A BYTE BOOK

>>$9.95

# HOW TO EXCEL ON YOUR ATARI 600XL AND 800XL

Timothy O. Knight

# HOW TO EXCEL ON YOUR ATARI 600XL AND 800XL

# HOW TO EXCEL ON YOUR ATARI 600XL AND 800XL

**Timothy O. Knight**

**McGRAW-HILL BOOK COMPANY**

New York St. Louis San Francisco Auckland Bogotá
Guatemala Hamburg Johannesburg Lisbon London Madrid
Mexico Montreal New Delhi Panama Paris San Juan São Paulo
Singapore Sydney Tokyo Toronto

The author has carefully reviewed the programs provided in this book to ensure their performance in accordance with the specifications described. Neither the author nor McGraw-Hill, however, makes any warranties whatever concerning the programs. They assume no responsibility or liability of any kind for errors in the programs or for the consequences of any such errors.



A BYTE Book

1 2 3 4 5 6 7 8 9 SEM SEM 8 7 6 5 4
ISBN 0-07-035104-X

LIBRARY OF CONGRESS CATALOGING IN PUBLICATION DATA

Knight, Timothy O.
How to excel on your Atari 600XL and 800XL.

Includes index.
1. Atari 600XL (Computer)—Programming. 2. Atari 800XL (Computer)—Programming. 3. Basic (Computer program language) I. Title.
QA76.8.A813K58 1984 001.64 84-7153
ISBN 0-07-035104-X

Editor: John A. Aliano
Editing Supervisor: Margery Luhrs
Designer: Richard Glassman

*To my good friend Darren LaBatt,*
*whose innovative ideas and*
*hard work have helped*
*both of us in our endeavors.*

# Contents

# Preface

When I first received my Atari 600XL and 800XL home computers, I was eager to begin programming; however, there was practically no documentation with the machines. There was a user's guide, but it was more like an advertisement than a manual, and the "Basic Reference Guide" listed the keywords but only explained briefly what they did. No real, comprehensive manual accompanied the computers, so the machines I had purchased, evidently, had a lot of features, but I didn't know how to access them.

Fortunately, I've been using personal computers for about four years; in fact I've written software, books, and taught classes about computers. With my prior knowledge, I was able to read the Basic Reference Guide booklet and figure out how the machines worked. However, I knew that almost everybody who bought either the 600XL or 800XL model would be confused unless they knew a lot about computers.

Of course, anyone who bought the computer *could* go out, as Atari recommends, and purchase books about the Atari 400 or 800 computers whose BASIC language is similar to that of the 600XL and 800XL models. Still, those books weren't written specifically for the new XL computer line, and most of them are

already outdated. (Many new technologies and applications for computers have come along during the years since the Atari 400 and 800 computers were introduced.) For this reason, I decided there was a need for a book about how the Atari 600XL and 800XL computers work. I knew the book should cover the details of the Atari's version of the BASIC language, explain some of the uses for computers, and discuss ways of expanding your system.

The book you are holding is the final product of my idea, and it contains information on just about everything you need to know about computers, but specifically the Atari 600XL and 800XL computers. It can be used as the manual for these machines; however, it contains a lot of additional bonus information which you don't ordinarily find in computer company manuals.

I hope that this book will not only teach you about your computer's language, uses, and potential, but will also encourage you to find out more about computers and the impact they will have on you, your work, and your world in the coming years.

*Timothy Knight*

## Chapter One

# Getting Started with the Atari

Ever since the personal computer "revolution" began around 1977, these amazing machines have been increasing in quality and decreasing in price. This seeming paradox is due to mass production, more advanced high technology, and competition among computer companies.

When home computers were first introduced, they had crude black-and-white graphics, no sound, a limited amount of software and memory, and were expensive both to purchase and to maintain. Today, you can purchase a home computer like the Atari 600XL or 800XL (Fig. 1-1), together with a printer, a disk drive, and a high-quality color monitor, for about $1000. These powerful, inexpensive, and versatile home computers have incredible 256-color graphics, four-voice sounds and music, extensive software (over 1000 programs), and enough memory and power to support many programs which will be useful to you, your family, and your friends. In addition, they are easy and inexpensive to maintain.

