meca™

MICRO EDUCATION CORPORATION OF AMERICA

BASIC BUILDING BLOCKS™

with BASIC DESIGN TOOL™

User's Guide

ADVANCING TOMORROW'S IDEAS™

UNLAWFUL SOFTWARE DUPLICATION

Micro Education Corporation of America
285 Riverside Avenue, Westport, Connecticut 06880
A Subsidiary of Marketing Corporation of America

Printed in the United States of America

meca™
MICRO EDUCATION CORPORATION OF AMERICA

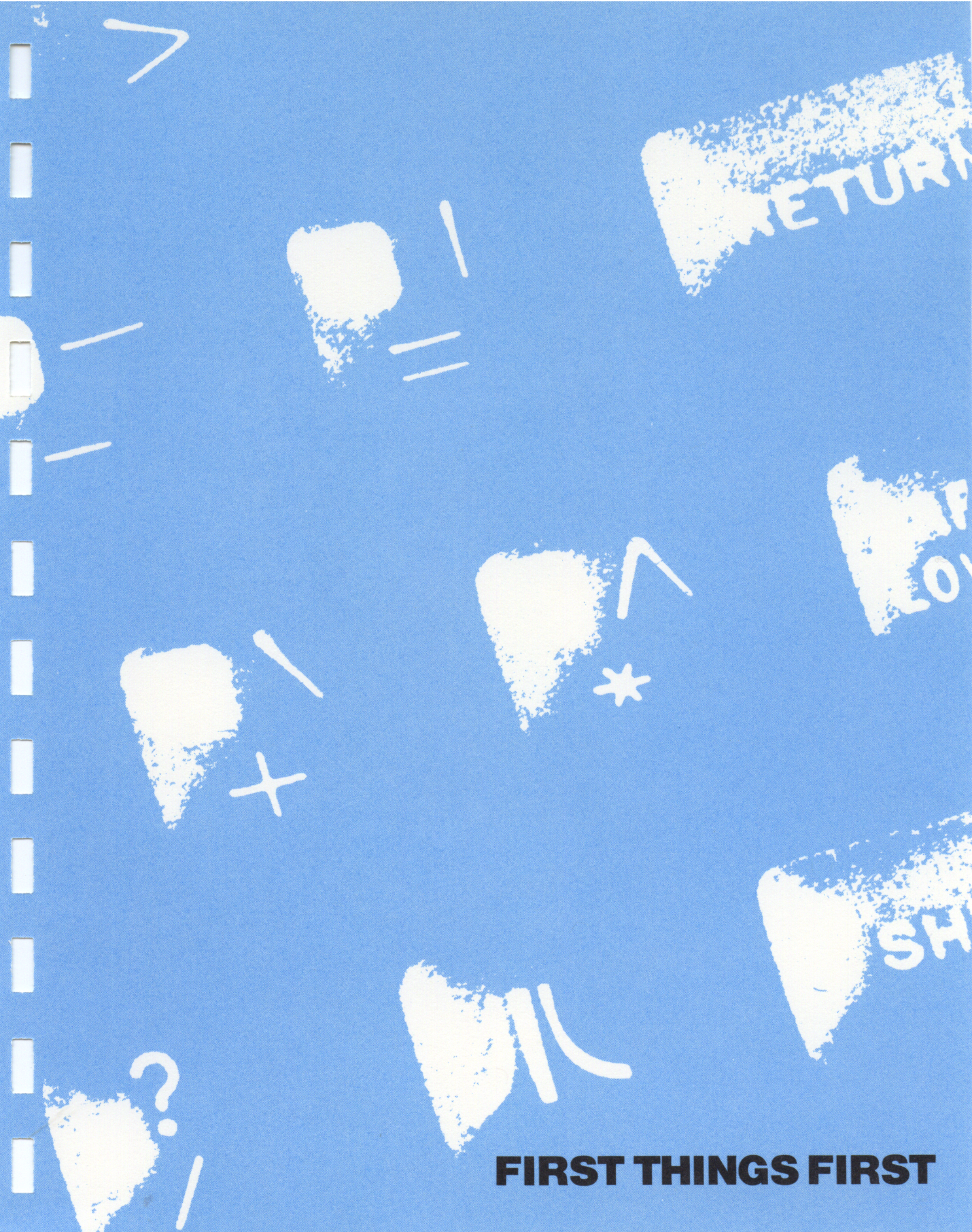
BASIC BUILDING BLOCKS™

with BASIC DESIGN TOOL™

ATARI®
USER'S GUIDE

ADVANCING TOMORROW'S IDEAS™

TABLE OF CONTENTS



FIRST THINGS FIRST

CREDITS FOR BASIC BUILDING BLOCKS™

PROGRAM DESIGN AND CODING:	KEN LIPNICKEY JERRY RUBIN GRANT SCHENCK CHRIS SURA
EDITING AND USER'S GUIDE:	JOHN HAWKINS PAM NEIDITZ
MUSIC:	RICHARD ISEN
ART:	JOHN PARKINSON
PHOTO:	MANUEL ALONSO
DESIGN:	PENNY B. HAVARD

WHAT IS INCLUDED

You should have 2 disks (one marked DISK 1, one marked DISK 2) and a USER'S GUIDE. There should also be a CUSTOMER SERVICE booklet which includes a LIMITED WARRANTY on the diskette media, a REPLACEMENT ORDER CARD and PRODUCT REGISTRATION CARD.

If any of these are not included, please call Micro Education Corporation of America (MECA) at (203) 222-1000, or see your dealer.

HOW TO GET STARTED

BASIC BUILDING BLOCKS is designed to be so easy to use that you can start using it without needing to read the manual first.

All you need to begin is an Atari 800, 1200 or XL series computer with at least 48K of RAM and one disk drive. An Atari 400 will work if it has been expanded to 48K capacity.

You will also need the Atari BASIC Cartridge if you do not have Atari BASIC built in.

Just turn the computer on with Disk 1 of the two MECA BASIC BUILDING BLOCKS disks in the drive. Be sure that you have turned the power on for the drive. If you have two drives, you can put Disk 1 in drive 1 and Disk 2 in drive 2. The program will automatically find the correct disk.

The program takes about 35 seconds to load, but it's doing a lot of work. You won't have to wait very long after it loads the first time. If you haven't put the Atari BASIC Cartridge in, the program will prompt you to do so.

Disk 1 of BASIC BUILDING BLOCKS is copy protected to prevent unlawful copying. Disk 2 is not copy protected so that you can copy STANDALONE BASIC DESIGN TOOL for your own personal use.

After a brief and entertaining introduction, the TABLE OF CONTENTS will appear. Just follow the directions shown on the screen. If you have any questions press the ? key to get a page full of HELP.

NOTE TO XL-SERIES OWNERS:

If you have any XL-Series computer, the SHIFT-ATARI keys should be replaced by SHIFT-INVERSE keys throughout the manual and tutorial program.

AN OVERVIEW OF THIS PROGRAM

This program has been carefully designed and tested to help you to learn the commands and key concepts of programming in BASIC by actually watching programs execute BASIC instructions one-by-one.

You can move through the program to learn at your own pace because the program is designed to allow you to move forward, backward, to the beginning of a chapter and to the Table of Contents at any time. If you don't understand something, you can go back and try it again until you feel you understand how it works. There is no time limit and no pressure. The computer will wait.

This program will take you step-by-step through over 40 actual programs which you can watch executing line-by-line. (These programs are listed in Appendix I.) Be sure to study the programs by going back and re-running them until you are satisfied that you know how each one is working.

This program is constructed to build advanced concepts from simpler ones, so don't go on to a new chapter until you are comfortable with what you have already studied.

As you gain new skills, you will be asked to practice those skills by writing small programs on your own. Test your own creations in the same manner that you have been learning the programming concepts in this lesson.

And there is more. When you have completed the Tutorial, you will also have learned to use what programmers call a *debugging program.* Debugging is a term which means finding program errors so that a program will work correctly. Even veteran programmers have a hard time finding errors and use a debugging program to help them.

We call our debugging program the BASIC Design Tool™ (or BDT for short). It is the same tool that you have been using to watch BASIC programs, but with some powerful extensions to help you write and debug your own programs. These extensions are explained in more detail in the section on the STANDALONE BDT TUTORIAL.

Topics covered in each chapter on the TABLE OF CONTENTS screen follow on the next page.

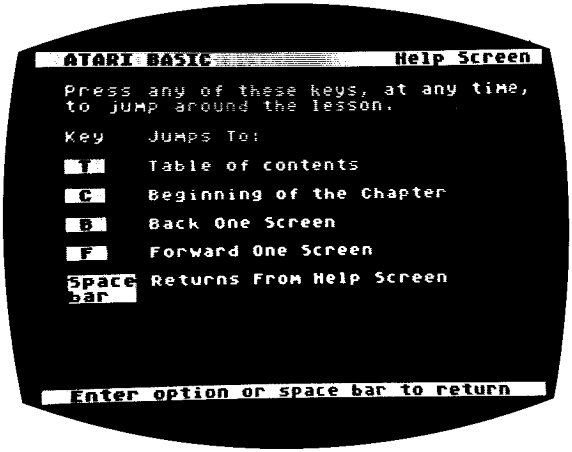
PROGRAM CHAPTER OVERVIEWS

HOW TO MOVE AROUND THE PROGRAM

When you get to the TABLE OF CONTENTS, you will note that the bottom of the screen is telling you what to do next—in this case "ENTER YOUR CHOICE." If this is your first time through the program, we suggest you begin at the beginning...CHAPTER A.

Once you enter a chapter, the bottom of each screen says "PRESS ? FOR HELP." If you hit the question mark key, even without pressing the shift key at the same time, you will get a HELP screen that will tell you how to move around the Tutorial.

This is what the HELP screen looks like:



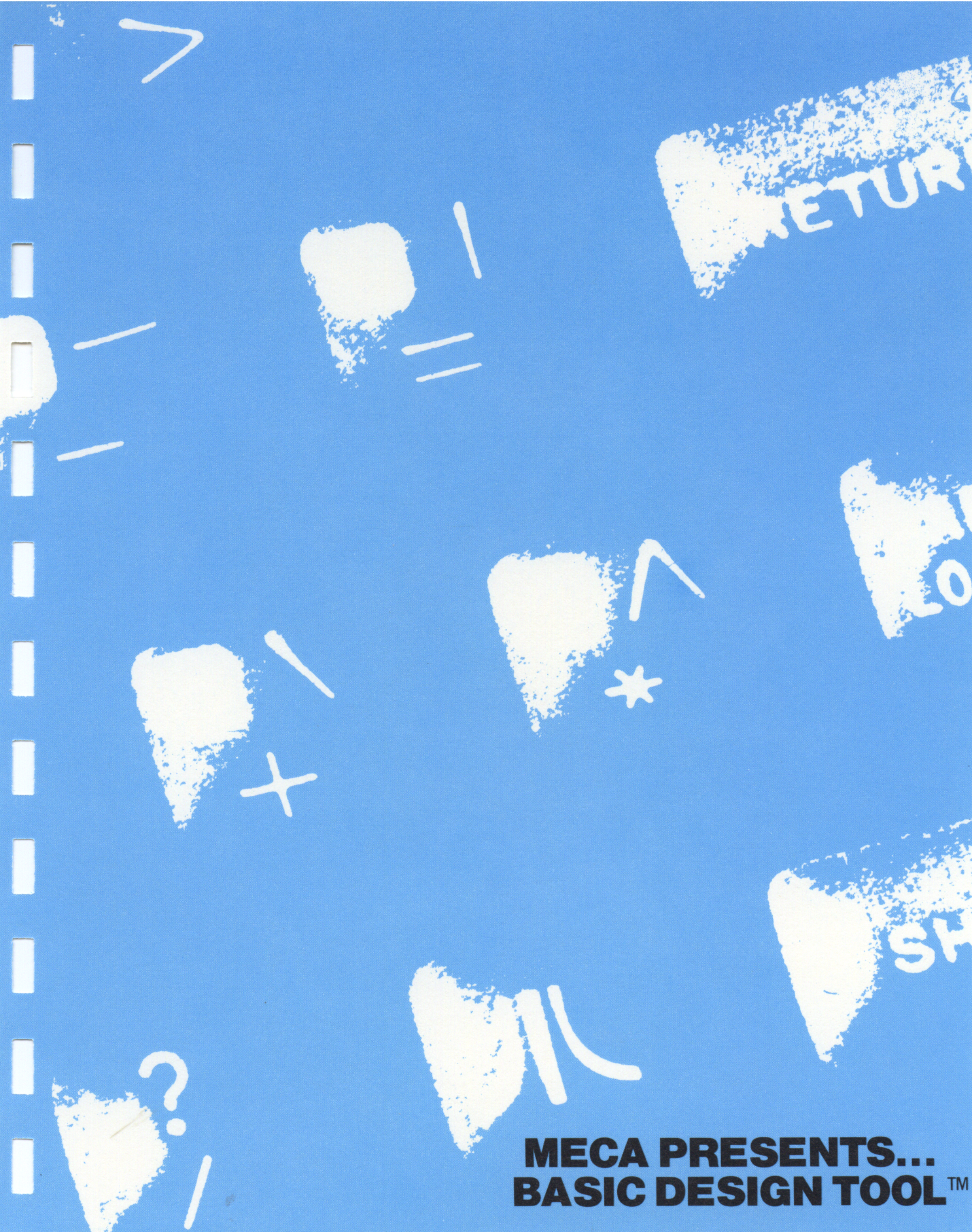
```
ATARI BASIC                 Help Screen

Press any of these keys, at any time,
to jump around the lesson.

Key     Jumps To:

 T      Table of contents

 C      Beginning of the Chapter

 B      Back One Screen

 F      Forward One Screen

Space   Returns From Help Screen
bar

   Enter option or space bar to return
```

That's all there is to it. You can use these keys at any time while you are in the lesson to move from page to page. They even work if they are pressed from the HELP page.

