ATARI® 65XE™

PERSONAL COMPUTER

OWNER'S MANUAL

IMPORTANT INFORMATION

Like any electrical appliance, this ATARI 65XE Computer uses and produces radio-frequency energy. If it is not installed and used according to the instructions in this guide, the equipment may cause interference with your radio or television reception.

If you believe that this equipment is causing interference with your radio or television reception, try turning the equipment off and on. If the interference problem stops when the equipment is turned off, then the equipment is probably causing the interference. With the equipment turned on, you may be able to correct the problem by trying one or more of the following measures:

- Reorient the radio or television antenna.
- Reposition the equipment in relation to the radio or television set.
- Move the equipment away from the radio or television.
- Plug the equipment into a different wall outlet so that the equipment and the radio or television are on different branch circuits.

If necessary, consult your ATARI Computer retailer or an experienced radio and television technician for additional suggestions.

Another helpful resource is How to Identify and Resolve Radio-TV Interference Problems, a booklet prepared by the Federal Communications Commission. This booklet is available from the U.S. Government Printing Office, Washington, DC 20402, Stock No. 004-000-00345-4.

WARNING: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. These rules are designed to provide reasonable protection against such interference when the equipment is used in a residential setting. However, there is no guarantee that interference will not occur in a particular home or residence. Only peripherals (computer input/output devices, terminals, printers, etc.) that have been certified to comply with the Class B limits may be attached to this computer.

Operation of noncertified peripherals with this computer is likely to result in interference with radio and TV reception.

ATARI® 65XE™
PERSONAL COMPUTER

The Right Computer for the Task at Hand

Welcome to the world of home computing. Your new ATARI® 65XE™ is one of the most powerful and versatile small computers you can buy for your home.

Its 64K RAM memory, full-stroke keyboard, sound and graphics capabilities, and an array of compatible software programs and hardware accessories make the ATARI 65XE the perfect computer for entertainment, educational, and business uses. And with the built-in ATARI BASIC programming language, you can begin immediately to write your own computer programs.

This manual is arranged for easy access to the information you need, whether you are a beginning or an advanced programmer. Part 1 shows you how to set up your ATARI 65XE Computer, check whether it's functioning properly, use the keyboard, load software cartridges, and expand the system by adding peripheral devices.

Part 2 is an introductory lesson in BASIC programming. If you already know how to program, you can go directly to the sample programs and reference materials in Part 3. However, the beginning programmer should work through the lessons. Your ATARI 65XE has many applications, and understanding its built-in language will make your computer more fun and more useful.

Once you have the ATARI 65XE set up and working, look at the Resources section of the manual. You'll find that there are unlimited resources for the ATARI Computer owner—from books and programs to users groups and magazines. You will find a whole realm of new activities to do with your ATARI Computer.

Table of Contents

PART 1:GETTING STARTED 1

Meeting the ATARI 65XE 2
Hooking It Up 5
Checking It Out 9
Understanding Your Keyboard 13
Introducing ATARI Graphics 17
Adding an International Character Set 19
Exploring ATARI BASIC 21
Using Software 23
Expanding Your System 25

PART 2: PROGRAMMING WITH ATARI BASIC 27

Editing with the ATARI 65XE Keyboard 29
Writing a Simple BASIC Program:
NEW,LIST,PRINT,RUN 35
Creating an Interactive Loop:
GOTO,DIM,INPUT 41
Using Random Numbers and Mathematical Functions:
RND,+,-,*,/ 47
Making Decisions and Solving Problems:
IF-THEN, FOR-NEXT 55
Producing Sound and Graphics:
SOUND, SETCOLOR, COLOR 65

PART 3: APPENDICES 77

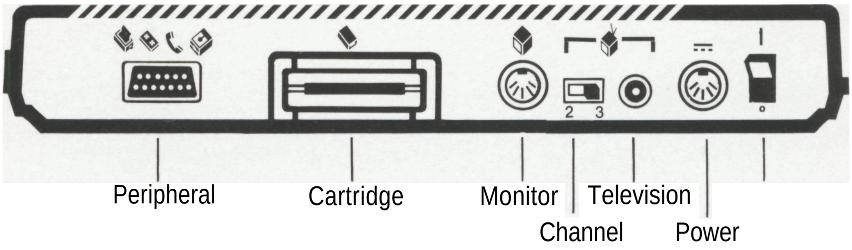
A. Sample Programs 79
B. BASIC Reserved Words 93
C.ATASCII Character Set 101
D. Error Messages 111
E. Troubleshooting 115
F. Resources 117
G. ATARI 65XE Specifications 119

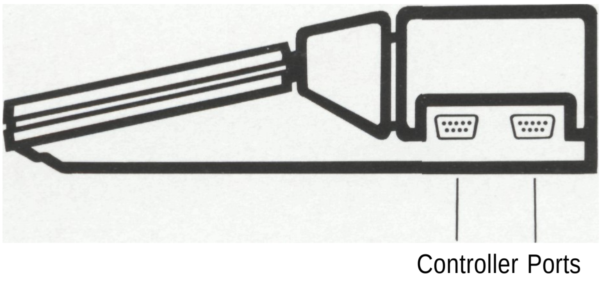
INDEX 121

Part 1

Getting Started

Meeting the ATARI 65XE





Peripheral Port

Connects peripheral equipment, such as disk drives, program recorders, printers, and modems.

Cartridge Slot

Allows the use of plug-in cartridges for quick and easy program loading.

Monitor Jack

Provides access to a monitor's superior resolution.

TV Channel Select Switch

Selects channel 2 or 3 for the clearest picture.

Television Jack

Provides a plug for the connecting cable from the TV Switch Box.

Power Adapter Plug

Connects the computer to the AC power supply.

On/Off Switch

Controller Ports

Connects touch tablets, numeric keypads, joysticks, and paddle controllers.

Hooking It Up

When you unpack your ATARI 65XE, make sure you have the following items:

- ATARI 65XE Computer
- AC Power Supply
- TV Switch Box (NOT FOR EUROPEAN COUNTRY)
- Connecting cable from computer to TV Switch Box
- 65XE Owner's Manual
- Warranty/Registration Card

If you are missing any of these items, contact your dealer. Another good idea is to save the packing materials in case you want to transport the computer or send it in the mail.

Setting up your ATARI Computer system is easy. All you need is a television set and a screwdriver; Atari Corp. provides the rest. Your ATARI 65XE can be connected to an ordinary black-and-white or color television, or you can buy a computer monitor to use solely as a display screen.

To connect the computer to a monitor, follow the installation instructions that come with the monitor. You will need a connecting cable (usually supplied with the monitor) to attach the monitor to the ATARI 65XE Monitor jack on the back of your computer.

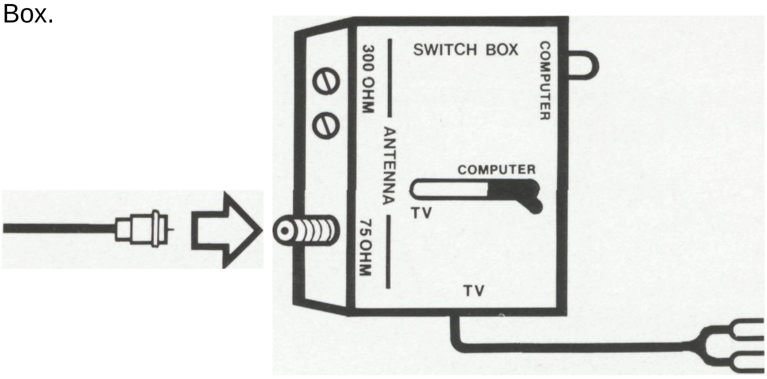
If you are going to use a television as a display screen, locate the TV Switch Box and the connecting cable. The Switch Box attaches directly to your television, but it will not interfere with normal television reception. With the sliding switch in the COMPUTER position, your television will display signals from your ATARI 65XE. In the TV position, signals come from your television antenna.

INSTALLING THE TV SWITCH BOX

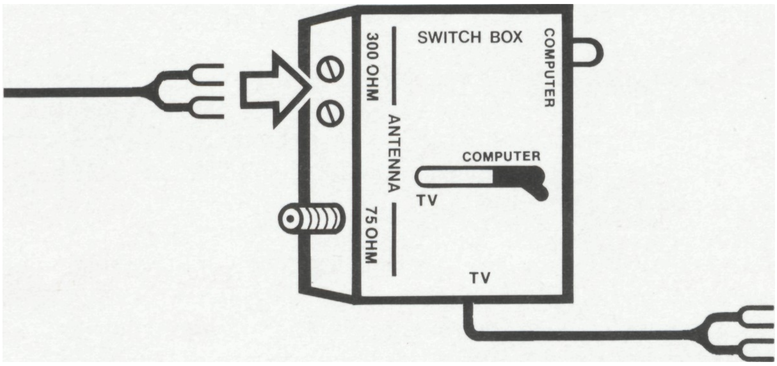
How you install the Switch Box will depend on the kind of antenna connection that you have. There are two types of connections: a round, 75-OHM connection and a flat, 300-OHM twin-lead connection. Follow the steps below for either type of antenna connection:

1. Disconnect your present VHF antenna or your cable TV line from the terminals on the back of your television. If the antenna cable is the

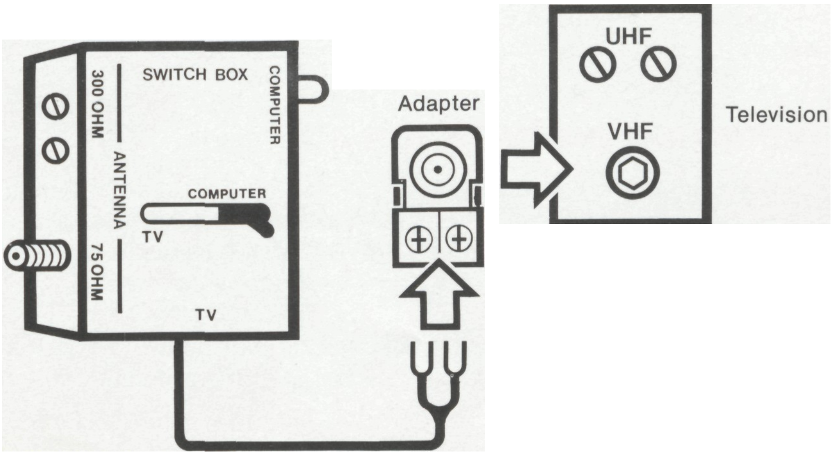
75-OHM round variety with a screw-on connector, attach it to the threaded 75-OHM connector labeled ANTENNA on the side of the TV Switch Box.



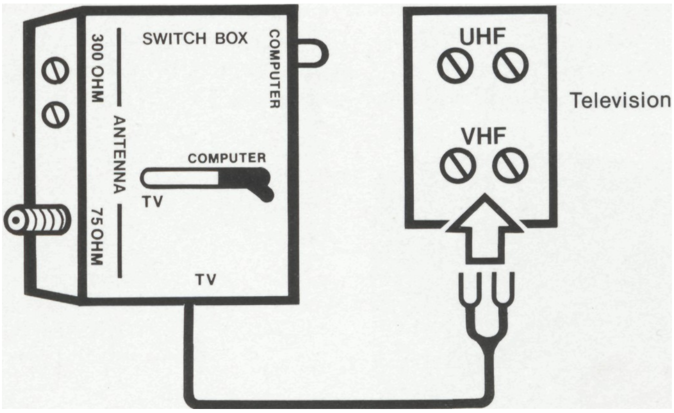
If the antenna cable is a flat twin-lead 300-OHM cable, attach it to the 300-OHM screw terminals on the side of the TV Switch Box.



2. If your television has a threaded VHF connection, attach a threaded VHF adapter to the twin-lead cable labeled TV at the bottom of the Switch Box and attach it to your television. (The adapter should be supplied with your television; if not, you can buy one at a video or television store.)



If your television has two VHF screw terminals, attach the twin-lead cable labeled TV at the bottom of the Switch Box directly to the terminals on your television.

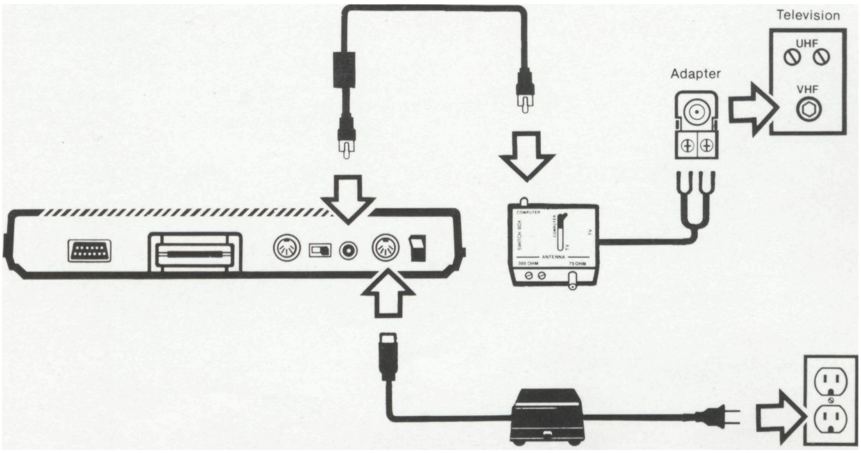


Note: If the terminal connections on your television are different from those shown, refer to the television manual or contact a service center.

3. Remove the cover from the adhesive square on the back of the Switch Box and secure the Switch Box to the back of the television.

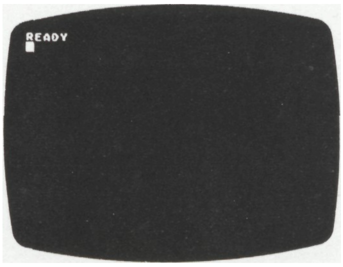
CONNECTING THE COMPUTER TO THE TV SWITCH BOX

4. Plug one end of the connecting cable into the COMPUTER connection on the TV Switch Box and the other end into the Television jack on your computer. Move the sliding switch on the TV Switch Box to the COMPUTER position.



5. With the power switch on the computer in the off position, insert the round plug of the AC adapter into the Power jack on your computer. Plug the other end into an electrical outlet.

6. Turn on your television and select channel 2 or 3, whichever is weaker in your area. Set the Channel switch on the back of your computer to the same channel. Turn on your power switch on the computer. In a few seconds you will see the word READY.



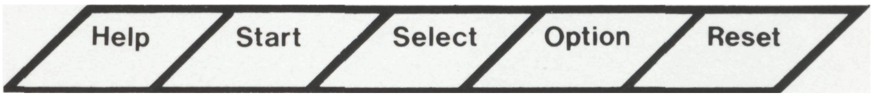
The Ready prompt tells you that ATARI BASIC is ready for your instructions and that your computer is hooked up properly. If the Ready prompt does not show up on your screen, try adjusting your television picture controls. (For more information, see Appendix E: Troubleshooting.)

When you leave your computer on for more than nine minutes without using the keyboard, the computer will start changing colors to protect the television from "color burn." Whenever you turn your computer off, wait three to five seconds before turning it on again.

CARING FOR YOUR ATARI 65XE

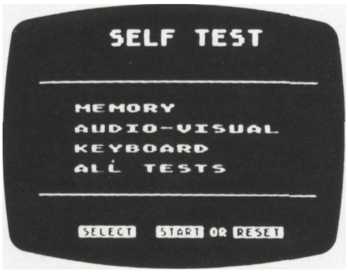
- Wipe off dust with a moist, lint-free cloth.
- Do not use household solvents or cleansers on the computer.
- Keep liquids away from the work area.
- Avoid smoking near the computer.
- Don't move the computer more than is necessary.

Checking It Out



When you turn on your ATARI 65XE, it automatically tests itself. But you can run three separate tests just to insure that your computer is working properly. Running all three tests after setting up your computer for the first time is recommended. To run the tests, turn the computer on while you are holding down the Option key. You may also get to the test program by typing BYE when ATARI BASIC gives the Ready prompt. (If you have a disk drive hooked up to your computer, make sure it is turned off when you run the Self Test.)

A Self-Test menu will appear on the screen. Like a restaurant menu, this menu offers a choice of selections: MEMORY TEST, AUDIO-VISUAL TEST, KEYBOARD TEST, and ALL TESTS. Press Select to choose a test, then press Start. Press Help to return to the Self-Test menu. When you select ALL TESTS, the computer runs all three tests for you. The tests repeat until you press the Help key. Press Reset to stop the tests and return to ATARI BASIC.

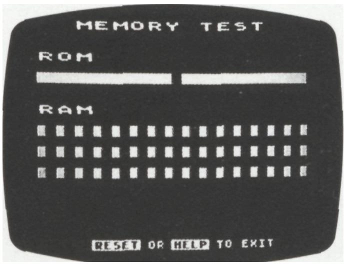


```
SELF TEST

MEMORY
AUDIO-VISUAL
KEYBOARD
ALL TESTS

SELECT  START OR RESET
```

MEMORY TEST

This test checks Read Only Memory (ROM) and Random Access Memory (RAM). ROM is noneraseable memory containing the computer's operating system. RAM is the memory for the programs that you use or write yourself.

The Memory Test works a little like a stoplight. Two bars appear on the screen when the ROM is tested. If the bars turn green, the ROM is okay, and you may proceed. Red means stop; the ROM is out of order, and you must contact your ATARI retailer.

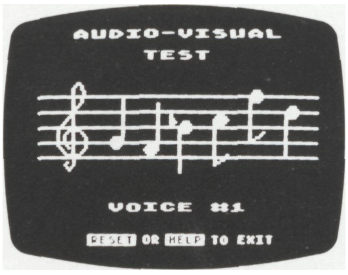


During the RAM test, 48 squares—each representing a RAM section-are tested. The remaining 16K RAM is available only with certain software programs. If a square turns white, then green, the corresponding section is good. If no square turns red, the RAM is okay.

Note: If you enter the test program by typing BYE, you'll get 40 squares during the Memory Test, not 48.

AUDIO-VISUAL TEST

The Audio-Visual Test checks the four programmable sound voices and the color and graphics capabilities. A musical staff and treble clef appear on the screen above the number of the voice being tested. Six notes are played and displayed on your screen. The six notes repeat once for each of the four voices.

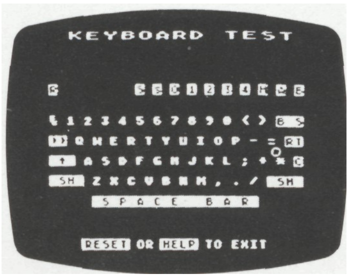


If a voice number appears but you don't hear music, that voice is not working properly. Make sure that the volume on your television is turned up.

The colors displayed on the screen should be consistent during each test.

THE KEYBOARD TEST

When you select this test, you work with two keyboards—the one under your fingertips and a facsimile on the screen. When you press a key, a corresponding key on the screen flashes in inverse video (a blue character on a white background), and a note sounds. The space bar and special-function keys are already in inverse video, so pressing any of them changes the matching key on the screen to normal video. If nothing happens on the screen when you press a key, the key is not working.



The keys in the top row of the screen keyboard are located on the top of your computer keyboard. Disregard keys numbered 1 to 4. Reading from left to right, the keys in the top row on the screen include Reset Start Select Option, four numbered keys, Help, Inverse Video ◩ and Break. Otherwise, the two keyboards have the same arrangement of keys.

The Shift and Control keys flash only when they are pressed simultaneously with another key.

There are three keys that do not flash or sound and that terminate the test when they are pressed: Help returns you to the Self-Test menu; Reset returns you to BASIC; and Break evokes no response. The Break key does not function in the Self Test.

In addition to the optional Self Test, your ATARI 65XE automatically runs the Memory Test each time you turn the computer on. If the computer's circuits fail the test, your screen will display the words MEMORY TEST, along with red and green squares and rectangles. As long as you experience trouble-free operation, running the Self Test every few months should be sufficient.

