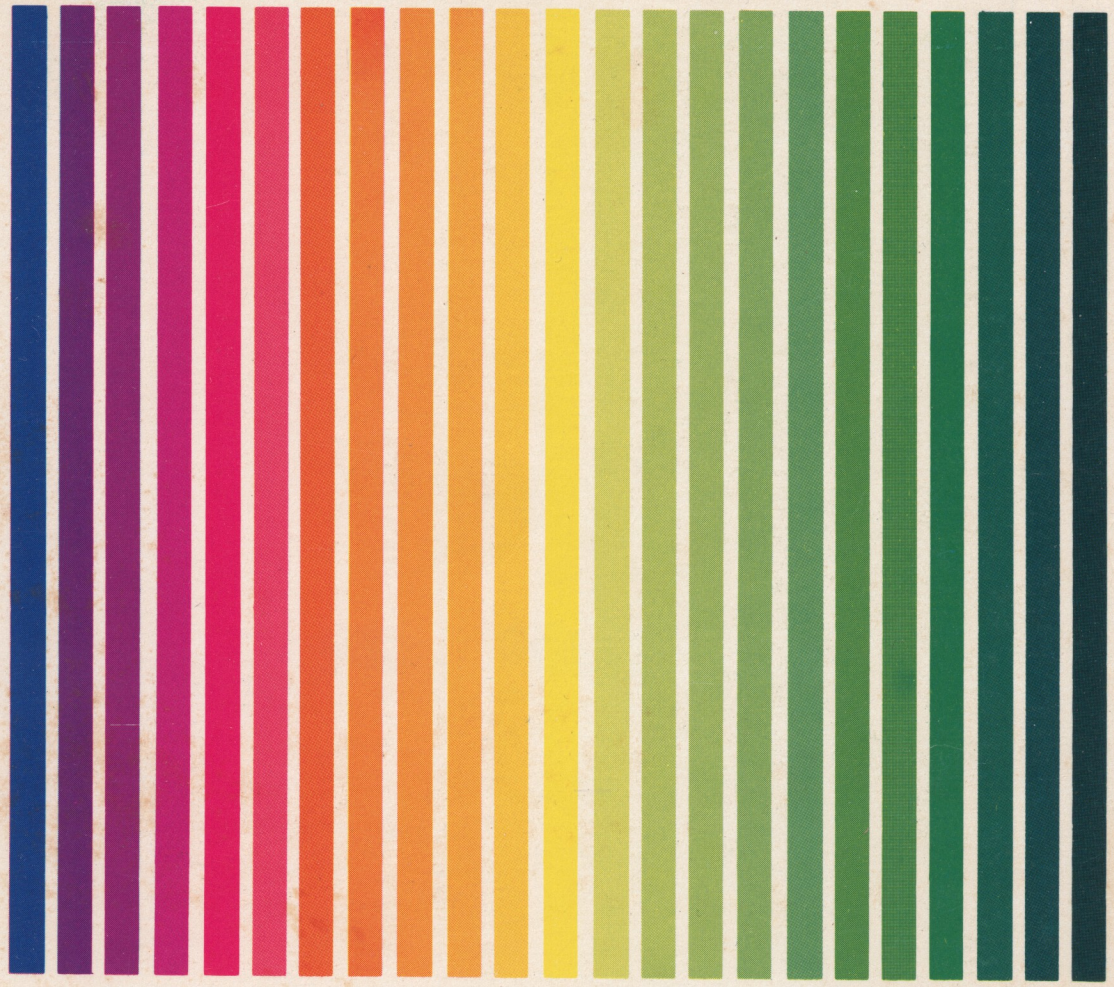
APX ATARI® PROGRAM EXCHANGE



Thomas Newton

BASIC/XA

Development tools for ATARI BASIC programmers

Cassette: 16K (APX-10177) Diskette: 24K (APX-20177)

User-Written Software for ATARI Home Computers

Thomas Newton

BASIC/XA

Development tools for ATARI BASIC programmers

Cassette: 16K (APX-10177) Diskette: 24K (APX-20177)

BASIC/XA

by

Thomas D. Newton

Distributed By

The ATARI Program Exchange
P.O. Box 3705
Santa Clara, CA 95055

To request an APX Product 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, 408/727-5603

Trademarks of Atari

The following are trademarks of Atari, Inc.

ATARI®
ATARI 400™ Home Computer
ATARI 800™ Home Computer
ATARI 410™ Program Recorder
ATARI 810™ Disk Drive
ATARI 820™ 40-Column Printer
ATARI 822™ Thermal Printer
ATARI 825™ 80-Column Printer
ATARI 830™ Acoustic Modem
ATARI 850™ Interface Module

Printed in U.S.A.

TABLE OF CONTENTS

INTRODUCTION

OVERVIEW

EXTENDED ATARI BASIC (BASIC/XA) helps you write programs in ATARI BASIC. It can tell you what variables you have used in your program, their values and dimensions, and which lines use them. BASIC/XA also lets you change variable names, delete a range of lines from your program, renumber your program, and check your program for bad GOTO statements and syntax errors.

This automatically loading program is written in machine language and uses about 4000 bytes of memory. A BASIC program included with both versions lets you relocate the diskette version for your system and provides cassette owners with a way to transfer BASIC/XA to diskette.

To use BASIC/XA, just type DOS when the READY prompt displays. The screen will clear and display your choices, in the style of the DOS menu. Displaying the real DOS menu is one of BASIC/XA's menu choices.

REQUIRED ACCESSORIES

ATARI BASIC Language Cartridge

Cassette version

16K RAM
ATARI 410 Program Recorder

Diskette version

24K RAM
ATARI 810 Disk Drive

OPTIONAL ACCESSORIES

ATARI printer or equivalent printer

RELATED PUBLICATIONS

1. ATARI BASIC Reference Manual

The following publications aren't needed for using BASIC/XA, but advanced assembly language programmers who want to study the listings in the ADVANCED TECHNICAL INFORMATION section should have these materials.

2. ATARI Operating System User's Manual and Hardware Manual (CO16555)

3. Winner, Lane, "The Atari Tutorial, Part 6: ATARI BASIC," BYTE, February 1982, pp. 91-118. This material also appears in De Re ATARI (APX-90008).

4. ATARI Disk Operating System II Reference Manual (CO16347)

CONTACTING THE AUTHOR

Users wishing to contact the author about BASIC/XA may write to him at:

P.O. Box 513
Wrightsville Beach, NC 28480

GETTING STARTED

LOADING BASIC/XA INTO COMPUTER MEMORY

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

2. If you have the cassette version of BASIC/XA:

   a. Have your computer turned OFF.

   b. Insert the BASIC/XA cassette into the program recorder's cassette holder and press REWIND on the recorder until the tape rewinds completely. Then press PLAY to prepare the program recorder for loading the program.

   c. Turn on the computer while holding down the START key, and then turn on your TV set.

   d. When you hear a beep, release the START key and press the RETURN key. The program will load into computer memory and start automatically.

If you have the diskette version of BASIC/XA:

   a. Have your computer turned OFF.

   b. Turn on your disk drive.

   c. When the BUSY light goes out, open the disk drive door and insert the BASIC/XA diskette with the label in the lower right-hand corner nearest to you. (Use disk drive one if you have more than one drive.)

   d. Turn on your computer and your TV set. The program will load into computer memory and start automatically.

THE FIRST DISPLAY SCREEN

After BASIC/XA loads into computer memory, you'll see:

```
*** EXTENDED ATARI BASIC
*** Version 1.1
*** Copyright 1982 Thomas Newton

READY
```

USING BASIC/XA

BASIC/XA'S DISPLAY SCREEN

To use BASIC/XA, load your program into computer memory. Then type DOS and press the RETURN key to display this menu:

```
EXTENDED ATARI BASIC VERSION 1.1
COPYRIGHT 1982 THOMAS NEWTON

A. LIST VARIABLES          F. RENUMBER
B. VARIABLE VALUES         G. CHECK PROGRAM
C. CHANGE NAME             H. NEW OUTPUT FILE
D. CROSS REFERENCE         I. RETURN TO BASIC
E. DELETE LINES            J. GO TO DOS MENU

SELECT ITEM OR RETURN FOR MENU
```

To select a command, type its letter and press RETURN. Press RETURN again to redisplay the menu.

COMMANDS

The description of each command is divided into the following sections:

1. Function -- describes what the command does.

2. Using it -- shows the questions you must answer when you use the command. Your responses are underlined. If you're in BASIC's READY mode, you should type DOS to display the BASIC/XA menu.

3. Error messages -- explains the error messages that display if you make a mistake or if the computer can't print to your NEW OUTPUT FILE.

4. Warning messages -- explains the warning messages you may see when you use the RENUMBER and CHECK PROGRAM commands. These messages tell you about expressions used as line numbers and errors in your program.