Here are some tips on using these keys:

F **FORWARD**
Press this key when you want to move Forward through a lesson without going through the sample programs again. This key will skip to the next page after the sample program, if you press it on the page before you would normally enter a sample program.

B **BACKWARD**
Press this key when you want to go Back and re-run a sample BASIC program in the step-by-step mode. If you press this key on the page after the sample BASIC program, for example, it will jump you to the tutorial page just before the sample program.



MECA PRESENTS...
BASIC DESIGN TOOL™

HOW TO STEP THROUGH A SAMPLE BASIC PROGRAM

This is the really powerful and fun part of using the BASIC BUILDING BLOCKS Tutorial. After you have been taught a new BASIC concept, you will enter the step-by-step mode under the control of the BASIC Design Tool. The program gives very explicit instructions on how to enter this mode and what to do next, but these instructions are reviewed in detail here just in case you don't trust your computer yet.

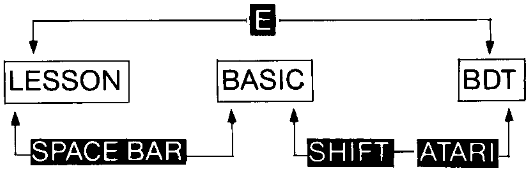
When you get to the point in the program where you will use BDT, you will see the prompt: SPACE BAR FOR EXAMPLE OR ? FOR HELP. Press the SPACE BAR as indicated.

If you are using a TV set or a color monitor the resulting screen will be the Atari blue BASIC screen. Now you should press the SHIFT—ATARI keys to enter BDT.

Remember: To get the BASIC Design Tool up and running after the lesson puts you on the blue BASIC screen, you must press the SHIFT key and the ATARI key at the same time. SHIFT—ATARI *will get you on the BASIC Design Tool screen**.

Finally, to return to the LESSON, you would PRESS E to EXIT BDT.

If you could look at your computer's memory, you would see three environments. The diagram below reviews how you move between them:

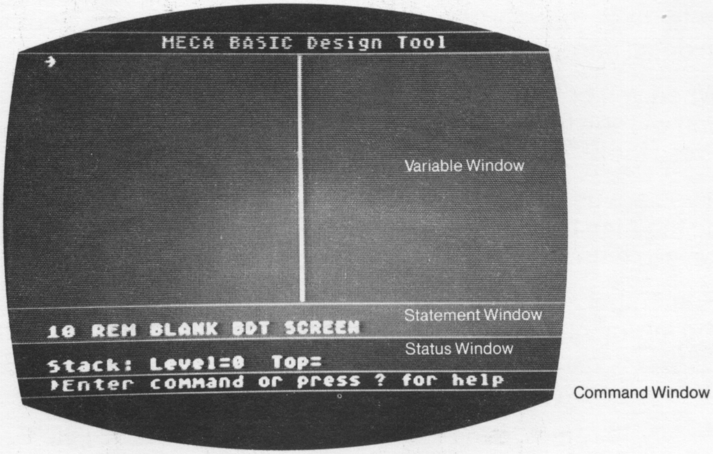


Note: If you encounter an error while on the BASIC screen, you can restart BDT by pressing SHIFT—ATARI. For example, if you gave an improper response to an INPUT statement, you would get an error message and the program would stop. Just press SHIFT—ATARI and try the program again.

***REMINDER TO XL-SERIES OWNERS:**
If you have any XL-Series computer, the SHIFT-ATARI keys should be replaced by SHIFT-INVERSE keys throughout the manual and tutorial program.

THE BASIC DESIGN TOOL SCREEN (BDT)

After you have pressed SHIFT — ATARI, you will see the BASIC Design Tool screen. This is what it looks like and what it will do...


```
MECA BASIC Design Tool
→
10 REM BLANK BDT SCREEN
Stack: Level=0  Top=
▸Enter command or press ? for help
```

Variable Window: Up to 16 variables and their values can be displayed in the window. The variables are shown on the left side of the window; the values of the variables on the right side.

Statement Window: Two lines are displayed here. The top line represents the statement which has just been executed; the bottom line displays the statement which is about to be executed.

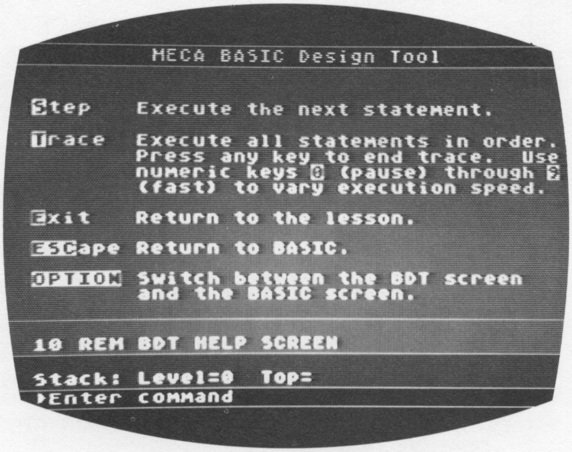
Status Window: This window shows advanced topics like the use of Breakpoints, the Gosub and the For...Next stack. Don't worry about this at first.

Command Window: This one line window shows the current command that is being executed. Also, it shows that you can press ? to get a HELP SCREEN for BDT.

BASIC DESIGN TOOL TUTORIAL COMMANDS

Once you are in BDT you can press the ? key to get a HELP SCREEN just like the one in the Tutorial, but this one will give you the commands for using the BASIC Design Tool.

This is what the BASIC Design Tool HELP SCREEN looks like:



```
            MECA BASIC Design Tool

Step     Execute the next statement.

Trace    Execute all statements in order.
         Press any key to end trace.  Use
         numeric keys 0 (pause) through 9
         (fast) to vary execution speed.

Exit     Return to the lesson.

ESCape   Return to BASIC.

OPTION   Switch between the BDT screen
         and the BASIC screen.

10 REM BDT HELP SCREEN

Stack: Level=0  Top=
▸Enter command
```

SOME POINTS THAT ARE NOT SELF EXPLANATORY

STEP: Pressing the Letter S invokes the SINGLE STEP MODE. The statement in the " About To Be Executed " Window is executed. Each time S is pressed, another statement is executed.

TRACE: Pressing the Letter T invokes the TRACE MODE. Program execution begins at the statement "About To Be Executed" in the Command Window. While you TRACE a program, you may adjust the speed of execution by pressing the numeric keys from 0 (pause) to 9 (fast). The Command Window will display "TRACE (Speed=N)" where N is the last TRACE speed.

The TRACE MODE remains in effect until (1) Any key (other than a number) is pressed, (2) a STOP or END statement is executed in the program, or (3) an error is detected.

Note: In some sample programs, TRACE will cause the program to loop endlessly. Remember to press any key (but not when you are in an input statement) to stop TRACE. Then press E to return to the lesson.

ESCAPE: When this key is pressed, BDT puts you on the BASIC screen with BASIC in control. To get back into BDT press SHIFT—ATARI.

OPTION KEY: When you hit this key, the BDT screen flips to show you the BASIC screen, but BDT is in control (unlike ESC which shows the BASIC screen with BASIC in control). Hit it again and you flip back. By using this key you can watch a program execute either line by line on the BDT or on the BASIC screen.

EXIT: This key will EXIT you from BDT back to the BASIC Tutorial. You must be under the control of BDT to EXIT to the tutorial. (If you are in BASIC, press SHIFT—ATARI and then E to return to the Tutorial.)

HOW TO TYPE-IN AND TEST YOUR OWN PROGRAMS

From time to time, you will be asked to test your comprehension of certain concepts by writing your own programs. In these cases, you will get suggestions at the end of a chapter on programs to write. Then you will enter BASIC by pressing the SPACE BAR.

At this point you will be on the blue BASIC screen and can type in your own program. When you press SHIFT—ATARI you can use BDT to watch your program run just as you have watched the sample programs. To get back to BASIC to make changes in your program, just hit the ESCAPE ESC key.

Possible solutions to SUGGESTED EXERCISES are in APPENDIX II.

Typing In Your Own Programs

1. **Entering BASIC Lines**
   Each BASIC line should start with a line number, like 10, 20, 30. This is so BASIC knows in what order to execute the statements. Lines are entered by typing the number, then the BASIC statements, and finally, pressing the RETURN key to enter the line into memory.

2. **Deleting Lines**
   To delete a line you can type the number of the line and press RETURN. Alternately, you can just type a new line starting with the same number. The new line will replace the old.

3. **Listing Lines**
   To list your whole program to the screen, just *type LIST* and press RETURN.

   You can list one line by typing LIST 10 (or any line number) and RETURN or you can list a series of lines by typing LIST 10,200 and RETURN. This would list lines 10 to 200 inclusive.

4. **Error Messages**
If you get an error message, figure out what is wrong or just try something different and retype the line with the change. If you get an error when running under BDT, you can restart BDT by pressing SHIFT—ATARI.

5. **Erasing a Program**
To erase a program from the computer's memory just type NEW and press RETURN.

NOTE: In order to use BDT, you must type at least one BASIC line before pressing SHIFT —ATARI. If you haven't typed at least one line, SHIFT — ATARI will put you back in the lesson.

NOTE: Remember, to get back to the lesson just get into BDT (SHIFT— ATARI) and press E to exit.

HOW TO EDIT YOUR PROGRAMS

Your job of editing can be made easier by using a few built-in screen editing commands.

You can edit any part of a BASIC program when it is on the screen. This would save you from retyping everything to make minor changes. For example, if you want to edit lines 10 through 60 of a program, you could just list them on the screen by typing LIST 10, 60 and RETURN.

You can move the cursor (the white box) to anywhere on the screen without changing anything by using the CTRL and arrow keys ↓↑ → ←. Just hold down the CTRL and simultaneously press one of the arrow keys. The cursor will move in the direction of the arrow.

CHANGING CHARACTERS: Position the cursor over the character and type in a new character. When you hit RETURN (anywhere on the program line), the change will be made in the line, and the cursor will move to the next program line.

INSERTING CHARACTERS: Position the cursor to where you want to insert a character and press CTRL and INSERT simultaneously. A blank space will be inserted. You can type in a character. When you hit RETURN the changes will be made.

DELETING CHARACTERS: Works the same as inserting, except you press CTRL and DELETE.

These are the most handy editing features. Please consult your ATARI documentation to learn about others.

NOTE: In the LESSON, the CAPS/LOWER key and ATARI key (inverse) have been disabled to prevent accidental presses.

STANDALONE BDT TUTORIAL

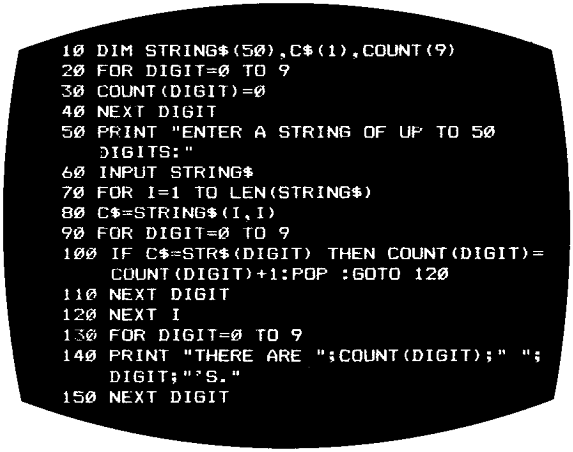
There are two versions of BDT in BASIC Building Blocks. The first, you've already seen as part of the LESSON. This is a simple version which contains a limited number of commands.