This book will tell you how to make the most of your Atari 600XL or 800XL by showing you the following:

1. *How to effectively program your computer in BASIC.* Programming is simply instructing the computer to carry out

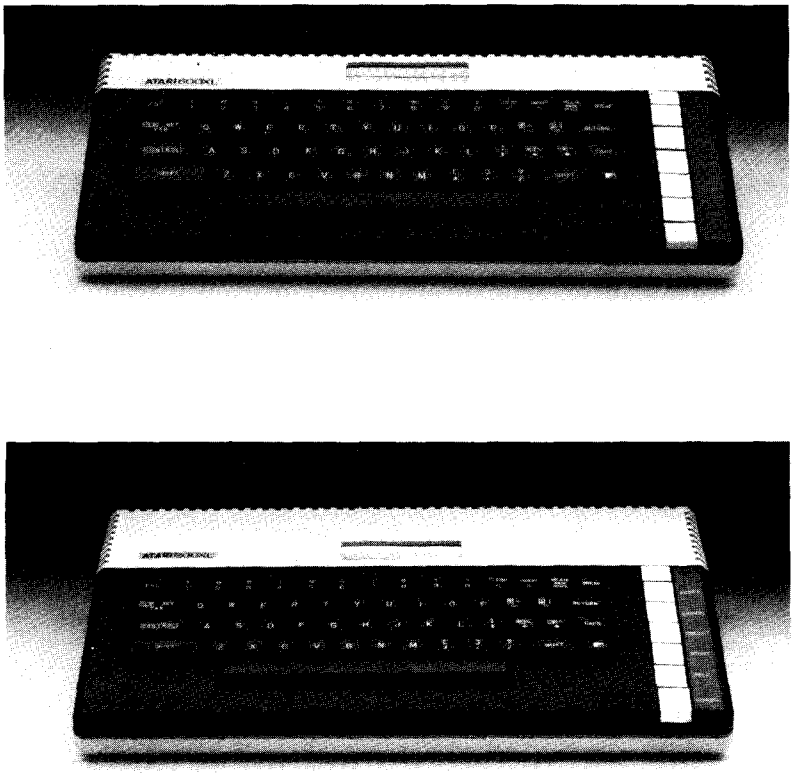

**Fig. 1-1 The Atari 600XL and 800XL computers** ***(Courtesy of Atari, Inc.)***

certain tasks. In other words, it is the art of creating a series of instructions, numbers, and commands for the computer so that it can do something useful for you (entertain you with a game, manage your home budget, or use the telephone to communicate with another computer).

2. *What hardware and software is available. Hardware* is the "real" part of a computer system (the computer itself, a printer, or a disk drive, for example); *software* is the instruc-

tions given to the computer such as the program package (the Star Raiders game from Atari or an educational program from Spinnaker Software, for example). Deciding what hardware and software to buy can be a difficult decision for anyone not familiar with the terms used to describe hardware and software. This book should help guide you through the jargon used in the industry.

3. *What you can do with your Atari home computer.* A computer has many uses, including entertainment, word processing, and computer communications.
4. *How you can make money with your computer.* You may think of your Atari as simply an expensive form of entertainment at this point, but it can be a "money machine" if you learn how to create programs, articles, and even books which will sell to other computer users.
5. *How to program graphics and sounds on your computer, and how to use these features to their full potential.* This is an especially exciting aspect of programming because graphics and sounds programs can show off some of the flashier features of your Atari home computer.

These and many other subjects will be covered in this book. In this first chapter, however, we'll be looking at the "basics" of the machine, such as the terms related to the Atari 600XL or 800XL you'll need to know, uses for your Atari computer, and recommended additional equipment that's worth buying.

## How to Use Your Computer

If you have just purchased an Atari 600XL or 800XL, some people may say, "Great—but what good is it?" If you're fairly new to computers, you might try to explain: "Well, it's great at playing games . . . and I think it can do something with business programs, and . . . uhh . . . hmmm," while your friends look at you with doubt and confusion.

True, the Atari 600XL or 800XL is capable of playing the best of games because it has outstanding graphics and sound capabili-