5. Example -- demonstrates the use of the command with a small BASIC program.

6. Notes -- lists additional information about the command that you should know.

IMPORTANT NOTES

If your program uses IOCB #5 (OPEN #5, PRINT #5, and so on), you should CLOSE it before typing DOS to display the BASIC/XA menu.

If you have the diskette version of BASIC/XA, you must have a MEM.SAV file on the diskette in disk drive one when you go to the DOS menu with BASIC/XA's J command. While in DOS, you must not give permission for DOS to use the program area. The same diskette must be in drive one when you return to BASIC as when you went to the DOS menu.

If you want to use the DUPLICATE DISK command, turn the computer off and turn it back on using a diskette that doesn't contain BASIC/XA. You can now allow DOS to use the program area (if the diskette does not have a MEM.SAV file, DOS will use it automatically). You may also want to follow this procedure when using the COPY FILE and DUPLICATE FILE commands, since allowing DOS to use all of memory results in far less diskette swapping.

COMMAND A: LIST VARIABLES

Function

This command lists the variable names you've used in your program. It also tells you how many variable names are in the table. If you've used command H, NEW OUTPUT FILE, the table prints to the file you selected (e.g., to the printer).

Using it

```
SELECT ITEM OR RETURN FOR MENU
A
```

Error messages

INPUT/OUTPUT ERROR -- the computer could not print to your file OR you pressed the BREAK key. If you were sending output to a file, the program closes the file, and the rest of the table prints on the TV screen.

Example

Suppose you enter the following program (your typing is underlined):

```
READY
NEW

READY
10 DIM C$(10),A(20)
20 FOR X=1 TO 10
30 A(X)=X*X*X
40 NEXT X
```

Now you want to see all the variable names used in the program:

```
DOS

(screen clears and the BASIC/XA menu appears)

SELECT ITEM OR RETURN FOR MENU
A

VARIABLE NAME TABLE

C$
A(
X

3 VARIABLES USED

SELECT ITEM OR RETURN FOR MENU
I
```

```
READY
```

Notice that the name of array A ended with a "(" character. The name of an array or matrix always ends with a "(", and the name of a string, like C$, always ends with a dollar sign.

Variable names appear in the table in the order you used them in the program.

COMMAND B: VARIABLE VALUES

Function

This command lists the variables used in your program. It also prints the value of simple variations (such as X) and the dimensions of arrays, matrices, and strings. If you've used command H, NEW OUTPUT FILE, the table prints to the file you selected.

Using it

```
SELECT ITEM OR RETURN FOR MENU
B
```

Error messages

INPUT/OUTPUT ERROR -- the computer couldn't print to your file OR you pressed the BREAK key. If you were sending output to a file, the program closes the file, and the rest of the table prints on the TV screen.

Example

Suppose you enter the following program (your typing is underlined):

```
READY
NEW

READY
10 DIM C$(10),X(5)
20 FOR B=1 TO 5
30 X(B)=B*B
40 NEXT B
```

Now you want to list the value of B and the dimensions of C$ and X(.

```
DOS

(screen clears and the BASIC/XA menu appears)

SELECT ITEM OR RETURN FOR MENU
B

VARIABLE VALUES

C$(...)
X(...)
B = 0

3 VARIABLES USED
```

The computer typed the (...) by C$ and X to let you know that these

variables haven't been dimensioned. It also told you that B is zero. Now let's RUN the program and try it again.

```
SELECT ITEM OR RETURN FOR MENU
I

READY
RUN

READY
DOS

(screen clears and the BASIC/XA menu appears)

SELECT ITEM OR RETURN FOR MENU
B

VARIABLE VALUES

C$(10)
X(5)
B = 6

3 VARIABLES USED
```

This time, the computer told you the dimensions of C$ and X(. It also told you that B is equal to 6.

```
SELECT ITEM OR RETURN FOR MENU
I

READY
PRINT B:REM make sure it is 6
6

READY
```

Notes

VARIABLE VALUES is most useful when you're debugging a program. You can press the BREAK key to stop a running program, go to BASIC, and CONTinue running the program.

Although the computer prints the DIMension of strings, it doesn't print their length. If X$ has been dimensioned, the BASIC command PRINT LEN(X$) will print the length of X$.

COMMAND C: CHANGE NAME

Function

This command lets you change the name of any variable in your program. You can make long variable names short, or change short names such as F into more descriptive names like FOOD.

Using it

```
SELECT ITEM OR RETURN FOR MENU
C

CHANGE NAME--OLD VARIABLE NAME?
the name the variable has now

NEW NAME (MUST BE SAME TYPE)?
the name you want to change it to
```

To return to the SELECT ITEM prompt without changing any names, press RETURN or BREAK in response to either question.

Remember to end the name of an array with the "(" character; for example, array X is named X(. The name of a string variable ends with a dollar sign (example: C$).

Error messages

LINE TOO LONG -- you entered a name that is more than one screen line (38 characters) long. CHANGE NAME can only handle names that are 38 characters long or less.

NOT USED IN PROGRAM -- a variable name you entered is not used in the program, so it is impossible to change it.

BAD VARIABLE NAME -- a variable name must start with a letter and consist of letters and numbers. The name of an array ends with a "(" character, and the name of a string ends with a dollar sign. You also get this message if you type spaces before the variable name.

TYPES DO NOT MATCH -- you can only change a simple variable to a simple variable, an array to an array, or a string to a string.

NAMES ALREADY EXISTS -- you cannot change a variable name to a name already used in the program. This prevents you from having two variables with the same name. You will also get this message if you try to change a variable name to itself. It is possible to have a variable named A, an array A, and a string A$, since their respective names are A, A(, and A$--all different.

NOT ENOUGH MEMORY -- there isn't enough memory to change the variable name. This happens when the new name is too long or when you

have less than 20 bytes of free memory.

Any error message returns you to the SELECT ITEM OR RETURN FOR MENU prompt without changing the variable name.

Example

Suppose you enter the following program (your typing is underlined):

```
READY
NEW

READY
10 DIM A(10),D$(5)
20 FOR C=1 TO 10
30 B=C*C:PRINT C,B
40 NEXT C
```

Now you want to change the name of array A, string D$, and variable C:

```
DOS
(screen clears and the BASIC/XA menu appears)

SELECT ITEM OR RETURN FOR MENU
C

CHANGE NAME--OLD VARIABLE NAME?
A(

NEW NAME (MUST BE SAME TYPE)?
ARRAY(

SELECT ITEM OR RETURN FOR MENU
C

CHANGE NAME--OLD VARIABLE NAME?
D$

NEW NAME (MUST BE SAME TYPE)?
STAT$

SELECT ITEM OR RETURN FOR MENU
C

CHANGE NAME--OLD VARIABLE NAME?
C

NEW NAME (MUST BE SAME TYPE)?
B

NAME ALREADY EXISTS
```

Since variable name B is already used, the computer doesn't permit the change. Instead, it prints an error message.

```
SELECT ITEM OR RETURN FOR MENU
C

CHANGE NAME--OLD VARIABLE NAME?
C

NEW NAME (MUST BE SAME TYPE)?
COUNT

SELECT ITEM OR RETURN FOR MENU
I

READY
LIST

10 DIM ARRAY(10),STAT$(5)
20 FOR COUNT=1 TO 10
30 B=COUNT*COUNT:PRINT COUNT,B
40 NEXT COUNT

READY
```

Notes

Although you can use variables names like INPUT, PRINT, and so on, avoid doing so, because BASIC may not let you edit your program. If you make this mistake, use CHANGE NAME to change the variable name to one that BASIC likes, such as X or INP.

If you use long lines in your program and you make your variable names too long, you won't be able to edit some of the lines in your program because they'll be more than three screen lines long. To edit these lines, you'll have to shorten the names.

COMMAND D: CROSS REFERENCE

Function

This command lists the variable names used in your program and the lines that use them. It also tells you how many variable names have been used. If you've used command H, NEW OUTPUT FILE, the table prints to the file you selected.

Using it

```
SELECT ITEM OR RETURN FOR MENU
D
```

Error messages

INPUT/OUTPUT ERROR -- the computer could not print to your file OR you pressed the **BREAK** key. If you were sending output to a file, the program closes the file, and the rest of the table prints on the TV screen.

Example

Suppose you enter the following program (your typing is underlined):

```
READY
NEW

READY
10 FOR C=1 TO 10
20 B=C*C:PRINT C,B
30 NEXT C
PRINT A
0

READY
```

Now you want to list the variable in the program and the lines that use them:

```
DOS
(screen clears and the BASIC/XA menu appears)

SELECT ITEM OR RETURN FOR MENU
D
```

```
VARIABLE CROSS REFERENCE TABLE

C
      10     20     30
B
      20
A

3 VARIABLES USED

SELECT ITEM OR RETURN FOR MENU
I

READY
```

Note that variable A is not used in the program, since there are no line numbers following its name. However, it is taking up space and pushing the program closer to ATARI BASIC's limit of 128 variable names.