The other is the STANDALONE VERSION which you will find on Disk 2. The STANDALONE VERSION is an [AUTORUN.SYS] file which will automatically load when you boot your system with a disk that has DOS and the [AUTORUN.SYS] file on it. BDT [AUTORUN.SYS] is copyable so you can place it on your own disks for your convenience and personal use.

Before you go through this Tutorial, you should know and understand the lesson version of BDT's commands. (i.e., STEP, TRACE, The Windows, ESC and OPTION.)

We will walk you through a session using STANDALONE BDT. You should follow what is happening on your screen. A summary of all the commands for the STANDALONE BDT is listed on pages 23 and 24.

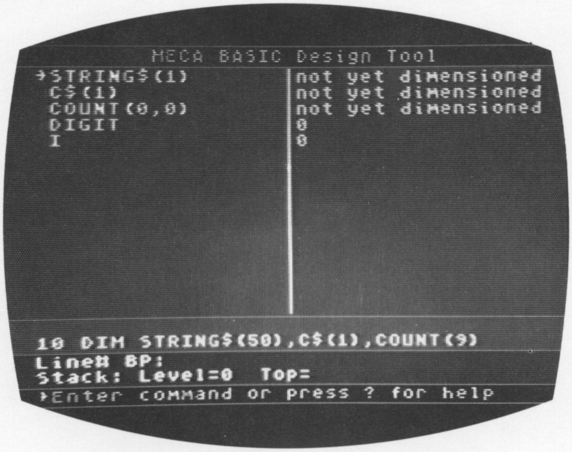
Put Disk 2 into drive 1 and turn on the computer. DOS will load BDT. You will see the title and copyright notice, and eventually a READY prompt. BDT is now installed in your machine with BASIC running. Next, LOAD the example program we have included to help you. Type: LOAD " D:SAMPLE.BAS and RETURN. LIST the program to see what it does. It should look like this:



```
10 DIM STRING$(50),C$(1),COUNT(9)
20 FOR DIGIT=0 TO 9
30 COUNT(DIGIT)=0
40 NEXT DIGIT
50 PRINT "ENTER A STRING OF UP TO 50
   DIGITS:"
60 INPUT STRING$
70 FOR I=1 TO LEN(STRING$)
80 C$=STRING$(I,I)
90 FOR DIGIT=0 TO 9
100 IF C$=STR$(DIGIT) THEN COUNT(DIGIT)=
    COUNT(DIGIT)+1:POP :GOTO 120
110 NEXT DIGIT
120 NEXT I
130 FOR DIGIT=0 TO 9
140 PRINT "THERE ARE ";COUNT(DIGIT);" ";
    DIGIT;"'S."
150 NEXT DIGIT
```

The program asks for a string and counts the number of 0's, 1's, 2's...9's.

Now let's go into BDT. There are several ways you can enter BDT. The simplest is the one you already know from the lesson : SHIFT – ATARI. Press SHIFT – ATARI and you should see the following:

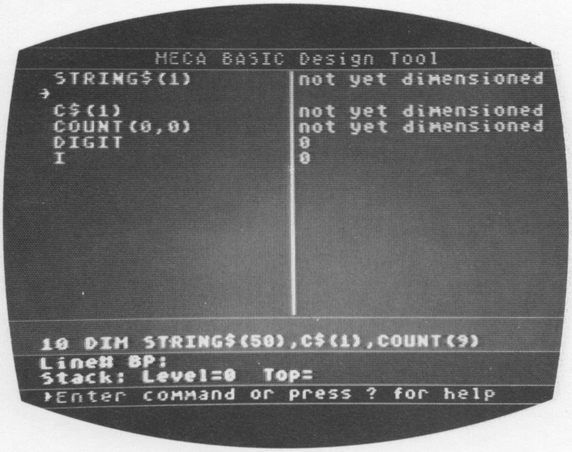


```
          MECA BASIC Design Tool
>STRING$(1)           |not yet dimensioned
 C$(1)                |not yet dimensioned
 COUNT(0,0)           |not yet dimensioned
 DIGIT                |0
 I                    |0

10 DIM STRING$(50),C$(1),COUNT(9)
Line# BP:
Stack: Level=0  Top=
>Enter command or press ? for help
```

Before you start executing the program, you should learn how to set up the Variable Table. When you boot-up BDT, it is set up to display the first 16 variables defined in your program. You will only be able to see the first 20 characters of string variables and only element zero of the arrays. You can alter this display to show what you want to see, including strings longer than 20 characters, and numeric array elements other than zero.

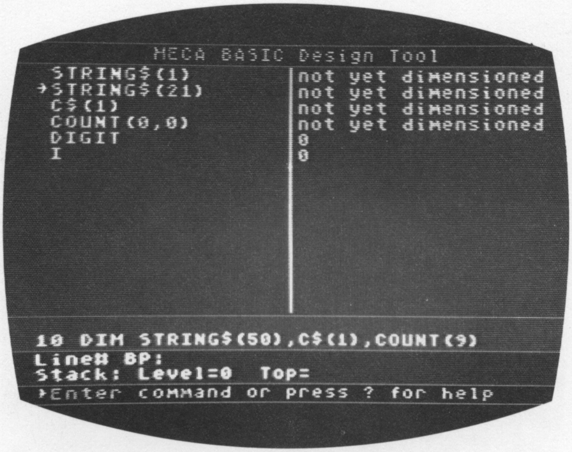
The string, STRING$, will be dimensioned to 50 characters. Therefore, we want to display 3 'substrings' of STRING$; the first 20 characters, the second 20 and the last 10.

To do this, press the down arrow key ↓ once. *(Don't use* CTRL or SHIFT.) This will move the small arrow (→) next to the STRING$(1) entry of the Variable Window down one line. Now press the right arrow key →. This should insert a blank line between STRING$(1) and C$(1). The screen should look like this:



```
           MECA BASIC Design Tool
  STRING$(1)            |not yet dimensioned
→                       |
  C$(1)                 |not yet dimensioned
  COUNT(0,0)            |not yet dimensioned
  DIGIT                 |0
  I                     |0

10 DIM STRING$(50),C$(1),COUNT(9)
Line# BP:
Stack: Level=0  Top=
▸Enter command or press ? for help
```

Now you need to SELECT STRING$ to be displayed in these two lines. Press the yellow SELECT key and STRING$ will appear in inverse on the second line of the display. Now press the right arrow key ➡ until the dimension displayed reaches 21. If you overshoot, you can always back up with the left arrow key ⬅. Once you have it at 21, press SELECT again and the line will return to a normal display. The screen should now look like this:



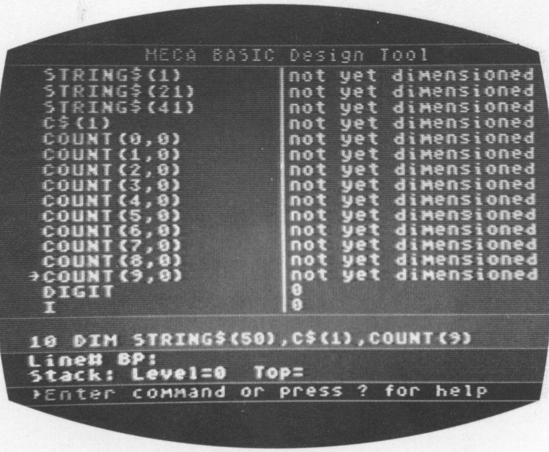
```
            MEGA BASIC Design Tool
  STRING$(1)             |not yet dimensioned
→STRING$(21)             |not yet dimensioned
  C$(1)                  |not yet dimensioned
  COUNT(0,0)             |not yet dimensioned
  DIGIT                  |0
  I                      |0

10 DIM STRING$(50),C$(1),COUNT(9)
Line# BP:
Stack: Level=0   Top=
→Enter command or press ? for help
```

Repeat the operation, i.e., down arrow ⬇, right arrow ➡, SELECT, right arrow ➡ until the dimension reaches 41, and SELECT. You should now have 3 substrings of STRING$.

Next you should set up the display of the elements in the array named COUNT. This will be dimensioned with 10 elements, 0 thru 9. At present you can only see the value of COUNT(0). The process is similar to selecting strings for display. Move the arrow pointer down to the variable DIGIT. Then insert 9 blank lines by pressing the right arrow key ➡ 9 times.

Now press SELECT and STRING$(1) should appear in inverse, because this is the first variable in BASIC's Variable Table. We now need to move through the Variable Table to locate the array COUNT. Press the down arrow key ↓ twice. This should bring COUNT(0,0) into the inversed Variable Window. Then press the right arrow key → once to select the first element of COUNT. Finally, finish the selection by pressing SELECT again. Repeat this selection 8 more times until COUNT(0,0) through COUNT(9,0) are displayed.

The screen should look like this now:

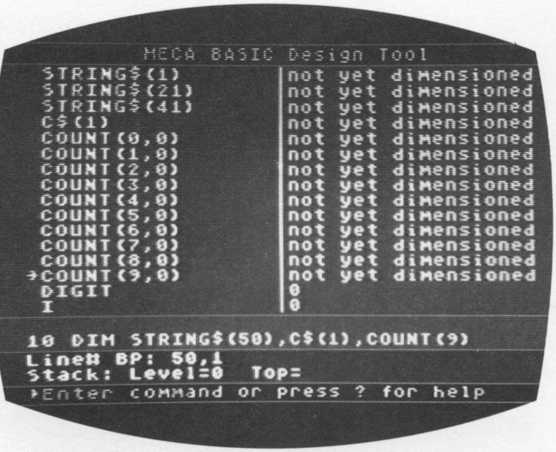


```
           MECA BASIC Design Tool
   STRING$(1)          |not yet dimensioned
   STRING$(21)         |not yet dimensioned
   STRING$(41)         |not yet dimensioned
   C$(1)               |not yet dimensioned
   COUNT(0,0)          |not yet dimensioned
   COUNT(1,0)          |not yet dimensioned
   COUNT(2,0)          |not yet dimensioned
   COUNT(3,0)          |not yet dimensioned
   COUNT(4,0)          |not yet dimensioned
   COUNT(5,0)          |not yet dimensioned
   COUNT(6,0)          |not yet dimensioned
   COUNT(7,0)          |not yet dimensioned
   COUNT(8,0)          |not yet dimensioned
  →COUNT(9,0)          |not yet dimensioned
   DIGIT               |0
   I                   |0

 10 DIM STRING$(50),C$(1),COUNT(9)
 Line# BP:
 Stack: Level=0   Top=
 ›Enter command or press ? for help
```

By using the above process, you can arrange your variables any way you want to. The only limitation is that no more than 16 different entrys can be displayed at a time.

The first thing this sample program does is to set each element in the array equal to zero. What you should do is set a Breakpoint at line 50 so you can run at full speed until the program has completed this operation.

In BDT, press B and the word Breakpoint appears in the Command Window at the bottom of the screen. Now press the right arrow key → once. A small arrow will appear next to Line# BP:. Press SELECT and a box will appear. Type in 50 for line 50, and RETURN. A comma and a smaller box will appear. BDT wants you to type in the number of times you want to pass line 50 before it will stop. Just press RETURN and BDT will supply one. The screen should look like this:



```
          MEGA BASIC Design Tool
   STRING$(1)         |not yet dimensioned
   STRING$(21)        |not yet dimensioned
   STRING$(41)        |not yet dimensioned
   C$(1)              |not yet dimensioned
   COUNT(0,0)         |not yet dimensioned
   COUNT(1,0)         |not yet dimensioned
   COUNT(2,0)         |not yet dimensioned
   COUNT(3,0)         |not yet dimensioned
   COUNT(4,0)         |not yet dimensioned
   COUNT(5,0)         |not yet dimensioned
   COUNT(6,0)         |not yet dimensioned
   COUNT(7,0)         |not yet dimensioned
   COUNT(8,0)         |not yet dimensioned
  →COUNT(9,0)         |not yet dimensioned
   DIGIT              |0
   I                  |0
 10 DIM STRING$(50),C$(1),COUNT(9)
 Line# BP: 50,1
 Stack: Level=0   Top=
 ▸Enter command or press ? for help
```

You're ready to go so press R and you'll run at full speed on the BASIC screen until the program hits the Breakpoint at line 50. Almost instantly, you'll hear a beep and you'll be back in BDT.