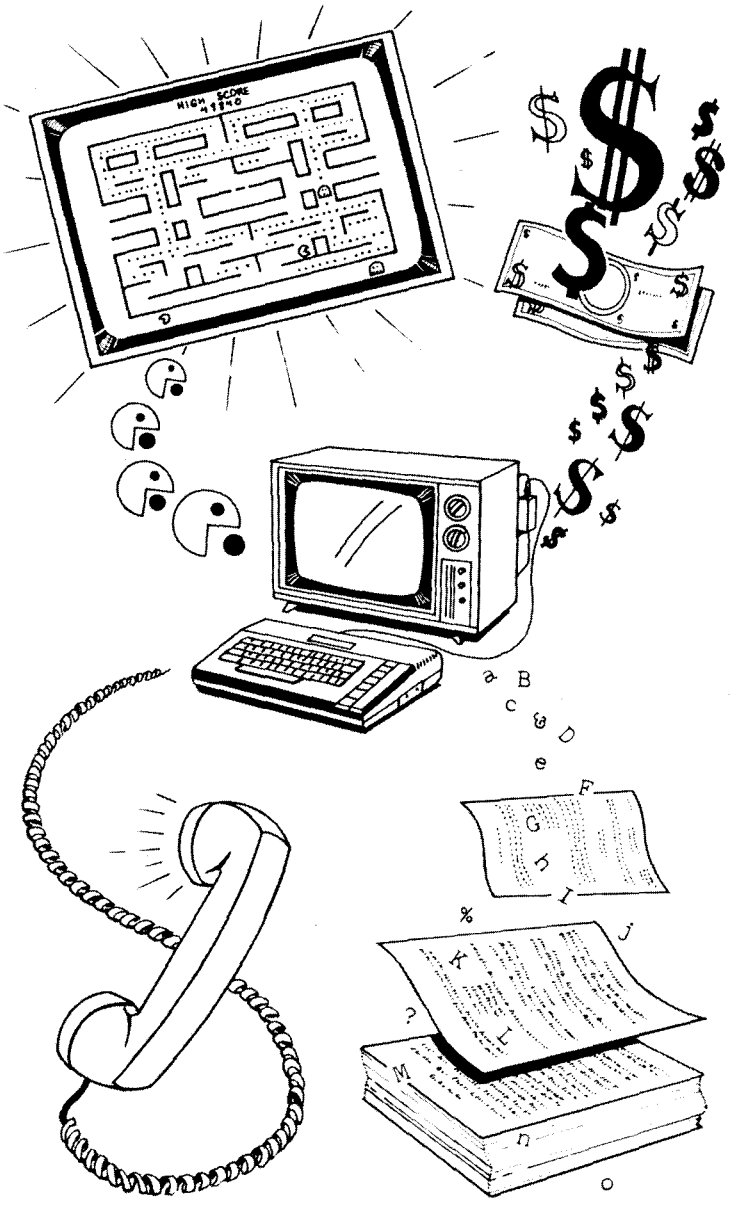


**Fig. 1-2 The uses of computers**

ties in addition to enough memory to support the more sophisticated and enjoyable game programs. A great number of game programs for the 600XL or 800XL are already available from companies such as Sirius Software, Broderbund, and Atari itself. But the Atari 600XL and 800XL have a great many uses besides their game capabilities (Fig. 1-2).

The Atari is an excellent educational tool. The computer is very effective for "drill and practice" work; and its excellent graphics and sounds make learning about the alphabet, numbers, and simple math and reading enjoyable. Teenagers and adults can use the foreign-language tutorials, educational science programs, and there are even software packages to improve their Scholastic Aptitude Test (SAT) scores.

In addition, your computer has many applications in the areas of home finance, record keeping, and home-office business. Although the 600XL and 800XL lack the memory to support the highly sophisticated business and management programs, they can be used to balance your personal budget (checkbook, credit card, tax, expense, and income); keep records of your financial transactions, important dates, and home inventory; and manage mailing lists and accounts payable/accounts receivable lists. Perhaps most important, however, is its potential use as a word processor because it can greatly reduce the amount of time you spend putting your thoughts into written words. And there are even programs to help correct your spelling, such as SpellStar. After you have used a word processor for a while, you'll wonder why you ever used a conventional manual or electric typewriter. They will seem almost archaic to you (Fig. 1-3).

Computer communications is another rapidly growing use for your computer. When you are able to communicate with other computers, you can send and receive electronic mail; "chat" with other computer users; receive accurate, up-to-the-minute financial, weather, and news information; and even shop without leaving your chair. Using a computer for communications brings the world to your fingertips and enables you to meet a lot of other computer users.