Notes

On the screen, the cross reference table has four columns per line. When you use command H, NEW OUTPUT FILE, to send the table to the printer or a file, the table has ten columns per line (for an 80-column printer). To adjust BASIC/XA for a 40-column printer, see CUSTOMIZING THE PROGRAM.

To remove unused variable names from your program, follow this procedure:

1. Load the program into memory.

2. If you have a cassette recorder:

   a. Type LIST "C:" and press RETURN. When you hear two beeps, place a blank tape in the recorder. Then press the PLAY and RECORD buttons on the recorder, and press the RETURN key on the computer console. The computer will list the program to the cassette.

   b. Type NEW and press RETURN to remove the tokenized version of the program and the variable names from memory.

   c. Rewind the cassette and press PLAY on the recorder to prepare it for loading the program. Then type ENTER "C:" and press RETURN. When you hear a beep, press RETURN again. The computer will get your program from the cassette. When the READY prompt appears, press STOP on the recorder.

   If you have a disk drive:

   a. Place a diskette with plenty of free space in drive one. Then type LIST "D:TEMP" and press RETURN. The computer will list

the program to diskette.

b. Type NEW and press RETURN to remove the tokenized version of the program and the variable names from memory.

c. Type ENTER "D:TEMP" and press RETURN. The computer will get your program from the diskette.

d. To remove the "D:TEMP" file from the diskette, type XIO 33,#1,0,0, "D:TEMP" and press RETURN.

4. Save your program to cassette or diskette. The unused variable names will now be gone.

Normally, BASIC stores your program in a "tokenized" form, meaning that commands like PRINT are stored as a single number. Variable names are stored in a table and referred to by a number. For example, the command PRINT X is translated into two numbers, the number for PRINT and the position of X in the variable name table. If you stop using a variable name, BASIC still keeps it in the table. When you LIST the program to tape or diskette and then ENTER it, BASIC translates the program into numbers all over again. Since unused variable names do not show up in the listing, they are removed from the variable name table.

COMMAND E: DELETE LINES

Function

This command lets you delete a range of lines from your program.

Using it

```
SELECT ITEM OR RETURN FOR MENU
E

DELETE--START, END LINES?
starting line, ending line
```

To return to the SELECT ITEM prompt, press RETURN or BREAK instead of entering the starting and ending lines.

Error messages

BAD NUMBER -- you didn't type two numbers separated by a comma.

LINE TOO LONG -- your response was longer than one line on the screen.

NUMBER OUT OF RANGE -- one of the numbers you typed was negative or greater than 32767. BASIC uses line numbers from 0 to 32767.

SECOND LINE # MUST BE LARGER -- the first line number must be smaller than or equal to the second one.

When the program prints any error message, you return to the SELECT ITEM or RETURN FOR MENU prompt without deleting any lines.

Example

Suppose you enter the following program (your typing is underlined):

```
READY
NEW

READY
10 REM THIS LINE WILL REMAIN
20 REM THESE LINES WILL BE DELETED
22 REM
27 REM
30 REM THIS LINE WILL REMAIN
```

Now you want to delete lines 20 through 27:

```
DOS
(screen clears and the BASIC/XA menu appears)

SELECT ITEM OR RETURN FOR MENU
E

DELETE--START, END LINES?
20,27

SELECT ITEM OR RETURN FOR MENU
I

READY
LIST

10 REM THIS LINE WILL REMAIN
30 REM THIS LINE WILL REMAIN

READY
```

Notes

If you have many lines to delete, DELETE LINES may take several minutes. DO NOT press SYSTEM RESET or you will lose your program and lock up the computer. Just wait for the program to finish and return to the SELECT ITEM OR RETURN FOR MENU prompt.

For example, typing "400,32767" in response to the DELETE prompt will delete from line 400 to the end of the program.

If you delete a large section of the program, you might remove several variable names from the program. However, the names will still be stored by BASIC. Use the CROSS REFERENCE command to see how many unused variable names are in the table. You can remove unused names by following the steps listed in the notes for the CROSS REFERENCE command.

COMMAND F: RENUMBER

Function

This command lets you renumber your program. You choose the new starting line number and the spacing between lines.

Using it

```
SELECT ITEM OR RETURN FOR MENU
F

RENUMBER--NEW STARTING LINE, SPACING?
new starting line number, spacing between line numbers
```

If you press RETURN without entering numbers, RENUMBER uses 10 for the new starting line number and 10 for the spacing between lines.

To return to the SELECT ITEM prompt without renumbering, press the BREAK key when you see the RENUMBER prompt.

Error messages

These messages appear if you answer the RENUMBER prompt incorrectly. You return to the SELECT ITEM prompt without renumbering the program.

LINE TOO LONG -- your answer was longer than one screen line.

BAD NUMBER -- you didn't type two numbers separated by a comma.

NUMBER OUT OF RANGE -- one of the numbers you typed was negative or greater than 32767. BASIC uses line numbers from 0 to 32767.

SPACING CAN'T BE ZERO -- you cannot have zero spacing between lines, since all lines would have the same line number.

CAN'T RENUMBER -- renumbering would result in a line number within SPACING of 32767. For example, if the SPACING was 10, you would get CAN'T RENUMBER if renumbering would result in a line number greater than 32757. Try renumbering the program again with a smaller SPACING.

Warning messages

These messages appear during renumbering. They tell you to check lines in your program.

EXPRESSION FOUND IN LINE xxx -- an expression or a negative number follows a GOTO, GO TO, GOSUB, TRAP, RESTORE, LIST,

IF/THEN, ON/GOTO, or ON/GOSUB statement in line xxx. You must update the expression; if it was in a LIST, ON/GOTO, or ON/GOSUB statement, you must update line numbers following it in the statement.

LINE #yyy, FOUND IN LINE xxx, DOES NOT EXIST -- a line number in a GOTO, GOSUB, etc., statement does not correspond to any line of the program (for example, GOTO 100 when there is no line 100). The line number is left unchanged.

BAD LINE NUMBER IN LINE xxx -- the line number in a GOTO, GOSUB, etc., statement is greater than 32767 (greater than 65535 for TRAP). When you RUN the program, this line will cause an ERROR - 7. The line number is left unchanged.

Example

Suppose you enter the following program (your typing is underlined):

```
READY
NEW

READY
10 INPUT A
20 ON A GOSUB 100,200,300
21 PRINT "TEST PROGRAM"
22 TRAP 40000
23 GOTO 23
100 REM SUB1
110 RETURN
300 REM SUB3
305 RETURN
```

Now you want to renumber the program to make room for some statements betweeen lines 21 and 22:

```
DOS

(screen clears and the BASIC/XA menu appears)

SELECT ITEM OR RETURN FOR MENU
F

RENUMBER--NEW STARTING LINE, SPACING?
(you press the RETURN key)

LINE #200, FOUND IN LINE 20,
DOES NOT EXIST

SELECT ITEM OR RETURN FOR MENU
I

READY
LIST
```

```
10 INPUT A
20 ON A GOSUB 60,200,80
30 PRINT "TEST PROGRAM"
40 TRAP 40000
50 GOTO 50
60 REM SUB1
70 RETURN
80 REM SUB3
90 RETURN

READY
```

Since TRAP with a number between 32768 and 65535 cancels previous TRAPs, RENUMBER left the TRAP 40000 alone. In line 20, although the 200 caused an error, the other line numbers were adjusted for the renumbered program.

Notes

DO NOT press SYSTEM RESET or BREAK while RENUMBER is renumbering the program. Most renumbering jobs take only a few seconds.

If you use the LIST command in your program (e.g., 10 LIST), you will get a false EXPRESSION FOUND message if the LIST is to a device, and the line numbers following the device name will NOT be changed. For example, 10 LIST "P:",100,200 will cause an EXPRESSION FOUND message, and the LIST command will not be updated to reflect the new line numbers.

COMMAND G: CHECK PROGRAM

Function

This command lets you check your program for line numbers that don't exist, bad line numbers, INPUT statements without variables, and lines with syntax error. It also tells you about expressions used as line numbers.

Using it

```
SELECT ITEM OR RETURN FOR MENU
G
```

Warning messages

INPUT/OUTPUT ERROR -- you pressed the BREAK key. If you were sending output to a file, the file is closed and the output of LIST VARIABLES, VARIABLE VALUES, and CROSS REFERENCE will be sent to the TV screen.

EXPRESSION FOUND IN LINE xxx -- an expression or a negative number follows a GOTO, GO TO, GOSUB, TRAP, RESTORE, LIST, IF/THEN, ON/GOTO, or ON/GOSUB statement in line xxx. Line numbers following the expression in the same statement are not checked. This is not really an error, but you will get this message again when you renumber the program.

LINE #yyy, FOUND IN LINE xxx, DOES NOT EXIST -- a line number in a GOTO, GOSUB, etc., statement does not correspond to any line in the program (for example, GOTO 100 when there is no line 100). This line will cause an ERROR - 12 when the program is RUN and reaches the line.