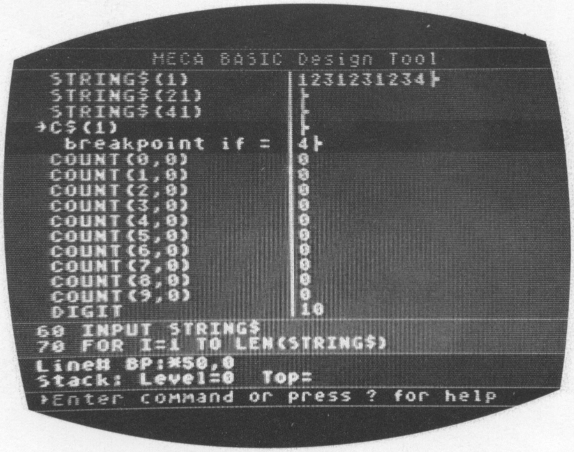
In front of the 50 on the Breakpoint line should appear a highlighted asterisk (*). This is BDT's way of saying it has hit the Breakpoint, and it is now disabled.

Try this again. Press the yellow START key to reset the program to the beginning and reset the Breakpoint. Now press T for TRACE and watch it count digits from 0 to 9. Eventually, you'll hit the line number Breakpoint and the TRACE will stop. Press S twice, enter 1231231234 in response to the INPUT statement, and hit RETURN.

Now, you should set another kind of Breakpoint—a Variable Breakpoint. Press B for Breakpoint and position the small arrow to the variable C$(1) using the up ↑ and down ↓ arrow keys. Now press SELECT and a dark blue two line window will appear with C$(1) highlighted. At this point you could select a different substring from the string starting at 1. However, this is what you want, so press SELECT again.

The words Breakpoint if <= will appear with the <= highlighted. Press the down arrow key ↓ until the = sign appears. Then press SELECT again, and a long window will light up. Type in the string that the Breakpoint should stop at. Type a 4 and RETURN.

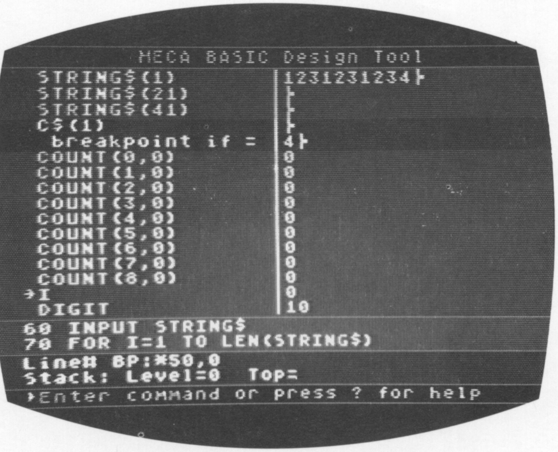
The screen should look like this:



```
                  MECA BASIC Design Tool
  STRING$(1)               |1231231234▸
  STRING$(21)              |▸
  STRING$(41)              |▸
 →C$(1)                    |▸
   breakpoint if =         |4▸
  COUNT(0,0)               |0
  COUNT(1,0)               |0
  COUNT(2,0)               |0
  COUNT(3,0)               |0
  COUNT(4,0)               |0
  COUNT(5,0)               |0
  COUNT(6,0)               |0
  COUNT(7,0)               |0
  COUNT(8,0)               |0
  COUNT(9,0)               |0
  DIGIT                    |10
60 INPUT STRING$
70 FOR I=1 TO LEN(STRING$)
Line# BP:*50,0
Stack: Level=0  Top=
▸Enter command or press ? for help
```

Did you notice that when you created the line number Breakpoint the entry for Variable "I" disappeared off the bottom of the screen? This is because a Breakpoint entry takes up two lines. You should get rid of COUNT(9,0), so you can see the value of "I." Move the arrow pointer to COUNT(9,0) using the up ↑ and down ↓ arrow keys. Then press the left arrow key ←. This will remove COUNT(9,0)'s entry.

Insert a blank line by pressing the right arrow key →. Then press SELECT and the up ↑ or down ↓ arrow key until "I" appears in the window. Finally complete the operation by pressing SELECT again. The screen should now look like this:



```
            MECA BASIC Design Tool
  STRING$(1)              |1231231234├
  STRING$(21)             |├
  STRING$(41)             |├
  C$(1)                   |├
   breakpoint if =        |4├
  COUNT(0,0)              |0
  COUNT(1,0)              |0
  COUNT(2,0)              |0
  COUNT(3,0)              |0
  COUNT(4,0)              |0
  COUNT(5,0)              |0
  COUNT(6,0)              |0
  COUNT(7,0)              |0
  COUNT(8,0)              |0
 →I                       |0
  DIGIT                   |10
60 INPUT STRING$
70 FOR I=1 TO LEN(STRING$)
Line# BP:*50,0
Stack: Level=0   Top=
›Enter command or press ? for help
```

Now, you should run the program at full speed again. *But,* instead of R for Run, type C for Continue. The difference is that C runs from the present line number while R starts from the beginning.

The BASIC screen should flash up and then you should hear a beep. You should see a highlighted asterisk next to the word Breakpoint in the Variable Window. This indicates that the Breakpoint occurred.

Let's watch that again with a TRACE T. First, you should reenable the Breakpoint. Make sure that the small arrow is pointing at C$(1). If it isn't, move it there with the up-down arrow keys ↑↓.

Press the TAB key and the asterisk (*) should disappear. This means that the Breakpoint is enabled.

Now let's go back to just after the INPUT statement. Press G for GOTO and the command line will display GOTO with an inverse box next to it. Type in 70 and RETURN. Note that line 70 appears in the about to be executed line of the Statement Window. Now TRACE T and sit back and wait for the breakpoint to happen.

Woops! It happened right away. This is because C$ still contains "4." Press ESC to go into BASIC. Then type an immediate command C$="0" and RETURN. Then go back to BDT, but *don't use* SHIFT — ATARI. Instead, hold down the CTRL key and press ATARI, or CTRL — ATARI. This will return you to BDT at the same place you left.

Now TRACE T, possibly at speed 9, until you hit the Variable Breakpoint you previously set.

Press C and you'll continue all the way to the end of the program.

You may want to disable breakpoints. Hit the START key to reset the program to the beginning. To delete the Line # Breakpoint, press B and the right arrow key → once. The inverse box will appear next to the arrow at Line # BP: . Just press RETURN and the Line # Breakpoint will be deleted.

To delete the Variable Breakpoint, press B again. This time move the arrow on the Variable Window to the Breakpoint Variable. Hit SELECT once and the Breakpoint will disappear (but the Variable will stay).

To RUN again, press START to re-initialize the program and then press R.

SUMMARY OF BDT COMMANDS

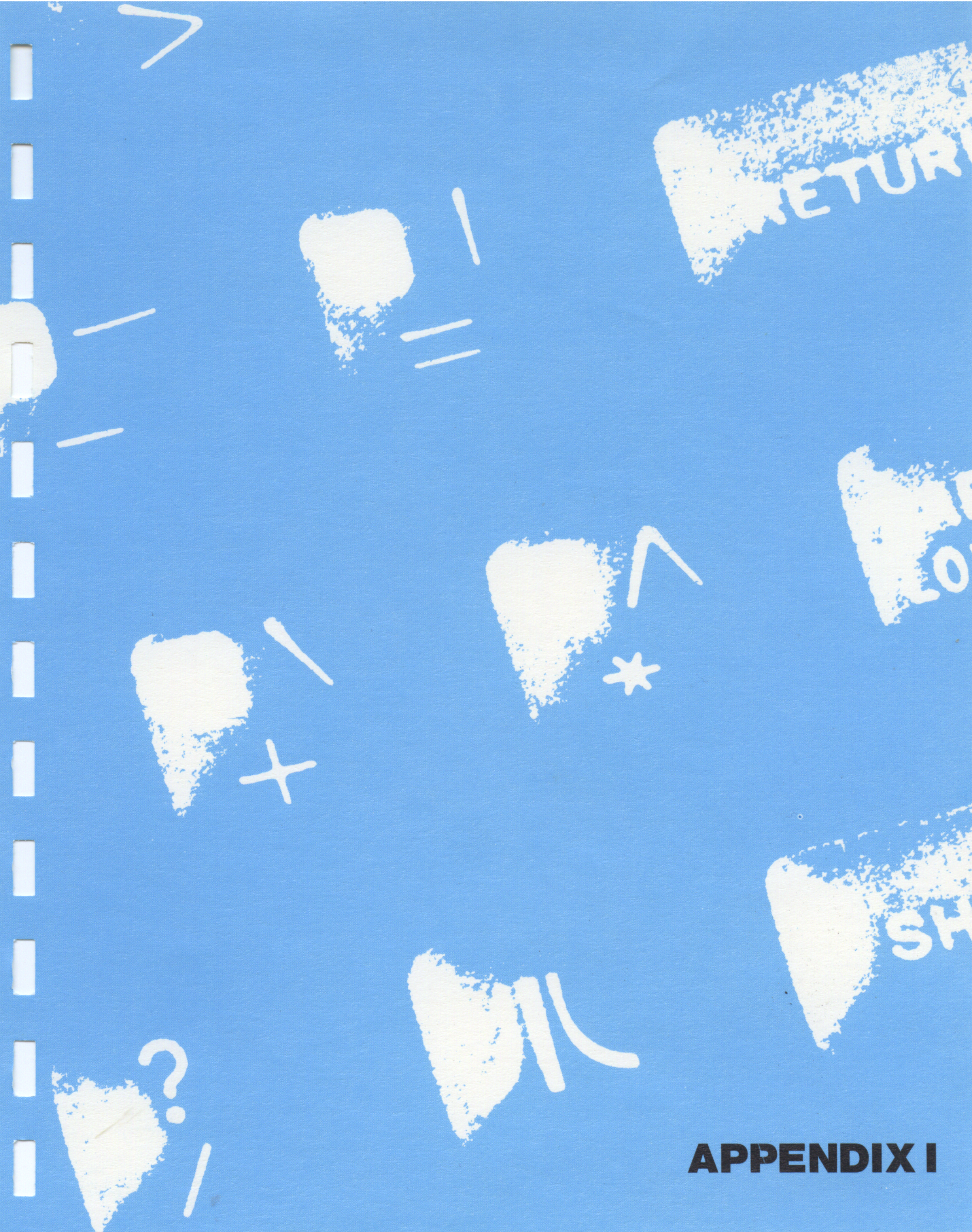
COMMANDS IN LESSON BDT

SHIFT – ATARI	Puts BASIC program under control of BDT
STEP S	Single step through a BASIC program
TRACE T	Execute all statements in order. Use numeric key 0 (Pause) through 9 (Fast) to vary execution speed. Press any key to end Trace.
EXIT E	Not a command in STANDALONE BDT
ESCAPE ESC	Return to BASIC from BDT. ESC is also used to stop a R RUN or C CONTINUE command in BDT (see below)
OPTION	In BDT, switch between the BDT screen and the BASIC screen.

NEW COMMANDS IN STANDALONE BDT

RUN R	Starts a program at the first line in the program, displaying it on the BASIC screen. To stop, press ESC, which returns to BDT.
CONTINUE C	Restarts a program from the line it was stopped at and displays the program on the BASIC screen. To stop, press ESC which returns to BDT.
CONTROL–ATARI CTRL – ATARI	Puts BASIC program under control of BDT, but does not start program at beginning like SHIFT – ATARI.

GO TO G	Puts the indicated line number in the about to be executed line of the Statement Window. Program will proceed from this line number.
BREAKPOINT B —LINE #	Right arrow key selects Line # Breakpoint. Press SELECT, enter line #, and then number of times you want to pass the line # before it activates.
—VARIABLE	Use up and down arrow keys ↑↓ to point to a variable. Press SELECT. Use up & down keys ↑↓ to select substring. Press SELECT. Type in string. Press RETURN.
ARRANGING VARIABLES/STRINGS	Use up and down arrows ↑↓ to position variable screen arrow. Press right arrow → to get new line. (Left arrow ← will delete a variable from the screen.) Press SELECT. Use up and down arrow keys ↑↓ to find the variable or string you want. Use right and left arrows → ← to change the first character of a string variable which will be shown on the screen. (For example, A$(1) will show characters 1-20, A$(10) will show 10-30; etc). Press SELECT to accept.



APPENDIX I

APPENDIX I

THE SAMPLE BASIC PROGRAMS

Each of the sample BASIC programs used in the Tutorial is listed below with a CHAPTER, SCREEN reference and the NEW CONCEPT it introduces. The program appears after the page listed.