You will find that there are multiple uses and benefits to own-

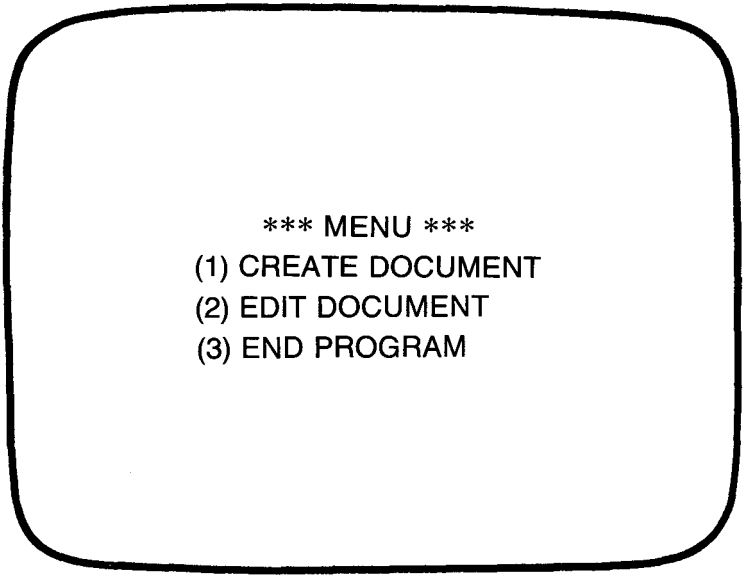


**Fig. 1-3 A typical document menu**

ing a home computer. It may make your business more efficient; organize your house; entertain you; allow you to receive important information; or simply make your thinking more logical, efficient, and analytical. Computers have become inexpensive, beneficial, and versatile machines, and this book should help you to unlock their powers and potentials.

## What Terms You Need to Know

You may be confused by some of the terms used in reference to your Atari 600XL or 800XL home computer either in the written material which comes with your computer or in other references. You need to understand these "computer words" in order to operate your machine, but they are not as mysterious as they might first appear. Here are some concise definitions of the more important words you'll need to know:

**ALPHANUMERIC:** consisting of the letters A to Z, the digits 0 to 9, and special symbols, such as &, $, and #.

**ASCII:** (pronounced askee) a code system that correlates numbers from 0 to 255 with the symbols, letters, and numbers a computer uses. The ASCII (American Standard Code for Information Interchange) code number corresponding to the capital letter A, for example, is 65. Therefore, if you told the computer to print the character represented by the ASCII code 65, **PRINT CHR$**(65), the Atari would respond by printing the letter A.

**ASSEMBLY LANGUAGE:** a very fast, efficient, but difficult language to learn, which is usually reserved for advanced programmers. You can use assembly language on the Atari 600XL or 800XL in the future, but for now, we'll be exploring the far easier BASIC language.

**BIT:** the smallest amount of information a computer can hold. It does this by using a binary code. The bit can be either *on* (represented by the digit 1) or *off* (represented by the digit 0).

**BUG:** an error in a program, usually causing the program to work improperly, if at all. Almost everyone who ever created a program, no matter how simple, has encountered bugs which must be found and removed before the program can be used.

**BYTE:** a storage unit of information, presently equal to eight bits. One alphanumeric character is stored as a byte (as an 8-digit binary number or eight bits). A byte might contain the digits *00101011, 11110101,* or any other combination of eight bits. The Atari 600XL has about 16,000 (16K) bytes; the 800XL has over 64,000 (64K) bytes.

**CARTRIDGE:** a plug-in device that has a program installed in it which can be "instantly" loaded into your Atari home computer. Programs like AtariWriter are on cartridge so that they can be instantly ready to run. Cartridges are much faster than, say, diskettes because programs on disk take about 10 seconds to load into the computer.

**COMMAND:** an instruction issued to a computer. You may tell a computer to **RUN** a program, for example, or **SAVE** data. These commands may be followed by numbers or words. For instance, if you wanted to **LIST** the program line 100, you would type **LIST** 100. If you wanted to save your program with the name "Program1", you would type **SAVE**"D:PROGRAM1" to tell the computer to save it (**SAVE**) to the disk drive (D:) as PROGRAM1.

**CURSOR:** a square on your monitor that shows where the next typed character will be displayed. If the cursor is at the top of the screen when you begin typing, the lines you're typing will appear at the top of the screen. If the cursor is in the middle of the screen, the words, letters, or symbols you type will be in the middle of the screen. You can move your cursor around the screen to change the position of the lines you're typing. An advantage of having a cursor you can move around the screen (by means of the arrow keys located at the right-hand side of the keyboard) is that you can edit (change) any part of your program displayed on the screen simply by moving the cursor over the part you want changed and typing in the change.

**DATA:** pieces of information, including numbers, letters, and words. The numbers 1, 5, 23, 523.3, or −8923; the letters A, GED, STYSG, or HI THERE; or the symbols &, @#%$, or *(*(# are all different pieces of information. Once you place data in your program, the computer can read that data at any time. For example, you could store the names of the capital cities of all 50 states as data in a program and the computer would be able to read that data later when your program needed them.

**DEBUG:** to remove the bugs in a program (often the most arduous and time-consuming part of creating a piece of software). Debugging involves (1) going through an entire program, looking for typographical errors, misspellings, and misused commands, and (2) using the program in every conceivable way in order to make sure that nobody else will encounter problems in your program.

**EDIT:** to make changes in data or in a program. Editing usually involves moving the cursor around the screen in order to change either the contents of a program line, a line number in a program, or a word in a letter you are writing to someone. Editing a program may mean changing your instructions to the computer or simply debugging the program; editing a document can mean changing a letter, a word, a sentence, or any amount of text within a manuscript.