BAD LINE NUMBER IN LINE xxx -- the line number in a GOTO, GOSUB, etc., statement is greater than 32767 (greater than 65535 for TRAP). When you RUN the program, this line will cause an ERROR - 7.

INPUT BY ITSELF IN LINE xxx -- the INPUT statement in line xxx is not followed by a variable name. When you RUN the program, this line will lock the computer up, losing your program. Although BASIC checks for syntax errors, this is the one syntax error it doesn't catch until it's too late.

SYNTAX ERROR IN LINE xxx -- when you entered the line, BASIC told you it contained a syntax error and you didn't fix the line. When you RUN the program, this line will cause an ERROR - 17.

Example

Suppose you enter the following program (your typing is underlined):

```
READY
```

```
NEW

READY
10 PRINT "YOUR NUMBER"
20 INPUT
30 IF A=3 THEN 20
40 IF A=4 THEN 100
50 TRAP 100000
```

Now you want to check the program for the errors listed above:

```
DOS

(screen clears and the BASIC/XA menu appears)

SELECT ITEM OR RETURN FOR MENU
C

LINE #100, FOUND IN LINE 40,
DOES NOT EXIST
BAD LINE NUMBER IN LINE 50
INPUT BY ITSELF IN LINE 20

SELECT ITEM OR RETURN FOR MENU
I

READY
```

CHECK PROGRAM caught three errors: the branch to line 100 in line 40, the TRAP 100000 in line 50, and the INPUT in line 20. If you had RUN this program immediately, the computer would have locked up. Now let's fix the errors and try it again:

```
20 INPUT A
40 IF A=4 THEN PRINT "OK"
50 TRAP 40000
DOS

(screen clears and the BASIC/XA menu appears)

SELECT ITEM OR RETURN FOR MENU
C

SELECT ITEM OR RETURN FOR MENU
I

READY
```

This time, CHECK PROGRAM did not find any errors.

Notes

CHECK PROGRAM is most useful for catching bad GOTO and GOSUB statements. Often, you don't find these mistakes until you've RUN a

program several times. CHECK PROGRAM catches all these mistakes at once.

CHECK PROGRAM cannot find logical errors in your program, such as setting A to 5 when it should be 6.

COMMAND H: NEW OUPUT FILE

Function

This command lets you send the LIST VARIABLES, VARIABLE VALUES, and CROSS REFERENCE tables to the printer, a cassette file, or a diskette file. Having a printed copy of these tables is very useful.

Using it

```
SELECT ITEM OR RETURN FOR MENU
H

NEW OUTPUT FILE (RETURN FOR SCREEN)?
filename
```

The filename may be any of the following:

P:	Sends tables to the printer
C:	Sends tables to the cassette recorder. When you hear two beeps, place a blank tape in the recorder, press PLAY and RECORD, and press RETURN.
D:filename.ext	Sends tables to a disk file. Anything that was in the file before will be lost.
Rn:	Sends tables to serial port #n of the ATARI 850 Interface. You may have to condition the port with XIO statements before selecting it for printing.
E: or RETURN	Sends tables to the screen

If you were already sending output to a file, the program closes the previous file before opening the new file.

Error messages

INPUT/OUTPUT ERROR -- you pressed the BREAK key while the computer was trying to open the file.

LINE TOO LONG -- the filename you gave was more than one screen line too long.

CAN'T OPEN FILE -- when the computer tried to open the file, an error happened. The most common errors are:

1. Device timeout -- you selected the printer and a printer was not connected or it was not turned on. You also may have left your drive off when you tried to send output to a diskette file.

2. Locked file -- you selected a diskette file and it was locked.

3. Diskette write-protected -- the write-protect notch on the diskette is covered or the diskette does not have a write-protect notch.

4. Bad filename -- you forgot the D: for a diskette filename or the filename was bad.

Example

```
SELECT ITEM OR RETURN FOR MENU
H

NEW OUTPUT FILE (RETURN FOR SCREEN)?
P:

SELECT ITEM OR RETURN FOR MENU
E

SELECT ITEM OR RETURN FOR MENU
D
```

Both the VARIABLE VALUES table and the CROSS REFERENCE table will be sent to the printer.

Notes

If there is an error when the computer tries to print to the file, you will see the INPUT/OUTPUT ERROR message on the screen. The program automatically closes the file and sends the rest of the listing to the screen.

When you select command I, RETURN TO BASIC, or command J, GO TO DOS MENU, the program closes the file. If you select command H again, the program closes the file before opening the new one.

If you have any file open, don't press SYSTEM RESET, because you may lose the file. Once you return to BASIC, go to the DOS menu, or select NEW OUTPUT FILE and send output to the screen, the program closes the file, making it safe to press SYSTEM RESET.

COMMAND I: RETURN TO BASIC

Function

This command lets you return to ATARI BASIC from BASIC/XA.

Using it

```
SELECT ITEM OR RETURN FOR MENU
I

READY
```

Notes

If you were sending output to a file, the file will be closed.

COMMAND J: GO TO DOS MENU

Function

This command lets you go to the DOS menu from BASIC/XA.

Using it

```
SELECT ITEM OR RETURN FOR MENU
J

(DOS menu appears on the screen)
```

Notes

The diskette must have a MEM.SAV file. While you are in DOS, you must not give DOS permission to use the program area. When you return to BASIC (DOS option B or SYSTEM RESET), you must have the same diskette in drive one as when you went to the DOS menu. See IMPORTANT NOTES for more information.

If you were sending output to a file, the file will be closed.

CUSTOMIZING THE PROGRAM

USING CROSS REFERENCE WITH A 40-COLUMN PRINTER

The CROSS REFERENCE command normally prints ten numbers to a printer line. This is fine for 80-column printers, but messy for 40-column ones. To adjust CROSS REFERENCE for a 40-column printer:

1. If you have the cassette version of BASIC/XA (WARNING: Do NOT use this POKE unless you have loaded BASIC/XA. If you haven't, you could lock up the computer.):

   a. Type POKE 5681,4 if you have a 40-column printer.

   b. To reset CROSS REFERENCE for 80 columns, type POKE 5677,10.

2. If you have the diskette version of BASIC/XA (WARNING: Do NOT use this POKE unless you have loaded BASIC/XA. If you haven't, you could lock up the computer.)::

   a. Type POKE 11321,4 if you have a 40-column printer.

   b. To reset CROSS REFERENCE for 80 columns, type POKE 11317,10.

Each time you load BASIC/XA, you must do the POKE if you want 40-column output.

If you have the diskette version of BASIC/XA, these steps modify the program for your printer:

1. Prepare a diskette that contains the DOS.SYS, DUP.SYS, and MEM.SAV files. The diskette should not have an AUTORUN.SYS file (it will be replaced). It should have at least 36 free sectors (DOS prints the number of free sectors at the end of the directory).

2. If you have 24K of RAM, turn your computer off and turn it back on using a diskette that does not have a copy of BASIC/XA. If you have an ATARI 850 Interface Module, leave it off. These steps are necessary because CUSTOM.BAS uses all 24K of memory.

3. Insert the BASIC/XA diskette into drive one. Type RUN "D:CUSTOM.BAS" and press RETURN. The program loads into memory and displays:

```
EXTENDED ATARI BASIC VERSION 1.1
Copyright 1982 Thomas Newton

This program lets you relocate EXTENDED ATARI
BASIC for your system.
```

```
Please hold on while I get ready...

Do you want to relocate the program
(type Y or N)?
```

Type N and press RETURN.

4. The computer asks:

```
Do you have an 80-column printer
(type Y or N)?
```

If you have a 40-column printer, type N and press RETURN. If you have an 80-column printer, type Y and press RETURN.

5. The computer asks:

```
Do you want the program to check for
an ATARI 850 Interface included with
EXTENDED ATARI BASIC (Type Y or N)
?
```

Type Y and press RETURN.

6. The screen clears and displays:

```
EXTENDED ATARI BASIC VERSION 1.1
Copyright 1982 Thomas Newton

Program loads at: 7420
Program ends at : 11491
Columns per line: 4      (10 for an 80  column
printer)

Place a system diskette (one that has
a copy of DOS) in drive one and press
RETURN to write the AUTORUN.SYS file.
Press any other key to quit without
writing the AUTORUN.SYS file.
```

7. Insert the diskette that you prepared in step 1 into drive one and press RETURN. The computer saves a copy of BASIC/XA on the diskette.

8. The computer types:

```
Your disk now contains a copy of
EXTENDED ATARI BASIC.  To use the
program, place the disk in drive one
when you turn your system on.

READY
```

RELOCATING THE PROGRAM FOR AN ALTERED DOS

If you change the number of drive buffers or file buffers that DOS uses (described in the DOS II Reference Manual), you must relocate BASIC/XA to work with your version of DOS. To make a copy of BASIC/XA for your system:

1. Prepare a diskette that has the DOS.SYS, and MEM.SAV files. This diskette contains your version of DOS. It should not have an AUTORUN.SYS file. It should have at least 36 free sectors (DOS prints the number of free sectors at the end of the directory).