CHAPTER B

NEW CONCEPT: BDT
FROM SCREEN: B.8

```
10 REM FROM BDT, PRESS 'E' TO EXIT
```

NEW CONCEPT: LET
FROM SCREEN: B.14

```
10 REM STEP THIS PROGRAM 6 TIMES
20 LET QUESTIONS=10
30 LET CORRECT=7
40 LET WRONG=QUESTIONS-CORRECT
50 LET GRADE=CORRECT/QUESTIONS*100
60 REM PRESS 'E' TO EXIT
```

CHAPTER C

NEW CONCEPT: GOTO
FROM SCREEN: C.4 and C.5

```
10 COUNT=0
20 COUNT=COUNT+1
30 GOTO 20
```

NEW CONCEPT: GOTO
FROM SCREEN: C.6

```
10 A=6
20 GOTO 40
30 A=12
40 END
```

CHAPTER D

NEW CONCEPT: PRINT/LOOP
FROM SCREEN: D.5

```
10 PRINT "THE FOLLOWING IS A LIST OF NUMBERS"
20 A=0
30 A=A+1
40 PRINT A
50 REM OPTION TO SEE BASIC SCREEN
60 GOTO 30
```

NEW CONCEPT: INPUT
FROM SCREEN: D.8

```
10 INPUT GRADE
20 INPUT AGE
30 INPUT SCORE
40 INPUT GRADE
50 END
```

NEW CONCEPT: PRINT/INPUT FROM SCREEN: D.9

```
10 PRINT "ENTER NUMBER OF MILES DRIVEN"
20 INPUT MILES
30 PRINT "ENTER GALLONS OF GASOLINE USED"
40 INPUT GALLONS
50 MILEAGE=MILES/GALLONS
60 PRINT "MILEAGE=";MILEAGE
70 END
```

NEW CONCEPT: PRINT/INPUT FROM SCREEN: D.10

```
10 PRINT "INPUT DEGREES FAHRENHEIT:";
20 INPUT FAHRENHEIT
30 LET CELSIUS=(FAHRENHEIT-32)*5/9
40 PRINT
50 PRINT FAHRENHEIT;" DEGREES FAHRENHEIT
   IS ";CELSIUS;" CELSIUS"
60 GOTO 10
```

NEW CONCEPT: PRINT/INPUT/LOOP
FROM SCREEN: D.11

```
10 PRINT "ENTER STARTING BALANCE IN DOLLARS:";
20 INPUT BALANCE
30 PRINT "ENTER INTEREST RATE(.07 FOR 7%):";
40 INPUT INTEREST
50 YEAR=0
100 REM LOOP
110 YEAR=YEAR+1
120 BALANCE=BALANCE*(1+INTEREST)
130 PRINT "BALANCE AFTER YEAR ";YEAR;" IS ";BALANCE
140 GOTO 100
```

CHAPTER E

NEW CONCEPT: IF...THEN
FROM SCREEN: E.9

```
10 A=1
20 B=2
30 C=3
40 IF A<B THEN PRINT "A IS LESS THAN B"
45 REM OPTION TO SEE BASIC SCREEN
50 IF B>C THEN PRINT "THIS SHOULDN'T HAPPEN"
60 IF A+B=C THEN PRINT "EXPRESSIONS CAN
   BE IN CONDITIONS"
70 IF A=1 THEN PRINT "A IS EQUAL TO 1"
80 IF A THEN PRINT "ONE IS CONSIDERED 'TRUE'"
```

NEW CONCEPT: IF...THEN
FROM SCREEN: E.10

```
10 PRINT "ENTER COST OF PURCHASE:";
20 INPUT COST
30 PRINT "AMOUNT PAID:";
40 INPUT PAID
50 IF PAID<COST THEN PRINT "THAT'S NOT
   ENOUGH MONEY":GOTO 10
60 CHANGE=PAID-COST
70 IF CHANGE=0 THEN PRINT "THANK YOU":GOTO 10
80 PRINT "YOUR CHANGE IS $ ";CHANGE
90 PRINT "THANK YOU"
100 END
```

NEW CONCEPT: IF...THEN
FROM SCREEN: E.11

```
10 PRINT "ENTER TEST SCORES (0-100)"
20 PRINT "STOP BY ENTERING -1"
30 COUNTER=0
40 TOTAL=0
50 PRINT "ENTER GRADE:";
60 INPUT GRADE
70 IF GRADE=-1 THEN 110
80 COUNTER=COUNTER+1
90 TOTAL=TOTAL+GRADE
100 GOTO 50
110 AVERAGE=TOTAL/COUNTER
120 PRINT "YOUR AVERAGE IS: ";AVERAGE
130 END
```

NEW CONCEPT: IF...THEN/GOTO
FROM SCREEN: E.12

```
10 PRINT "ENTER  1  FOR FAHRENHEIT TO CELSIUS"
20 PRINT "   OR  2  FOR CELSIUS TO FAHRENHEIT"
30 PRINT "   OR  0  TO STOP."
40 INPUT CHOICE
50 PRINT
60 IF CHOICE=1 THEN 110
70 IF CHOICE=2 THEN 200
80 IF CHOICE=0 THEN STOP
90 PRINT "INVALID CHOICE"
100 GOTO 10
110 PRINT "INPUT DEGREES FAHRENHEIT:";
120 INPUT FAHRENHEIT
130 CELSIUS=(FAHRENHEIT-32)*5/9
140 GOTO 300
200 PRINT "ENTER DEGREES CELSIUS:";
210 INPUT CELSIUS
220 FAHRENHEIT=CELSIUS*9/5+32
300 PRINT
310 PRINT FAHRENHEIT;" FAHRENHEIT = ";CELSIUS;
    " CELSIUS."
320 PRINT
330 PRINT
340 GOTO 10
```

CHAPTER F

NEW CONCEPT: FOR...NEXT
FROM SCREEN: F.2

```
10 FOR NUMBER=1 TO 10
20 PRINT NUMBER
30 NEXT NUMBER
40 END
```

NEW CONCEPT: NESTED FOR...NEXT
FROM SCREEN: F.7

```
10 FOR NUMBER1=1 TO 4
20 FOR NUMBER2=1 TO 3
30 COUNT=COUNT+1
40 NEXT NUMBER2
50 NEXT NUMBER1
```

NEW CONCEPT: ON...GOTO
FROM SCREEN: F.11

```
10 PRINT "ENTER A NUMBER FROM 1-4:";
20 INPUT I
30 ON I GOTO 100,200,300,400
40 PRINT "OUT OF RANGE"
50 GOTO 10
100 PRINT "WENT TO 100"
110 GOTO 10
200 PRINT "WENT TO 200"
210 GOTO 10
300 PRINT "WENT TO 300"
310 GOTO 10
400 PRINT "WENT TO 400"
410 GOTO 10
```

NEW CONCEPT: GOSUB...RETURN
FROM SCREEN: F.14

```
10 A=10
20 B=30
30 GOSUB 1000
110 A=21
120 B=67
130 GOSUB 1000
210 A=6.023
220 B=3.14159
230 GOSUB 1000
240 END
1000 REM SUBROUTINE
1010 C=(A+B)/2
1020 PRINT "AVERAGE=";C
1030 RETURN
```

CHAPTER G

NEW CONCEPT: READ...DATA
FROM SCREEN: G.2

```
10 DATA 1,5,7,13,17
20 FOR I=1 TO 5
30 READ NUMBER
40 PRINT NUMBER
50 NEXT I
```

NEW CONCEPT: DIMENSION AN ARRAY
FROM SCREEN: G.6

```
10 DIM CARPRICE(7)
20 DATA 4995,6215,6999,7885,9500,10750,11250,
   11995
30 FOR I=0 TO 7
40 READ TEMP
50 CARPRICE(I)=TEMP
60 NEXT I
70 PRINT "WHAT CAR PRICE DO YOU WANT? (0-7)"
80 INPUT NUMBER
90 IF NUMBER<0 OR NUMBER>7 THEN PRINT "ILLEGAL
   NUMBER":GOTO 70
100 PRINT "CAR ";NUMBER;" COSTS ";
   CARPRICE (NUMBER)
110 GOTO 70
```

NEW CONCEPT: BUBBLE SORT
FROM SCREEN: G.8, G.11, G.12

```
10 PRINT "ENTER 8 NUMBERS"
20 DIM NUMBS(7)
30 FOR I=0 TO 7
40 PRINT "ENTER NUMBER #";I;
50 INPUT NUMB
60 NUMBS(I)=NUMB
70 NEXT I
100 REM SORT STARTS HERE
110 FOR I=0 TO 7
120 FOR J=0 TO 6
130 IF NUMBS(J)>NUMBS(J+1) THEN HOLD=NUMBS(J)
    :NUMBS(J)=NUMBS(J+1):NUMBS(J+1)=HOLD
140 NEXT J
150 NEXT I
160 END
```

NEW CONCEPT: MODIFIED BUBBLE SORT FROM SCREEN: G. 13

```
10 PRINT "ENTER 8 NUMBERS"
20 DIM NUMBS(7)
30 FOR I=0 TO 7
40 PRINT "ENTER NUMBER #";I;
50 INPUT NUMB
60 NUMBS(I)=NUMB
70 NEXT I
100 REM SORT STARTS HERE
110 FOR I=0 TO 7
120 SWAPS=0
130 FOR J=0 TO 6
140 IF NUMBS(J)>NUMBS(J+1) THEN HOLD=NUMBS(J)
    :NUMBS(J)=NUMBS(J+1):NUMBS(J+1)=HOLD
    :SWAPS=1
150 NEXT J
160 IF SWAPS=0 THEN 180
170 NEXT I
180 END
```

NEW CONCEPT: BUBBLE SORT— "SHORTENED PATH"
FROM SCREEN: G.15

```
10 PRINT "ENTER 8 NUMBERS"
20 DIM NUMBS(7)
30 FOR I=0 TO 7
40 PRINT "ENTER NUMBER #";I;
50 INPUT NUMB
60 NUMBS(I)=NUMB
70 NEXT I
100 REM SORT STARTS HERE
110 TOP=0:BOT=6
120 SWAPS=0
130 FOR I=TOP TO BOT
140 GOSUB 1000
150 NEXT I
160 IF SWAPS=0 THEN 240
170 SWAPS=0
180 FOR I=BOT-1 TO TOP STEP -1
190 GOSUB 1000
200 NEXT I
210 IF SWAPS=0 THEN 240
220 TOP=TOP+1:BOT=BOT-1
230 GOTO 130
240 END
1000 REM COMPARISON/SWAP SUBROUTINE
1010 IF NUMBS(I)>NUMBS(I+1) THEN HOLD=NUMBS(I)
     :NUMBS(I)=NUMBS(I+1):NUMBS(I+1)=HOLD
     :SWAPS=1
1020 RETURN
```

CHAPTER H

NEW CONCEPT: SQR
FROM SCREEN: H.3

```
10 PRINT "ENTER NUMBERS"
20 PRINT "TO FIND THEIR SQUARE ROOTS"
30 PRINT "ENTER NEGATIVE TO STOP"
40 INPUT NUMB
50 IF NUMB<0 THEN STOP
60 S=SQR(NUMB)
70 PRINT "SQUARE ROOT OF ";NUMB;" IS ";S
80 GOTO 40
```

NEW CONCEPT: RND
FROM SCREEN: H.5

```
10 X=RND(0)
20 DIE=1+INT(X*6)
30 GOTO 10
```

NEW CONCEPT: INT/SQR/RND
FROM SCREEN: H. 7

```
10 NUMERATOR=0
20 DENOMINATOR=0
30 X=(RND(0)*2)-1
40 Y=(RND(0)*2)-1
50 DENOMINATOR=DENOMINATOR+1
60 IF SQR(X*X+Y*Y)<1 THEN NUMERATOR=NUMERATOR+1
70 PI=4*NUMERATOR/DENOMINATOR
80 GOTO 30
```

NEW CONCEPT: PEEK
FROM SCREEN: H.12

```
10 PRINT "ENTER CURRENT TIME AS HOURS"
20 PRINT "MINUTES AND SECONDS"
30 INPUT HOURS,MINUTES,SECONDS
40 REM GOSUB TO GET ATARI'S TIME
50 GOSUB 220
60 OTIME=TIME
100 GOSUB 220
110 IF TIME=OTIME THEN 100
120 SECONDS=SECONDS+TIME-OTIME
130 IF SECONDS<60 THEN 200
140 SECONDS=SECONDS-60
150 MINUTES=MINUTES+1
160 IF MINUTES<60 THEN 200
170 MINUTES=MINUTES-60
180 HOURS=HOURS+1
190 IF HOURS>23 THEN HOURS=0
200 OTIME=TIME
210 GOTO 100
220 TIME=INT(((PEEK(18)*65536)+(PEEK(19)
    *256)+PEEK(20))/60)
230 RETURN
```