**GRAPHICS:** pictures that can be created on the screen by the computer. The Atari 600XL and 800XL have excellent graphics capabilities, which means that a wide variety of low- and high-resolution pictures can be drawn on them. When you are playing an arcade game, the pictures of the ships and laser fire you see are graphics. In a business program, the line graphs and bar charts are examples of graphics. *Graphics* are the pictures on the screen; *text* is the letters, numbers, and symbols on the screen.

**HARD COPY:** something printed on paper, such as a program listing, that would otherwise be visible only on the video monitor. For example, if you are using a word processor to type a letter, what you see on the screen is *soft copy*; the actual letter when it is printed on paper is *hard copy*.

**HARDWARE:** the physical part of a computer system (the computer itself, a screen, or a printer, for example).

**INPUT:** the information you give to the computer. If, for example, your Atari asks you to give it your name, the name you type in is *input* to the computer.

**INVERSE VIDEO:** dark text (letters, numbers, and symbols) on a light background. Inverse video is effective for highlighting.

**KEYWORD:** a word that has a specific meaning to the computer (**PRINT**, **LET**, or **GOTO**, for example) and cannot be used by you for a variable name. *Keywords* are specific commands to the computer. *Variables* are letters or groups of letters and/or numbers which represent other numbers (such as A representing the

number 12 or B1 representing −23.532). You cannot, for example, tell the computer that a variable called **RUN** is equal to a certain number because you cannot name a variable after a keyword.

**KILOBYTE:** 1024 bytes. The Atari 600XL has 16 kilobytes (16K) for you to use; the 800XL has 64 kilobytes (64K). The more memory your Atari has, the larger the programs it can use and the more information it can retain.

**MENU:** a list of options shown on the screen. Suppose you had a program that could solve mathematical problems. Your menu might look like this:

```
PLEASE CHOOSE AN OPTION

(1) ADDITION
(2) SUBTRACTION
(3) MULTIPLICATION
(4) DIVISION

ENTER CHOICE - - -
```

You would select a number from 1 to 4 from this menu to gain access to the desired function.

**MODEM:** a device that enables your computer to communicate with other computers. It sends and receives information by converting the digital pulses of the computer to the audio tones of a standard telephone. *Modem* is derived from *MO*dulator/*DEM*odulator.

**MONITOR:** a high-resolution screen dedicated mainly to computer use. Your video screen, CRT, television set, or whatever you look at while you are using your computer, is called a "monitor."

**PERIPHERALS:** "extras" in your computer system (a printer, a modem, or a plotter, for example). *Peripherals* are simply pieces of hardware that can be connected to your Atari 600XL or 800XL home computer.

**RAM:** the computer's working storage, or main memory. The information stored in the RAM (Random Access Memory) can be changed or accessed. It is also erased when the computer is turned off. When you write a program, it is stored in RAM only until you turn the Atari off. Because of this, it's a good idea to transfer a program to a cassette recorder or a disk drive if you want to keep it.

**ROM:** a permanent memory chip for program storage. Only the computer can use the information that is stored in the ROM (Read Only Memory) at the time of manufacture. It will not be lost when you turn your machine off. The Atari's BASIC language is in the ROM.

**SOFTWARE:** the programs available for your computer. When you buy a software package, you'll usually receive a disk or cassette with a program or programs on it, *documentation* (instructions) telling you how to use the software, and perhaps a warranty card stating how long your software is protected by the software company's limited warranty.

**STRING:** a sequence of digits, letters, or other characters. A string could be "12345," "This is a test," or "I have 8 pigs." In BASIC, a variable that represents a string has a dollar sign at the end (A$ or YH$).

**USER-FRIENDLY:** software that is easy to use and understand. AtariWriter, for example, is a relatively user-friendly piece of software because the various options are displayed on the screen and you don't have to study the manual all day to learn how to use the program effectively.

**VARIABLE:** a representation of a number or a string of characters. For example, the letter X could represent the number 13, or the symbol A$ could represent the word "computer." The difference between these two is that the first is a *numeric variable* and the second is a *string variable.*

A numeric variable is a letter or any combination of letters and numbers (A, B1, TESTER, NUMBER, Z12) which represents a number. A1 might equal 10, B might equal 98, and Y64 might

represent −23542.23. Furthermore, you can change the value of any variable at any time by simply telling the computer to let the variable equal another number. If you wanted the variable A to equal the number 400, you would simply type A = 400.