2. Turn your computer off. Turn it back on using the diskette you prepared in step 1. If you have an ATARI 850 Interface Module, leave it off. Type PRINT PEEK(743)+256*PEEK(744) , press RETURN, and write down the number on the screen.

3. Turn your computer off. Turn it back on using the DOS II Master Diskette. If you have an ATARI 850 Interface Module, leave it off.

4. Insert the BASIC/XA diskette into drive one. Type RUN "D:CUSTOM.BAS" and press RETURN. The program loads into memory and displays:

```
EXTENDED ATARI BASIC VERSION 1.1
Copyright 1982 Thomas Newton

This program lets you relocate EXTENDED ATARI
BASIC for your system.

Please hold on while I get ready...

Do you want to relocate the program
(type Y or N)?
```

5. Type Y and press RETURN. The computer asks:

```
Where should the program start
(give address in decimal)?
```

Type the number you wrote down in step 2 and press RETURN. The computer types:

```
Hold on while I relocate the program
```

6. After the computer finishes, it asks:

```
Do you have an 80-column printer
(type Y or N)?
```

Answer Y or N and press RETURN.

7. The computer asks:

```
Do you want the program to check for
an ATARI 850 Interface included with
EXTENDED ATARI BASIC (Type Y or N)
?
```

Type Y and press RETURN.

8. The screen clears and displays:

```
EXTENDED ATARI BASIC VERSION 1.1
Copyright 1982 Thomas Newton

Program loads at: xxxxx
Program ends at : yyyyy
Columns per line: zz

Place a system diskette (one that has
a copy of DOS) in drive one and press
RETURN to write the AUTORUN.SYS file.
Press any other key to quit without
writing the AUTORUN.SYS file.
```

9. Insert the diskette that you prepared in step 1 into drive one and press RETURN. The computer saves a copy of BASIC/XA on the diskette.

10. After the computer saves the program, it types:

```
Your disk now contains a copy of
EXTENDED ATARI BASIC.  To use the
program, place the disk in drive one
when you turn your system on.
```

TRANSFERRING THE CASSETTE VERSION TO DISKETTE

INTRODUCTION

Although you can't use the program on the first side of the BASIC/XA cassette with a diskette, BASIC/XA includes a second program for use with diskettes.

TRANSFERRING BASIC/XA

To transfer BASIC/XA to diskette, you need DOS II and at least 24K of RAM.

1. Prepare a diskette.

   a. Insert the ATARI BASIC Language Cartridge in the cartridge slot.

   b. Insert the DOS II Master Diskette in disk drive one.

   c. Turn on your computer. Note. If you have an ATARI 850 Interface Module, leave it off.

   d. When you see the READY prompt, type DOS and press RETURN.

   e. When the screen displays the DOS menu, place a new (blank) diskette in drive one and type these underlined responses:

```
SELECT ITEM OR RETURN FOR MENU
I
WHICH DRIVE TO FORMAT?
1
TYPE "Y" TO FORMAT DISK 1
Y
```

The disk drive whirs and clicks for a little while.

```
SELECT ITEM OR RETURN FOR MENU
H
WHICH DRIVE TO WRITE DOS FILES TO?
1
TYPE "Y" TO WRITE DOS FILES TO DRIVE 1?
Y
WRITING NEW DOS FILES

SELECT ITEM OR RETURN FOR MENU
N
TYPE "Y" TO CREATE MEM.SAV
Y
```

You have now prepared the diskette.

```
SELECT ITEM OR RETURN FOR MENU
B

READY
```

2. Place the BASIC/XA cassette in the program recorder and rewind it to the beginning of side 2.

3. Press PLAY on the recorder. Type CLOAD and press RETURN. When you hear a bell, press RETURN again. The computer loads the program into memory and types READY when through.

4. Type SAVE "D:CUSTOM.BAS" and press RETURN to save the program on diskette. To create the AUTORUN.SYS file that loads BASIC/XA:

a. Type RUN. The program displays:

```
EXTENDED ATARI BASIC VERSION 1.1
Copyright 1982 Thomas Newton

This program lets you relocate EXTENDED ATARI
BASIC for your system.

Please hold on while I get ready...

Do you want to relocate the program
(type Y or N)?
```

Type N and press RETURN.

b. The computer asks:

```
Do you have an 80-column printer
(type Y or N)?
```

If you have a 40-column printer, type N and press RETURN. If you have an 80-column rinter, type Y and press RETURN. If you don't have a printer, it doesn't matter which way you respond.

c. The computer asks:

```
Do you want the program to check for
an ATARI 850 Interface included with
EXTENDED ATARI BASIC (Type Y or N)
?
```

Type Y and press RETURN.

d. The screen clears and displays:

```
EXTENDED ATARI BASIC VERSION 1.1
Copyright 1982 Thomas Newton

Program loads at: 7420
```

```
Program ends at : 11491
Columns per line: 4 (10 for an 80-column
printer)

Place a system diskette (one that has
a copy of DOS) in drive one and press
RETURN to write the AUTORUN.SYS file.
Press any other key to quit without
writing the AUTORUN.SYS file.
```

e. Press **RETURN**. The computer saves a copy of BASIC/XA on the diskette. Then it types:

```
Your disk now contains a copy of
EXTENDED ATARI BASIC.  To use the
program, place the disk in drive one
when you turn your system on.
```

Your diskette is the same as the diskette version of BASIC/XA. Follow the instructions for the diskette version of the program.

ADVANCED TECHNICAL INFORMATION

COMBINING BASIC/XA WITH OTHER AUTORUN.SYS PROGRAMS

Since BASIC/XA is relocatable, you can combine it with many other AUTORUN.SYS programs. However, there are some restrictions:

1. The other program must not be copy-protected. You will need to make a copy of the other diskette when you combine the programs.

2. The other program must fit entirely on page six or entirely above DOS.

3. If the other program contains the code to check for the ATARI 850 Interface Module, you should remove it. The PREPARE.BAS program described below can do this job for you.

To combine the programs:

1. Make a copy of the other program diskette and remove the ATARI 850 Interface Module code using the PREPARE.BAS program. Turn your computer off. Insert the ATARI BASIC Language Cartridge in the cartridge slot of your computer, place the diskette into disk drive one, and turn the computer on. Type PRINT PEEK(743)+256*PEEK(744) and write down the number on the screen.

2. Turn the computer off. Place a diskette without an AUTORUN.SYS file in drive one and turn the computer back on. This frees memory for CUSTOM.BAS.

3. Place the diskette you used in step 1 into drive one. Type XIO 32,#1,0,0,"D:AUTORUN.SYS,PROG2" to rename the other program.

4. Place the BASIC/XA diskette in drive one and type RUN "D:CUSTOM.BAS". The program loads and displays:

```
EXTENDED ATARI BASIC VERSION 1.1
Copyright 1982 Thomas Newton

This program lets you relocate EXTENDED ATARI
BASIC for your system.

Please hold on while I get ready...

Do you want to relocate the program
(type Y or N)?
```

5. Type Y and press RETURN. The computer asks:

```
Where should the program start
(give address in decimal)?
```

Type the number you wrote down in step 1 and press RETURN. The computer types:

```
Hold on while I relocate the program
```

6. After the computer finishes, it asks:

```
Do you have an 80-column printer
(type Y or N)?
```

Answer Y or N and press RETURN.

7. The computer asks:

```
Do you want the program to check for
an ATARI 850 Interface included with
EXTENDED ATARI BASIC (Type Y or N)
?
```

Type Y and press RETURN.

8. The screen clears and displays:

```
EXTENDED ATARI BASIC VERSION 1.1
Copyright 1982 Thomas Newton

Program loads at: xxxxx
Program ends at : yyyyy
Columns per line: 10     (4 for a 40-column
                  printer)

Place a system diskette (one that has
a copy of DOS) in drive one and press
RETURN to write the AUTORUN.SYS file.
Press any other key to quit without
writing the AUTORUN.SYS file.
```

9. Place the diskette you used in step 3 in drive one and press RETURN. The computer saves a copy of BASIC/XA on the diskette.