CHAPTER I

NEW CONCEPT: STRINGS/SUBSTRING FROM SCREEN: I.7

```
10 DIM STRING$(20),TEMP$(10)
20 STRING$="THE QUICK BROWN FOX"
30 TEMP$=STRING$(5,9)
40 TEMP$=STRING$(11)
50 TEMP$(7)="CAT"
60 I=1
70 J=5
80 TEMP$(I,J)="GREEN"
90 END
```

NEW CONCEPT: LEN FROM SCREEN: I.8

```
10 DIM STRING$(10)
20 PRINT "ENTER 0-10 CHARACTERS"
30 INPUT STRING$
40 LENGTH=LEN(STRING$)
50 GOTO 20
```

NEW CONCEPT: CONCATENATION FROM SCREEN: I.9

```
10 DIM STRING1$(15),STRING2$(15)
20 STRING1$="Hello there"
30 STRING2$=""
40 FOR I=LEN(STRING1$) TO 1 STEP -1
50 STRING2$(LEN(STRING2$)+1)=STRING1$(I,I)
60 NEXT I
70 END
```

NEW CONCEPT: STRINGS/RND/INT/DIM, READ...DATA

FROM SCREEN: I.10

```
10 DIM WORDS$(100),WORD$(10),TEST$(10),G $(1)
20 FOR I=1 TO 100:WORDS$(I,I)=" ":NEXT I
30 FOR I=1 TO 10
40 READ WORD$
50 DATA ANIMAL,JOYSTICK,PENCIL,SEASHORE,
   JUPITER,FOOTBALL,SHOELACES,TRUMPET,SCHOOL,
   EAGLE
60 WORDS$((I-1)*10+1,I*10)=WORD$
70 NEXT I
80 I=INT(RND(0)*10)+1
100 WORD$=WORDS$((I-1)*10+1,I*10)
110 FOR I=1 TO 10
115 TEST$(I,I)=" "
120 IF WORD$(I,I)<>" " THEN TEST$(I,I)=" _"
130 NEXT I
135 MISSES=0
140 PRINT :PRINT TEST$:PRINT "ENTER YOUR
    1-LETTER GUESS ";
150 INPUT G$
160 IF G$=" " THEN GOTO 140
165 SW=0
170 FOR I=1 TO 10
180 IF WORD$(I,I)=G$ THEN SW=1:TEST$(I,I )=G$
```

```
190 NEXT I
200 IF SW=1 THEN GO TO 270
205 MISSES=MISSES+1
210 PRINT "WRONG.  MISS #";MISSES
230 IF MISSES<10 THEN GOTO 140
240 PRINT "YOU ARE HANGED."
250 PRINT "THE WORD WAS ";WORD$
260 GOTO 310
270 PRINT "GOOD GUESS."
290 IF WORD$=TEST$ THEN PRINT "THAT'S IT ,
    ";TEST$:GOTO 310
300 GOTO 140
310 PRINT :PRINT :PRINT "DO YOU WANT TO
    PLAY AGAIN (Y,N)";
320 INPUT G$
330 IF G$="N" THEN STOP
340 GOTO 80
```

NEW CONCEPT: STR$/VAL/ASC/CHR$
FROM SCREEN: I.12

```
10 REM STR$ AND VAL DEMONSTRATION
20 DIM STRING$(10)
30 A=10
40 STRING$=STR$(A+7)
50 B=VAL(STRING$)
100 REM ASC AND CHR$ DEMONSTRATION
110 DIM CHARACTER$(1)
120 FOR I=1 TO 12
130 DATA A,B,C,a,b,c,1,2,3,#,$,%
140 READ CHARACTER$
150 CODE=ASC(CHARACTER$)
160 NEXT I
170 FOR CODE=33 TO 122
180 CHARACTER$=CHR$(CODE)
190 PRINT "ASC OF ";CHARACTER$;" = ";CODE
200 NEXT CODE
```

NEW CONCEPT: STRING SORT FROM SCREEN: I.13

```
10 DIM WORDS$(100),WORD$(10),BLANK$(10)
20 BLANK$="          "
30 FOR I=0 TO 9
40 WORDS$(I*10+1,I*10+10)=BLANK$
50 NEXT I
90 PRINT "ENTER 9 WORDS TO BE SORTED:"
100 FOR I=1 TO 9
110 PRINT "ENTER WORD #";I;" -->";
120 INPUT WORD$
130 GOSUB 200
140 NEXT I
150 END
200 REM INSERTION SORT ROUTINE
210 REM PRESENTLY I-1 STRINGS
220 REM IN WORDS$
230 FOR J=I-1 TO 0 STEP -1
240 IF WORD$<WORDS$(10*J+1,10*J+10) THEN 280
250 WORDS$(10*J+11,10*J+20)=BLANK$
260 WORDS$(10*J+11,10*J+20)=WORD$
270 RETURN
280 WORDS$(10*J+11,10*J+20)=BLANK$
290 WORDS$(10*J+11,10*J+20)=WORDS$(10*J+1,10*J+10)
300 NEXT J
310 RETURN
```

CHAPTER J

NEW CONCEPT: SAVING DATA TO DISK/OPEN/CLOSE
FROM SCREEN: J.9

```
10 DIM WAIT$(1)
20 PRINT "PLACE AN INITIALIZED DISK IN DRIVE
   ONE"
30 PRINT "PRESS RETURN WHEN YOU HAVE";
40 INPUT WAIT$
50 OPEN #1,8,0,"D:SAMPLE.DAT"
60 PRINT "ENTER YOUR SECRET NUMBER:";
70 INPUT NUMBER
80 PRINT #1,NUMBER
90 CLOSE #1
100 END
```

NEW CONCEPT: RETRIEVING DATA FROM DISK/OPEN/CLOSE
FROM SCREEN: J.12

```
10 DIM WAIT$(1)
20 PRINT "PLACE THE DISK THAT YOU SAVED"
30 PRINT "SAMPLE.DAT ON IN DRIVE ONE."
40 PRINT "PRESS RETURN WHEN READY";
50 INPUT WAIT$
60 OPEN #1,4,0,"D:SAMPLE.DAT"
70 INPUT #1,NUMBER
80 CLOSE #1
90 PRINT "ENTER SECRET NUMBER:";
100 INPUT TEST
110 IF TEST<>NUMBER THEN 90
120 PRINT "CORRECT"
130 END
```

NEW CONCEPT: GET
FROM SCREEN: J.15

```
10 OPEN #1,4,0,"K:"
20 PRINT "SHOULD I STOP?"
30 GET #1,A
40 IF A=ASC("Y") THEN GOTO 60
50 GOTO 20
60 CLOSE #1
70 END
```

CHAPTER K

NEW CONCEPT: PADDLES/TRIG/PTRIG
FROM SCREEN: K.4

```
10 PRINT "Experiment with your paddles"
20 PAD0=PADDLE(0)
30 PAD1=PADDLE(1)
40 TRIG0=PTRIG(0)
50 TRIG1=PTRIG(1)
60 GOTO 20
```

NEW CONCEPT: JOYSTICK/TRIG/STRIG
FROM SCREEN: K.6

```
10 PRINT "Experiment with your joystick"
20 JOY=STICK(1)
30 TRIG=STRIG(1)
40 GOTO 20
```

NEW CONCEPT: SOUND (KEYBOARD) FROM SCREEN: K.8

```
10 PRINT "PressĘ↑to increase pitch"
20 PRINT "PressĘ↓to decrease pitch"
30 PRINT "PressĘ← to increase volume"
40 PRINT "PressĘ→ to decrease volume"
50 PRINT "Voice starts at 0 and changes"
60 PRINT "when you press space bar"
70 IN=4:OUT=8:APPEND=9:INOUT=12
80 OPEN #1,IN,0,"K:"
90 VOICE=0:PITCH=128:VOLUME=8
100 GET #1,KEY
110 IF KEY=ASC("↑") OR KEY=ASC("-") THEN
    PITCH=PITCH-1:IF PITCH<0 THEN PITCH=0
120 IF KEY=ASC("↓") OR KEY=ASC("=") THEN
    PITCH=PITCH+1:IF PITCH>255 THEN PITCH=255
130 IF KEY=ASC("→") OR KEY=ASC("*") THEN
    VOLUME=VOLUME+1:IF VOLUME>15 THEN VOLUME=15
140 IF KEY=ASC("←") OR KEY=ASC("+") THEN
    VOLUME=VOLUME-1:IF VOLUME<1 THEN VOLUME=1
150 IF KEY=ASC(" ") THEN VOICE=VOICE+1:IF
    VOICE>3 THEN VOICE=0
160 SOUND VOICE,PITCH,10,VOLUME
170 GOTO 100
```

NEW CONCEPT: SOUND (PADDLE) FROM SCREEN: K.8

```
10 PRINT "Use paddle Ø to control volume"
20 PRINT "and paddle 1 to control pitch"
30 PRINT "Distortion=1Ø"
40 PRINT "Voice starts at Ø and changes"
50 PRINT "with each trigger press on"
60 PRINT "either paddle Ø or 1"
90 VOICE=Ø
100 PADØ=PADDLE(Ø)
110 PAD1=PADDLE(1)
120 TRIGØ=PTRIG(Ø)
130 TRIG1=PTRIG(1)
140 IF TRIGØ=Ø OR TRIG1=Ø THEN VOICE=VOICE+1:
    IF VOICE>3 THEN VOICE=Ø
150 SOUND VOICE,(PAD1-1)*255/227,1Ø,
    ((PADØ-1)*14/227)+1
160 GOTO 1ØØ
```

NEW CONCEPT: SOUND (JOYSTICK) FROM SCREEN: K.8

```
10 PRINT "Push stick forward to increase pitch"
20 PRINT "backward to decrease pitch."
30 PRINT "Push stick to right to increase
   volume"
40 PRINT "left to decrease volume."
50 PRINT "Voice starts at 0 and changes"
60 PRINT "with each trigger press."
70 VOICE=0
80 PITCH=128
90 VOLUME=8
100 STK=STICK(1)
110 TRIG=STRIG(1)
120 IF TRIG=0 THEN VOICE=VOICE+1:IF VOICE>3
    THEN VOICE=0
130 IF STK=10 OR STK=14 OR STK=6 THEN PITCH
    =PITCH-1:IF PITCH<0 THEN PITCH=0
140 IF STK=9 OR STK=13 OR STK=5 THEN PITCH
    =PITCH+1:IF PITCH>255 THEN PITCH=255
150 IF STK=10 OR STK=11 OR STK=9 THEN VOLUME
    =VOLUME-1:IF VOLUME<1 THEN VOLUME=1
160 IF STK=6 OR STK=7 OR STK=5 THEN VOLUME
    =VOLUME+1:IF VOLUME>15 THEN VOLUME=15
170 SOUND VOICE,PITCH,10,VOLUME
180 GOTO 100
```

NEW CONCEPT: SOUND
FROM SCREEN: K.10

```
10 PRINT
12 PRINT "Enter speed to play at, 1-100 -->";
14 INPUT SPEED
16 IF SPEED<1 OR SPEED>100 THEN PRINT "Enter
   1 to 100 only.":GOTO 10
20 PRINT "Which tune do you want to hear ?"
30 PRINT "1  Mary had a Little Lamb"
40 PRINT "2  Twinkle Twinkle Little Star"
50 PRINT "3  Happy Birthday"
60 PRINT "Enter your choice -->";
70 INPUT CHOICE
80 ON CHOICE GOTO 1000,2000,3000
90 PRINT "Enter 1,2 or 3 only."
100 GOTO 10
500 REM PLAY A SONG
510 READ P
515 IF P=0 THEN RETURN
518 READ D
520 SOUND 0,P,10,10
530 FOR DELAY=1 TO D
540 FOR TEMPO=1 TO SPEED
545 NEXT TEMPO
550 NEXT DELAY
555 SOUND 0,0,0,0
560 GOTO 510
1000 REM MARY HAD A LITTLE LAMB
1010 RESTORE 1020
```

continued...

```
1020 DATA 64,4,72,4,81,4,72,4,64,4,64,4,
   64,8,72,4,72,4,72,8,64,4,53,4,53,8
1030 DATA 64,4,72,4,81,4,72,4,64,4,64,4,
   64,4,64,4,72,4,72,4,64,4,72,4,81,8,0
1040 GOSUB 500
1050 GOTO 10
2000 REM TWINKLE TWINKLE LITTLE STAR
2010 RESTORE 2020
2020 DATA 121,4,121,4,81,4,81,4,72,4,72,
   4,81,8,91,4,91,4,96,4,96,4,108,4,108,4,
   121,8,0
2030 GOSUB 500
2040 GOTO 10
3000 REM HAPPY BIRTHDAY
3010 RESTORE 3020
3020 DATA 81,3,81,1,72,4,81,4,60,4,64,8,
   81,2,81,1,72,4,81,4,53,4,60,8
3030 DATA 81,3,81,4,40,4,47,4,60,4,64,4,
   72,8,45,3,45,1,47,4,60,4,53,4,60,8,0
3040 GOSUB 500
3050 GOTO 10
```