A string variable is also a letter or combination of letters and numbers, but it is always followed by a dollar sign (A$, Y1$, or MONEY$). String variables can represent anything at all, from words to sentences to symbols. If, for example, you wanted T1$ to equal the word "Atari," you would tell the computer T1$ = "ATARI". The quotes are very important because the computer will make the variable equal whatever is inside the quotes.

Numeric variables are like the symbols used in algebra ($X$ = 123, for example), while string variables are unique to computers.

This glossary should give you an idea of what you need to know in order to understand the subjects in this book and in other books about computers. There are some other terms you need to know, of course, but these will be defined as they come along in the pages of this book.

## An Extension of Your Mind

You may see your computer as a machine that can perform a number of useful functions. You're right, of course, but a computer is more than just an appliance like a dishwasher or a lawn mower.

The computer itself is not intelligent, but *you* are, and because you are, you can make the computer perform intelligent functions through your own programming. The computer can be an extension of your own mind; a tool that you can use to speed up the process of changing your ideas into reality.

What if you are a good writer, but you don't have the patience to type rough drafts and then revisions and, finally, "clean" copy, with the aid of "white-out"? What if you are able to absorb information quickly but can't find a class that moves at

your speed? Or perhaps most classes are too fast for you. What if you have the skills of an artist, but you can't use or you don't have an artist's conventional tools? Each of these stumbling blocks can be overcome with one powerful tool—the home computer. With a home computer, you can use a word processor to produce manuscripts, substitute an educational program for classes, or use a high-resolution graphics drawing program to develop your skills as an artist.

The purpose of this book is to help you understand the power of your computer, to show how you can unleash that power, and to give you some idea of how your new Atari 600XL or 800XL can help you to use your own mind more efficiently and productively. We'll begin by investigating the area of programming.

## Chapter Two

# Excel in Programming

Writing a program is easy. Writing a *good* program isn't. To write a good program, you must know the different elements of programming, how you can combine these elements to produce a quality program, and what BASIC commands are used for these different elements.

The subjects that are going to be covered in this chapter are:

1. The details of programming—that is, the technical aspects of writing a program, including line numbers, special keys used on the computer, and how to store and retrieve programs
2. Mathematics in programming, including the "mathematical hierarchy" the computer uses when you include math in your programs
3. Variables and arrays, including numeric variables and string variables
4. Loops, including the **FOR** and **NEXT** statements of the BASIC language
5. Branching (using the **GOTO** statement) to different parts of your program, subroutines (**GOSUB/RETURN**), and decision making
6. Input from your data statements, from the keyboard, and from the **INPUT** statement itself; and output to the screen and to the printer

Once you have mastered these concepts, you should find it easy to take an idea for a program and develop it into a working piece of software for your computer.

## Details of Programming

You probably already know some of the fundamentals of programming, but the following should serve as a "refresher course."

First of all, you must know some of the commands used directly in the BASIC language. BASIC (Beginner's All-purpose Symbolic Instruction Code) is already installed in your Atari 600XL or 800XL computer. The language is simple, easy to understand, and very versatile; and because it is made up of English words, it is very easy to learn. Here are some of the commands used when the computer is in command mode—that is, not running a program. Remember that after every command you enter, you must press the RETURN key.

**CONT:** is used if you press the BREAK key while a program is running and you want to restart the program from where you stopped it. This will continue the program, with all of its variables intact, *if* you did not insert any new program lines or perform any modifications to the program after you halted it.

**LET:** assigns values to variables in the command mode. **LET** A = 2 gives the variable A a value of 2. In fact, you don't need to use the **LET** statement because the computer would recognize A = 2, but beginning programmers may find it easier to assign keywords to all statements.

**LIST:** lists program lines. Simply typing **LIST** and pressing the RETURN key will list the entire program, while typing **LIST** followed by a number will list that line number. Typing **LIST** followed by a number, a comma, and then another line number will result in the lines from the first number to the second being listed. Here are some examples:

| | |
|---|---|
| To list the entire program, type: | **LIST** |
| To list line 110, type: | **LIST** 110 |
| To list lines 100 to 200, type: | **LIST** 100,200 |

**NEW:** a command you should be careful with because it will erase any BASIC program in memory, along with its variables. If you think you might want to use your program later, save it before you type **NEW**.

**RUN:** when this command is followed by the RETURN key, it causes the computer to start executing your program from its smallest line number. You can also run a file that is on a diskette by typing **RUN**"D:filename", such as **RUN**"D:MYPROGRAM" or **RUN**"D:PROPROG".

When you want to write a program, you must use BASIC keywords, line numbers, and alphanumerics—that is, letters of the alphabet, digits, and punctuation marks. BASIC keywords will be thoroughly described in this book, but for the time being, think of them as commands which are already within the Atari's BASIC program, commands such as **PRINT** to print something onto the screen, **READ** to read data from another part of your program, or **STOP** to stop a program. You can see how straightforward these BASIC keywords are.