10. The computer types:

```
Your disk now contains a copy of
EXTENDED ATARI BASIC.  To use the
program, place the disks in drive one
when you turn your system on.
```

11. Type DOS to go to the DOS menu. When the SELECT ITEM OR RETURN FOR MENU prompt appears:

a. If the other program fits entirely on page six:

```
SELECT ITEM OR RETURN FOR MENU
C
```

```
COPY--FROM, TO?
PROG2,AUTORUN.SYS/A

SELECT ITEM OR RETURN FOR MENU
D
DELETE FILESPEC
PROG2
TYPE "Y" TO DELETE...
D:PROG2
Y
```

b. If the other program loads above DOS:

```
SELECT ITEM OR RETURN FOR MENU
C
COPY--FROM, TO?
AUTORUN.SYS,PROG2/A

SELECT ITEM OR RETURN FOR MENU
D
DELETE FILESPEC
AUTORUN.SYS
TYPE "Y" TO DELETE...
D:AUTORUN.SYS
Y

SELECT ITEM OR RETURN FOR MENU
E
RENAME, GIVE OLD NAME, NEW
PROG2,AUTORUN.SYS
```

THE PREPARE.BAS PROGRAM

This program will remove the code that checks for the ATARI 850 Interface Module from any AUTORUN.SYS file. If you have the diskette version of BASIC/XA, this program is saved on the program diskette as PREPARE.BAS.

```
100 REM ********************************
110 REM **  EXTENDED ATARI BASIC      **
120 REM **       Version 1.1          **
130 REM **                            **
140 REM **  Program:  PREPARE.BAS     **
150 REM **  Thomas Newton, 7/1982     **
160 REM ********************************
170 REM
180 REM
190 DIM A$(1)
```

```
200 GRAPHICS 0:PRINT "EXTENDED ATARI BASIC VERSION 1.1"
210 PRINT "Copyright 1982 Thomas Newton"
220 PRINT:PRINT "This program remove the code that"
230 PRINT "checks for the ATARI 850 from any"
240 PRINT "AUTORUN.SYS file."
250 PRINT:PRINT "Insert your diskette in drive one"
260 PRINT "and press any key to remove the ATARI"
270 PRINT "850 code."
280 POKE 764,255
290 IF PEEK(764)=255 THEN 290
300 POKE 764,255
310 PRINT:PRINT "Working...":PRINT
320 REM
330 REM Rename AUTORUN.SYS file
340 XIO 36,#1,0,0,"D:AUTORUN.SYS":REM unlock file if locked
350 XIO 32,#1,0,0,"D:AUTORUN.SYS,AUTORUN.TMP"
360 REM
370 REM Copy program to AUTORUN.SYS,
380 REM except for ATARI 850 program
390 OPEN #1,4,0,"D:AUTORUN.TMP"
400 OPEN #2,8,0,"D:AUTORUN.SYS"
410 TRAP 640:REM End-of-file trap
420 GET #1,X:PUT #2,X:GET #1,X:PUT #2,X:REM Copy header
      bytes to file
430 REM
440 REM Block copy loop
450 GET #1,X:GET #1,Y:START=256*Y+X
460 IF START=65535 THEN 450
470 GET #1,A:GET #1,B:ADEND=256*B+A
480 REM
490 REM Check for ATARI 850 INIT addr
500 IF START<>738 OR ADEND<>739 THEN 540
510 GET #1,C:GET #1,D:IF (256*D+C)=14336 THEN 440:
    REM Skip 850 INIT address
520 PUT #2,X:PUT #2,Y:PUT #2,A:PUT #2,B,C:PUT #2,D:
   GOTO 440:REM Regular INIT--copy all bytes to output
530 REM
540 REM Check START and ADEND
550 FLAG=1:IF START=14336 AND ADEND=14411 THEN FLAG=0
560 IF FLAG THEN PUT #2,X:PUT #2,Y:PUT #2,A:PUT #2,B:
  REM write block addr
570 REM
580 REM Loop for all bytes
590 FOR ADDR=START TO ADEND
600 GET #1,BYTE:IF FLAG THEN PUT #2,BYTE
610 NEXT ADDR
620 GOTO 440
630 REM
640 REM Error trap--EOF
650 X=PEEK(195):REM get error number
660 CLOSE #1:CLOSE #2:XIO 33,#1,0,0,"D:AUTORUN.TMP":
  REM Close files and delete old program
670 IF X<>136 THEN PRINT "Disk ERROR ";X;" in line" ;
```

```
   PEEK(186)+256*PEEK(187)
680 IF X=136 THEN PRINT "Through."
```

THE ATARI 850 INTERFACE HANDLER

After the computer loads DOS and/or an autoloading program, it checks for the presence of an ATARI 850 Interface Module. If it finds one, it loads the device driver over the serial bus. The device driver uses parts of page six while loading and relocates itself at LOMEM. It uses about 2K of memory.

On a cassette-based ATARI Computer, the Operating System checks for the 850 when you turn the computer on. An autoboot cassette file will load before the interface handler.

The sequence changes for a disk-based ATARI Computer. The computer loads DOS, but due to a bug in the Operating System does not check for the interface. To fix this bug, the AUTORUN.SYS file on the Master Diskette contains a program to check for the interface module. Here is a disassmebly (labels come from the Operating System User's Manual:)

```
3800        LDA #$50          ;Device and unit numbers
3802        STA DDEVIC        ;   for R1:
3805        LDA #$01
3807        STA DUNIT
380a        LDA #$3F          ;Unknown command (probably an
380c        STA DCOMND        ;  INIT or UPLOAD command)
380F        LDA #$40          ;Will read a data frame from
3811        STA DSTATS        ;  the device
3814        LDA #$05
3816        STA DTIMLO        ;Timeout = 5/60th of a second
3819        STA DBUFHI        ;Load address = $0500
381c        LDA #$00
381e        STA DBUFLO
3821        STA DBYTHI
3824        STA DAUX1         ;Auxillary bytes set to zero
3827        STA DAUX2
382a        LDA #$0c          ;Transfer 12 bytes
382c        STA DBYTLO
382F        JSR SIOV          ;Call Serial Bus handler
3832        BPL GO            ;Return if error (which
3834        RTS               ;  means no response)
3835 GO     LDX #$0B          ;Else copy these bytes
3837 LOOP   LDA $0500,X       ;  as the new serial
383a        STA $0300,X       ;  bus commands
383d        DEX
383e        BPL LOOP
3840        JSR SIOV          ;Load driver
3843        BMI RET           ;Return on error
3845        JSR $0506         ;Init RS-232 code
3848        JMP (DOSINI)      ;Restart DOS
384b RET    RTS
```

The program loads near the top of memory in a 16K ATARI Computer (DOS requires at least 16K of memory). It is relocatable--if you move the program up in memory, you don't need to change any instructions.

When the ATARI 850 Interface Module is present, the device handler takes about 2K of memory. Since BASIC/XA requires at least 24K of memory.

SUBROUTINES IN BASIC/XA

Below is a description of each subroutine in BASIC/XA, its interface with the rest of the program, and its purpose.

```
Name:  TITLE
Entry conditions:  program just loaded
Exit conditions :  part of page six used, then set to zero:
                      cassette:  locations $0689 to $06FF
                      diskette:  locations $0600 to $0669
                   After BASIC/XA loads into memory, it
                      does not use any part of page six.  By using
                      page six during loading, I added the title
                      message with no loss of user memory.
                   Because of the nature of the autoloaded process,
                      it is impossible to tell that page six was
                      altered, since the program sets it back to
                      zero.
Purpose:  TITLE prints the title message and copyright:

          ***EXTENDED ATARI BASIC
          ***Version 1.1
          ***Copyright 1982 Thomas Newton
```

```
Name:  INIT
Entry conditions:  program just loaded or SYSTEM RESET pressed
Exit conditions :  MEMLO  = address of first byte after program
                   DOSINI = address of INIT's SYSTEM RESET routine
                   DOSVEC = address of NEWDOS (when user types DOS,
                      BASIC jumps through DOSVEC to BASIC/XA)
                   For program just loaded:
                      OLDINI = old contents of DOSINI
                      OLDDOS = old contents of DOSVEC
                   SYSTEM RESET while in BASIC/XA:
                      all registers restored
                      Screen Editor address/buffer length restored
                      contents of PTR and PTR2 restored
Purpose:  INIT links the program with the Operating System and DOS.
```

```
Name:  NEWDOS
Entry conditions:  user typed DOS while in BASIC
Exit conditions :  jumps to MENU code after
                   * saving registers, Screen Editor address/buffer length
                       and the contents of PTR and PTR2
                    * setting the INUSE flag to $FF (program in use)
                    * setting NUMIOCB to $FF (print numbers to file)
```

* setting the output file as the screen and the cross reference command for 40 columns
* setting the screen margins to (2,39) and clearing the screen with a GRAPHICS 0

While you are in the BASIC/XA menu, the program uses zero-page locations $CB through $CE as pointers. When you return to BASIC or go to the DOS menu, their contents are restored.

Name: MENU
Entry conditions: NEWDOS has just finished
Exit conditions : BASIC and DOS actually exit from the menu by popping the return address from the stack and calling the RESTORE subroutine to restore registers and pointer
Calls: PRINT, INPUT,LISTV, VALUE, CHANGE, XREF, DELLIN, RENUM, CHECK, OUTPUT, BASIC, and DOS
Except for PRINT and INPUT, MENU calls these routines by copying their addresses from a table and modifying a JSR at the end of the MENU loop.
Purpose: MENU displays the BASIC/XA menu, gets the user's choice, and calls the appropriate subroutine.