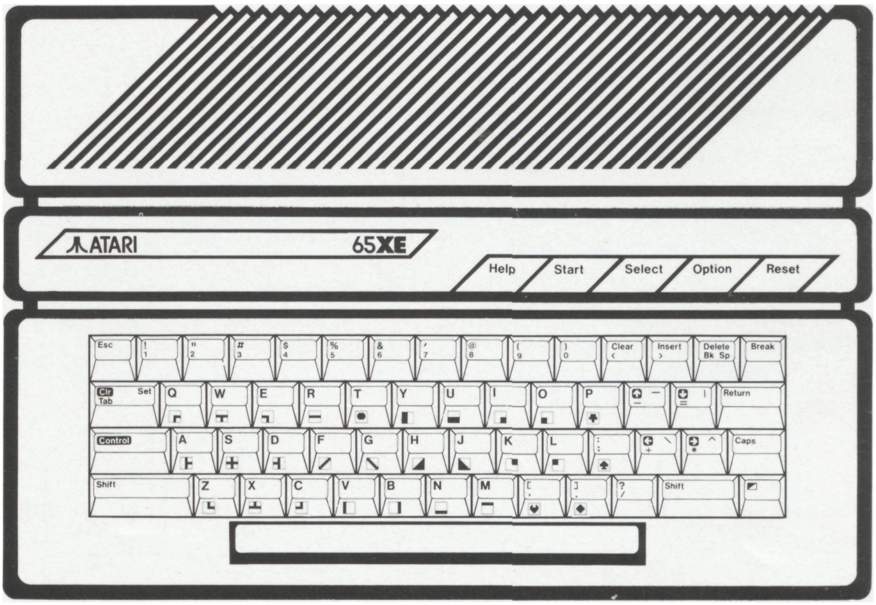
LOADING SOFTWARE CARTRIDGES

After you have tested your ATARI 65XE, you may want to load a software cartridge. To load a software program, make sure that the computer is turned off, hold the cartridge so that the label is facing up, push the cartridge firmly into the Cartridge slot on the back of the computer, and turn on the computer. If you have already been using another software program, turn off the computer, remove the first cartridge, insert the new cartridge, wait several seconds, and turn on the computer again. (For more information about software cartridges, see Using Software.)

If you don't want to use ATARI BASIC (the programming language that is built into your ATARI 65XE), you can temporarily disable the language in several ways: plug in a software cartridge; hold down the Option key when you turn on the computer; type BYE to exit to the Self Test; or, if the computer is connected to a disk drive, type DOS to activate the Disk Operating System. (For more details, see Exploring ATARI BASIC.)

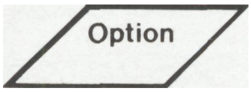
Understanding Your Keyboard

Although the ATARI 65XE keyboard looks and works much like a typewriter, some special keys allow you to reach into a bag of special editing tricks. Some of the keys,like Shift and Control are used in combination with other keys. These double-key combinations provide extra commands and functions without increasing the size of the keyboard.



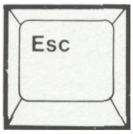
The following general descriptions explain the special keys and how they are used. Because the function of a key may change from one program to another, it is important to consult the manual that accompanies each program you use. For a detailed description of how the keyboard functions with built-in ATARI BASIC, see the chapter Editing with the ATARI 65XE Keyboard in Part 2: Programming with ATARI BASIC.

Stops the computer in the middle of an activity and returns the program to the opening screen. Pressing [Reset] produces the same effect as turning the computer off and on again—with two exceptions. First, in most programs [Reset] will not erase the computer's RAM memory, whereas turning the computer off and on will. And second, use of the [Reset] key will save wear and tear on the power supply and the on/off switch.



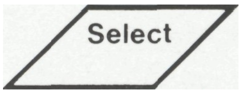
Temporarily disables BASIC when pressed as the computer is turned on. In some programs, chooses among program variations.

Usually interrupts whatever function the computer is doing. Refer to individual program instructions.

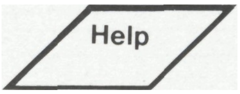


Varies from program to program but is often used to go (escape) from one menu to another.

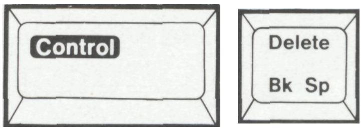
Usually tells the computer to begin running a game or a program. Consult the software manual.



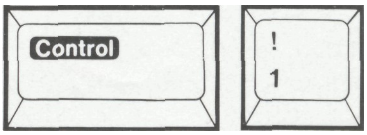
Often used to select one of several applications within a program. Because its function varies, consult the software manual.



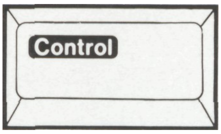
Gives you instructions in some programs when you need help.



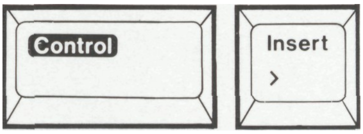
Deletes the character underneath the cursor and shifts the remaining characters on the line to close up the empty space.



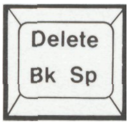
In most programming languages, stops a moving screen display when you want to view a listing. Press [Control] [1] again to continue.



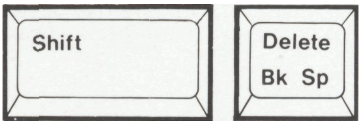
Always used in conjunction with another key. Prints special graphics characters when used with the alphabet keys in ATARI BASIC.



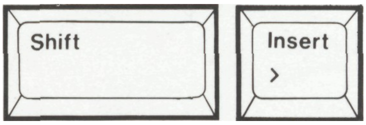
Inserts a space between characters in ATARI BASIC, although its use varies from program to program.



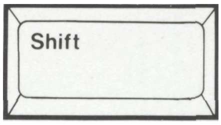
In most programs, including ATARI BASIC, deletes (erases) the character to the left of the cursor and moves the cursor one space to the left. Does not close up the space caused by the deletion.



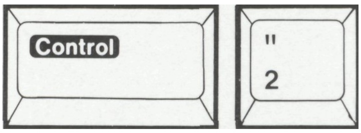
Deletes a program line from the screen. The program statement remains in the computer's memory.



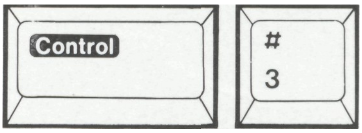
Inserts a blank line in ATARI BASIC.



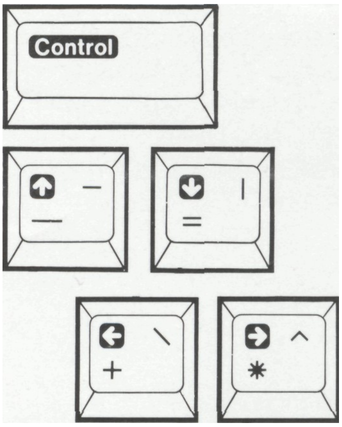
In conjunction with other keys, types uppercase characters without leaving the lowercase mode.



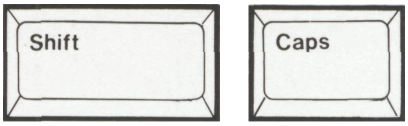
Sounds the buzzer. (The TV speaker must be turned up for the buzzer to be heard.)



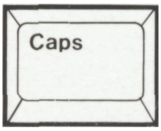
Produces an end-of-file (EOF) response to a program that is reading input from the keyboard. (Used by more experienced programmers.)



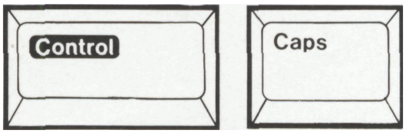
When used with the Control key, the arrow keys move the cursor up, down, left, and right.



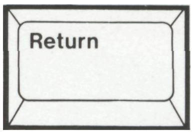
Locks the computer into the uppercase mode for alphabet characters. You will still need to press Shift to enter the uppercase mode on the numeric and symbol keys.



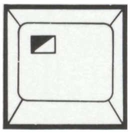
Shifts the computer between uppercase and lowercase character modes in ATARI BASIC and other programs. Sometimes used in conjunction with the Shift key to change modes. In ATARI BASIC, exits from the Control Lock (graphics character) mode.



Locks the computer into the Control mode. Used when entering a series of commands that require pressing the Control key, such as creating graphics characters in ATARI BASIC.



Tells the computer that you are done typing or editing a program line. Returns the cursor to the left margin.

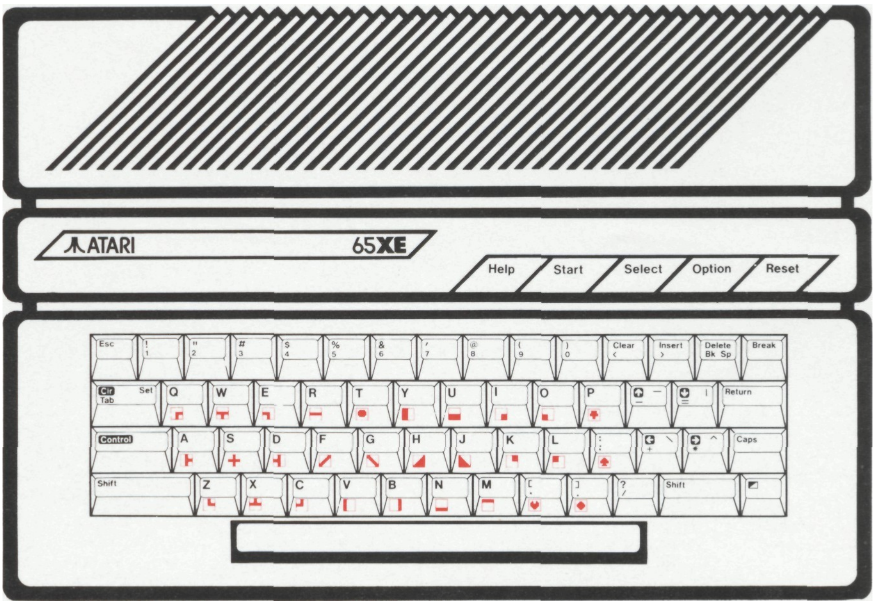


Turns the Inverse Video mode on and off. On ATARI® 400™ and ATARI® 800™ models, this key was referred to as the ATARI logo key

Most keys on the 65XE will automatically repeat when they are held down for more than half a second.

Introducing ATARI Graphics

The ATARI 65XE keyboard has 29 built-in graphics characters. They can be used to brighten up a chart or to create a work of art. To display the graphics characters on your screen, press the Control key and any of the keys shown below. If you intend to use several characters, it may be more convenient to lock in the Control mode by pressing Control and Caps. Press Caps to exit from the Control Lock mode and return to alphabet characters.



Adding an International Character Set

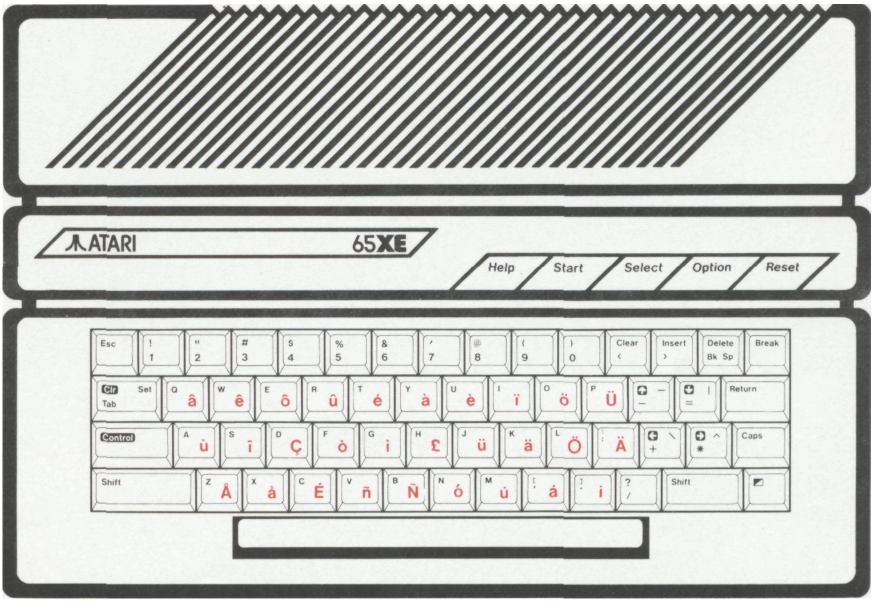
Ordinarily when you type pages in a foreign language, you go back and laboriously add accent marks, cedillas, and other diacritical marks by hand. You are spared that trouble with the ATARI 65XE Computer, which has an international character set.

International characters are available when you use ATARI BASIC. The diagram below shows the international characters that are associated with the letter keys. To activate the international characters, type the statement below and press the Return key:

```
POKE 756,204
```

Pressing Control and any of the keys shown below will produce an international character instead of a graphics character. To return to the graphics character mode and normal keyboard utilities, type the following line and press the Return key:

```
POKE 756,224
```



Exploring ATARI BASIC

Your ATARI 65XE Computer has built-in ATARI BASIC, a version of one of the most popular programming languages. BASIC (Beginner's All-purpose Symbolic Instruction Code) was developed at Dartmouth College in the 1960s to teach computer programming to beginning college students. Since then, it has become the most commonly used programming language for home computers.

Although BASIC is a single language, each version of BASIC language is slightly different, and ATARI BASIC has some important and unique features. For instance, special words in ATARI BASIC make producing sounds and creating color graphics easy. Also, ATARI BASIC is specially designed for the beginning programmer. Unlike many versions of BASIC, ATARI BASIC will check each program line that you write and tell you if you have made an error in grammar or syntax. And, of course, when you learn to program with ATARI BASIC, learning other versions of BASIC will be easier.

ATARI BASIC is immediately available when you turn on the computer. If you don't want to use BASIC, hold down the Option key when you turn on the computer. Another way to exit from BASIC is to type BYE, which activates the Self Test, or DOS, which activates the Disk Operating System (when the computer is connected to a disk drive).

The second part of this manual, Programming with ATARI BASIC, is a simple tutorial for the first-time BASIC programmer. Unlike most tutorials, which make you study the syntax of the language first, this approach lets you immediately begin to write programs and word games, solve mathematical problems, and make use of the ATARI 65XE sound and graphics capabilities. When you are done with this step-by-step lesson, you will understand much better the functions of the keys and the workings of the computer. The tutorial will help you get the most out of your new ATARI 65XE.

If you already know how to program in BASIC, the appendices give you most of the reference tools you will need to work at your own speed. Appendix A: Sample Programs includes programs of varying levels of difficulty to give you some practice before you start writing your own programs with ATARI BASIC. Appendix B: BASIC Reserved Words offers a list of the commands used in ATARI BASIC and a brief

description of what each one instructs the computer to do. For advanced-level programmers, Appendix C: ATASCII Character Set lists the decimal and hexadecimal locations of characters in the ATASCII code. Whenever Error messages appear on the screen, you can refer to Appendix D: Error Messages to find out what went wrong.
Check Appendix F: Resources for the names of other valuable sources of information about ATARI BASIC.

Using Software

Your ATARI 65XE and your television are the hardware components of your system. Software refers to the computer programs that tell your ATARI 65XE what to do and how to do it. Some software programs are built into your computer, some you can buy, and some you can write yourself.

BUILT-IN SOFTWARE

Built-in software is permanently encoded in the Read Only Memory (ROM) chips inside the computer. Your ATARI 65XE contains a ROM chip encoded with the ATARI BASIC programming language and the Operating System program. The Operating System in the ATARI 65XE contains programs that allow the keyboard, display screen, program recorder, and most ATARI printers to communicate with one another.

COMMERCIAL SOFTWARE

Commercial software programs are usually written by professional programmers and are available from retail outlets. Video games, spreadsheet programs, word processing programs, and programmed math lessons are just a few of the many types of software available for entertainment, business, and educational applications.

The ATARI 65XE and the ATARI® 800XL™ Computers are completely compatible. You select whatever program you want to use, as long as it is designed for use with the ATARI 65XE or the ATARI 800XL Computer. Software and hardware for the 800XL can be used with the 65XE.

Your ATARI 65XE is equipped to use software cartridges. If you want to use software in diskette or audiocassette form, you will need to purchase a disk drive or a program recorder.

Note: Some software programs written for the earlier ATARI 400/800 models may not work with your ATARI 65XE. If you have a disk drive, this problem can be remedied with a Translator Disk. The disk is available from ATARI Customer Relations, P.O. Box 61657, Sunnyvale, CA 94088. Cost: $9.95 plus $2.50 for shipping. California residents add 6.5 percent tax.

USER-WRITTEN SOFTWARE

Once you learn a programming language, you can write your own programs. The ATARI 65XE comes with the ATARI BASIC programming language already built into the computer. The tutorial in Part 2 of this manual teaches you how to use this language to write BASIC programs. However, you can write programs with other computer languages, such as ATARI Logo, by inserting a cartridge containing the programming language of your choice.

SAVING YOUR WORK

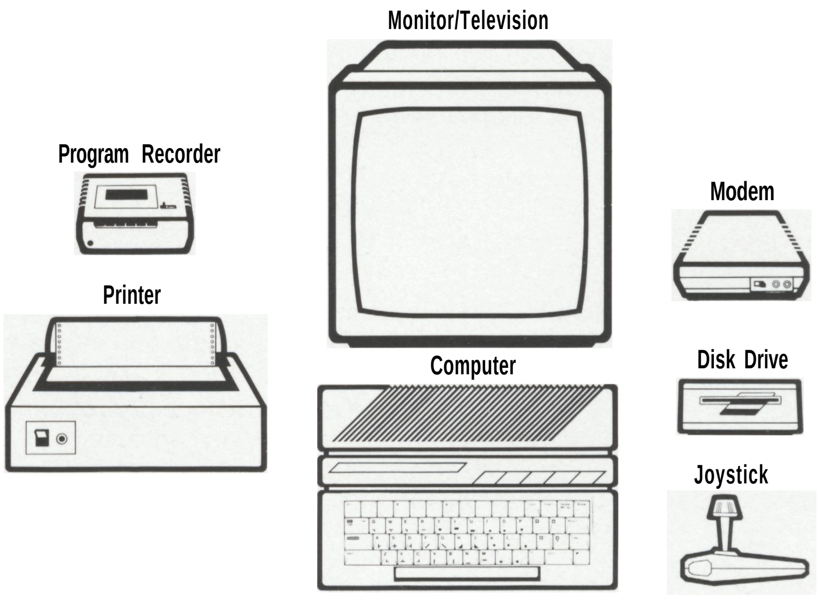
If you write your own programs, you will probably want to store them so that you can use them again. And if you use your computer for word processing, you will probably want to save that work, too. Just turning off the computer will automatically erase your work, but connecting a disk drive or a program recorder to your computer enables you to store and retrieve your work later. The disk drive saves information on magnetic diskettes, and the program recorder uses ordinary blank audiocassettes. The instructions that come with the disk drive and the program recorder will tell you how to use these devices. (For more information, see Expanding Your System.)

LOADING SOFTWARE CARTRIDGES

Loading ready-to-use commercial software cartridges into the ATARI 65XE is a simple operation. Software that comes in a cartridge is inserted into the Cartridge slot at the back of the computer console. First make sure that the computer is turned off. Inserting or removing a cartridge while the computer is on can damage the computer and the cartridge. Hold the cartridge with the label facing upward. Insert the cartridge securely into the slot. Then turn the computer on. If you have just been using another cartridge, wait several seconds between the time you insert the new cartridge and the time you turn the computer back on.

Expanding Your System

Your ATARI 65XE Computer is the center of a powerful and versatile system. Whatever job you need to do or game you want to play, the 65XE has a peripheral to help. The most popular peripherals are illustrated below.



TV OR MONITOR

A TV or monitor is the display screen for your computer. Many people prefer to use a monitor because it provides a sharper picture and does not conflict with television usage. Either a color or monochromatic (black-and-white) monitor or television can be used. However, many software programs are designed to showcase the colorful graphics of the ATARI 65XE Computer.

DISK DRIVE

A disk.drive lets you store and retrieve programs and other information that you create on your computer. Also, because many programs are available in diskette form, the disk drive provides access to an extended library of ready-to-use programs in the areas of word processing, financial management, education, and entertainment.

PROGRAM RECORDER

The program recorder is an inexpensive storage device. Like a disk drive, it lets you save programs and other information. Although not as efficient as a disk drive for business and professional use, it does provide access to a library of ready-to-use educational and entertainment programs.

PRINTER

As an ATARI Computer owner, you have a variety of printers to choose from. To print out everyday reports and letters and your programs, you can use an inexpensive dot-matrix printer. For printing high-quality business letters, a letter-quality printer might be better. And for printing your graphics, a variety of color and graphics printers will work with your ATARI 65XE.

MODEM

A modem, your computer, and a telephone line will give you access to a wide world of electronic communications. If you and a friend have modems connected to your computers, you can send letters to each other. And if you are connected to some of the many data base and information services that are available, you can take advantage of users groups, electronic mail services, news services, and reference materials.

JOYSTICK

Joysticks are popular tools for entertainment and educational programs. Using a joystick, you can control your computer without touching the keyboard.

Part 2

Programming with ATARI BASIC

Editing with the ATARI 65XE Keyboard

No matter how well you type on a typewriter, you will need to familiarize yourself with the special features of the ATARI 65XE keyboard before you begin editing with ATARI BASIC.

AUTO REPEAT FUNCTION

Begin by typing the letter A:

```
A
```

Continue to hold down the [A] key and watch the rows of A's appear. When a line is filled, the cursor automatically drops down to the next row. There is no need to press [Return]

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

You are using the built-in auto repeat function of the ATARI 65XE keyboard. Most of the keys, including the space bar, have auto repeat. Did you hear a buzzer when the A's almost filled the third line? That warning buzzer—a built-in function of ATARI BASIC—indicates that the instruction line is getting too long. An instruction line can be no more than three lines long.