*Line numbers* are the numbers preceding the commands you give the computer. Line numbers are important because they specify the sequence in which the commands are to be executed. For example, suppose you wanted to instruct the computer in your own language (not in BASIC language) to buy something for you. You might give it the following instructions:

```
PAY FOR THE PRODUCT
FIND THE PRODUCT
PICK UP THE PRODUCT
ENTER THE STORE
LEAVE THE STORE
```

All of these commands together will accomplish the task, but they are in the wrong order (perhaps in trying to remember all of the steps involved in buying a product, you did not think of them in order). To let the computer know in what order it should perform these steps, you must give each line a number, with the lowest number being the first step and the highest number being the last step:

```
40 PAY FOR THE PRODUCT
20 FIND THE PRODUCT
30 PICK UP THE PRODUCT
10 ENTER THE STORE
50 LEAVE THE STORE
```

It's a good idea to select line numbers in increments of 10: line 10, line 20, line 30, and so on. This will enable you to insert additional lines into a program, should it become necessary to do so, without renumbering existing lines.

The computer will then put the instructions in numerical order, as follows:

```
10 ENTER THE STORE
20 FIND THE PRODUCT
30 PICK UP THE PRODUCT
40 PAY FOR THE PRODUCT
50 LEAVE THE STORE
```

This is what line numbers are for.

If you want to make a program which asks a person what his name is and then says hello to that person, using his name, you would type in this program, even if you didn't type the lines in the correct order:

```
30 PRINT"Hello,";N$
10 PRINT"Hi. What is your name?";
20 INPUT N$
```

The computer would then put your program lines in order, as follows:

```
10 PRINT"Hi. What is your name?";
20 INPUT N$
30 PRINT"Hello,";N$
```

If you are not sure what the above lines mean, here is a short explanation:

*Line 10:* tells the computer to print something. That "something" can be either a variable or something contained in quotes. In this case, the "something" is the sentence "Hi. What is your name?", which is printed onto the screen.

*Line 20:* uses the **INPUT** statement (which lets the computer accept input from the keyboard) and gives that input the variable name N$. When the computer encounters this statement, it prints a question mark on the screen, indicating that it is waiting for some information (in this case, a name), to substitute for the variable N$.

*Line 30:* prints the word "Hello," followed by the variable N$, which represents a name. Suppose you give the computer the name "Mark." N$ will therefore equal "Mark," and when the computer encounters the statement in line 30, it will print "Hello, Mark" because it was instructed to print the word "Hello," followed by the variable N$.

Besides giving instructions a particular order, line numbers serve another important purpose: you can use them as a reference point in a program. For instance, you can make the computer go back to the instructions in line 10 by typing "**GOTO** 10". You might make this "**GOTO**" statement line 40:

```
40 GOTO 10
```

Or you might tell the computer to go to another line, such as line 30.

```
40 GOTO 30
```

If you do, the computer will keep returning to line 30 and printing "Hello,";N$ repeatedly.

We see, therefore, that line numbers have two main purposes: order and a point of reference. Line numbers must precede any instruction(s) which are added to your BASIC program (Fig. 2-1). We will explore some BASIC instructions later, but let's first take a look at editing features.

Editing is helpful when you need to make a modification in your program or when you have simply made a mistake. To edit something, you must move your cursor around the screen. The *cursor* is a blinking box on the video display that indicates where you are. For example, if you begin typing at the very top of the screen, your cursor will be at the top. If your cursor is at the