Name: PRINT
Entry conditions: Accumulator holds message #
All registers must be preserved
IOCB is $00 (screen) or $50 (file); all printing directed to the "file" goes to channel #IOCB.
Tables PRADDR and PRLEN hold the addresses and lengths of all messages.
Table PRIOCB holds one byte for each message:
$00 means always to send the message to the screen.
$FF means to send the message to file #IOCB.
Exit conditions : All registers are preserved
Input/output errors are handled internally. When an error (including BREAK) occurs, PRINT closes file #5, resets IOCB and XMAX for the screen, prints "I/O ERROR", and reprints the message.
Purpose: PRINT prints every message used by BASIC/XA (except for the title message when you load the program).

Name: OUTCHR
Entry conditions: Accumulator holds character
Exit conditions : All registers are preserved
Purpose: OUTCHR prints a single character to file #IOCB. It uses PRINT to do the actual work.

```
Name:  LISTV
Called by:  MENU
Entry conditions:  none
Exit conditions :  none
Calls:  PRINT, DUMP
Purpose:  LISTV is selection A on the BASIC/XA menu.
             It sets DUMPPTR to point to a RTS (do-nothing subroutine),
             then calls DUMP to list the variable names.

-------------------------------------------------------------------------

Name:  DUMP
Called by:  LIST,  VALUE, and XREF
Entry conditions:  DUMPPTR must be set to the address of a subroutine
Action:  For each variable name, DUMP
            prints the variable name
            calls subroutine (DUMPPTR) with
               PTR pointing to start of variable name
               VNUM = variable number (0 to 127)
               register Y holding length of name
               all registers can be altered
             prints a carriage return
          After printing the names, DUMP prints the number of variable
            names in the table.
Exit conditions :  All registers destroyed
                   PTR altered

-------------------------------------------------------------------------

Name:  VALUE
Called by:  MENU
Entry conditions:  none
Action:  Prints heading
         Changes DUMPPTR to point to VALSUB
         Calls DUMP to print the variable value table
Exit conditions :  none
Purpose:  VALUE is selection B on the BASIC/XA menu.
             It prints the variable value table.

-------------------------------------------------------------------------

Name:  VALSUB
Called by:  DUMP (through DUMPPTR)
Entry conditions:  PTR is off-limits
                   VNUM holds number (0-127)
Action:  VALSUB checks the variable value table entry for variable VNUM,
            then prints its value or dimension(s).
Exit conditions :  All registers destroyed
                   PTR2 altered
Purpose:  VALSUB prints the variable values after the variable names.

-------------------------------------------------------------------------

Name:  PINT
```

```
Entry conditions:  FR0 and FR0+1 hold a 16-bit integer in low,high form
Exit conditions :  All registers destroyed
                   MNUM holds length of ASCII representation
Purpose:  PINT prints the integer in FR0 to the output file (NUMIOCB=$FF,
,              which is most of the time), or the screen (NUMIOCB=$00).
```

```
Name:  PFLT
Entry conditions:  FR0 holds a floating-point number
Exit conditions :  All registers destroyed
                   MNUM holds length of ASCII representation
Purpose:  PFLT prints the number in FR0 to the output file (NUMIOCB=$FF,
             which is most of the time), or the screen (NUMIOCB=$00).
```

```
Name:  CHANGE
Called by:  MENU
Entry conditions:  none
Exit conditions :  none
Calls:

     1) GETVAR
          Entry:  none
          Return: BUF hold variable name with bit 7 of last character set.
          Y register holds length of variable name.
          A register holds last character (with bit 7 set).
          Carry set if error, clear if no error.  GETVAR prints
             its own error messages.
          Purpose:  GETVAR gets a variable name for CHANGE and puts it
                       in the format used by the variable name table.

     2) FINDVAR
          Entry:  BUF holds variable name to be found
                  Y register holds length of variable name
          Return: Carry set if name not found.
                  If name found (carry clear), PTR points to the start of
                     the name in BASIC's variable name table.

     3) DELETE2
          Entry:  PTR points to start of name to delete
                  LEND holds length of variable name
          Return: Variable name deleted and BASIC's pointers adjusted.
                  PTR2 is altered.

     4) INSERT
          Entry:  PTR = where to insert new variable name
                  BUF holds variable name with bit 7 of last character set
                  LENI is the length of the new variable name.
          Return: Variable name inserted into BASIC's variable name table
                  PTR2 is altered.

Purpose:  CHANGE is selection C on the BASIC/XA menu.  It lets
             the user change variable names.
```

```
---------------------------------------------------------------------------

Name: GETVAR
Called by: CHANGE
Entry/exit: described above
Purpose:  GETVAR gets a variable name from the user and puts it in the
             correct format for BASIC's variable name table.

---------------------------------------------------------------------------

Name:  FINDVAR
Called by:  CHANGE
Entry/exit:  described under CHANGE
Purpose:  FINDVAR finds the name in BUF in BASIC's variable name table.
             CHANGE uses FINDVAR twice--to find the old name in the
             table, and to make sure the new name is unused.

---------------------------------------------------------------------------

Name:  DELETE2
Called by:  CHANGE
Entry/exit:  described under CHANGE
Purpose:  DELETE2 deletes a variable name from BASIC's variable
          name table.  CHANGE uses DELETE2 to remove the old variable
          name before inserting the new one.

---------------------------------------------------------------------------

Name:  INSERT
Called by:  CHANGE
Entry/exit:  described under CHANGE
Purpose:  INSERT puts the new variable name in BASIC's variable name table

---------------------------------------------------------------------------

Name:  DELETE
Called by:  DELETE2, DELLIN
Entry conditions:  PTR points to start of delete area
                   LEND holds number of bytes to delete
Exit conditions :  PTR unchanged; PTR2 altered
                   All registers destroyed
                   BASIC pointers common to both variable names and
                      program lines adjusted for deletion
Purpose:  DELETE remove program lines and variable names from the program.

---------------------------------------------------------------------------

Name:  LINES
Called by:  DUMP, RENUM, CHECK
Entry conditions:  LINPTR = address of subroutine to call for each stmt.
Exit conditions :  PTR2 is altered by LINES
                   PTR may be altered by subroutine (LINPTR)
                   All registers destroyed
Calls:  XREFSUB, RENSUB, and CHSUB (through LINPTR)
```

```
     Conditions:  LINENO = current line number (low byte, high byte)
                  LINELEN = length of current line (one byte)
                  INDEX = offset to statement length byte from start
                          of current line
                  CMDBYT and register Y = offset to statement command
                          byte from start of current line.
                  LIMIT = offset to next statement length byte (LIMIT
                          = LINRLRN if current statement is the
                          last statement in the line)
                  PTR2 points to the start of the current line.
                  None of these variables may be altered.
                  The subroutine does not need to preserve any registers.
Purpose:  LINES loops through all the statements in the program.  By
          breaking a line into statements, LINES simplifies the jobs
          of XREFSUB, RENSUB, and CHSUB.

-------------------------------------------------------------------------

Name:  XREF
Called by:  MENU
Entry conditions:  none
Exit conditions :  none
Calls:  DUMP, PRINT
Action:  XREF prints the heading "VARIABLE CROSS REFERENCE TABLE"
         It changes DUMPPTR to LINES and LINPTR to XREFSUB
         XREF then calls DUMP:
            For each variable name, DUMP prints the name and calls LINES:
               XREFSUB checks to see if variable VNUM is used in the
               current statement and prints the line number if so.
            After printing the names, DUMP prints the number of variable
            names in the table.
Purpose:  XREF is selection D on the BASIC/XA menu.  It
          prints a cross-reference listing of variable names and
          line numbers.

-------------------------------------------------------------------------

Name:  XREFSUB
Called by:  LINES (through LINPTR)
Entry conditions:  see calling conditions for DUMP and LINES, also
                   OLDVAR = last variable for which XREFSUB printed
                      a cross reference.  OLDVAR is normally 0 to 127,
                      but XREF sets it to 255 so that the first cross-
                      reference starts a new line.
                   OLDLIN = last line number printed
                   XCNT = number of references printed on current line
                   XMAX = maximum number of cross-references per line.
                      When XCNT = XMAX, XREFSUB starts a new line before
                      printing a cross-reference.
Calls:  TOKEN, PRINT, PINT
Exit conditions :  All registers destroyed
                   OLDVAR, OLDLIN, and XCNT updated when XREFSUB prints
                      a line number
Action:  XREFSUB checks the current statement for variable #VNUM
```

```
          If the variable is in the statement:
             If VNUM is not equal to OLDVAR or LINENO is not equal
             to OLDLIN:
                Start new printing line if VNUM and OLDVAR are different.
                Start new printing line if XCNT = XMAX.
                Print LINENO plus enough spaces to pad the field to
                   seven characters
                Let XCNT = XCNT + 1
                Let OLDVAR = VNUM and OLDVIN = LINENO
Purpose:  XREFSUB prints the line numbers in the cross-reference table.
             The width of the table is set by XMAX; XREFSUB will print
             up to XMAX references per line, for a width of 7*XMAX+6
              characters (XMAX=4 for the screen, and XMAX=4 or 10 for
              printouts).

-------------------------------------------------------------------------

Name:  TOKEN
Called by:  XREFSUB, RENSUB
Entry conditions:  see calling conditions for LINES, also
                   Y register holds offset to current token from
                      the start of the line
Exit conditions :  OLDY = contents of Y register on entry
                   Y register points to next token (if any) or
                      the same one (if none)
                   A register holds symbol (if any left)--for a
                      numeric constant or string constant, the
                      A register holds the first byte.
                   Carry set if there were no tokens left in the
                      statement on entry
                   All registers destroyed
Purpose:  TOKEN gets the next token in the current statement.  Since
            XREFSUB and RENSUB need to get the next token in the current
            statement, I put the code in a subroutine.  This also
            made it easier to handle statements such as IF A=B THEN
            PRINT A, where BASIC treats the line as two statements --
            IF A=B THEN and PRINT A -- but does not put an end-of-
            statement byte between them.

-------------------------------------------------------------------------

Name:  DELLIN
Called by:  MENU
Entry conditions:  none
Exit conditions :  none
Calls:  GETTWO, DELETE, PRINT
Purpose:  DELLIN is selection E on the BASIC/XA menu.
          It deletes a range of lines from the program.

-------------------------------------------------------------------------

Name:  GETTWO
Called by:  DELLIN, RENUM
Entry conditions:  none
```

```
Exit conditions :  Carry set if error (error messages handled internally)
                   If no error, carry clear and NUM1/NUM2 hold numbers
                      in the range of 0 to 32767.
                   All registers destroyed
Purpose:  GETTWO gets two numbers from the user and checks to make sure
             that they are in the range of 0 to 32767.
```

```
Name:  RENUM
Called by:  MENU
Entry conditions:  none
Exit conditions :  none
Calls:  PRINT, LINES, GETTWO
Purpose:  RENUM is selection F on the BASIC/XA menu.  It
             renumbers a BASIC program the following way:

                   Calls PRINT and GETTWO for starting line number
                      and increment.  If the user just presses RETURN,
                      RENUM uses 10 for both numbers.
                   Checks to see if renumbering is possible by setting
                      a temporary variable to NUM1 and adding NUM2 for
                      each line of the program.  If the sum exceeds
                      32767, RENUM prints the CAN'T RENUMBER message.
                   *** IF RENUMBERING IS POSSIBLE ***
                   Changes LINPTR to RENSUB and calls LINES.  LINES calls
                      RENSUB for each statement, and RENSUB changes the
                      line reference in that statement.  To save memory,
                      RENSUB uses the line numbers at the beginning of
                      each line (which have not been changed yet) to
                      calculate new line numbers.
                   Changes the line numbers at the start of each line.
```

```
Name:  RENSUB
Called by:  LINES (through LINPTR)
Entry conditions:  see calling conditions for LINES
Exit conditions :  All registers destroyed; PTR altered
Calls:  GETNEW, PRINT, PINT
Action:  If RFLAG is zero, RENSUB updates GOTO, GOSUB, etc., references
            in the current statement and checks them for errors.  If
            RFLAG is 255, RENSUB just checks the statement, but does
            not update it.
         The combination of LINES and RENSUB checks every statement
            in the program.
Purpose:  RENSUB updates GOTO, GOSUB, etc., references for RENUMBER.
          It also checks GOTO, GOSUB, etc., references for CHECK PROGRAM.
```

```
Name:  GETNEW
Called by:  RENSUB
Entry conditions:  PTR2 off-limits
```

```
                    FLIN holds line number to find
Exit conditions :   PTR altered
                    If line FLIN exists, FRO = new line number after
                    renumbering
                    Carry set if line FLIN does not exist
```

```
Name:  CHECK
Called by:  MENU
Entry conditions:  none
Exit conditions :  none
Calls:  LINES
Action:  CHECK sets LINPTR to RENSUB, sets RFLAG to 255, and calls LINES
            to check all GOTO, GOSUB, etc., references.
         Then CHECK sets LINPTR to CHSUB and calls LINES to check for
            syntax errors and INPUT statements without variable names.
Purpose:  CHECK is selection G on the BASIC/XA menu.
          It checks the program for common errors.
```

```
Name:  CHSUB
Called by:  LINES (through LINPTR)
Entry conditions:  see calling conditions for LINES
Exit conditions :  All registers destroyed
Purpose:  CHSUB checks the current statement for syntax errors.
```

```
Name:  OUTPUT
Called by:  MENU
Entry conditions:  none
Exit conditions :  none
Calls:  INPUT, Operating System
Action:  OUTPUT closes file #5 and sends output to the screen.
         Then it prompts the user for a filename.  OUTPUT attempts
            to open the file.  If successful, it sets IOCB to $50
            (file number * 16, as required by the Operating System).
         There are two bugs in the cassette handler:  (1) sometimes
            incorrect tones are written on the tape leader, and (2)
            the motor does not stop after an OPEN for writing.  OUTPUT
            contains code to defeat these bugs (however, you will still
            encounter them in your programming).
Purpose:  OUTPUT is selection H on the BASIC/XA menu.
          It sends output from LIST VARIABLES, VARIABLE VALUES, and
             CROSS REFERENCE to the screen, printer, or tape/diskette file
```

```
Name:  BASIC
Called by:  MENU
Action:  The subroutine pulls the return address off the stack, calls
            RESTORE, and does a RTS to return to BASIC.
```

```
Purpose:  Returns to BASIC from the menu.

------------------------------------------------------------------------------

Name:  DOS
Called by:  MENU
Action:  The subroutine pulls the return address off the stack, calls
           RESTORE, and does a JMP (OLDDOS) to go to the DOS menu.
Purpose:  Goes to the DOS menu from the BASIC/XA menu.

------------------------------------------------------------------------------

Name:  RESTORE
Called by:  BASIC, DOS, INIT
Entry conditions:  none
Exit conditions :  File #5 closed
                   Screen Editor buffer and length restored
                   Contents of PTR and PTR2 restored
                   All registers restored to original values
Purpose:  RESTORE restores the state of the ATARI Computer before returnir
to
             DOS, the Operating System, or BASIC.

------------------------------------------------------------------------------

Name:  INPUT
Called by:  many subroutines
Entry conditions:  A register = maximum number of characters (incl. RETURN
Exit conditions :  BMI on error; Y register holds status/error code
Purpose:  INPUT gets a line of input from the user and masks lower case
             and inverse video.

------------------------------------------------------------------------------
```