ERROR MESSAGES

Find the [Return] key and press it. You should see the word ERROR on the screen, followed by the three lines of A's that you typed. Your computer is interacting with you now. It is telling you that it doesn't understand what you are typing because the rows of A's aren't part of the BASIC language. Clear your screen by pressing [Return] until the Error message no longer appears. To avoid getting Error messages while you are following the directions for editing, do not press the [Return] key until you are instructed to do so.

UPPERCASE AND LOWERCASE

To make lowercase A's, press the [Caps] key once and hold down the [A] key. You should see this:

```
aaaaaaaa
```

To return to uppercase letters, press the Caps key and type more A's. You should see this:

```
AAAAAAA
```

Try typing a word—a word beginning with A, such as ATARI. Type in the following words, switching between upper- and lowercase letters with the Caps key:

```
ATARI 65XE atari 65xe
```

The numbers appear the same whether you are typing capitals or lowercase letters. Unlike a typewriter, the computer has separate keys to control capitalizing and shifting. In both the lowercase and the uppercase modes, the symbol that appears is always the one shown on the bottom of a key. To get the symbol on the top half of a key, use the Shift key. There are two Shift keys on the keyboard. You can use either one.

Using the Caps, Shift, and 1 keys, try typing this:

```
!!! ATARI 65XE !!!  !!! atari 65xe !!!
```

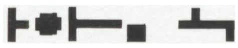
Experiment with different words, letters, and punctuation marks.

GRAPHIC SYMBOLS

Many of the keys have two or three symbols on them. Each letter key has a letter on the top side and a graphic symbol on the front side. Some of the other keys have three symbols or words, all on the top side. One function of a key is activated by pressing the key only, another by pressing Shift and the key, and the third by pressing Control and the key. Graphic symbols are produced by pressing Control and the key.

To type a graphic symbol (the symbol on the front side of a letter key), use the Control key on the left side of the keyboard. First press the Control key. While you are still holding down the Control key, press a graphic symbol key. Then release both keys.

Hold down the Control key and try typing ATARI 65XE. You should see this:



Only seven characters appear on the screen. The number keys do not have graphic symbols. If you use the [Control] key when you press a number key, no graphic symbols appear.

Graphic symbols are most useful in making screen designs, borders, and simple artwork. You can lock the keyboard into graphic symbols by holding down [Control] and then pressing [Caps]. Pressing [Caps] just once will put you back into the lowercase mode.

CURSOR CONTROL

The [Control] key is used most frequently for directing the movement of the cursor. The cursor is the small white square that marks your place on the screen. Find the Up Arrow key [↑] next to the letter P. The arrow, like the [Control] key, is outlined in white. This white marking indicates that the arrow function is activated only by using the [Control] key. Press the [Control] key and then the Up Arrow key [↑] and watch the cursor move up to the top of the screen. When it reaches the top, the cursor returns to the bottom of the screen and starts moving up again. Now try out the other directional arrow keys: [↓], [→], [←]. Remember to use the [Control] key.

CLEARING THE SCREEN

The [Control] key is frequently used in conjunction with the [Clear] key to erase everything on the screen. Press and hold the [Control] key, then press the [Clear] key. This action should clear your screen and return the cursor to the upper left-hand corner of the screen. Try it again.

Now fill up the screen with more letters, numbers, words, and graphic symbols. This time use the [Shift] key with the [Clear] key to clear the screen. Both [Shift] [Clear] and [Control] [Clear] empty the screen and return the cursor to the upper left-hand corner.

INSERTING

The [Control] key is used with the [Insert] key to insert spaces in a line. To practice this function, type

! ! ! ATARI 65XE ! ! !

Position the cursor on top of the first letter A in ATARI. Holding down the [Control] key, press the [Insert] key 11 times. You should see this:

! ! !   ATARI 65XE ! ! !

Eleven blank spaces have been added in the middle of the line. This function is very useful for inserting words. Using the cursor control keys (the arrows), return to the space next to the third exclamation mark on the line. Press the space bar once and type THIS IS AN in the blank spaces as shown below:

! ! ! THIS IS AN ATARI 65XE ! ! !

To add blank lines, rather than individual blank spaces, hold down the [Shift] key, then press the [Insert] key. A whole new blank line will appear on the screen. Insert a few more blank lines, but don't insert so many that you have a blank screen. Keep the sentence on the screen so that you can proceed to the next exercise.

DELETING

Using the [Control] key with the [Delete Bk Sp] key makes deleting jus as easy as inserting. Position the cursor on the T in the word THIS. Holding down the [Control] key, press the [Delete Bk Sp] key 11 times. Your screen should look like this:

! ! ! ATARI 65XE ! ! !

You now know how the [Control] and [Delete Bk Sp] keys work. To discover what the [Delete Bk Sp] key does when pressed by itself, position the cursor on the first A and press the [Delete Bk Sp] key four times. Your screen should look like this:

ATARI 65XE ! ! !

When used alone, the [Delete Bk Sp] key moves the cursor to the left, erasing as it goes, but it does not close up the space. Using the [Control] key with the [Delete Bk Sp] key erases the characters to the right and closes up the gap.

The third function of the [Delete Bk Sp] key requires the use of the [Shift] key. Pressing and holding [Shift] and then pressing [Delete Bk Sp] deletes an entire line and returns the cursor to the left margin. It does not matter where the cursor is positioned on the line when you press [Shift] [Delete Bk Sp] ; the entire line is erased and is not retrievable.

TABS

On a blank screen move the cursor to the left margin and type an asterisk. Press the Tab key. Every time the cursor stops, type an asterisk. You should have six asterisks spaced across the screen as shown below:

```
*    *       *       *       *       *
```

Press the Tab key only and notice that it stops at the same preset tab marks every time. The first preset tab is five spaces from the left margin (a normal paragraph indention), and the following tabs are eight spaces apart. Position the cursor on top of the first asterisk and move it in three spaces. Press and hold Shift and then press Tab to activate the Set Tab function. Move the cursor back to the left margin, then press the Tab key. The cursor jumps to the newly set tab position.

Continue to press the Tab key. It continues to go to all the preset tab positions, in addition to the new one. When the cursor jumps down to the next line, it ignores the new tab position. (But on all the following lines the cursor will go to all the tab positions—the new one and the preset ones.) Return the cursor to the first asterisk and press the Tab key. The new tab mark is still there.

Return the cursor to the left margin. Press Tab to move it to the first tab mark (three spaces in). Use the Control key with the Tab key to activate the Clr Tab function. Press the Tab key to get to the next tab position and clear that one also. Move the cursor back to the left margin of the same line and press the Tab key only. The cursor should skip two tab positions. Continue pressing the Tab key until the cursor drops to the next line. Clr Tab did not clear the second tab position on this line. (However, both tab positions have been cleared from all the following lines.)

INVERSE VIDEO

Type the word ATARI. Find the Inverse Video key ◪ and press it just once. Type ATARI again. Press the Inverse Video key again and type ATARI again. Your screen should look like this:

ATARI ATARI ATARI

Inverse Video creates blue letters on a white background, the inverse of the normal screen colors. This function is very useful for highlighting letters in your programs. Just one touch of the Inverse Video key changes the way the letters are displayed.

MISCELLANEOUS KEYS

Another important key is the Escape key $\boxed{\text{Esc}}$, When you press it once, nothing happens. When you press it twice or more, this graphic appears on the screen: ␛ . Press $\boxed{\text{Return}}$ and try again. In later sections you will need to use the $\boxed{\text{Esc}}$ key.

The $\boxed{\text{Break}}$ key is in the upper right-hand corner. When you press this key, the cursor drops one line and moves to the left margin. In the chapter about looping, you will learn how to use the $\boxed{\text{Break}}$ key.

$\boxed{\text{Help}}$ $\boxed{\text{Start}}$ $\boxed{\text{Select}}$ $\boxed{\text{Option}}$ Press each of these keys. Nothing happens. These keys are programmable and often have functions in software programs.

After you press the $\boxed{\text{Reset}}$ key, the screen will turn blank for a second or two, and the Ready prompt will appear. The $\boxed{\text{Reset}}$ key restarts the system. You should use this key very sparingly because, in many programs, the information that you are entering or have entered will be lost.

Writing a Simple BASIC Program: NEW, LIST, PRINT, RUN

Once you know your way around the computer keyboard, it's easy to write your first program. To begin, clear the screen and make sure the cursor is on the left-hand margin.

NEW: CLEARING THE COMPUTER'S MEMORY

Type in the word NEW, then press [Return] :

```
NEW
```

NEW tells the computer to get ready for a new set of instructions by erasing any old instructions that might be in the computer's memory.

LIST: CHECKING THE COMPUTER'S MEMORY

To make sure nothing is in the computer's memory, ask the computer to list any instructions that it might be storing. Type LIST on a line by itself and press [Return] :

```
LIST
```

If you typed NEW correctly, nothing appears on your screen. Now you can begin a new program. Type in the first line of instruction to the computer. Type in the line exactly as it appears below and press [Return] after the last quotation mark:

```
10 PRINT "I HEARD OF A POET NAMED SAM"
```

All instruction lines in BASIC programs are numbered. When you type this one-line program, make sure that the 1 and the 0 in the number 10 are numerals, not letters. If you used letters instead of numbers, you will get an Error message.

A numbered instruction line in a program can be longer than one line on the screen. When the cursor runs out of space on one line, it automatically drops down to the next line. You should press [Return] only at the end of an instruction line to tell the computer that you are done typing the instruction and that it should store the instruction in its memory. Nothing dramatic happens when you press [Return] ; the cursor merely returns to the left margin so that you can begin another line in the program.

RUN: EXECUTING INSTRUCTIONS

To make the computer execute your program, you have to type RUN. The RUN command tells the computer to carry out its instructions. Type RUN and press [Return] to see what happens:

```
RUN
I HEARD OF A POET NAMED SAM
```

The computer's first and only instruction, line 10, was to print the words inside the quotation marks. Clear the screen, type RUN again, and press [Return] . The computer follows its instruction again and prints I HEARD OF A POET NAMED SAM.

Even though the instruction is no longer on the screen, the computer remembers what to do. Your program is stored in RAM (Random Access Memory), the programmable section of the computer's memory. When you type LIST, the computer shows on the screen all the instructions stored in the RAM portion of its memory. Type LIST. Your screen should look like this:

```
LIST
10 PRINT "I HEARD OF A POET NAMED SAM"
```

If your screen looks different, you might have forgotten to press [Return] at the end of each entry or to type LIST on a line by itself. Type in the line below, then give the RUN command:

```
20 PRINT "I MET HIM ONE DAY, AND TO MY DISMAY, "
RUN
```

The words enclosed in the quotation marks in both lines of the program appear on the screen. Type LIST to see the instructions that the computer has stored in RAM. Both lines 10 and 20 appear.

LINE NUMBERING: CREATING ORDER

Each instruction line in a BASIC program must have a number in front of it. The numbers are called "line numbers." The computer executes the instructions, beginning with the smallest number and continuing through the program until all the instructions have been carried out. The usual procedure is to number lines by tens so that enough numbers are available for inserting additional lines later, if desired. Try inserting a line now. Add line 15 (shown on the next page) and instruct the computer to run the program. Your screen should look like the following program:

```
15 PRINT "WHOSE POEMS WERE THE TALK OF THE LAND. "
RUN
I HEARD OF A POET NAMED 5AM
WHOSE POEMS WERE THE TALK OF THE LAND .
I MET HIM ONE DAY, AND TO MY DISMAY,
```

The computer automatically inserted line 15 between lines 10 and 20. Write another line:

```
30 PRINT "HIS BRAINS WERE SILICON-SAND. "
RUN
LIST
```

The RUN and LIST commands cause all four lines of PRINT instructions to appear on the screen.

ERROR MESSAGE: COMPUTER TALK FOR "I DON'T UNDERSTAND"

PRINT simply tells the computer to print whatever is inside quotation marks on the screen. The computer doesn't care what words or symbols are inside the quotation marks; the words don't need to be spelled correctly or make sense. Try out the instructions below:

```
40PRINT"AYESAYHYE.:USAYBI."
RUN
```

Even when the quotation marks enclose a nonsense sentence containing misspelled words, the computer does what it is told to do. However, try misspelling PRINT as shown below and see what happens:

```
50 PRIMT "I SAY HI; YOU SAY BYE. "
```

The computer sends you an Error message. The computer checks up on only those instructions that are outside the quotation marks because those instructions are intended for the computer. Instructions that are inside the quotation marks are intended for you, so the computer copies them exactly. Move to a blank line but do not clear the screen. Run the program to see what happens.

Error message 17 appears at line 50, the line in which you intentionally misspelled PRINT. Error message 17 is called the "syntax error." It indicates that the instructions were undecipherable to the computer. (For a complete listing of Error messages, see Appendix D.)

There are several ways to correct an Error message. The easiest solution is to move the cursor to the line that contains the typing error. Place the cursor on the offending M in PRIMT and change it to N.

Press [Return]. (In this case, you can press [Return] regardless of the cursor's position on the line, even if it is in the middle of the word PRINT.) No new Error message appears this time. Clear the screen and run the program. The screen should not show any Error messages.

Another way to correct an Error message is to erase the offending line. To practice this technique, type another line that has an intentional error. This time omit the quotation marks in the PRINT statement below, then run and list the program:

```
60 PRINT I ONCE HAD A PROGRAM CALLED BOZON
RUN
LIST
```

An Error message appears when you press [Return] and when you try to run and list the program. To erase the offending line, simply type the line number and press [Return] :

```
60
RUN
LIST
```

Now the program runs and lists without errors, although line 60 does not contain any instructions. The line I ONCE HAD A PROGRAM CALLED BOZON has been erased. Typing the line number and pressing [Return] erases a line entirely from the computer's memory. Type the line correctly as shown below:

```
60 PRINT "I ONCE HAD A PROGRAM CALLED BOZON"
RUN
```

PRINT: CREATING BLANK LINES

Inserting a blank line after the poem would make the poem more readable. Type in the following instructions to create a blank line between lines 30 and 40:

```
35 PRINT
RUN
LIST
```

When nothing follows the PRINT command, the computer creates a blank line. Insert another blank line between lines 50 and 60. Use 55 for the line number and type only the word PRINT after it.

? : ABBREVIATION FOR PRINT

You can save time and effort by substituting a question mark (?) for PRINT. Try the next program line below:

```
70 ? "THAT RAN FROM DUSK UNTIL DAWN. "
RUN
LIST
```

The program runs the same with ? as with PRINT. The question mark is just a convenient shortcut. For clarity, all the following PRINT statements in this tutorial use the word PRINT, but you can substitute a question mark.

LOGICAL LINE LENGTH

Sometimes the quotation marks contain too many characters to fit on one or two lines. Make sure the sound is audible on your television or monitor before you type the following sample:

```
80 PRINT "IT WOULDN'T RESPOND TO ESCAPE, BREAK, CONTROL,
OR LIST.. AND IT WAS STILL RUNNING WHEN I TURNED OFF THE
SWITCH."
```

When the cursor reaches the third line, a buzzer sounds. The buzzer indicates when you are approaching the maximum length of an instruction line. An instruction line can be no longer than three screen lines. This limit is called a "logical line." (You may wish to turn down the volume now.)

SCREEN DISPLAY

Words are often broken in awkward places when the cursor reaches the end of a line on the screen. Also, the spacing between words when you type in the program lines is different from the word spacing when the computer runs out the program. To avoid both of these problems, determine what you want each line to look like and type separate PRINT statements for each line. Retype the sentence in line 80 so that it appears in a poem format:

```
80 PRINT "IT WOULDN'T RESPOND"
90 PRINT "TO ESCAPE, BREAK, CONTROL, OR LIST,"
100 PRINT "AND IT WAS STILL RUNNING"
110 PRINT "WHEN I TURNED OFF THE SWITCH.
RUN
LIST
```

PRINTING GRAPHIC SYMBOLS

You can also use graphic symbols in PRINT statements to produce simple artwork. To set off the poem, type the lines below. Use [Control] [J] and [Control] [H] to create the graphics:

```
58 PRINT "◣"
115 PRINT "◢"
```

PRINT "↖":CLEARING THE SCREEN

You can make your program look even better by making sure the screen is clear when you start. Type a line number, PRINT, and the first quotation mark. Press the [Esc] key once lightly. Then press either the [Shift] and [Clear] keys or the [Control] and [Clear] keys. A bent arrow appears on the screen. Type another quotation mark and press [Return] . Then run and list the program:

```
5 PRINT "↖"
RUN
LIST
```

[Control] [1] : STOPPING THE SCREEN DISPLAY

Now the program looks better, but it is too long for all the lines to appear together on the screen. When the computer lists the program, you can stop the lines as they move up and off the screen by pressing the [Control] and the [1] keys. Type LIST. Use two fingers on your left hand to press the [Control] and [1] keys and one finger on your right hand to press [Return]. [Control] [1] both starts and stops the LIST function.

Creating an Interactive Loop

GOTO, DIM, INPUT

Loops tell the computer to go back and repeat instructions in the program automatically. The GOTO command saves you the trouble of typing the same instruction lines over and over again. The DIM and INPUT commands allow you to interact with your computer on a question-and-answer basis. Putting these three commands together—GOTO, DIM, INPUT—lets you have an ongoing conversation with your computer.

GOTO: A COMPUTER'S MAP

The simplest computer loop is the GOTO loop. GOTO is always followed by a line number that tells the computer where to go on the next command. You need just two commands to create a loop. Type in the program below to produce an infinite loop:

```
NEW
110 PRINT "CONGRATULATIONS!"
120 GOTO 110
RUN
```

To break this infinite loop, turn off the machine or use the Break key. When you stop the loop with the Break key, one of the following messages appears:

```
STOPPED AT LINE 110
```
or
```
STOPPED AT LINE 120
```

The computer is telling you where it was when it received the command to stop.

Comma: A Tab Maker

The GOTO loop puts out an endless amount of work with just two lines of instruction. To make the program fancier, list your program, position the cursor in the space next to the last quotation mark, insert a comma, and press Return . Run the program and watch the special effects:

```
LIST
110 PRINT "CONGRATULATIONS!",
120 GOTO 110
RUN
```

The comma acts like a tab. Each time the computer moves down to the next line and prints CONGRATULATIONS!, it moves to the next tab position. The result is a barber-pole effect. Remember to break the loop with the Break key.

Semicolon: Computer Glue

A semicolon produces another kind of effect. List the program, change the comma in line 110 to a semicolon, press Return , and run the program:

```
LIST
110 PRINT "CONGRATULATIONS! ";
120 GOTO 110
RUN
```

The semicolon glues the PRINT statements together with no space in between. To put some space between the words, go back and edit line 110 so that it looks like this:

```
110 PRINT"CONGRATULATIONS!    ";
RUN
```

Colon: A Separator

The colon is a separator. It permits two instructions to be placed on one line. Change the semicolon in line 110 to a colon and add the PRINT statement below:

```
110PRINT"CONGRATULATIONS!":PRINT"YOU JUST WON THE
LOTTERY."
RUN
```

As you progress in your programming ability, conserving space in the computer's memory becomes important. Consolidating commands on one line with a colon is one way to help save free bytes of RAM memory. To see how much memory is conserved, type the following statement:

```
PRINT FRE (0)
```

The computer will answer with a number. Reprogram line 110 so that the two PRINT statements take up two program lines:

```
110PRINT"CONGRATULATIONS!"
115 PRINT "YOU JUST WON THE LOTTERY."
PRINT FRE (0)
```

Compare the two numbers of free bytes available. The second number is two or three less than the first. Because simplicity is more important to a beginning programmer than conservation of computer memory, the program lines in this section will usually contain only one statement per line. One exception will be a PRINT statement that inserts a blank line between segments of the program. Type in the new line below to see the effect:

```
110 PRINT:PRINT "CONGRATULATIONS!"
RUN
```

DIM AND INPUT: DIMENSIONING AND INPUTTING STRING VARIABLES

The computer must be programmed to respond to a question. You can use a PRINT command to ask a question and an INPUT command to enter a response into the computer. However, when you give the computer an answer, the computer must know where to put it. It places it in a spot called a "variable" in RAM memory. If the answer is composed of letters, numbers, or both, it is called a "string variable." Your ATARI 65XE Computer needs to know how much space you will need for an answer so that it can reserve space for it. This process is called "dimensioning the string variable."

The DIM (dimensioning) command always accompanies the INPUT command for string variables because DIM determines the expected size of the answers. For variables, the size refers to the number of characters, including blanks, that are needed. You have to tell the computer the maximum number of spaces that the answer can occupy.