CHAPTER L

NEW CONCEPT: GRAPHIC TEXT MODES FROM SCREEN: L.5

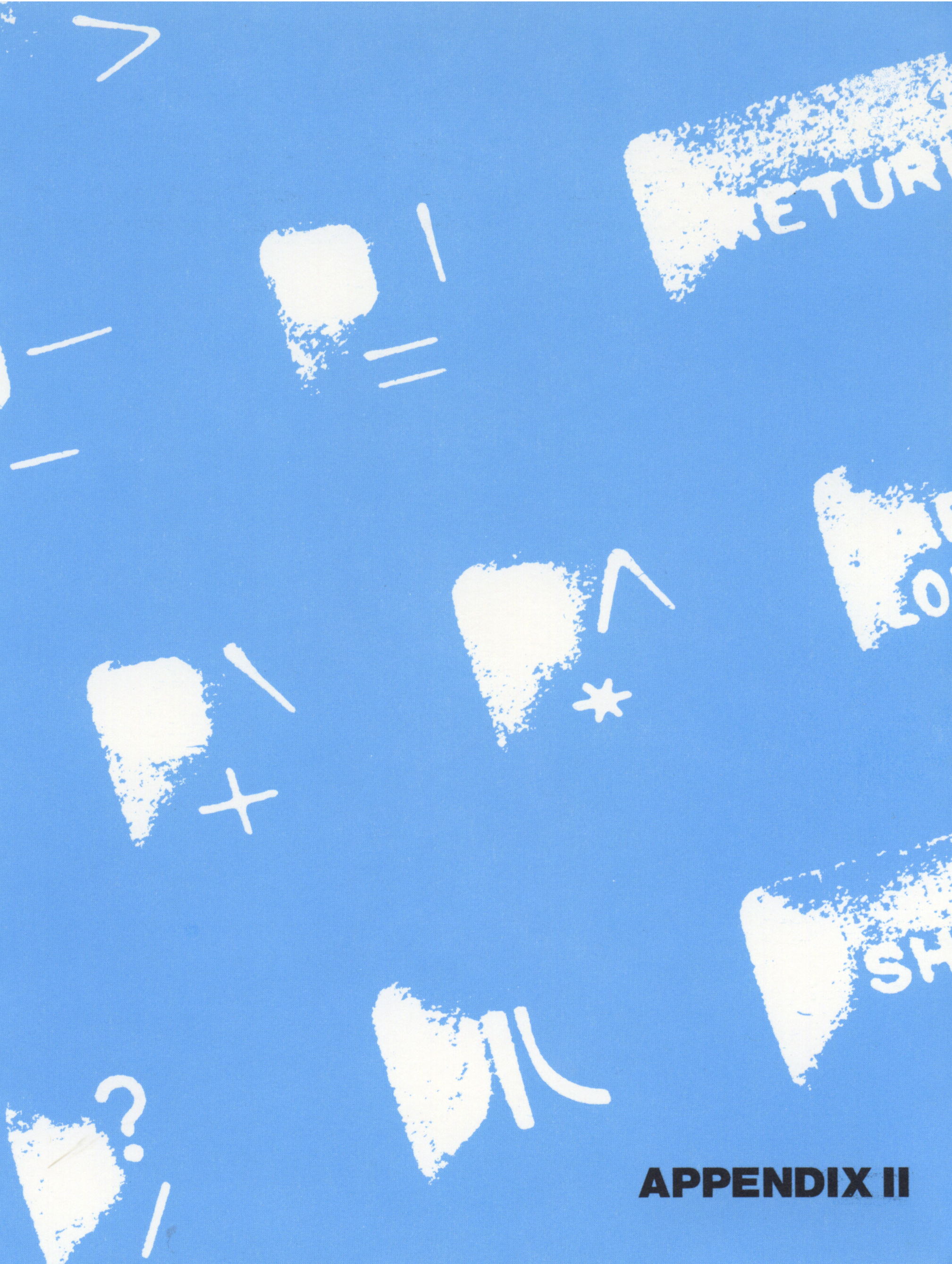
```
10 GRAPHICS 1
20 PRINT #6;"THE MODE 1 PART"
30 PRINT "THE NORMAL TEXT PART"
40 GRAPHICS 2
50 PRINT #6;"THE MODE 2 PART"
60 PRINT "THE NORMAL TEXT PART"
65 POSITION 7,2
70 PRINT #6;"NORMAL"
75 POSITION 7,4
80 PRINT #6;"INVERSE"
85 POSITION 5,6
90 PRINT #6;"lower case"
95 POSITION 1,8
100 PRINT #6;"inverse lower case"
110 GRAPHICS 2+16
120 PRINT #6;"THIS IS FULL SCREEN TEXT MODE 2"
130 GRAPHICS 0
140 END
```

NEW CONCEPT: COLOR/PLOT/DRAWTO/SETCOLOR
FROM SCREEN: L.9

```
10 GRAPHICS 5
20 PRINT "COLOR 1":COLOR 1
30 PRINT "PLOT 0,0":PLOT 0,0
40 PRINT "COLOR 2":COLOR 2
50 PRINT "PLOT 79,39":PLOT 79,39
60 PRINT "COLOR 3":COLOR 3
70 PRINT "PLOT 79,0":PLOT 79,0
80 PRINT "DRAWTO 0,39":DRAWTO 0,39
90 PRINT "COLOR 1":COLOR 1
100 PRINT "PLOT 1,39":PLOT 1,39
110 PRINT "DRAWTO 79,1":DRAWTO 79,1
120 PRINT "COLOR 2":COLOR 2
130 PRINT "PLOT 79,2":PLOT 79,2
140 PRINT "DRAWTO 2,39":DRAWTO 2,39
150 PRINT "COLOR 3":COLOR 3
160 PRINT "PLOT 3,39":PLOT 3,39
170 PRINT "DRAWTO 79,3":DRAWTO 79,3
180 PRINT "COLOR 0":COLOR 0
190 PRINT "PLOT 79,39":PLOT 79,39
200 PRINT "DRAWTO 0,0":DRAWTO 0,0
210 PRINT "COLOR 1":COLOR 1
220 PRINT "DRAWTO 0,15":DRAWTO 0,15
230 PRINT "DRAWTO 15,15":DRAWTO 15,15
240 PRINT "DRAWTO 15,0":DRAWTO 15,0
250 PRINT "DRAWTO 0,0":DRAWTO 0,0
260 PRINT "SETCOLOR 0,9,8":SETCOLOR 0,9,8
270 PRINT "SETCOLOR 0,9,0":SETCOLOR 0,9,0
280 PRINT "SETCOLOR 0,9,14":SETCOLOR 0,9,14
290 END
```

NEW CONCEPT: PLOT
FROM SCREEN: L.10

```
10 PRINT "Use keys to draw"
20 PRINT "I-up      O-up&right"
30 PRINT "K-right   ,-down&right"
40 PRINT "M-down    N-down&left"
50 PRINT "J-left    U-up&left"
60 PRINT "space bar color change"
70 PRINT
80 PRINT "Press any key to begin"
100 IN=4:OUT=8:APPEND=9:INOUT=12
110 OPEN #1,IN,0,"K:"
120 GET #1,P
130 GRAPHICS 5
140 X=40:Y=20:COL=1
150 GET #1,P
160 IF P=32 THEN COL=COL+1:IF COL>3 THEN COL=0
170 IF P=85 OR P=73 OR P=79 THEN Y=Y-1:
  IF Y<0 THEN Y=0
180 IF P=79 OR P=75 OR P=44 THEN X=X+1:
  IF X>79 THEN X=79
190 IF P=78 OR P=77 OR P=44 THEN Y=Y+1:
  IF Y>39 THEN Y=39
200 IF P=85 OR P=74 OR P=78 THEN X=X-1:
  IF X<0 THEN X=0
210 COLOR COL
220 PLOT X,Y
230 GOTO 150
```



APPENDIX II

APPENDIX II

SOLUTIONS TO SUGGESTED EXERCISES

Each of the SUGGESTED EXERCISES has a possible solution presented below. ***Remember:*** there are many ways to write a program. If yours is not like ours, they may both be right. Do you get the requested response?

CHAPTER D
INPUT/OUTPUT SUGGESTED EXERCISES FROM SCREEN D.12 and D.13

PROBLEM:

Input 2 numbers. Put them in variables A and B. Calculate the average and store it in variable C. Print out the average.

POSSIBLE SOLUTION:

```
10 PRINT "ENTER A ";
20 INPUT A
30 PRINT "ENTER B ";
40 INPUT B
50 C=(A+B)/2
60 PRINT "AVERAGE = ";C
70 END
```

PROBLEM:

Input your age. Store it in a variable called YEARS. Calculate your age in months (YEARS*12), and store it in a variable called MONTH. Print out your age in months.

POSSIBLE SOLUTION:

```
10 PRINT "ENTER YOUR AGE IN YEARS ";
20 INPUT YEARS
30 MONTH=YEARS*12
40 PRINT "YOU ARE ";MONTH;" MONTH OLD."
50 END
```

CHAPTER E
DECISIONS
SUGGESTED EXERCISES FROM SCREEN E.13

PROBLEM:
Input a number and print it out only if it is greater than zero.

POSSIBLE SOLUTION:
```
10 PRINT "ENTER A NUMBER ";
20 INPUT NUMBER
30 IF NUMBER>0 THEN PRINT "THE NUMBER IS ";NUMBER
40 END
```

PROBLEM:
Input two numbers. Print out the smaller of the two.

POSSIBLE SOLUTION:
```
10 PRINT "ENTER FIRST NUMBER ";
20 INPUT NUMBER1
30 PRINT "ENTER SECOND NUMBER ";
40 INPUT NUMBER2
50 PRINT "THE SMALLER NUMBER IS ";
60 IF NUMBER1<NUMBER2 THEN PRINT NUMBER1:GOTO 80
70 PRINT NUMBER2
80 END
```

PROBLEM:
Print out the numbers 1-10 using GOTO and a loop.

POSSIBLE SOLUTION:
```
10 COUNT=0
20 COUNT=COUNT+1
30 PRINT COUNT
40 IF COUNT<10 THEN GOTO 20
50 END
```

CHAPTER F
MORE BRANCHING SUGGESTED EXERCISES FROM SCREEN F.17

PROBLEM:

Using a FOR/NEXT loop and step, print 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 BLAST OFF!

POSSIBLE SOLUTION:

```
10 FOR COUNT=10 TO 0 STEP -1
20 PRINT COUNT
30 NEXT COUNT
40 PRINT "BLAST OFF!"
50 END
```

PROBLEM:

Write a Subroutine that will take NUMBER and set SQUARE equal to NUMBER*NUMBER. In a FOR/NEXT loop, GOSUB this subroutine and print NUMBER and SQUARE for Number=1 to 100.

POSSIBLE SOLUTION:

```
10 FOR NUMBER=1 TO 100
20 GOSUB 100
30 PRINT "NUMBER = ";NUMBER;", SQUARE = ";SQUARE
40 NEXT NUMBER
50 END
100 REM SQUARE SUBROUTINE
110 SQUARE=NUMBER*NUMBER
120 RETURN
```

CHAPTER G
ARRAYS
SUGGESTED EXERCISES FROM SCREEN G.16

PROBLEM:

In a FOR/NEXT loop, input ten numbers into an array. After the tenth, print them out in another FOR/NEXT loop.

POSSIBLE SOLUTION:

```
10 DIM A(9)
20 PRINT "ENTER 10 NUMBERS"
30 FOR I=0 TO 9
40 INPUT NUM
50 A(I)=NUM
60 NEXT I
70 PRINT "HERE ARE THE NUMBERS YOU ENTERED:"
80 FOR I=0 TO 9
90 PRINT A(I)
100 NEXT I
```

PROBLEM:

Initialize an array of ten numbers using READ and DATA inside a FOR/NEXT loop. Use INPUT to read a number from 0-10. If 0, stop, otherwise, print that array element and loop back.

POSSIBLE SOLUTION:

```
10 DIM A(10)
20 DATA 5,-3.2,6.023,543,7
30 DATA 7.23,-9876,0.001,76,3
40 FOR I=1 TO 10
50 READ NUM
60 A(I)=NUM
70 NEXT I
80 PRINT "WHICH ARRAY ELEMENT";
90 INPUT I
100 IF I<0 OR I>10 THEN PRINT "ENTER 0-10 ONLY"
    :GOTO 80
110 IF I=0 THEN STOP
120 PRINT "ARRAY(";I;")=";A(I)
130 GOTO 80
```

CHAPTER H
FUNCTIONS
SUGGESTED EXERCISES FROM SCREEN H.13

PROBLEM:
In a FOR/NEXT loop, print out the square (SQR), the SINE (SIN) and any other functions you care to of the numbers 1-100.