Limited Warranty on Media and Hardware Accessories. We, Atari, Inc., guarantee to you, the original retail purchaser, that the medium on which the APX program is recorded and any hardware accessories sold by APX are free from defects for thirty days from the date of purchase. Any applicable implied warranties, including warranties of merchantability and fitness for a particular purpose, are also limited to thirty days from the date of purchase. Some states don't allow limitations on a warranty's period, so this limitation might not apply to you. If you discover such a defect within the thirty-day period, call APX for a Return Authorization Number, and then return the product along with proof of purchase date to APX. We will repair or replace the product at our option.

You void this warranty if the APX product: (1) has been misused or shows signs of excessive wear; (2) has been damaged by use with non-ATARI Home Computer products; or (3) has been serviced or modified by anyone other than an Authorized ATARI Computer Service Center. Incidental and consequential damages are not covered by this warranty or by any implied warranty. Some states don't allow exclusion of incidental or consequential damages, so this exclusion might not apply to you.

Disclaimer of Warranty and Liability on Computer Programs. Most APX programs have been written by people not employed by Atari, Inc. The programs we select for APX offer something of value that we want to make available to ATARI Home Computer owners. To offer these programs to the widest number of people economically, we don't put APX products through rigorous testing. Therefore, APX products are sold "as is," and we do not guarantee them in any way. In particular, we make no warranty, express or implied, including warranties of merchantability and fitness for a particular purpose. We are not liable for any losses or damages of any kind that result from use of an APX product.

For the complete list of current APX programs, ask your ATARI retailer for the APX Product Catalog

APX ATARI® PROGRAM EXCHANGE
P.O. Box 3705
Santa Clara, CA 95055

Review Form

We're interested in your experiences with APX programs and documentation, both favorable and unfavorable. Many of our authors are eager to improve their programs if they know what you want. And, of course, we want to know about any bugs that slipped by us, so that the author can fix them. We also want to know whether our instructions are 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 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 programs)
   _____ 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 program or user instructions:

From

STAMP

ATARI Program Exchange
P.O. Box 3705
Santa Clara, CA 95055

[seal here]