Change the loop program to a program that asks a question and expects an answer. There is no need to rewrite the program; just write in the new lines—lines 10, 120, 130, and 140—as shown below. (Typing in the new line 120 automatically erases the old line 120.)

```
10 DIM ANSWER$ (100)
110 PRINT: PRINT"CONGRATULATIONS!"
115 PRINT "YOU JUST WON THE LOTTERY."
120 PRINT: PRINT "HOW DOES THAT MAKE YOU FEEL?"
130 INPUT ANSWER$
140 PRINT "I THOUGHT YOU WOULD SAY THAT. "
RUN
```

Line 10 tells the computer to save enough space in its memory for an answer that is a maximum of 100 characters. The variable in this program has been named ANSWER. The variable is going to store letters and numbers, so it is a string variable. String variables are designated by a dollar sign after the last letter of the variable name.

Line 130 allows you to enter an answer. When you run the program, the computer displays the question on the screen, and you then type in your answer. That answer is stored in the string variable called ANSWER$. If the DIM statement, line 10, was omitted, an Error message would occur, and the INPUT statement wouldn't work.

?: Courtesy of INPUT

Run the program again. Two question marks will appear on the screen. The second question mark will be on the line next to the left margin. List your program and notice that you typed only one question mark in the program. The INPUT command always puts a question mark on the screen for you. Type the variation of line 120 below:

```
120 PRINT  HOW DOES THAT MAKE YOU FEEL" ;
```

Run the program and type in your response when the computer asks its question. Now only one question mark appears, and your answer immediately follows the question on the same line. Create some more dialogue by dimensioning more string variables and inserting more INPUT statements. The DIM statements should be at the beginning of the program:

```
20  DIM  DATE$  (25)
140 PRINT:PRINT "WHEN WOULD YOU LIKE TO COME AND PICK UP
YOUR  PRIZE";
150 INPUT DATE$
RUN
```

String Variables in PRINT Statements

The computer program now asks two questions but doesn't respond to your last answer. To get a response, you can place the string variable in the PRINT statement in the following way:

```
160 PRINT "I'M SORRY BUT, OUR OFFICES ARE ALWAYS CLOSED
ON ";DATE$;" . TOO BAD! "
```

The semicolon glues the string variable between two phrases in quotation marks. Run the program. If the words are not spaced correctly, compare your line to the line above. You probably left out a space after the N of ON or forgot the period and the space before TOO BAD!. Those spaces are important. Practice with another string variable input:

```
30 DIM NAME$ (1)
170 PRINT "BY THE WAY.. WHAT IS YOUR NAME";
180 INPUT NAME$
190 PRINT "WELL, "; NAME$;", I BET YOU WOULD LIKE TO KNOW
HOW MUCH YOU WON . FIRST YOU HAVE TO ANSWER A QUESTION . "
```

Run the program. Even though you typed in a full name, the computer printed only the first initial. That happened because the area dimensioned in RAM memory for the name was too small. Most people's names are longer than one character. Change line 30 to a more reasonable number of spaces and run the program:

```
30 DIM NAME$ (25)
RUN
```

Inputting Numeric Variables

So far you have been working with alphanumeric string variables-variables composed of letters, numbers, or both. For instance, the computer would accept the name R2-D2 or 007 as a string variable. However, the number name would be used only as a name, not as a number in any math problems. Now try some numeric variables that can be used in mathematical calculations. Numeric variables do not need a DIM command or a dollar sign. Enter the following program lines:

```
200 PRINT:PRINT "HOW OLD ARE YOU";
210 INPUT AGE
220 PRIZE=AGE*1000
230 PRINT : PRINT "YOU HAVE JUST WON $" ; PRIZE ;" FROM THE
LOTTERY. YOU CAN COLLECT DURING OFFICE HOURS."
```

In this program, the age that you enter is stored in the numeric variable called AGE. Line 220 creates another variable called PRIZE. Line 220' allows the computer's built-in calculator to calculate the prize money, which is $1000 multiplied by the age of the winner. (To the computer, * means multiply.) The program does the math for you and stores the answer in PRIZE. Line 230, which places the numeric value inside the PRINT statement in the same manner as string variables, tells you what the answer is.

INPUT LOOPS

To repeat your conversation with the computer, add a loop command to the program again. A GOTO statement at the end will make the computer repeat the program from the beginning. For program readability, use a REM statement to show where the main conversation portion of the program begins. A REM (remark) statement functions like a label for the programmer. The computer does not carry out REM commands but only prints them when you list your program.

```
100 REM *** CONVERSATION LOOP ***
240 GOTO 100
```

The computer must return to line 100, rather than line 10, because it cannot go back over the DIM statements for string variables. If it loops over the same DIM statements, you will receive an Error message.

Using Random Numbers and Mathematical Functions:

RND, +,-, *,/

Initially computers were developed to process numbers quickly and easily. To take advantage of the computer's ability to calculate a math answer in a few milliseconds, you must know how to speak to a computer.

NUMBERS

Type the statement below and press [Return] :

```
PRINT 10
```

The computer should print the number 10. Make sure you use the numerals 1 and 0, not letters. Practice printing the following numbers:

```
PRINT1000000000
PRINT-100000000
```

Use the minus sign (-) on the Up Arrow key to indicate negative numbers. Do not use commas in numbers. Type the statements below to see what happens when commas are used:

```
PRINT  9,876,543,210
PRINT 9,876,543,210
```

In both examples, the computer interprets the commas as separators in a series of numbers. It spaces the numbers out across the screen according to its preset tab positions. To the computer, the 9 is not 9 billion, just the number 9 followed by a series of other numbers.

SCIENTIFIC NOTATION

The computer may not understand commas when it prints numbers, but it does understand exponents. Often it will automatically translate a large number into an exponential form. Try the numbers below:

```
PRINT 99999999999
PRINT  55555555555
PRINT  11111111111
PRINT  -11111111111
PRINT  -98765432112
```

These numbers are large or small enough that the computer prefers to rewrite them in scientific notation. Familiarity with scientific notation is not essential for understanding the computer, or even this chapter.

Scientific notation expresses a large number as a number between 0 and 10 multiplied by a power of 10. An exponent specifies the power of 10. In the following example, E + 13 means that the exponent is 13:

$2.5E+13 = 2.5 \times 10^{13} = 25000000000000$

You can use exponents to talk to your computer. The caret on the Right Arrow key is the symbol for exponents. You must use the Shift key to print the caret. Try the following computations:

```
PRINT 2 ^ 1
PRINT 2 ^ 2
PRINT 2 ^ 3
PRINT 2 ^ 4
PRINT 2 ^ 64
```

The first notation is 2 to the first power; the second, 2 to the second power; and so on. The last notation is 2 to the sixty-fourth power, which is a large enough number that the computer needs to express it in scientific notation.

Unless you are a physicist timing electrons in their orbits or an astronomer calculating the size of the universe, you will rarely need to use scientific notation. But if you ever do, the computer is capable of doing your calculations with even these often unwieldy numbers.

THE COMPUTER AS A CALCULATOR

The computer can perform the same functions as a calculator. Use the plus (+) sign on the Left Arrow key to type the statement below:

```
PRINT 1 + 1
```

When you press Return, the computer immediately gives you an answer, just like a calculator. Invent your own addition problems now. Make the numbers big or small, and try a long series of numbers to add up. Experiment with lots of variations.

Use the minus sign (-) on the Up Arrow key for subtraction problems. Try the three versions of the same problem below:

```
PRINT 4 - 1
PRINT  4-1
PRINT4-1
```

The same answer appears for each example as soon as you press Return. The spacing in math problems is unimportant to the computer. Try out problems of your own. Make long problems that combine subtraction and addition functions.

The multiplication sign—the asterisk (*)—is located on the Right Arrow key . The division sign is the slash (/) on the Question Mark key . Type the following statements:

```
PRINT 2 * 2
PRINT (2*2)
PRINT 6 / 3
PRINT (6/3)
```

The computer not only understands the use of parentheses in math problems but needs them when the problems become complex. Notice what happens in this problem with and without parentheses:

```
PRINT 3* (2+2)
PRINT 3*2+2
```

The answer to the first problem is 12; the answer to the second problem is 8. In the first problem, the computer first adds 2 and 2, then multiplies by 3 to arrive at 12. In the second problem, the computer multiplies 3 and 2 first, then adds 2 to arrive at 8. Whenever the computer encounters parentheses in a math problem, it does the computa tions inside the parentheses first and then finishes the rest of the calculations.

Try out the problems below to discover some other interesting facts about how your computer works. See if you can predict the answers before you press Return:

```
PRINT (2+2)*3
PRINT 2+2*3
```

In the first problem, the computer does the computation inside the parentheses first. In the second problem, the computer does the multiplication first, then the addition. The computer executes these mathematical functions according to rules of order: first, computations inside parentheses; second, exponential functions; third, multiplication and division functions as they appear in the problem from left to right; and last, addition and subtraction functions from left to right. The rules are summarized in the following table:

Order of Mathematical Execution

1.	( )	Computations in parentheses	
2.	^	Exponential functions	
3.	*	Multiplication	In order of appearance
	/	Division	from left to right
4.	+	Addition	In order of appearance
	−	Subtraction	from left to right

RANDOM NUMBERS

The computer can perform other functions that your calculator most likely cannot do. For example, your computer can pick random numbers for you. Type the program below:

```
NEW
10 PRINT RND (0)
20 GOTO 10
RUN
```

RND is the command for generating random numbers. The infinite loop in the program above will generate random numbers endlessly. Remember to break the loop with the [Break] key. To make changes in the program, you can just list the program and use the cursor keys to insert characters, rather than retype entire lines. Try out the various programs below:

```
10  PRINT RND (1)
RUN
10  PRINT  RND  (123)
RUN
10  PRINT  RND  (50)
RUN
10  PRINT  RND  (50000)
RUN
```

All four variations of line 10 generate random numbers between 0 and 1. The decimal point is always before the first digit in a random number. The few random numbers that have a number on the left side of the decimal point are still between 0 and 1 but are so small that the computer has written them in scientific notation.

The number in the parentheses is called a "dummy variable." It does not matter what number is used as the dummy variable, but it is important that the parentheses appear and that they enclose something (any number or letter). For typing ease, 0 is usually placed in the dummy variable position. Change line 10 again as shown below:

```
10 PRINT (RND(0) * 10)
RUN
10 PRINT (RND(0) * 100)
RUN
10 PRINT (RND(0) * 1000)
RUN
```

Each program generates a different range of random numbers. PRINT (RND(0) * 10) generates numbers up to 10 because the statement instructs the computer to multiply the random number by 10. Multiplying by 10 moves the decimal point over one place. In PRINT (RND(0) * 100), multiplying by 100 moves the decimal point over two places, and in PRINT (RND(0) * 1000), multiplying by 1000 moves the decimal point over three places. If you want, you can multiply by much larger numbers to generate large random numbers.

Because long numbers with many digits after the decimal point are cumbersome, the computer has an instruction that tells it to print only integers. Integers are whole numbers without any decimal points. The instruction INT tells the computer to drop everything after the decimal point. Reprogram the three variations of line 10 above and compare the results:

```
10 PRINT INT (RND(0)*10)
RUN
10 PRINT INT (RND(0)*100)
RUN
10 PRINT INT (RND(0)*1000)
RUN
```

The programs generate numbers in the same ranges as before, but the numbers are more readable without the digits after the decimal.

To generate numbers in a more specific range, try the examples below:

```
10 PRINT INT (RND(0)*3)
RUN
10 PRINT INT (RND(0)*12)
RUN
10 PRINT INT (RND(0)*25)
RUN
```

The program generates random numbers that are always one less than the number by which it is multiplied. The first line 10 generates the numbers 0, 1, and 2. To generate random numbers 0, 1, 2, and 3, the program would be written this way:

```
10 PRINT INT (RND(0)*4)
RUN
```

To generate only the numbers 1, 2, and 3, the program should look like this:

```
10 PRINT INT (RND(0)*3)+1
RUN
```

To generate three numbers starting at 20, write the program this way:

```
10 PRINT INT (RND(0)*3)+20
RUN
```

Random Number Game

Random-number programs are very flexible. You can even use them to play games with your computer. Type the following program. Remember that to get the bent arrow in line 5, press Esc , hold down Shift or Control , and press Clear

```
NEW
1 REM *** NUMBER.GAM ***
5 PRINT "↖"
10 SECRETNUM=INT(RND(0)*3)+1
20 PRINT: PRINT "I AM THINKING OF A NUMBER, EITHER 1, 2,
OR 3. TRY TO GUESS IT."
30 INPUT GUESS
40 IF GUESS=SECRETNUM THEN PRINT "YOU WON."
50 IF GUESS<>SECRETNUM THEN PRINT "YOU LOST."
60 GOTO 10
```

Line 10 assigns the random number to the numeric variable called SECRETNUM. Line 30 lets the user type in a guess and assigns this number to the numeric variable called GUESS. (Remember that numeric variables do not need to be dimensioned or tagged at the end the way that string variables do.) Line 40 compares the guess to the secret number. If they equal each other, the computer prints "YOU WON." Line 50 also compares the guess to the secret number. If they are not equal (the symbols < > mean not equal to), the computer prints "YOU LOST." Line 60 makes a loop so that you can play the game again. (The next chapter explains IF-THEN statements in more detail.)

MATH PROGRAMS

The computer's mathematical functions can be used for work purposes, as well as for play. If you were a chef who prepared food for banquets, you might need a computer to expand your recipes. For instance, suppose that you are trying to figure out how many pounds of sea scallops to buy to serve Coquilles St. Jacques at a dinner for 62 guests. Your recipe indicates that 1 1/2 pounds of scallops feeds 5 people. The program below would tell you how many pounds to buy:

```
NEW
1 REM *** COQUILLE ***
10 PRINT "↖"
20 GUESTS=62
30 POUNDSTOBUY=1.5/5 * GUESTS
40 PRINT:PRINT "BUY";POUNDSTOBUY;"POUNDS OF
SCALLOPS."
50 END
```

The program produces the answer (18.6 pounds of scallops), but a calculator would achieve the same result with less work To make the program more useful, allow a variation in the number of guests by inserting an INPUT statement. Type in the additional lines below:

```
15 PRINT : PRINT "HOW MANY GUESTS DO YOU EXPECT?"
20 INPUT GUESTS
```

Run the program several times, entering a different number of guests each time. The amount of scallops needed changes each time. For 200 guests, 60 pounds of scallops are required; for 436 guests, 130.8 pounds. The INPUT function makes the program more practical.

Making Decisions and Solving Problems: IF-THEN, FOR-NEXT

The IF-THEN and FOR-NEXT commands enable you to write programs that mimic the way humans approach a decision or a problem. Especially useful for games and logic puzzles, the commands let you, the programmer, make the choices for the computer.

IF-THEN COMMANDS

To practice the IF-THEN statement, type in the following program:

```
NEW
1 REM *** BRNPROBE.QZ ***
5 PRINT "↖"
10 DIM RAIN$ (3)
20 PRINT:PRINT "YES OR NO, IF IT HERE RAINING OUTSIDE,
WOULD YOU GO OUT WITH AN UMBRELLA";
30 INPUT RAIN$
40 IF RAIN$="YES" THEN PRINT "YOU HAVE A FORMIDABLE IQ."
50 IF RAIN$="NO" THEN PRINT "YOU ARE A BORN RISK TAKER."
```

The Brainprobe Quiz evaluates your answer. In line 40, if the answer stored in the string variable RAIN$ is yes, the computer prints the IQ message. If the answer is not yes, the computer reads the next line, line 50, and evaluates the string variable RAIN$ again. If the answer is NO, the computer prints the risk-taker message. However, if you answer neither yes nor no, the program just ends. The program has no instructions for responding to an indefinite answer. Try it out.

One way to encourage an expected reply is to create an infinite loop. Insert the additional line below:

```
60 GOTO 20
```

Evaluating with IF-THEN

Another way to encourage a correct answer is to provide hints. The following program uses numeric variables to elicit a correct response:

```
NEW
1 REM *** NUMBER.QZ ***
5 PRINT "↖"
10 SECRETNUM=INT(RND(0)*10)+1
20 PRINT:PRINT "GUESS A SECRET NUMBER BETWEEN 1 AND 10."
```

```
30 PRINT
40 PRINT "YOUR GUESS";
50 INPUT GUESS
60 PRINT
70 IF GUESS=SECRETNUM THEN PRINT "YOU GOT IT!":END
80 IF GUESS<SECRETNUM THEN PRINT "TOO LOW. TRY AGAIN.":
GOTO 40
90 IF GUESS>SECRETNUM THEN PRINT "TOO HIGH. TRY
AGAIN.":GOTO40
```

Lines 80 and 90 evaluate the guess as greater than or less than the secret number. The PRINT statement provides a hint that the next guess should be higher or lower. The GOTO commands in lines 80 and 90 create an infinite loop if you continue to guess incorrectly.

Ending the Program

The Number Quiz is programmed to stop only when you discover the secret number. When you enter the correct answer, line 70 gives the computer the instruction to end. END stops the program, and the Ready prompt appears on your screen.

Trapping Errors

If you accidentally enter a letter instead of a number for GUESS, the computer sends an Error message, and the program ends abruptly. Make an intentional error by typing a letter key or pressing the Return key only. To avoid ending the program, you can use a TRAP command to trap the Error message. Add the lines below and run the program again:

```
45 TRAP 100
100 PRINT:PRINT "PLEASE ENTER A NUMBER ONLY."
110 GOTO 30
```

In line 45, the TRAP command tells the computer not to stop the program when a mistake is entered and sends the computer to line 100. Line 100 tells the computer to print the directions for correcting the mistake. Line 110 returns the computer to the place where it left off. The TRAP statement always comes before the INPUT statement, and it always contains the number of the line that will resolve the problem.

Quiz Writing with IF-THEN

A program can easily provide hints when the correct answer is a number, including a date. The following program uses IF-THEN statements and the TRAP command to evaluate guesses:

```
NEW
1 REM *** LOVELACE . QZ ***
5 PRINT "↖"
10 PRINT:PRINT "ADA LOVELACE, DAUGHTER OF THE POET LORD
BYRON, WAS MATHEMATICALLY BRILLIANT. "
20 PRINT
30 PRINT "IN WHAT YEAR DID SHE WRITE HER AMAZINGLY
ACCURATE DESCRIPTION OF THE FUTURE USES OF THE
COMPUTER";
40 TRAP 200
50 INPUT GUESS
60 IF GUESS=1842 THEN GOTO 100
70 IF GUESS<1842 THEN GOTO 110
80 IF GUESS>1842 THEN GOTO 120
100 PRINT: PRINT "CONGRATULATIONS! YOU GUESSED THE YEAR
CORRECTLY.":END
110 PRINT:PRINT "THAT WAS TOO EARLY. TRY AGAIN.":GOTO 20
120 PRINT:PRINT "THAT WAS TOO LATE. TRY AGAIN.":GOTO 20
200 PRINT:PRINT "PLEASE ENTER A NUMBER ONLY."
210 GOTO 20
```

In the Lovelace Quiz, the placement of the PRINT messages associated with the IF-THEN statements is different from their placement in the Brainprobe Quiz and the Numbers Quiz. This difference illustrates that there is often more than one way to achieve the same results in programming.

Computer Bugs

The TRAP statement makes the Lovelace Quiz more errorproof, but it still is not perfect. Because the computer evaluates the date as a number, it will accept 1842.78 as incorrect but 1842,78 as correct. Most programs have "bugs," or problems. When you can figure out the bugs and fix them, you have really learned to program. Every beginner encounters many bugs and makes many mistakes. To become a better programmer, study this manual, refer to the ATARI BASIC Reference Guide, and perhaps have a more experienced person look over your shoulder occasionally. You will learn how to identify bugs so that you can avoid similar mistakes in future programs.

FOR-NEXT LOOP: THE COUNTING LOOP

You are already familiar with the infinite GOTO loop. Another kind of loop is the FOR-NEXT loop. The FOR-NEXT loop is a counting loop, which is not infinite. Type NEW and enter the following program:

```
NEW
10 FOR X=1 TO 4
20 PRINT "POTATO"
30 NEXT X
RUN
```

POTATO appears on the screen four times. Change line 10 to read like this:

```
10 FOR X=1 TO 7
```

When you run the program this time, the screen shows POTATO seven times. The computer is looping seven times through lines 10, 20, and 30. FOR tells the computer how many times to loop, and NEXT tells the computer to go back to the top and start again. NEXT is similar to GOTO. X is a variable. You can use anything to represent the variable. Try this name for the variable:

```
10 FOR NUM=1 TO 7
30 NEXT NUM
```

When you run the program, there is no difference from the previous program. Change the variable name again:

```
10 FOR JKL=1 TO 7
30 NEXT JKL
```

JKL is a nonsense name for the numeric variable in the FOR-NEXT loop. Run the program to see that it, too, runs the same as before. Now add this line:

```
15 PRINT JKL,
RUN
```

The PRINT statement in line 15 shows the value of the variable. (Put the comma in for readability.) Each time the computer repeats the FOR-NEXT loop, the variable takes on the value of the next number in the series specified in line 10. The first time, the variable is 1; the second time, 2; and so on. The last number in the FOR statement controls the number of times the computer loops through the program. Change that number in line 10 as shown below:

```
10 FOR JKL=1 TO 50
RUN
10 FOR JKL=1 TO 200
RUN
10 FOR JKL=1 TO 500
RUN
```

Starting Point

List the program. The first number in the FOR line is the starting point for the count, and the last number is the stopping point. Even negative numbers can be the starting point for the count. Try these variations for line 10:

```
10 FOR JKL=1 TO 5
RUN
10 FOR JKL=0 TO 5
RUN
10 FOR JKL=3 TO 5
RUN
10 FOR JKL=-10 TO 5
RUN
```

STEP: Counting Incrementally

List the program, delete the PRINT statement in line 20 and the comma in line 15, and run the program. The computer counts and prints the numbers very quickly. Use the STEP command to make the computer count in increments. Try the program below:

```
10 FOR JKL=0 TO 500 STEP 5
RUN
10 FOR JKL=0 TO 500 STEP 2
RUN
10 FOR JKL=0 TO 500 STEP 100
RUN
10 FOR JKL=0 TO 500 STEP 7
RUN
```

The computer will obligingly count by any sequence you specify.