POSSIBLE SOLUTION:
```
10 FOR I=1 TO 100
20 PRINT SQR(I),SIN(I),LOG(I)
30 NEXT I
40 END
```

PROBLEM:
Roll a 20 sided die, 100 times.

POSSIBLE SOLUTION:
```
10 FOR I=1 TO 100
20 DIE=INT(RND(1)*20)+1
30 PRINT DIE
40 NEXT I
50 END
```

PROBLEM:
Input NUMBER. Roll a Number-Sided die. Try again.

POSSIBLE SOLUTION:
```
10 PRINT "HOW MANY SIDED DIE";
20 INPUT NUMBER
30 DIE=INT(RND(1)*NUMBER)+1
40 PRINT DIE
50 GOTO 10
```

CHAPTER I STRINGS SUGGESTED EXERCISES FROM SCREEN I.13

PROBLEM:

Input a 10-character long string. Print it out reversed. i.e., ABCDEFGHIJ gives JIHGFEDCBA.

POSSIBLE SOLUTION:

```
10 DIM S$(10),REVERSED$(10)
20 PRINT "ENTER A 10 CHARACTER STRING -->";
30 INPUT S$
40 IF LEN(S$)<>10 THEN PRINT
   "10 CHARACTERS PLEASE!":GOTO 20
50 FOR I=1 TO 10
60 REVERSED$(I,I)=S$(11-I,11-I)
70 NEXT I
80 PRINT S$;" REVERSED IS ";REVERSED$
90 END
```

PROBLEM:

Read in a string up to 100 characters long. Print out how many A's it has, B's, etc.

POSSIBLE SOLUTION:

```
10 DIM S$(100)
20 PRINT "ENTER A STRING UP TO 100 CHARACTER."
30 INPUT S$
40 FOR I=ASC("A") TO ASC("Z")
50 COUNT=0
60 FOR J=1 TO LEN(S$)
70 IF S$(J,J)=CHR$(I) THEN COUNT=COUNT+1
80 NEXT J
90 PRINT "THERE ARE ";COUNT;" ";CHR$(I);"'S."
100 NEXT I
110 END
```

CHAPTER J
DISK INPUT/OUTPUT SUGGESTED EXERCISES FROM SCREEN J.16

PROBLEM:

Write a program to write your name, address and telephone number to a disk file called DIRECT.DAT.

POSSIBLE SOLUTION:

```
10 DIM IN$(38)
20 PRINT "PLACE AN INITIALIZED DISK INTO"
30 PRINT "THE DISK DRIVE."
40 PRINT "PRESS RETURN WHEN READY -->";
50 INPUT IN$
60 OPEN #1,8,0,"D:DIRECT.DAT"
70 PRINT "ENTER YOUR NAME"
80 INPUT IN$
90 PRINT #1;IN$
100 PRINT "ENTER YOUR ADDRESS"
110 INPUT IN$
120 PRINT #1;IN$
130 PRINT "ENTER YOUR PHONE NUMBER"
140 INPUT IN$
150 PRINT #1;IN$
160 CLOSE #1
170 END
```

PROBLEM:

Read in DIRECT.DAT and print your name, address and telephone number.

POSSIBLE SOLUTION:

```
10 DIM IN$(38)
20 PRINT "PLACE THE DISK WITH DIRECT.DAT"
30 PRINT "IN THE DISK DRIVE."
40 PRINT "PRESS RETURN WHEN READY -->";
50 INPUT IN$
60 OPEN #1,4,0,"D:DIRECT.DAT"
70 INPUT #1,IN$
80 PRINT "YOUR NAME IS:"
90 PRINT IN$
100 INPUT #1,IN$
110 PRINT "YOUR ADDRESS IS:"
120 PRINT IN$
130 INPUT #1,IN$
140 PRINT "YOUR PHONE NUMBER IS:"
150 PRINT IN$
160 END
```

PROBLEM:

SAVE one of the programs you wrote above. Type NEW (the command used to clear the program in Memory) and LOAD the program back in. LIST it to make sure you got it back.

CHAPTER K
STICKS AND SOUNDS SUGGESTED EXERCISES FROM SCREEN K.11

PROBLEM:

Using two FOR/NEXT loops, one counting up followed by another counting down, play every pitch from 0 to 255 then 255 to 0.

POSSIBLE SOLUTION:

```
10 FOR PITCH=0 TO 255
20 SOUND 0,PITCH,10,8
30 NEXT PITCH
40 FOR PITCH=255 TO 0 STEP -1
50 SOUND 0,PITCH,10,8
60 NEXT PITCH
70 END
```

PROBLEM:

Get a value from a paddle when the user presses the trigger switch. Play pitches from 0 to that value using a FOR/NEXT loop.

POSSIBLE SOLUTION:

```
10 IF PTRIG(0)<>0 THEN GOTO 10
20 PADDLE0=PADDLE(0)
30 FOR PITCH=0 TO PADDLE0
40 SOUND 0,PITCH,10,8
50 NEXT PITCH
60 SOUND 0,0,0,0
70 GOTO 10
```

CHAPTER L
GRAPHICS
SUGGESTED EXERCISES FROM SCREEN L.11

PROBLEM:

In GRAPHICS 5, use the RND function to generate and PLOT random points on the 80 by 40 screen in random COLOR from 0 to 3.

POSSIBLE SOLUTION:

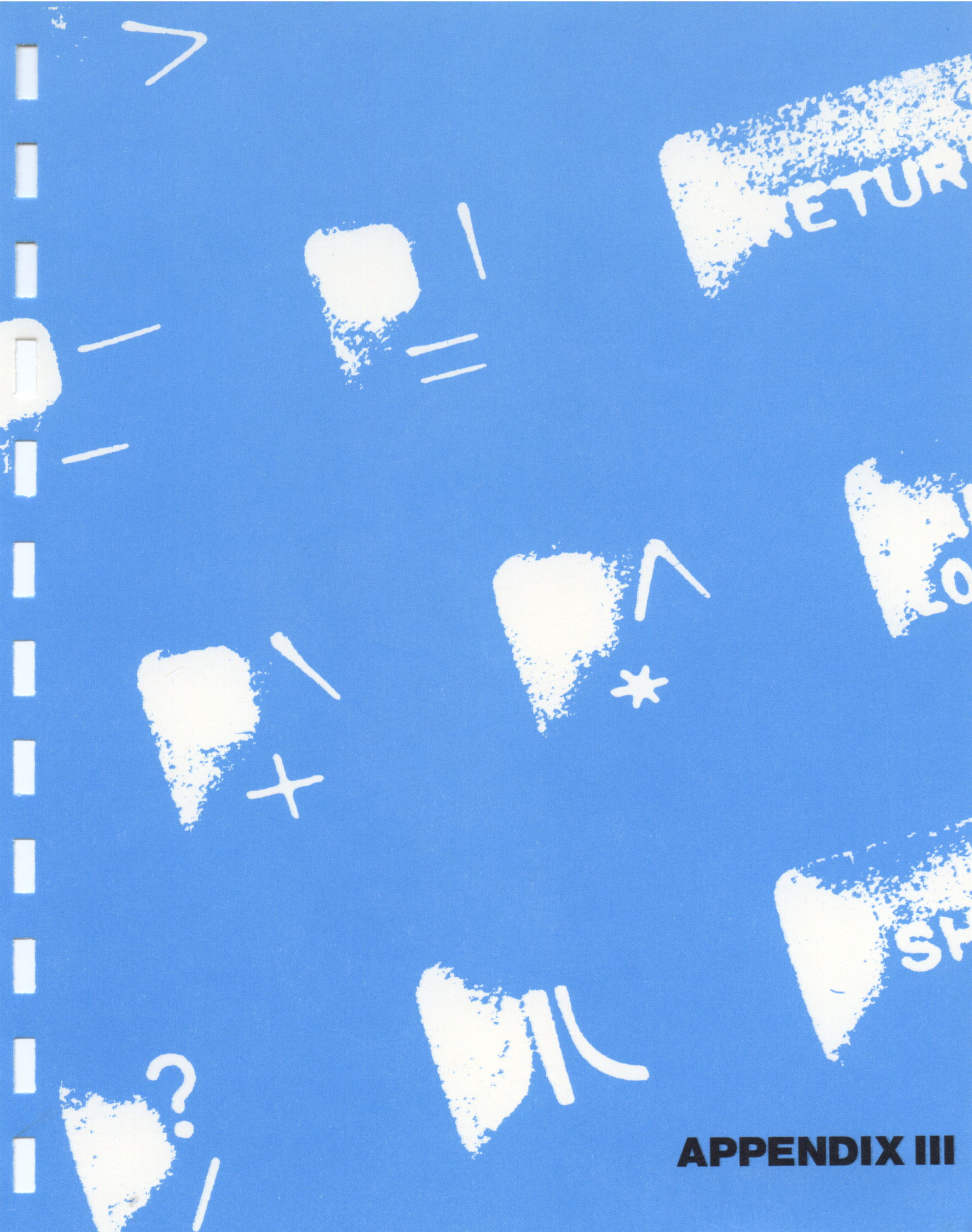
```
10 GRAPHICS 5+16
20 X=INT(RND(1)*80)
30 Y=INT(RND(1)*48)
40 C=INT(RND(1)*4)
50 COLOR C
60 PLOT X,Y
70 GOTO 20
```

PROBLEM:

In GRAPHICS 3, use nested FOR loops to print a series of color bars vertically down the 40 by 20 screen.

POSSIBLE SOLUTION:

```
10 GRAPHICS 3
20 FOR Y=0 TO 16 STEP 4
30 FOR C=0 TO 3
40 COLOR C
50 PLOT 0,Y+C
60 DRAWTO 39,Y+C
70 NEXT C
80 NEXT Y
```



APPENDIX III

ATARI BDT TECHNICAL NOTES

The primary design criterion was that BDT should not alter the operation of a BASIC program. However, it is still a program and therefore it takes up space and does interact with normal operation. Here are some key facts about BDT's operation which will be useful for programmers who are using BDT.

Memory

BDT is a large program. It occupies 12K of RAM just under the BASIC cartridge (from $7000-$9FFF). BASIC screen RAM area is now under $7000 (instead of $A000) and HIMEM is below that. BDT uses 4 zero page locations: $CB, $CC, $CD and $CE. It uses these locations for temporary pointers so that a program can use them, but every time BDT is active these locations will be altered.

Keyboard Routine

After a key is pressed, the 6502 is interrupted and control is passed to a handler whose address is contained in VKEYBD ($208). The keyboard value is placed in CH($2FC) by the handler, and serves as a one character buffer. When BDT is installed, the address in VKEYBD is changed to point to BDT's handler. It is similar to Atari's handler, except that SHIFT—ATARI or CONTROL—ATARI causes a jump to BDT. When BDT is in the command mode, all characters are read from CH and CH is cleared.

BDT works this way in the trace mode unless a GET or INPUT statement is executing. In this case the character in CH goes to the GET or INPUT statement. Also, the one character buffer is gone because BDT will read CH if GET or INPUT doesn't

In the BDT run on continue mode, CH is always read but it is only cleared when the ESC KEY is pressed.

CRT Routine

Since the ANTIC ship and 6502 are both involved in screen display, care must be taken in writing programs which customize the ANTIC display list or which poke characters to the screen. If you follow these guidelines you should be OK:

1) The BASIC screen is not where it used to be. It now starts under $7000 (not $A000).
2) If you PEEK or POKE at the display list vector ($230), the vertical blank vector ($222) or the various color registers, then make sure the BASIC screen is being displayed when you execute that particular statement. Otherwise, you may end up modifying BDT!
3) The DOSINI vector ($0C) is permanently altered to trap the RESET key. If your program alters this vector then BDT will get clobbered every time you press RESET.

Things BDT Doesn't Change

BDT does not:

—use any sound registers
—use any player-missile graphics
—modify DOS
—interfere with DUP
—affect disk I/O
—modify BASIC's display list
—use CIO and IOCB's
—use SIO and DCB's
—hinder RS-232C (the "R:" driver is loaded if available from the Atari 850)
—delete your programs.

meca™

MICRO EDUCATION CORPORATION OF AMERICA
A SUBSIDIARY OF MARKETING CORPORATION OF AMERICA
285 RIVERSIDE AVENUE • WESTPORT • CONNECTICUT • 06880