Counting Backward

The computer can count backward if you use the STEP -1 command and the proper sequence of numbers (from larger to smaller) for starting and stopping the count. For example:

```
10 FOR JKL=500 TO 0 STEP -1
RUN
10 FOR JKL=10 TO 0 STEP -1
RUN
10 FOR JKL=-1 TO -19 STEP -1
RUN
```

The computer can count backward in increments also:

```
10 FOR JKL=500 TO 0 STEP -20
RUN
10 FOR JKL=500 TO 0 STEP -3
RUN
10 FOR JKL=0 TO -500 STEP -50
RUN
```

You can also instruct the computer to start and stop at any number you desire:

```
10 FOR JKL= 500 TO 300 STEP -10
RUN
10 FOR JKL=25 TO 0 STEP -1
RUN
```

Now you know how to instruct the computer to count forward and backward, to count consecutively and incrementally, and to start and stop at specified numbers.

The FOR-NEXT "Sandwich" Loop

List your program. FOR is on the top line, and NEXT is on the bottom line. Whatever you want the computer to do is sandwiched in between. Type in the lines below:

```
10 FOR JKL=1 TO 5
20 PRINT "AVOCADO"
```

The computer will carry out any instruction or number of instructions between the FOR and NEXT statements the specified number of times. Have the computer print other words:

```
16 PRINT "CHEESE"
17 PRINT"MAYONNAISE"
18 PRINT"MUSTARD"
19 PRINT"TOMATO"
21 PRINT "BACON BITS"
22PRINT"LETTUCE"
23 PRINT:PRINT
RUN
```

The computer prints and counts too quickly for anyone to read the screen clearly. Nonetheless, it prints the PRINT statement exactly five times as instructed in the FOR-NEXT statement. Other instructions, such as math computations and INPUT statements, can also be part of the FOR-NEXT sandwich loop.

Delay Loops

Erase all the PRINT statements so that absolutely nothing is in the FOR-NEXT sandwich loop, except the FOR and the NEXT statements. Run the program and see what happens:

```
15
16
17
18
19
20
21
22
23
LIST
RUN
```

Nothing happens. Change the number in line 10 and watch carefully again:

```
10 FOR JKL=1 TO 500
RUN
```

The Ready prompt takes a few seconds to appear. Change line 10 again:

```
10 FOR JKL=1 TO 5000
RUN
```

This time the Ready prompt takes considerably longer to appear. The computer is counting but not printing its calculations. The process is similar to counting silently to yourself. The time it takes the Ready prompt to appear on the screen is the time it takes the computer to count to 5000.

FOR-NEXT loops are excellent devices for keeping the computer from moving on. In fact, FOR-NEXT loops are used so frequently for this purpose that they are sometimes called "delay loops," and the common variable name is DELAY. Rewrite the FOR-NEXT loop, using DELAY as the variable name and different numbers in the FOR statement:

```
NEW
10 FOR DELAY=1 TO 300
20 NEXT DELAY
LIST
RUN
```

Sometimes the delay loop is sandwiched on the same program line:

```
NEW
10 FOR DELAY=1 TO 300:NEXT DELAY
LIST
RUN
```

Sample Programs

The programs below use FOR-NEXT loops in a variety of ways. The first program uses the FOR-NEXT loop as a simple delay loop to leave the word HI on the screen long enough to be read before line 30 clears the screen:

```
NEW
1 REM *** DLAYLOOP ***
5 PRINT "↖"
10 PRINT "HI"
20 FOR DELAY=1 TO 800:NEXT DELAY
30 PRINT " ↖"
40 PRINT "BYE"
50 FOR DELAY=1 TO 800 : NEXT DELAY
```

The next program uses a numeric variable in the FOR-NEXT loop. It also uses a TRAP command that refers the computer back to the previous line, giving no specific message about the error:

```
NEW
1 REM *** HOWHIGH? ***
10 DIM A$ (1), HH$(1)
20 PRINT " ↖ "
30 PRINT:PRINT "HOW HIGH DO YOU WANT TO COUNT";
40 TRAP 30
50 INPUT HH
55 HH$=STR$(HH) : IF HH$="0" THEN GOTO 30
60 FOR COUNT=1 TO HH
70 PRINT COUNT
80 NEXT COUNT
90 PRINT :PRINT "PLEASE ANSWER (Y/N) . WOULD YOU LIKE TO
COUNT AGAIN";
100 TRAP 90
110 INPUT A$
120 IF A$="Y" THEN GOTO 30
130 IF A$="N" THEN PRINT:PRINT "BYE":END
140 GOTO 90
```

The last program paraphrases an old rock 'n' roll song and uses "nested" FOR-NEXT loops. A nested FOR-NEXT loop is a smaller delay loop inside a larger FOR-NEXT loop. The program also uses OR to create multiple conditions in the IF-THEN statement:

```
NEW
1 REM *** CLOCKRCK ***
5 PRINT "↖"
10 FOR X=1 TO 9
20 PRINT X;
30 PRINT " O'CLOCK"
40 FOR DELAY=1 TO 500:NEXT DELAY
50 IF X=3 OR X=6 OR X=9 THEN PRINT "ROCK! " : FOR PAUSE=1
TO 500:NEXT PAUSE
60 NEXT X
70 PRINT:PRINT "WE'RE GOING TO ROCK"
80 PRINT "AROUND THE CLOCK"
90 PRINT "TONIGHT!"
```

Producing Sound and Graphics:

SOUND, SETCOLOR, COLOR

Creating sound and graphics on some computers is very complicated, but not on the ATARI 65XE. The SOUND command of ATARI BASIC, combined with some simple programming techniques, is all you need. Sound and graphics add new dimensions to your BASIC programs—anything from arcade-game zaps and cracks, musical themes, and songs to colorful graphic displays.

SOUNDING OFF

Your ATARI 65XE can play up to four sounds at one time. The four sound registers, or voices, are numbered 0, 1, 2, and 3. To select the first voice, you type SOUND 0; for the second, SOUND 1; for the third, SOUND 2; and for the fourth, SOUND 3.

The SOUND command in ATARI BASIC controls four elements:

voice (0-3)
pitch (0-255)
distortion (0-14)
volume (0-15)

The pitch, or frequency, of the sound is determined by a number from 0 to 255, giving you a total of 256 frequencies from which to choose. The pitch value is the second number in the SOUND command. SOUND 1,50 specifies the second voice with a pitch of 50. Make sure that the volume is turned up on your TV or monitor, then type

```
SOUND 1,50,0,8
```

Press Return. A great explosion, isn't it? To turn off the sound, you just turn down the volume on your television, or type either of the commands below and press Return:

```
END
SOUND 1,0,0,0
```

The purity, or distortion, of the sound is determined by any even number between 0 and 14. In the SOUND command, the purity of the sound is the third number. Try this:

```
SOUND 1,50,10,8
```

The number 10 produces a pure tone without distortion. To put in a little distortion, change the 10 to 06:

```
SOUND 1,50,06,8
```

The computer sounds as if it's ready for takeoff. Type END before the neighbors start complaining.

The last number in the SOUND command controls the volume. The number must be between 0 and 15. Number 8 is a good number for most uses. You risk damaging your TV speaker and your ears if you go above 12.

To try some four-part harmony, enter the following:

```
SOUND 0,50,10,8
SOUND l,100,10,8
SOUND 2,150,10,8
SOUND 3,200,10,8
```

Type END to stop the chorus.

Sounding Off with Variables

Variables in SOUND commands add versatility to your programs. Using variables,you can program the computer to change the voice, pitch, distortion, and volume of sustained sounds. Enter and run the following program:

```
NEW
10 REM * SET VARIABLES FOR SOUND VALUES
20VOICE=0:PITCH=100:TONE=8:VOL=8
30 SOUND VOICE,PITCH,TONE,VOL
40 GOTO 20
RUN
```

To stop the sound, press the Break key and type END. To sustain a sound, you need to repeat the SOUND command in the program. Two common methods are a FOR-NEXT loop or a GOTO loop like the one in the example above. The following program uses a variable for the pitch in a FOR-NEXT loop to produce the computer's entire range of pitches:

```
NEW
10 REM * SOUND EFFECTS WITH FOR-NEXT LOOP
20 VOICE=0:PITCH=0:TONE=10:VOL=8
30 FOR PITCH=0 TO 255
40 SOUND VOICE,PITCH,TONE,VOL
50 NEXT PITCH
RUN
```

Varying the volume in a program produces a variety of sounds. Change VOL = 8 to VOL = 0 and press Return. Then add the following line:

```
35 VOL=INT(RND(0)*16)
```

This line randomly selects a value between 0 and 15 for the volume variable. Run the program to find out how randomly changing the volume affects the sound.

Making Music

The SOUND command can produce musical tones as well. The following scale includes musical notes and their pitch values:

Note	Pitch
high C	29
B	31
A	35
G	40
F	45
E	47
D	53
C	60
B	64
A	72
G	81
F	91
E	96
D	108
middleC	121

Type and run the following program:

```
NEW
10 REM ** SIMPLE SONG
15 DIM PITCH$ (1)
20VOICE=0:PITCH=0:TONE=10:VOL=8
30 REM ** C=121:D=108:E=96:F=91
40 TRAP 300
50 PRINT "↖"
```

```
60 PRINT "NOTES FOR SIMPLE SONG"
65 FOR NOTE= 1 TO 8
70 READ PITCH
80 SOUND VOICE,PITCH,TONE,VOL
90 GOSUB 200
100 PRINT:PRINT PITCH$
110 FOR PAUSE=1 TO 500 : NEXT PAUSE
120 SOUND 0,0,0,0
130 NEXT NOTE
140 GOTO 300
150 REM ** DATA FOR NOTES
160 DATA 121,121,108,96,96,91,108,121
200 REM ** PRINT NOTES
210 IF PITCH=121 THEN PITCH$="C"
220 IF PITCH=108 THEN PITCH$="D"
230 IF PITCH=96 THEN PITCH$="E"
240 IF PITCH=91 THEN PITCH$="F"
250 RETURN
300 PRINT: PRINT "END OF SIMPLE SONG":END
RUN
```

The GOSUB-RETURN and READ-DATA commands allow the computer to produce different notes by inserting a series of values for the variable PITCH. GOSUB tells the computer to go to the "subroutine" that starts at line 200 and continues to line 250; the RETURN command sends the computer back to the line immediately below the GOSUB line. The READ command tells the computer to pick up an item in the DATA line and insert it into the variable. The computer continues to loop through the program until all the values in the DATA line have been used.

The program also uses a FOR-NEXT loop to determine how long the notes last. Using different FOR-NEXT loops, try modifying the program to produce whole notes, half notes, and other kinds of notes.

For more information about programming with sound, read some of the books and magazines listed in the Resources section of this guide.

COLORFUL GRAPHICS

Your ATARI 65XE has 16 graphics modes encompassing 128 colors. To get you started, this section presents 6 different modes and some of the most essential graphics commands.

The following chart lists the 16 basic colors and their corresponding number values. (The colors vary somewhat according to the adjustment of the hue control on your television set.)

0	Gray	8	Blue
1	Gold	9	Light blue
2	Orange	10	Turquoise
3	Red-orange	11	Green-blue
4	Pink	12	Green
5	Purple	13	Yellow-green
6	Red-orange	14	Orange-green
7	Blue	15	Light orange

The remaining 112 colors are obtained by adding a value for luminance, or brightness. The luminance must be an even number between 0 and 14. The higher the luminance number, the lighter and brighter the color.

Color registers are another important element in ATARI graphics. The color registers can be thought of as paint cans. Each register can hold any of the 128 colors. Because there are five registers, a maximum of five different colors can be displayed. The five color registers are numbered 0, 1, 2, 3, and 4.

SETCOLOR is an essential graphics command. The format is SETCOLOR 2,10,8: the first number is the color register; the second is the color number; and the third is the luminance.

Graphics Mode 0

The color registers function differently in different graphics modes. Their functions in graphics mode 0 (the text mode) are shown in the chart below:

Default Colors	**Register**	**Function**
	0	Not used
Light blue	1	Brightness of text
Dark blue	2	Background
	3	Not used
Black	4	Border

The default colors are the colors that the computer automatically uses unless you instruct it to use some other colors. Using SETCOLOR to change colors, type in the following:

SETCOLOR 2, 3, 4

When you press Return, the screen turns orange. The color transformation occurs because in the SETCOLOR command, the 2 represents the screen color, the 3 equals the color orange, and the 4 indicates the brightness. Change the 4 to a 6. The orange changes to a lighter orange. Change the 6 to a 7. Nothing happens because only the even numbers between 0 and 14 define the luminance. If you type an odd number, the computer uses the color of the previous even number. Change the 7 to an 8 and watch the color get lighter yet. The following program shows all 128 colors and luminances:

```
NEW
10 REM ** 128 ATARI COLORS
20 REM ** 16 COLORS
30 FOR COLOR =0 TO 15
40 REM ** 8 LUMINANCES
50 FORLUMINANCE=0TO 14 STEP 2
60 SETCOLOR 2,COLOR,LUMINANCE
65 PRINT "COLOR=";COLOR;" LUMINANCE=";LUMINANCE
70 REM ** PAUSE TO SEE COLOR
80 FOR PAUSE=1 TO 600:NEXT PAUSE
90 NEXT LUMINANCE
100 NEXT COLOR
RUN
```

When the luminance reaches number 10, the text disappears because the default luminance of the text is also 10. (The default luminance is the luminance that the computer automatically uses unless it is instructed to do otherwise.) Whenever the background luminance is the same as the text luminance, the text seems to disappear. Pay attention to background and text luminances as you work more with color and luminance in graphics mode 0. Type GR.0 (which is an abbreviation for graphics mode 0) to restore the normal screen colors.

Change SETCOLOR 2 to SETCOLOR 4 in line 60 and run the program again. Because register 4 governs the border, the border changes color this time instead of the background area. Type GR.0 to restore the normal screen colors.

Graphics Modes 1 and 2

Graphics modes 1 and 2 provide large-size text and color options. Graphics mode 2 is identical to graphics mode 1 except that each character is twice as tall. Mode 1 has 24 horizontal screen lines, and mode 2 has 12. To enter graphics mode 1, type

```
NEW
10 GRAPHICS 1
20 PRINT #6;"GRAPHICS MODE ONE"
```

Run the program. Graphics mode 1 is in orange text at the top of the screen. At the bottom is a blue strip containing the word READY. The blue strip is the text window and displays text in graphics mode 0. Type GR.0 to return to the text mode.

To print large text on the screen in graphics modes 1 and 2, use PRINT #6; followed by quotes and then the text that you want to print. This statement is a variation on the PRINT command that you learned earlier.

Now list the program. Change MODE to mode and run the program. MODE turns green. Type LIST 20. Using the Inverse Video key ◩ change mode in line 20 to MODE and run the program. MODE now turns blue. List the line again and change MODE to mode and run the program. Now MODE is red.

Enter and run the following program:

```
NEW
10 REM ** COLORFUL TEXT
20 GRAPHICS 1
30 PRINT #6; "ORANGE"
40 PRINT #6; "green"
50 PRINT #6; "DARK BLUE"
60 PRINT #6; "red"
70 PRINT "COLORFUL TEXT"
RUN
```

As you can see, graphics mode 1 is capable of displaying five colors at the same time—four different text colors and one background color. The colors can also be changed by using SETCOLOR according to the guidelines outlined in the following chart:

Register	Default Color	Character Style	Color#	LUM
0	Orange	Uppercase	2	8
1	Light green	Lowercase	12	10
2	Dark blue	Inverse uppercase	9	4
3	Red	Inverse lowercase	4	6
4	Black	Background	0	0

Type SETCOLOR 4,15,5. Register 4 (the background) changes to a reddish orange. But now the dark blue text is difficult to read. Use SETCOLOR to change it. According to the chart, register 2 controls the dark blue text. SETCOLOR 2,8,6 does the trick by making the dark blue text a little bit lighter. Add the following lines to the Colorful Text program:

```
100 FOR COLOR=0 TO 15
110 SETCOLOR 2,COLOR,8
120 FOR DELAY=1 TO 400:NEXT DELAY
130 NEXT COLOR
```

Run the program. The text window at the bottom of the screen changes color along with the dark blue text because register 2 governs the text window as well as the text display.

Getting Rid of the Text Window

Sometimes you may not want the text window to appear in your programs. To eliminate the text window, simply add 16 to the graphics mode number. Change line 20 to GRAPHICS 17 and delete line 70. The PRINT command will always print in graphics mode 0. If you are in modes 1 or 2, if you don't have a text window, and if you use the PRINT command and the PRINT #6; command, the computer gets confused and prints everything in mode 0. Add this line:

```
70 PRINT " WINDOW TEST"
```

Run the program to see what happens. If you use PRINT and PRINT #6; you must use a text window to have mode 1 show up on the screen.

Delete lines 100, 110, 120, and 130. Run the program. WINDOW TEST and then READY appear at the top of the screen. List the program. Line 20 specifies mode 17 (mode 1 without the text window), but where is it? Replace line 70 with this line:

```
70  GOTO 70
```

When you run the program, the mode 1 screen comes back. When you use mode 1 or 2 without a text window, you must use a GOTO loop to keep the display on the screen or it will flash by too fast to be seen. Pressing the Break key returns you to mode 0.

To see an example of mode 2, list the Colorful Text program and change line 20 to

```
20  GRAPHICS 18
```

Graphics 18 stands for mode 2 plus 16 (no text window). Run the program. Now you have LARGE colorful text.

To return the screen to its original colors, press the Reset button or type SETCOLOR 2,9,4. You will not lose your program when you press Reset in ATARI BASIC. However, that feature may not apply to other languages or programs.

Graphics Mode 3

The graphics mode 3 screen is a grid consisting of 40 columns and 24 rows (20 if you use the text window). Enter and run the following program:

```
NEW
10 GRAPHICS 3
20 COLOR 1
30 PLOT 0,0
RUN
```

In the upper left corner is an orange block. The block, or pixel, is one unit in the graphics screen. The COLOR command determines the color of the pixel. The number after the COLOR command determines which color register to use for the color of the pixel. The COLOR command does not place a color in the register; SETCOLOR does that. The COLOR command simply selects which register to use to plot the pixel, and the pixel becomes whatever color is in the register. To make this clearer, change line 20 to

```
20  COLOR 2
```

Run the program. The orange pixel is now light green. Think of each pixel as a text character. In modes 1 and 2, you used uppercase and lowercase characters and Inverse Video to select the colors of the text. In modes 3 and above, use the COLOR command to select the color for the pixels.

PLOT: Plotting Points on the Grid

PLOT is like the PRINT #6; command except that it prints pixels instead of letters and numbers. COLOR is like the upper/lower/inverse color selection method; it selects the register. The default colors are orange, light green, dark blue, and black. To change the color in any of the registers, use the SETCOLOR command.

The color registers are like four buckets of paint. SETCOLOR selects the color that goes into each of the four buckets, and COLOR selects the bucket into which the paintbrush will be dipped. PLOT determines where the brush will be positioned on the screen.

DRAWTO: Connecting the Dots

Add this line:

```
40  DRAWTO  39,0
```

Run the program. A light green line goes across the top of the screen. After plotting a pixel, use the DRAWTO command to plot a second pixel and draw a connecting line between the two. Line 40 tells the computer to plot a pixel at column 39, row 0, and then connect them. Now type

```
DRAWTO  39,19
```

The command plots a pixel in the bottom right corner of the graphics screen, just above the text window, and then draws a line to connect 39,0 to 39,19. Now type

```
DRAWTO  0,19
```

To complete the rectangle, type

```
DRAWTO  0,0
```

Now type GR.0 and list the program. Add these lines:

```
50 DRAWTO 39,19
60 DRAWTO 0,19
70 DRAWTO 0,0
```

SETCOLOR and COLOR

When you run the program, the computer draws a green rectangle again. To brighten up the screen, type

```
35 COLOR 1
45 COLOR 2
55 COLOR 1
65 COLOR 3
```

Run the program to see a rectangle of many colors.

To change the color in a register, use SETCOLOR. You might conclude that COLOR 1 selects the color for register 1 and that COLOR 2 selects the color for register 2. Unfortunately, that conclusion is not quite true. Mode 3 has four registers and four colors—but the registers are numbered 0, 1, 2, and 4, and the colors are numbered 0, 1, 2, and 3. To keep things straight, make a chart:

Color 0	=	Register 4	Black
Color 1	=	Register 0	Orange
Color 2	=	Register 1	Light green
Color 3	=	Register 2	Dark blue

Type GR.0, list the program, and change COLOR 2 in line 20 to COLOR 1. COLOR 1 selects register 0, and orange is the default color for register 0. To change the color in register 0, use the SETCOLOR command. Add the following line:

```
15 SETCOLOR 0,4,6
```

When you run the program, the orange lines change to a pinkish color. You have changed the color of the lines by using SETCOLOR to change the paint in the bucket (the color in the register), not by using COLOR to choose a different bucket (register). The color luminance of register 0 also affects the luminance of the text in the text window.

Now add

```
42 SETCOLOR 1,2,8
```

The light green at the right side of the box turns gold. Add one more line:

```
62 SETCOLOR 2,11,4
```

Run the program. Not only does the left side of the box change to green, but the text window also turns green. Therefore, register 2 also controls the color of the text window.

Now you should be able to use SETCOLOR and COLOR to achieve a wide variety of colors and hues in your programs.

Graphics Modes 5 and 7

The differences among modes 3, 5, and 7 can be illustrated very easily. Change line 10 to

```
10 GRAPHICS 5
```

Run the program. The rectangle is much smaller because the pixels are smaller. With the text window, the mode 3 grid has 39 columns and 20 rows. The mode 5 grid has 80 columns and 40 rows.

Now change line 10 to

```
10 GRAPHICS 7
```

When you run the program, an even smaller rectangle appears. The grid in mode 7 is 160 columns by 80 rows.

The smaller the pixels, the higher the resolution. Of the three modes, mode 3 is the lowest and mode 7 is the highest. Try drawing a rectangle around the screen borders in modes 5 and 7.

The following program illustrates all that you have tried in this section. Type it in and run it:

```
NEW
5 REM ** BILL'S BOX (PLOT AND DRAW)
10 PRINT "WHICH MODE (3,5, OR 7)";
20 LEFT=0:TOP=0
30 INPUT MODE
40 IF MODE = 3 THEN RIGHT=39:BOTTOM=19
50 IF MODE=5 THEN RIGHT=79:BOTTOM=39
6O IF MODE=7 THEN RIGHT=159 : BOTTOM=79
70 GRAPHICS MODE
80 PRINT" GRAPHICS MODE";MODE
90 FOR COUNT=1 TO 1000
100 COLOR 2
110 TRAP 240
115 REM ** DRAW BOX
120 PLOT LEFT,TOP
130 COLOR 1
140 DRAWTO RIGHT,TOP
150 COLOR 2
160 DRAWTO RIGHT,BOTTOM
170 COLOR 1
180 DRAWTO LEFT,BOTTOM
190 COLOR 3
200 DRAWTO LEFT,TOP
205 REM ** DELAY LOOP
210 FOR DELAY=1 TO 500:NEXT DELAY
215 REM ** SIZE OF NEXT BOX
220 LEFT=LEFT+2:TOP=TOP+2:RIGHT=RIGHT-2:BOTTOM=BOTTOM-2
230 NEXT COUNT
240 PRINT" THAT'S ALL FOLKS!"
250 END
```

Try using SETCOLOR to change the colors in the Bill's Box program.

You can learn a great deal more about ATARI graphics, including how to use other graphics modes and create animated characters. Refer to Appendix F: Resources to locate books, magazines, and users groups that can help you further explore the world of ATARI BASIC and your ATARI 65XE Computer.

Part3
Appendices

A. Sample Programs

Your ATARI Computer can work miracles with a little help from your imagination and the right programming techniques. These sample programs will show off the versatility of your ATARI 65XE and motivate you to try writing some programs yourself.

Just type in each program exactly as written, pressing Return at the end of every line. When you're finished, type the word RUN, press Return, and watch your ATARI Computer come to life.

Note: When spacing in program lines is critical, a note at the bottom of the program will specify the exact number of spaces needed.

THE ATARI CHOO-CHOO

Sound effects are an ATARI specialty. If you close your eyes when you run ATARI Choo-Choo, you might think you're on the Marrakesh Express.

```
10 POKE 765,255:POKE 580,1
20 GRAPHICS 17:POKE 712,148: POSITION 1,10 : PRINT #6;
"THE ATARI CHOO-CHOO"
30 FOR X=15 TO 0 STEP -1-P:SOUND 1,0,0,X
40 R=INT(RND(0)*300)+1
50 IF R=30 THEN SOUND 3,36,10,10: SOUND 2,48,10,
10:GOSUB 90: SOUND 3,0,0,0:SOUND 2,0,0,0
60 NEXT X:P=P+0.03
70 IF P>=5 THEN P=5
80 GOTO 30
90 POKE 77,0:POSITION 8,12: PRINT #6; "toot":FOR A=1 TO
400 : NEXT A : POSITION 8,12 : PRINT #6;"    ": RETURN
```

Note: Line 90 requires four blank spaces between the quotation marks.

THE BIG BANG

Close the door before you run the next program so that you won't disturb the neighbors.

```
10 POKE 764,255:POKE 580,1
20 GRAPHICS 17
30 FOR X=10 TO100:SOUND0,X,10,10:SOUND1,X-2,10,8:
SOUND2,X+2,l0,12:NEXT X
40 SOUND 1,0,0,0:SOUND 2,0,0,0
50 POSITION 4,11: PRINT #6; "BAROOOOMMM!"
60 FOR DECAY=15 TO 0 STEP -0.5:FOR B=1 TO 20:
SOUND 0,100,B,DECAY: POKE 712,B:NEXT B:NEXT DECAY
70 GRAPHICS 1+32:POKE 712,148
80 POKE 752,l:PRINT : PRINT " Press Start to set off
anotherexplosion."
90 IF PEEK(53279)<>6 THEN GOTO 90
100 GOTO 20
```

SORT THOSE WORDS

This sorting program puts words in their proper places—in alphabetic order. Replace the words in the DATA statements in lines 10 and 20 to sort words of your own choosing. Remember to separate each of your words with a comma.

```
10 DATA ATARI,DISK DRIVE, MONITOR,COMPUTER,TOUCH
TABLET,PRINTER,KEYBOORD
20 DATA SOFTWARE,PROGRAM RECORDER,WORD PROCESSING,
ACCOUNTING,DATA BASE,FUN
30 DIM Z$(1000),A(50),A$(20),S(10)
40 S(1)=1:FOR L=1 TO 9: S(L+1)=S(L)*3+1:NEXT L
50 TRAP 80:GRAPHICS 0:? "HERE IS THE LIST:"
60 READ A$:B=LEN(Z$):C=LEN(A$): Z$
(B+1,B+1)=CHR$(C):? A$
70Z$(B+2,B+1+C)=:A$:Q=Q+1:A(Q)=B+1:GOTO60
80 ? :? "READY TO SORT. . .",:P=0
90 P=P+1:IF S(P+2)<Q THEN 90
100 FOR I=P TO 1 STEP -1:S=S(I) : FOR J=S+1 TO
Q:L=J-S:A=A(J): B=A(L)
110 IF Z$(A+1,A+ASC(Z$(A,A)))>Z$ (B+1,B+ASC(Z$(B,B)))
THEN 130
120 A(L+S)=B:L=L-S:IF L>0 THEN B=A(L):GOTO 110
130 A(L+S)=A:NEXT J:NEXT I:? : ? "SORTED."
140 FOR L=1 TO Q:A=A(L) : ? Z$(A+1,A+ASC(Z$(A,A))):NEXT L
```

PLAYERS AND MISSILES

This program uses a technique called Player Missile Graphics to create a pink monster that moves across your screen in front of a blue vertical bar. If you want to make the monster scoot behind the blue bar, simply change line 150 to **150 POKE 623,4.**

```
10 POKE 764,255:POKE 580,1
20 GRAPHICS 3+16
30 FOR X=16 TO 24 : FOR Y=0 TO 23 : COLOR 3 : PLOT X, Y : NEXT
Y:NEXT X
40 MEMTOP=PEEK(741)+256*PEEK(742)-1
50PMBASE=INT((MEMTOP-1024)/1024)*1024
60 ADJTOP=PMBASE+384
70 POKE 742,INT (ADJTOP/256) :
POKE 741,ADJTOP-256*PEEK(742)
80 POKE 54279,PMBASE/256
90 POKE 53277,2
100 POKE 559,34+8
110 P0=PMBASE+512
120 FOR A=P0 TO P0+128:POKE A,0: NEXT A
130 FOR A=P0+60 TO P0+67:READ B: POKE A,B:NEXT A
140 POKE 53256,3
150 POKE 623,1
160 POKE 704,108
170 POKE 53248,PEEK(20) :GOTO 170
180 DATA 60,126,129,153,255,36,66,129
```

TOPSY-TURVY

When you run Topsy-Turvy, your screen will be filled with strange writing. To straighten it out, simply press Start. To mess things up again, press Select

```
10 POKE 764,255:POKE 580,1
20 GRAPHICS 18:POKE 712,128: POKE 755,5
30 POSITION 5,3:PRINT #6; "WELCOME TO"
40 POSITION 2,5:PRINT #6; "THE TOPSY-TURVY":POSITION
6,7:PRINT #6; "WORLD OF": POSITION 6,9
50 PRINT #6;"COMPUTERS"
60 IF PEEK(53279)=5 THEN POKE 755,5:POKE 712,128
70 IF PEEK(53279)=6 THEN POKE 755,1:POKE 712,99
80 GOTO 60
```

TYPE-A-TUNE

This program assigns musical note values to the keys on the top row of the keyboard. Press only one key at a time.

KEY	MUSICAL VALUE
Insert	B
Clear	B♭ (or A#)
0	A
9	A♭ (or G#)
8	G
7	F#(or G♭)
6	F
5	E
4	E♭ (or D#)
3	D
2	D♭ (or C#)
1	C

```
10  DIM  CHORD(37),TUNE(12)
20  GRAPHICS 0:?  :?  "      TYPE-A-TUNE PROGRAM"
25 ? :? "PRESS KEYS 1-9,0,<,> TO PRODUCE NOTES. ";
27 ? : ? "RELEASE ONE KEY BEFORE PRESSING THE   NEXT  . "
28 ? : ? "OTHERWISE, THERE MAY BE A DELAY . "
30  FOR X=1 TO 37:READA:CHORD(X)=A:NEXTX
40  FOR X=1 TO 12:READA:TUNE(X)=A:NEXTX
50  OPEN   #1,4,0,   "K: "
55  OLDCHR=-1
60  A=PEEK(764)  :IF A=255 THEN 60
63  IF  A=OLDCHR  THEN  100
65  OLDCHR=A
70   FOR  X=1  TO  12:IF  TUNE(X)=A  THEN  SOUND
0,CHORD(X),10,8:GOTO   100
80  NEXT X
100  I=INT(PEEK(53775)/4):IF  (I/2)=INT(I/2)  THEN 60
110  POKE  764,255:SOUND  0,0,0,0:OLDCHR=-1  :GOTO 60
200  DATA 243,230,217,204,193,182,173,162,153,144,136,
128,121,114,108,102,96,91,85,81,76,72,68,64,60
210 DATA 57,53,50,47,45,42,40,37,35,33,31,29
220 DATA 31,30,26,24,29,27,51,53,48,50,54,55
```

To play "Mary Had a Little Lamb," press the following keys:

5, 3, 1, 3, 5, 5, 5   3, 3, 3   5, 8, 8   5, 3, 1, 3, 5, 5, 5   5, 3, 3, 5, 3, 1

Note: Make sure you insert three spaces between THE and NEXT in line 27.

HIGHER MATH

Your ATARI Computer is a fancy calculator. When you enter two numbers into the program below, the computer will tell you their greatest common denominator. For example, if you enter 690911 and 11214017, you'll soon discover that their greatest common denominator is 53147.

```
10 ? CHR$(125) :? "Enter two numbers. Press Return after
eachentry."
20INPUTN1,N2
30 GOSUB 90
40 ? "Their GCD is "; :? AN
50 POKE 752,1:POSITION 10,10: ? "Press Start to
continue."
60 IF PEEK(53279)<>6 THEN GOTO 60
78 POKE 752,0:? CHR$(125) :GOTO 10
80 REM ****SUBROUTINE****
90 AN=0:POKE 195,0:TRAP 130: M=(N1>=N2)*N1+(N2>N1)*N2:
N=(M=N1)*N2+(M=N2)*N1
100 IF INT(N1)<>N1 OR INT (N2)<>N2 THEN RETURN
110P=M-INT(M/N)*N:M=N:N=P
120 IF P<>0 THEN GOTO 110
130AN=M*(PEEK(195)=0):RETURN
```

COMPUTER BLUES

This program generates random musical notes to "write" some very interesting melodies for the programmed bass.

```
1 GRAPHICS 0:? :? "  COMPUTER BLUES":?
2 PTR=1
3 THNOT=1
5 CHORD=1
6 PRINT "BASS TEMPO (1=FAST)";
7 INPUT TEMPO
8 GRAPHICS 2+16:GOSUB 2000
10 DIM BASE(3,4)
20 DIM LOW(3)
25 DIM LINE(16)
26 DIM  JAM(3,7)
30 FOR X=1 TO 3
40 FOR Y=1 TO 4
50 READ A:BASE(X,Y)=A
60 NEXT Y
70 NEXT X
80 FOR X=1 TO 3:READ A:LOW(X)=A
90 NEXT X
95 FOR X=1 TO 16:READ A:LINE(X)=A:NEXT X
96 FOR X=1 TO 3
97 FOR Y=1 TO 7
98 READA:JAM(X,Y)=A:NEXTY:NEXT X
100 GOSUB 500
110 T=T+1
115 GOSUB 200
120 GOTO 100
200 REM PROCESS HIGH STUFF
205 IF RND(0)<0.25 THEN RETURN
210 IF RND(0)<0.5 THEN 250
220 NT=NT+1
230 IF NT>7 THEN NT=7
240 GOTO 260
250  NT=NT-1
255 IF NT<1 THEN NT=1
260 SOUND 2,JAM(CHORD,NT),10,NT*2
280 RETURN
500 REM PROCESS BASE STUFF
510 IF BASS=1 THEN 700
520 BDUR=BDUR+1
530 IF BDUR<>TEMPO THEN 535
531  BASS=1:BDUR=0
535 SOUND 0,LOW(CHORD),10,4
540SOUND1,BASE(CHORD,THNOT),10,4
```

```
550 RETURN
700 SOUND 0,0,0,0
710 SOUND 1,0,0,0
720 BDUR=BDUR+1
730 IF BDUR<>1 THEN 800
740 BDUR=0:BASS=0
750 THNOT=THNOT+1
760 IF THNOT<>5 THEN 800
765 THNOT=1
770 PTR=PTR+1
780 IF PTR=17 THEN PTR=1
790 CHORD=LINE(PTR)
800 RETURN
1000 DATA 162,144,136,144,121,108,102,108,108,96,91,96
1010 DATA 243,182,162
1020 DATA 1,1,1,1,2,2,2,2,1,1,1,1,3,2,1,1
1030 DATA 60,50,47,42,40,33,29
1040 DATA 60,50,45,42,40,33,29
1050 DATA 81,68,64,57,53,45,40
2000 PRINT#6:PRINT#6:PRINT#6
2005 PRINT #6 ;" Computer"
2006 PRINT#6
2010 PRINT  #6;"  Blues"
2030 RETURN
```

UNITED STATES FLAG

This program involves switching colors to set up the stripes. It uses graphics mode 7 plus 16 so that the display appears as a full screen. Note the correspondence of the COLOR statements with the SETCOLOR statements. For fun and experimentation purposes, add a SOUND statement and use a READ/DATA combination to add "The Star Spangled Banner" after line 470.

```
10 REM DRAW THE UNITED STATES FLAG
20 REM HIGH RESOLUTION 4-COLOR GRAPHICS, NO TEXT WINDOW
30 GRAPHICS 7+16
40 REM SETCOLOR 0 CORRESPONDS TO COLOR 1
50 SETCOLOR 0,4,4:RED=1
60 REM SETCOLOR 1 CORRESPONDS TO COLOR 2
70 SETCOLOR 1,0,14:WHITE=2
80 REM SETCOLOR 2 CORRESPONDS TO COLOR 3
90 BLUE=3:REM DEFAULTS TO BLUE
100 REM DRAW 13 RED & WHITE STRIPES
110 C=RED
120 FOR 1=0 TO 12
130 COLOR C
140 REM EACH STRIPE HAS SEVERAL HORIZONTAL LINES
150 FOR J=0 TO 6
160 PLOT 0,I*7+J
170  DRAWTO  159,I*7+J
180 NEXT J
190 REM SWITCH COLORS
200 C=C+1:IF C>WHITE THEN C=RED
210 NEXT I
300 REM DRAW BLUE RECTANGLE
310 COLOR BLUE
320 FOR I=0 TO 48
330 PLOT 0,I
340  DRAWTO  79,I
350 NEXT I
360 REM DRAW 9 ROWS OF WHITE STARS
370 COLOR WHITE
380 K=0:REM START WITH ROW OF 6 STARS
390 FOR I=0 TO 8
395 Y=4+I*5
400 FOR J=0 TO 4:REM 5 STARS IN A ROW
410  X=K+5+J*14:GOSUB  1000
420 NEXT J
430 IF K<>0 THEN K=0:GOTO 470
```

```
440 REM ADD 6TH STAR EVERY OTHER LINE
450 X=5+5*14:GOSUB 1000
460 K=7
470 NEXT I
500 REM IF KEY HIT THEN STOP
510 IF PEEK(764)=255 THEN 510
515 REM OPEN TEXT WINDOW WITHOUT CLEARING SCREEN
520 GRAPHICS 7+32
525 REM CHANGE COLORS BACK
530 SETCOLOR 0,4,4:SETCOLOR l,0,14
550 STOP
1000 REM DRAW 1 STAR CENTERED AT X,Y
1010 PLOT X-1,Y:DRAWTO X+1,Y
1020 PLOT H,Y-l:PLOT X,Y+1
1030 RETURN
```

IGPAY ATINLAY

This short program converts words or sentences into pig Latin. One word of caution, though; don't enter any one-letter words like A or I.

```
10 DIM A$(256):S=2
20 ? "Type in a word or sentence. Please don't exceed
three lines of text."
30 INPUT A$
40 FOR X=1 TO LEN(A$)
50 IF A$(X,X)=CHR$(32) THEN PRINT
A$(S,X-1);A$(S-1,S-1);"AY";"  ";:S=X+2
66 IF X=LEN(A$) THEN PRINT A$(S,X); A$(S-1,S-1); "AY"
70 NEXT X
80 ? :? :? "THAT'S ALL FOLKS!"
```

GRAPHEEK

Just type this one in and watch the graphics action.

```
10 DIM A$(35)
20 GRAPHICS 1
25 TRAP 90
30 A$="THIS IS A GRAPHICS DEMONSTRATION."
40 FOR I=1 TO 33:? #6;A$(I,I);
50 S=PEEK(53770)
60 SOUND 0,S,10,14
70 FOR DELAY=0 TO 100:NEXT DELAY
80 NEXT I
90 SOUND 0,0,0,0:END
```

Note: Make sure you insert two spaces between GRAPHICS and DEMONSTRATION in line 30.

ESREVER

The title of this program is simply the word REVERSE printed in reverse. To print words spelled backward, just type in this short program. After you run it, a question mark will appear on your screen. Enter a word or a short sentence and let your ATARI 65XE do all the work.

```
10 DIM A$(180)
20 PRINT "Enter a word or short sentence and press
Return."
30 INPUT A$
40 FOR X=LEN(A$) TO 1 STEP -1
50 PRINT A$(X,X);
60 NEXT X
70 PRINT :PRINT :GOTO 20
```

PROTECTING YOUR PROGRAM

Ever wonder how you could protect your programs from prying eyes and quick fingers? A couple of programming tips can help keep pilferers out of your programs.

First type in this program:

```
10 FOR X=1 TO 50:POKE 710,X: NEXT X:GOTO 10
```

To protect the program, add another program line to disable the Break key. This line prevents someone from breaking into the program and listing it while it's running. Also, if you design a program that requires keyboard entry, disabling the Break key protects against "finger slip," that dreaded mishap when your finger accidentally hits the Break key and brings your program to a screeching halt.

Delete GOTO 10 from the colorful program and add this line:

```
20 POKE 16,64:POKE 53774,64:GOTO 10
```

Now run your new program and try to stop it by pressing the Break key. You can't get into it.

To be effective, the POKE statements must be inserted in your program after each graphics mode command.

Disabling the Break key has its limitations. Some smart programmer will figure out that he or she can break into your program and list it by simply pressing the Reset key. To foil this culprit, add this line to your program:

```
5   POKE   580,1
```

Now when the inquisitive intruder presses Reset, the flashing colors program is purged from the computer's memory—no program, no listing! The POKE statement should always be at the beginning of your program.

SEA GULL OVER OCEAN

This program combines graphics and sounds. The sounds are not "pure" sounds; they simulate the roar of the ocean and the gull's cry. To get the symbols in line 20, use [Control] [G] [Control] [F] [Control] [R] [Control] [R]

```
10 DIM BIRD$(4)
20 BIRD$="\/--"
30 FLAG=1:ROW=10:COL=10
40 GRAPHICS 1:POKE 756,226:POKE 752,1
50 SETCOLOR 0,0,0:SETCOLOR 1,8,14
60 PRINT  #6;"  the ocean"
70 R=INT(RND(0)*11)
80 POSITION 17,17
90 FOR T=0 TO 10
100 SOUND 0,T,8,4
110 FOR A=1 TO 50 : NEXT A
120 IF RND(0)>0.8 THEN FOR D=10 TO 5 STEP -1:SOUND
1,0,10,INT(RND(0)*10) :NEXT D: SOUND 1,0,0,0
130 GOSUB 200
140 NEXT T
150 FOR T=10 TO 0 STEP -1
160 SOUND 0,T,8,4
170 FOR A=1 TO 50 : NEXT A
175 IF RND(0)>0.8 THEN FOR D=10 TO 5 STEP
-1:SOUND 1,D,10,8:NEXT D:SOUND 1,0,0,0
180 FOR H=1 TO 10:NEXT H
185 GOSUB 200
190 NEXT T
195 GOTO 70
200 GOSUB 300
210 POSITION COL,ROW
220 PRINT #6;BIRD$(FLAG,FLAG+1)
230 FLAG=FLAG+2:IF FLAG=5 THEN FLAG=1
240 RETURN
300 IF RND(0)>0.5 THEN RETURN
310 POSITION COL,ROW
320 PRINT  #6;"  "
330  A=INT(RND(0)*3)-1
340  B=INT(RND(0)*3)-1
350  ROW=ROW+A
360 IF ROW=0 THEN ROW=1
370 IF ROW=20 THEN ROW=19
380 COL=COL+B
390 IF COL=0 THEN COL=1
400 IF COL>18 THEN COL=18
410 RETURN
```

Note: Two spaces are required between the quotation marks in line 320.

KINETIC ART

Put colors in motion with a program that creates a rainbow of continually moving lines.

```
10 REM KINETIC ART BY NEIL HARRIS
20 GRAPHICS 10
30 DIM A(3,50)
35 FOR L=0 TO 3:FOR M=0 TO 50:A(L,M)=0:NEXT M:NEXT L
40 HUE=INT(RND(1)*8+1):POKE
704+HUE,INT(RND(1)*8)*16+INT(RND(1)*4+4)
50 X1=INT(RND(1)*80):X2=INT(RND(1)*80)
 :Y1=INT(RND(1)*192):Y2=INT(RND(1)*192)
60 COLOR 0:PLOT A(0,WHICH),A(1,WHICH) :DRAWTO
A(2,WHICH),A(3,WHICH)
70  BOUNCE=BOUNCE-1:IF  BOUNCE>0  THEN 90
80 BOUNCE=INT(RND(1)*10+10):BX1=INT(RND(1)*9-4)
 :BX2=INT(RND(1)*9-4):BY1=INT(RND(1)*13-6)
 :BY2=INT(RND(1)*13-6)
90 CHANGE=CHANGE-1:IF CHANGE>0 THEN 110
100   CHANGE=INT(RND(1)*10+5):HUE=INT(RND(1)*8+1):POKE
704+HUE,INT(RND(1)*256)
110 COLOR HUE:PLOT X1,Y1:DRAWTO X2,Y2
120 A(0,WHICH)=X1:A(1,WHICH)=Yl:A(2,WHICH)=X2:A
(3,WHICH)=Y2
130  WHICH=WHICH+1:IF  WHICH>50  THEN WHICH=0
140 X1=X1+BX1:IF X1<0 OR X1>79 THEN BX1=-BX1:GOTO 140
150 X2=X2+BX2:IF X2<0 OR X2>79 THEN BX2=-BX2:GOTO 150
160 Yl=Yl+BY1:IF Y1<0 OR Y1>191 THEN BY1=-BY1:GOTO 160
170 Y2=Y2+BY2:IF Y2<0 OR Y2>191 THEN BY2=-BY2:GOTO 170
180 GOTO 60
```

B. BASIC Reserved Words

Note: The period is mandatory after all abbreviated keywords.

RESERVED WORD	**ABBREVIATION**	**BRIEF SUMMARY OF BASIC STATEMENTS**
ABS		Returns the absolute (unsigned) value of the variable or expression.
ADR		Returns the memory address of a string variable.
AND		Functions as a logical operator. The expression is true only if both subexpressions joined by AND are true.
ASC		Returns the numeric value of a single string character.
ATN		Returns the arctangent of a number or expression in radians or degrees.
BYE	B.	Exits from BASIC and returns to the resident operating system or console processor.
CLOAD	CLOA.	Loads data from the program recorder into RAM.
CHR$		Returns a single string byte equivalent to a numeric value between 0 and 255 in ATASCII code.
CLOG		Returns the base 10 logarithm of an expression.
CLOSE	CL.	Closes a file at the conclusion of I/O operations. Functions as an I/O command.
CLR		Performs the opposite function of DIM: undimensions all strings and matrices.

COLOR	C.	Chooses the color register to be used in color graphics work.
COM		Performs the same function as DIM.
CONT	CON.	Stands for "continue." Causes a program to restart execution on the next line after being stopped by the Break key or encountering STOP.
COS		Returns the cosine of the variable or expression in degrees or radians.
CSAVE		Outputs data from RAM to the program recorder for tape storage.
DATA	D.	As part of the READ-DATA combination, identifies the succeeding items (which must be separated by commas) as individual data items.
DEG	DE.	Tells the computer to perform trigonometric functions in degrees instead of radians. (The default measurement is in radians.)
DIM	DI.	Reserves the specified amount of memory for matrix, array, and string variables. (All string variables, arrays, and matrices must be dimensioned with a DIM statement.)
DOS	DO.	Stands for "Disk Operating System." Causes the menu to be displayed. (See DOS manual.)
DRAWTO	DR.	Draws a straight line between a plotted point and a specified point.
END		Stops program execution; closes files; turns off sounds. May be used more than once in a program. (CONT can be used to restart the program.)
ENTER	E.	Stores data or program in untokenized (source) form. Functions as an I/O command.

EXP		Returns e(2.7182818)raised to a specified power.
FOR	F.	Used with NEXT to establish FOR-NEXT loops. Introduces the range that the loop variable will operate in during the execution of the loop.
FRE		Returns the amount of remaining user memory in bytes.
GET	GE.	Used mostly with disk operations to input a single byte of data.
GOSUB	GOS.	Branches to a subroutine beginning at a specified line number.
GOTO	G.	Branches unconditionally to a specified line number.
GRAPHICS	GR.	Specifies one of the eight graphics modes. (GR.0 can be used to clear the screen.)
IF		Causes conditional branching or the execution of another statement on the same line (only if the first expression is true).
INPUT	I.	Causes the computer to ask for input from the keyboard. Execution continues only when the Return key is pressed after data has been inputted.
INT		Returns the next lowest whole integer below a specified value. (Rounding is always downward, even when the number is negative.)
LEN		Returns the length of the specified string in bytes or characters. (One byte contains one character.)
LET	LE.	Assigns a value to a specific variable name. (LET is optional in ATARI BASIC and can be omitted.)

LIST	L.	Displays or otherwise outputs the program list.
LOAD	LO.	Inputs from a disk into the computer.
LOCATE	LOC.	Stores in a specified variable the value that controls a specified graphics point.
LOG		Returns the natural logarithm of a number.
LPRINT	LP.	Commands the line printer to print a specified message.
NEW		Erases all contents of user RAM.
NEXT	N.	Causes a FOR-NEXT loop to terminate or continue, depending on the particular variables or expressions. (All loops are executed at least once.)
NOT		Returns a 1 only if the expression is not true; returns a 0 if it is true.
NOTE	NO.	Used only in disk operations. (See DOS manual.)
ON		Used with GOTO or GOSUB for branching purposes. (Multiple branches to different line numbers are possible, depending on the value of the ON variable or expression.)
OPEN	0.	Opens the specified file for input or output operations.
OR		Used as a logical operator between two expressions. If either one is true, a 1 is evaluated; if both are false, a 0 results.
PADDLE		Returns the position of the paddle game controller.

PEEK		Returns the decimal form of the contents of a specified memory location (RAM or ROM).
PLOT	PL.	Plots a single point at a specified X,Y location.
POINT	P.	Used with disk operations only.
POKE	POK.	Inserts the specified byte into the specified memory location. (May be used only with RAM. If you try to poke ROM, you'll get an Error message.)
POP		Removes the loop variable from the GOSUB stack. Used when departure from the loop is made in an other-than-normal manner.
POSITION	POS.	Sets the cursor at a specified screen position.
PRINT	PR. or ?	Causes output from the computer to the specified output device. Functions as an I/O command.
PTRIG		Returns the status of the trigger button on a game controller.
PUT	PU.	Causes output of a single byte of data from the computer to the specified device.
RAD		Tells the computer to give information in radians, rather than in degrees, for trigonometric functions. (The default measurement is radians. See DEG.)
READ	REA.	Reads the items in the DATA list and assigns them to specified variables.
REM	R.	Stands for "remarks." Does nothing but allows comments to be printed in the program list for the programmer's future reference. REM statements are not executed.

RESTORE	RES.	Allows data to be read more than once.
RETURN	RET.	Returns the computer from a sub-routine to the statement immediately following the one in which GOSUB appears.
RND		Returns a random number between 0 and 1, but never 1.
RUN	RU.	Executes the program; sets normal variables to 0; undims arrays and strings.
SAVE	S.	Causes data and programs to be recorded on disk under the filespec provided with SAVE. Functions as an I/O command.
SETCOLOR	SE.	Stores hue and luminance color data in a particular color register.
SGN		Returns + 1 if the value is positive, 0 if zero, -1 if negative.
SIN		Returns the trigonometric sine of a given value in degrees or radians.
SOUND	SO.	Controls register, pitch, distortion, and volume of a tone or note.
SQR		Returns the square root of a specified value.
STATUS	ST.	Calls status routine for a specified device.
STEP		Used with FOR-NEXT. Determines the quantity to be skipped between each pair of loop variable values.
STICK		Returns the position of the stick game controller.
STRIG		Returns 1 if stick trigger button is not pressed, 0 if pressed.

STOP	STO.	Causes the program to stop but does not close files or turn off sounds.
STR$		Returns a character string equal to the numeric value given. (For example, STR$(65) returns 65 as a string.)
THEN		Used with IF. If the expression is true, the THEN statements are executed. If the expression is false, control passes to the next line.
TO		Used with FOR, as in "FOR X = 1 TO 10." Separates the loop range expressions.
TRAP	T.	Takes control of the program in case of an INPUT error and directs execution to a specified line number.
USR		Returns the results of a machine-language subroutine.
VAL		Returns the equivalent numeric value of a string.
XIO	X.	Used with disk operations (see DOS manual) and in graphics work. Functions as a general I/O statement.

C.ATASCII Character Set

Decimal Code	Hexadecimal Code	ATASCII Character	Keystrokes	European Character
0	0	♥	Control ,	á
1	1	├	Control A	ù
2	2	▐	Control B	Ñ
3	3	┘	Control C	É
4	4	┤	Control D	Ç
5	5	┐	Control E	ô
6	6	╱	Control F	ò
7	7	╲	Control G	ì
8	8	◢	Control H	£
9	9	▗	Control I	ï
10	A	◣	Control J	ü
11	B	▝	Control K	ä
12	C	▘	Control L	Ö
13	D	▔	Control M	ú
14	E	▁	Control N	ó
15	F	▖	Control O	ö
16	10	♣	Control P	Ü
17	11	┌	Control Q	â

Notes:

1. ATASCII stands for ATARI ASCII. Letters and numbers have the same values as those in ASCII, but some of the special characters are different.
2. Except as shown, the characters from 128 to 255 are the reverse colors of 1 to 127.
3. Add 32 to the uppercase code to get the lowercase code for the same letter.
4. To get the ATASCII code, tell the computer (direct mode) to PRINT ASC ("________"). Fill the blank with a letter or a character. You must use the quotes!
5. The normal display keycaps are shown as white symbols on a black background; the inverse keycap symbols are shown as black symbols on a white background.

Decimal Code	Hexadecimal Code	ATASCII Character	Keystrokes	European Character
18	12	─	Control R	û
19	13	┼	Control S	î
20	14	●	Control T	é
21	15	▄	Control U	è
22	16	▏	Control V	ñ
23	17	┬	Control W	ê
24	18	┴	Control X	å
25	19	▌	Control Y	à
26	1A	└	Control Z	Å
27	1B	␛	Esc Esc	
28	1C	↑	Esc Control -	
29	1D	↓	Esc Control =	
30	1E	←	Esc Control +	
31	1F	→	Esc Control *	
32	20		Space bar	
33	21	!	Shift 1	
34	22	"	Shift 2	
35	23	#	Shift 3	
36	24	$	Shift 4	
37	25	%	Shift 5	
38	26	&	Shift 6	
39	27	'	Shift 7	
40	28	(	Shift 9	
41	29	)	Shift 0	
42	2A	*	*	
43	2B	+	+	
44	2C	,	,	
45	2D	-	-	
46	2E	.	.	

Decimal Code	Hexadecimal Code	ATASCII Character	Keystrokes	European Character
47	2F	/	/	
48	30	0	0	
49	31	1	1	
50	32	2	2	
51	33	3	3	
52	34	4	4	
53	35	5	5	
54	36	6	6	
55	37	7	7	
56	38	8	8	
57	39	9	9	
58	3A	:	Shift ;	
59	3B	;	;	
60	3C	<	<	
61	3D	=	=	
62	3E	>	>	
63	3F	?	Shift /	
64	40	@	Shift 8	
65	41	A	A	
66	42	B	B	
67	43	C	C	
68	44	D	D	
69	45	E	E	
70	46	F	F	
71	47	G	G	
72	48	H	H	
73	49	I	I	
74	4A	J	J	
75	4B	K	K	

Decimal Code	Hexadecimal Code	ATASCII Character	Keystrokes	European Character
76	4C	L	L	
77	4D	M	M	
78	4E	N	N	
79	4F	O	O	
80	50	P	P	
81	51	Q	Q	
82	52	R	R	
83	53	S	S	
84	54	T	T	
85	55	U	U	
86	56	V	V	
87	57	W	W	
88	58	X	X	
89	59	Y	Y	
90	5A	Z	Z	
91	5B	[	Shift ,	
92	5C	\	Shift +	
93	5D	]	Shift .	
94	5E	^	Shift *	
95	5F	_	Shift -	
96	60	♦	Control .	i
97	61	a	a	
98	62	b	b	
99	63	c	c	
100	64	d	d	
101	65	e	e	
102	66	f	f	
103	67	g	9	
104	68	h	h	
105	69	i	i	

Decimal Code	Hexadecimal Code	ATASCII Character	Keystrokes	European Character
106	6A	j	j	
107	6B	k	k	
108	6C	l	l	
109	6D	m	m	
110	6E	n	n	
111	6F	o	o	
112	70	p	p	
113	71	q	q	
114	72	r	r	
115	73	s	s	
116	74	t	t	
117	75	u	u	
118	76	v	v	
119	77	w	w	
120	78	x	x	
121	79	y	y	
122	7A	z	z	
123	7B	♣	Control ;	Ä
124	7C	|	Shift =	
125	7D	↖	Esc Control < or Esc Shift <	
126	7E	◀	Esc Delete Bk Sp	
127	7F	▶	Esc Tab	
128	80	♥	◩ Control ,	
129	81	┣	◩ Control A	
130	82	▐	◩ Control B	
131	83	┛	◩ Control C	
132	84	┫	◩ Control D	
133	85	┓	◩ Control E	
134	86	╱	◩ Control F	

Decimal Code	Hexadecimal Code	ATASCII Character	Keystrokes	European Character
135	87	╲	◩ Control G	
136	88	◢	◩ Control H	
137	89	▗	◩ Control I	
138	8A	◣	◩ Control J	
139	8B	▝	◩ Control K	
140	8C	▘	◩ Control L	
141	8D	▔	◩ Control M	
142	8E	▁	◩ Control N	
143	8F	▖	◩ Control O	
144	90	♣	◩ Control P	
145	91	┌	◩ Control Q	
146	92	─	◩ Control R	
147	93	┼	◩ Control S	
148	94	●	◩ Control T	
149	95	▄	◩ Control U	
150	96	▎	◩ Control V	
151	97	┬	◩ Control W	
152	98	┴	◩ Control X	
153	99	▌	◩ Control Y	
154	9A	└	◩ Control Z	
155	9B	EOL	◩ Return	
156	9C	↑	◩ Esc Shift Delete Bk Sp	
157	9D	↓	◩ Esc Shift >	
158	9E	←	◩ Esc Control Tab	
159	9F	→	◩ Esc Shift Tab	
160	A0	█	◩ Space bar	
161	A1	!	◩ Shift 1	
162	A2	"	◩ Shift 2	
163	A3	#	◩ Shift 3	

Decimal Code	Hexadecimal Code	ATASCII Character	Keystrokes	European Character
164	A4	$	◩ Shift 4	
165	A5	%	◩ Shift 5	
166	A6	&	◩ Shift 6	
167	A7	'	◩ Shift 7	
168	A8	(	◩ Shift 9	
169	A9	)	◩ Shift 0	
170	AA	*	◩ *	
171	AB	+	◩ +	
172	AC	,	◩ ,	
173	AD	-	◩ -	
174	AE	.	◩ .	
175	AF	/	◩ /	
176	B0	0	◩ 0	
177	B1	1	◩ 1	
178	B2	2	◩ 2	
179	B3	3	◩ 3	
180	B4	4	◩ 4	
181	B5	5	◩ 5	
182	B6	6	◩ 6	
183	B7	7	◩ 7	
184	B8	8	◩ 8	
185	B9	9	◩ 9	
186	BA	:	◩ Shift ;	
187	BB	;	◩ ;	
188	BC	<	◩ <	
189	BD	=	◩ =	
190	BE	>	◩ >	
191	BF	?	◩ Shift /	
192	C0	@	◩ Shift 8	
193	C1	A	◩ A	

Decimal Code	Hexadecimal Code	ATASCII Character	Keystrokes	European Character
194	C2	B	◩ B	
195	C3	C	◩ C	
196	C4	D	◩ D	
197	C5	E	◩ E	
198	C6	F	◩ F	
199	C7	G	◩ G	
200	C8	H	◩ H	
201	C9	I	◩ I	
202	CA	J	◩ J	
203	CB	K	◩ K	
204	CC	L	◩ L	
205	CD	M	◩ M	
206	CE	N	◩ N	
207	CF	O	◩ O	
208	D0	P	◩ P	
209	D1	Q	◩ Q	
210	D2	R	◩ R	
211	D3	S	◩ S	
212	D4	T	◩ T	
213	D5	U	◩ U	
214	D6	V	◩ V	
215	D7	W	◩ W	
216	D8	X	◩ X	
217	D9	Y	◩ Y	
218	DA	Z	◩ Z	
219	DB	[	◩ Shift ,	
220	DC	\	◩ Shift +	
221	DD	]	◩ Shift .	
222	DE	^	◩ Shift *	
223	DF	_	◩ Shift -	

Decimal Code	Hexadecimal Code	ATASCII Character	Keystrokes	European Character
224	E0	♦	◩ Control	
225	E1	a	◩ a	
226	E2	b	◩ b	
227	E3	c	◩ c	
228	E4	d	◩ d	
229	E5	e	◩ e	
230	E6	f	◩ f	
231	E7	g	◩ g	
232	E8	h	◩ h	
233	E9	i	◩ i	
234	EA	j	◩ j	
235	EB	k	◩ k	
236	EC	l	◩ l	
237	ED	m	◩ m	
238	EE	n	◩ n	
239	EF	o	◩ o	
240	F0	p	◩ p	
241	F1	q	◩ q	
242	F2	r	◩ r	
243	F3	s	◩ s	
244	F4	t	◩ t	
245	F5	u	◩ u	
246	F6	v	◩ v	
247	F7	w	◩ w	
248	F8	x	◩ x	
249	F9	y	◩ y	
250	FA	z	◩ z	
251	FB	♠	◩ Control ;	
252	FC	|	◩ Shift =	
253	FD	↰	Esc Control 2	

Decimal Code	Hexadecimal Code	ATASCII Character	Keystrokes	European Character
254	FE	◀	◩ Esc Control Delete Bk Sp	
255	FF	▶	◩ Esc Control >	

D. Error Messages

ERROR CODE NUMBER	ERROR CODE MESSAGE
2	**Insufficient Memory:** Not enough RAM memory is left to store the statement or the new variable name, or to dimension a new string variable.
3	**Value Error:** A value expected to be a positive integer is negative; a value is not within a specific range.
4	**Too Many Variables:** The maximum of 128 different variable names has been exceeded.
5	**String Length Error:** The user attempted to store string variables that exceeded the dimensioned string length.
6	**Out of Data:** The READ statement requires more data items than the DATA statement(s) supplied.
7	**Line Number Greater Than 32767:** The line number reference is greater than 32767.
8	**Input Statement Error:** The user attempted to input a nonnumeric value into a numeric variable.
9	**Array or String DIM Error:** The DIM size exceeded 5460 for numeric arrays or 32767 for strings; an array or string was redimensioned; reference was made to an undimensioned array or string.
11	**Floating Point Overflow/Underflow:** The user attempted to divide by zero or to refer to a number larger than 1 x 10^{98} or smaller than 1 x 10^{-99}.
12	**Line Not Found:** A GOSUB, GOTO, or THEN referenced a nonexistent line number.
13	**No Matching FOR Statement:** A NEXT was encountered without a previous FOR, or nested FOR/NEXT statements do not match properly.

(The Error message is reported at the NEXT statement, not at FOR.)

14 **Line Length Error:** The statement is too complex or too long for BASIC to handle.

15 **GOSUB or FOR Line Deleted:** A RETURN or NEXT statement was encountered, but the corresponding GOSUB or FOR has been deleted since the last RUN command.

16 **RETURN Error:** A RETURN was encountered without a matching GOSUB.

17 **Syntax Error:** The computer encountered a line with improper syntax.

18 **Invalid String Character:** The string in the VAL statement is not a numeric string.

Note: The following errors are INPUT/OUTPUT (I/O) errors that result during the use of disk drives, printers, or other accessory devices. Further information is provided with the auxiliary hardware.

19 **LOAD Program Too Long:** Insufficient memory remains to complete LOAD.

20 **Device Number Error:** The device number is larger than 7 or equal to 0.

21 **LOAD File Error:** The user attempted to load a non-load file, not a BASIC tokenized file. Tokenized files are created with the SAVE command.

128 **Break Abort:** The user hit the Break key during an I/O operation.

129 **IOCB**[1] **Already Open:** The Input/Output Control Block is already open.

130 **Nonexistent Device:** The user tried to access an undefined device (i.e., a device not in the handler table).

131 **IOCB**[1] **Write-Only Error:** A READ command has been sent to a write-only device (printer).

132	**Invalid Command:** The command is invalid for this device.
133	**Device or File Not Open:** No OPEN command has been specified for the device.
134	**Bad IOCB[1] Number:** The device number is illegal.
135	**IOCB[1] Read-Only Error:** A WRITE command has been sent to a read-only device.
136	**EOF:** The computer has reached the end of the file.
137	**Truncated Record:** This error typically occurs when the record being read is larger than the maximum record size specified in the call to CIO. (BASIC's maximum record size is 119 bytes.)
138	**Device Time-out:** The device doesn't respond.
139	**Device NAK:** Problems are located at the serial port or in the peripheral.
140	**Serial Bus Input Framing Error:** Information was lost from the peripheral to the computer.
141	**Cursor Out of Range:** The cursor is out of range for a particular mode.
142	**Serial Bus Data Frame Overrun:** Information was lost from the peripheral to the computer.
143	**Serial Bus Data Frame Checksum Error:** Information was lost from the peripheral to the computer.
144	**Device Done Error:** The user attempted to write on a write-protected diskette.
145	**Read After Write Compare Error:** The user tried to open the Screen Editor with an illegal graphics mode number.
146	**Function Not Implemented:** The function was not implemented in the handler.
147	**Insufficient RAM:** Not enough RAM memory is left for operating the selected graphics mode.

160	**Drive Number Error:** The user specified the wrong drive number.
161	**Too Many OPEN Files:** No sector buffer is available.
162	**Disk Full:** No free sectors are available.
163	**Unrecoverable System I/O Error:** The DOS version on disk may be damaged.
164	**File Number Mismatch:** The disk file may be damaged.
165	**File Name Error:** The file specification has illegal characters in it.
166	**POINT Data Length Error:** The second parameter of the POINT statement is too large.
167	**File Locked:** The user tried to access a locked file for purposes other than to read it.
168	**Invalid Command:** The command in a special operation code is invalid.
169	**Directory Full:** The user has used all the open space (64 file names) allotted for the directory.
170	**File Not Found:** The user tried to access a file that doesn't exist in the diskette directory.
171	**Invalid POINT:** The user tried to POINT to a byte in a file not opened for update.
172	**Illegal Append:** The user tried to use DOS II to open a DOS I file for append. DOS II cannot append to DOS I files. Using DOS II, copy the DOS I file to a DOS II diskette.
173	**Bad Sectors at Format Time:** The disk drive found bad sectors while it was formatting a diskette. Use another diskette because a diskette with bad sectors cannot be formatted. If this error occurs with more than one diskette, the disk drive may need repair.

[1]IOCB refers to Input/Output Control Block. The device number is the same as the IOCB number.

E. Troubleshooting

Q. I have set up my computer according to the instructions in the Hooking It Up section, but when I turn the computer on, nothing happens. What do I do now?

A. If the word READY doesn't appear when you turn on your computer, retrace your steps. Make sure that all cords and cables are plugged in securely and that power is coming into the system. Try adjusting the fine-tuning knob on your TV. If you still don't get the proper display or if the words MEMORY TEST appear on the TV screen, your computer may need service. For the location of the nearest ATARI Service center, contact your ATARI retailer or ATARI Customer Relations, P.O. Box 61657, Sunnyvale, CA 94088.

Q. What do I do if the computer is on but the TV picture is distorted?

A. There are a number of things you should check:

• Make sure that the cable is plugged into the TV Switch Box and that the Switch Box selector is turned to COMPUTER or GAME.

• The TV must receive a 300-OHM (300Ω) signal from the Switch Box to work properly with the computer. See if you have followed the instructions in the section Installing the TV Switch Box.

• Make sure that the Channel switch on the computer is turned to the same number as the channel selector on your TV set.

• Try a different cartridge in the Cartridge slot to determine whether the software is working. Make sure that the cartridge is pressed firmly into the slot.

Q. When I press the [Help] key, nothing happens. Why not?

A. [Help] is designed to work with specific programs. It gives you access to helpful information when you need assistance. If the computer doesn't respond, the program that you are using is probably not set up for use with [Help]

Q. I just tried the computer's built-in ATARI BASIC. I typed in a program, but it won't run. Why not?

A. Make sure you pressed Return after each program statement before typing RUN. When you press Return, you are telling the computer that you are entering information.

Another common mistake is to confuse zeros and capital letter O's. Though they look similar, the computer treats them differently.

F. Resources

ATARI Computer users don't have to work or play in isolation. Most of the problems that you might encounter have already been worked out by others. Much of this experience is documented, so plenty of support is available. The resources listed here are a few of the many aids that can guide you into new, rewarding directions in home computing. You can obtain the resources by visiting bookstores and ATARI Computer retailers or by writing to the addresses provided below.

USERS GROUPS

You can share information with other ATARI Computer owners by joining an ATARI users group. Users groups usually have monthly meetings and publish a newsletter. Whether you need help in programming one of several computer languages, advice about purchasing software, or news about the latest products for your ATARI 65XE, you can find it and more from your local users group. To get the address of the group nearest you or to find out how to start you own group, write ATARI Customer Relations, P.O. Box 61657, Sunnyvale, CA 94088.

BOOKS

ATARI BASIC Reference Manual. Part # C015307. Available from ATARI Customer Relations, P.O. Box 61657, Sunnyvale, CA 94088. Cost: $10.95 plus $2.50 for shipping. California residents add 6.5 percent tax. This manual is the most complete reference on ATARI BASIC. It features a complete description of every command and function and gives examples.

Inside ATARI BASIC: A Fast, Fun, and Friendly Approach by Bill Carris. Reston Publishing Co., 11480 Sunset Hill Road, Reston, VA 22090. This excellent book for beginners picks up where the *ATARI 65XE Owner's Manual* leaves off.

Your ATARI Computer by Lon Poole, Martin McNift, and Steven Cooke. Osborne/McGraw-Hill, 630 Bancroft Way, Berkeley, CA 94710. This general reference has a major emphasis on intermediate-to advanced-level programming in ATARI BASIC.

Dr. Wacko's Miracle Guide to Designing and Programming Your Own Arcade Games by David Heller, Addison-Wesley, Reading, MA 01867. This step-by-step tutorial for writing arcade-style games in ATARI BASIC is entertaining and educational.

MAGAZINE

ATARI EXPLORER. Every issue of the *ATARI EXPLORER* is designed to help you get more out of your ATARI Computer. Subscribe immediately and save 50 cents off the cover price—6 issues for $15.00. Or save 25 percent by subscribing for 18 issues for $39.95. Send a check or a money order to *ATARI EXPLORER,* P.O. Box 3427, Dept. X, Sunnyvale, CA 94088-3427.

CUSTOMER RELATIONS

For questions or problems regarding ATARI products, write to ATARI Customer Relations, P.O. Box 61657, Sunnyvale, CA 94088 or call (408)745-4851.

G.ATARI 65XE Specifications

Processor:	6502C Microprocessor, clock speed 1.79 MHz
Custom Chips:	GTIA chip—graphics display POKEY chip—sound generator and control ANTIC chip—screen and input/output ports FREDDY chip—memory system control
Memory:	64K RAM 24K ROM (operating system plus ATARI BASIC programming language)
Display:	11graphicsmodes 256 colors (128 displayable at one time) 320 x 192, highest graphics resolution 40 columns x 24 lines text display 5 text modes
Sound:	4 independent sound voices 3 1/2 octave range
Keyboard:	Full-stroke design 62 keys, including Help key and 4 special function keys International character set 29 graphics keys
Programming Features:	Built-in ATARI BASIC programming language Software compatibility with ATARI 800XL Computers
Input/Output:	Software cartridge slot TV output Monitor output 2 controller ports Serial I/O connector port

Index

A

addition function, 48
alphanumeric variables, 45
arrow keys, 15, 31, 47, 48, 49
ATARI BASIC (see BASIC)
Audio-Visual Test, 10
auto repeat, 29

B

BASIC
  built-in programming
    language, 21-22, 23
  disabling the language, 12, 21
bent arrow, 40
blank lines, 32
  in programs, 38
blank spaces, 31-32
Break key, 11, 14, 34, 41, 66, 72
bugs, 57
BYE, 9, 12, 21
byte, 42-43

C

Caps key, 16, 17, 29-30, 31
caret, 48
caring for your computer, 8
cartridges
  loading, 11-12, 24
  software, 23
Clear key, 31, 40
clearing the computer's memory, 35
clearing the screen, 31, 40
colon, 42
COLOR, 73, 74-75
color registers, 69
  changing colors in, 71, 73, 74-75
  designating, 73, 75
colors, 68-69
comma
  in numbers, 47
  with PRINT statements, 41-42
commands
  COLOR, 73, 74-75
  consolidating on one line, 42, 62
  DIM, 43, 44, 45, 46
  DRAWTO, 73-74
  END, 56, 65, 66
  FOR-NEXT, 57-63, 66, 68
  FRE, 42-43
  GOSUB-RETURN, 68
  GOTO, 41, 46, 55, 56, 66, 72
  IF-THEN, 52, 55-57
  INPUT, 43, 44, 53, 56
  INT, 51
  LIST, 35, 36
  NEW, 35
  PLOT, 73
  POKE, 19
  PRINT, 37, 38, 39, 40, 41, 42, 43,
    44, 71, 72
  PRINT#6;, 71, 72, 73
  READ-DATA, 68
  REM, 46
  RND, 50-52
  RUN, 36
  SETCOLOR, 69, 71, 72, 73, 74, 75
  SOUND, 65, 66
  STEP, 59-60
  TRAP, 56, 62, 68
compatibility, 23
Control key, 13, 15
  for graphic symbols, 17, 30-31
  for international characters, 19
  with arrow keys, 15, 31
  with Caps key, 16, 17, 31
  with Clear key, 31, 40
  with Delete Back Space key, 14, 32
  with Insert key, 15, 31
  with Tab key, 33
  with 1 key, 14, 40
  with 2 key, 15
  with 3 key, 15
counting loop, 57-63
cursor control, 31

D

DATA (see READ)
default color, 69, 71, 73, 75
default luminance, 70
delay loop, 61-63
Delete Back Space key, 14, 15, 32
deleting
  lines, 32
  program lines, 38
  spaces, 32
DIM (dimensioning), 43, 44, 45, 46
disk drive, 23, 24, 25
Disk Operating System, 12, 21
display screen, 25
distortion, 65-66
division sign, 49
Down Arrow key, 15, 31
DRAWTO, 73-74
dummy variable, 51

E

END, 56, 65, 66
erase
  computer's memory, 35
  program lines, 38
  screen, 31
  spaces and lines, 32
Error message, 29, 35, 37-38, 44, 46, 56, 68
Escape key, 14, 34, 40
exponent, 47-48

F

FOR, 58
FOR-NEXT, 57-63
  nested FOR-NEXT loops, 62
  with SOUND, 66, 68
FRE, 42-43
frequency, 65

G

garbage error, 37
GOSUB-RETURN, 68
GOTO, 41, 46, 55, 56
  to maintain graphics screen display, 72
  with SOUND, 66
graphic symbols, 17, 30-31, 40
graphics capabilities, 68-76
graphics modes, 68
  mode 0, 69, 72
  mode 1, 70-72, 73
  mode 2, 70-72, 73
  mode 3, 73-75
  mode 5, 75-76
  mode 7, 75-76

H

Help key, 9, 11, 14, 34

I

IF-THEN, 52, 55-57
increments in counting loops, 59-60
infinite loops, 41, 50, 55, 56
INPUT, 43, 44, 53, 56
inserting
  blank lines, 32
  blank program lines, 38
  blank spaces, 31-32
  program lines, 36
Insert key, 15, 31, 32
installation of your computer, 5-8
instruction line
  limit, 29, 39
  numbering, 35, 36
INT, 51
international characters, 19
Inverse Video, 33, 40
  to change colors in graphics modes, 71, 73
  to test keyboard, 11

J

joystick, 26

K

Keyboard Test, 11
keys, descriptions of, 13-16

L

Left Arrow key, 15, 31, 48
line breaks, 39
line numbering, 35, 36
LIST, 35, 36
logical line, 39
loop
  counting, 57-63
  delay, 61-63
  FOR-NEXT, 57-63, 66, 68
  GOTO, 41, 46, 55, 56, 66, 72
  infinite, 41, 50, 55, 56
lowercase, 29-30
  to change colors in graphics modes, 71, 73
luminance, 69, 70

M

math functions
  addition, 48
  division, 49
  multiplication, 49
  order of execution, 49-50
  subtraction, 48
math programs, 53
memory, 9-10, 23, 35, 36, 42-43
Memory Test, 9-10, 11
minus sign, 47, 48
modem, 26
monitor, 25
multiplication sign, 49
musical notes, 67-68

N

nested FOR-NEXT loops, 62
NEW, 35
NEXT, 58
  (see FOR-NEXT)
numbers, 47
numeric variables, 45, 52, 55, 58

O

Operating System, 23
Option key, 9, 12, 14, 21, 34
order of mathematical functions, 49 -50

P

parentheses
  for order of mathematical
    functions, 49-50
  with RND, 51
peripherals, 23, 24, 25, 26
pitch, 65, 67
pixel, 73
PLOT, 73
plus sign, 48
POKE, 19
PRINT, 37
  abbreviation for (?), 39
  in graphics modes, 71, 72
  to clear screen, 40
  to create blank lines, 38
  two statements on one line, 42-43
  with colon, 42
  with comma, 41-42
  with graphic symbols, 40
  with semicolon, 42
  with string variables, 44
PRINT#6;, 71, 72, 73
printer, 26
program recorder, 23, 24, 26

Q

question mark, 44
  abbreviation for PRINT, 39
Question Mark key, 49
quotation marks
  to clear screen, 40
  with PRINT, 37, 40

R

RAM, 9, 10, 36, 42, 43
random numbers, 50-52
READ-DATA, 68
register
  color, 69, 71, 73, 74, 75
  sound, 65
REM, 46
Reset key, 9, 11, 14, 34, 72
RETURN (see GOSUB)
Return key, 16, 35
Right Arrow key, 15, 31, 48, 49
RND, 50-52
ROM, 9, 10, 23
RUN, 36

S

scientific notation, 47-48
screen display
  format, 39
  maintaining a graphics display, 72
  stopping the LIST display, 40
Select key, 9, 14, 34
Self Test, 9-11
semicolon, 42, 44
SETCOLOR, 69, 71, 72, 73, 74-75
Shift key, 13, 15, 30
  with Caps key, 16
  with Clear key, 31, 40
  with Delete Back Space key, 15, 32
  with Insert key, 15, 32
  with Right Arrow key, 48
  with Tab key, 33
software
  audiocassettes, 23, 24
  built-in, 23
  cartridges, 11-12, 23, 24
  commercial, 23
  compatibility, 23
  diskettes, 23, 24
  storing, 24, 25, 26
  user-written, 21, 24
SOUND, 65-66
sound capabilities, 65-68
Start key, 9, 14, 34
starting point in counting loops, 59
STEP, 59-60
stopping point in counting loops, 59
stopping the screen display, 40
storing software. 24, 25, 26
string variables, 43, 44, 45, 55
subroutine, 68
subtraction function, 48

T

Tab key, 33
  clear tabs, 33
  set tabs, 33
test programs, 9-11
text mode, 69, 70
text window, 70, 72
THEN (see IF-THEN)
TRAP, 56, 62, 68

U

Up Arrow key, 15, 31, 47, 48
uppercase, 29-30
  to change colors in graphics
    modes, 71, 73

V

variables, 43
  dummy, 51
  in SOUND commands, 66-67, 68
  numeric, 45, 52, 55, 58
  string, 43, 44, 45, 55
voice, 65
volume, 65, 66, 67

TRAVEL BEYOND THE LEADING EDGE OF COMPUTING

Subscribe NOW to the ATARI® EXPLORER™
and save 50$ off the cover price.

Congratulations. You've just purchased the best personal computer on the market at the very best price. Now what are you going to do with it? Program? Write papers? Manage data? Play games? Maybe all of the above?

You'll want to dig into the latest news and information about your equipment, explore new ways to use your system, and discover how to expand it.

Naturally, the best place to get information is from the source. That's us. *Atari Explorer* magazine.

Every issue is filled with educational articles, reviews, tutorials, and tips to help you and your family get the most from your computer. The *Atari Explorer* is an adventure into the world of computers for business, education, and entertainment.

You can learn programming techniques for beginners and experts. Read in-depth reviews of the hottest new hardware and software. Get inside tips and strategies for winning your favorite video games. We'll even teach you how to make your own arcade games with tutorials from our own game-designing experts.

The *Explorer* features a section of do-it-yourself games and programs for parent-child interaction. You'll also find practical articles on the adult world of money-management, business, and home applications to help you with every-day life.

And it doesn't just cover the serious stuff. We also feature the human side of computing—real people using their computers for some very interesting projects.

The *Explorer* will intrigue, excite, and even pique your curiosity with world-wide coverage of Atari computing. So drop in and take a look around. You'll like what you read.

DETACH AND MAIL TODAY

YES, enter my subscription to the ATARI EXPLORER:

☐ 6 issues...$15.00* ☐ 18issues...$39.95* SAVE25%

☐ Payment Enclosed ☐ Visa ☐ MasterCard

Name____________________________________

Address____________________________________

City____________________________________

State__________________ Zip __________________

Acct. No. ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Expiration Date____________

Signature____________________________________

*This rate limited to USA and its Possessions. Canadian subscriptions, add $5 for postage and handling. Overseas add $10.00/16issues; $30.00/18 issues. Payments due in US funds.

MC01

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO.7283 SUNNYVALE,CA

POSTAGE WILL BE PAID BY ADDRESSEE

ATARI EXPLORER
P.O. BOX 3427
SUNNYVALE, CA 94088-3427

ATARI®

Printed in Taiwan
C072018-001 REV. B
K.I.2.1987