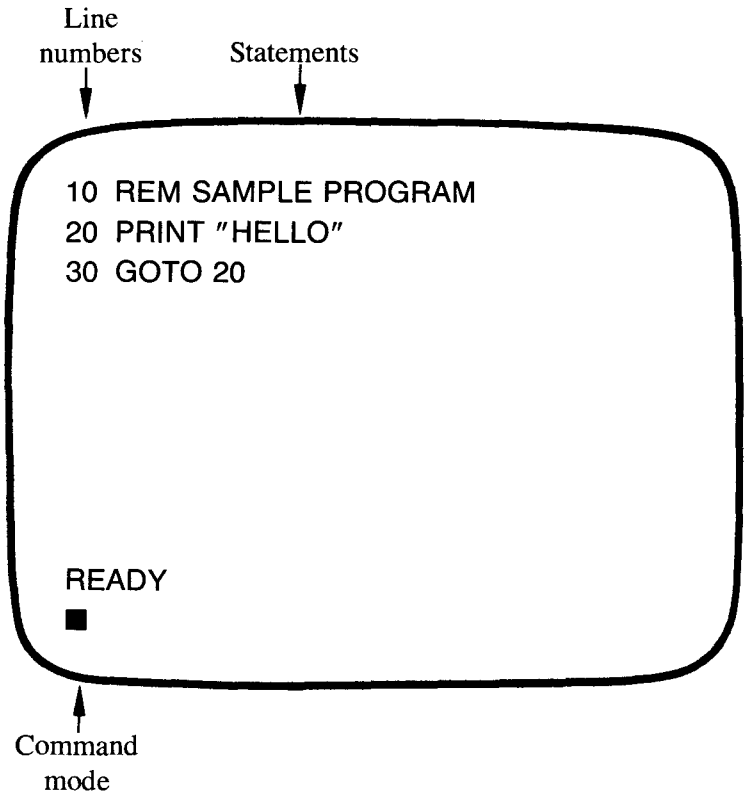


**Fig. 2-1 The elements of a program**

bottom of the screen when you begin to type, everything you type will be at the bottom of the screen. You can move your cursor around the screen to make modifications, and the cursor will show exactly where you are making changes.

The simplest way to move your cursor around the screen is to press the CONTROL key along with the arrow keys. There are four possible combinations of keys for the different movement commands:

| | |
|---|---|
| (CONTROL) (↑) means: | move cursor up one line |
| (CONTROL) (↓) means: | move cursor down one line |
| (CONTROL) (→) means: | move cursor right one character |
| (CONTROL) (←) means: | move cursor left one character |

The cursor will not destroy anything it encounters in its travels because its purpose is to "travel," not to delete. If after moving the cursor to a new location, however, you type over a program line and then press RETURN, you may change your program; so be sure you want to change something before you use your keyboard.

Suppose you typed in the program line

```
10 PRINT"Hello everybody"
```

when you meant to type in

```
10 PRINT"Hello everyone"
```

You could move your cursor up to the word "everybody" and type in "everyone," but you would still have the letter y left over because the first word is longer than the second. To delete this letter, you must press the CONTROL key *along with* the DE-

LETE/BACKSPACE key once. This "eats" the extra letter. The third step is to press the RETURN key to tell the computer you are through working with that line. Your program line is now correct.

You can delete more than one character by moving your cursor to the first character you want deleted and pressing the CONTROL and DELETE keys at the same time as many times as you like.

The insert command is very much like the delete command except that it performs the opposite function. Suppose you typed in the program line:

```
60 LET A = 10:PRINT"The value for the is";A
```

when you meant to type in

```
60 LET"A = 10:PRINT"The value for the variable is";A
```

Instead of retyping the whole program line, you can move your cursor up between the words "the" and "is" and press the keys CONTROL, INSERT (together) as many times as you need in order to insert the word "variable." Since the word "variable" has eight letters and you need an extra space following the word, you would hold down the CONTROL and INSERT keys until nine new spaces were available; then you could type in (*insert*) the word "variable."

There are some other handy editing features built into your 600XL or 800XL. For example, press:

| | |
|---|---|
| SHIFT, INSERT | to insert one line |
| SHIFT, DELETE | to delete one line |
| SHIFT, CAPS | to shift to the uppercase mode |
| BREAK | to stop the program while it is running |

## Mathematics in Programming

Most people think of computers as math machines. They are, indeed, good at math because that's the only thing they truly "understand." In fact, the computer understands only two things: the digit 0 and the digit 1, which are encoded as *on* and *off* conditions in the computer itself. However your Atari makes it easy for you by translating simple English words into its own "machine language." In this section, we'll take a look at the commands you can use to create your own mathematics programs. Keep in mind that these commands can be used in conjunction with numbers that you type into the keyboard (354, 333, or 736, for example) or numbers represented by variables such as AB or TK or YC.

Here are the BASIC mathematical keywords along with their functions:

**ABS:** returns the absolute value of a variable or number. If the number is negative, it will be returned as a positive; while a positive number will also be returned as a positive.

Example: **LET** A = **ABS**( – 12) would result in the variable A equaling 12.

**ATN:** returns the arc tangent of a number in degrees.

Example: **PRINT ATN**(15) would print the value of the arc tangent of 15.

**CLOG:** gives the base 10 logarithm of a number.

Example: **LET** X = **CLOG**(15) would make the variable X equal to the base 10 logarithm of 15.

**COS:** returns the cosine of a number. Telling the computer to **PRINT COS**(90) would result in the computer printing the number 0, which is the cosine of 90 degrees.

**DEG:** instructs the computer to perform trigonometric functions in degrees instead of radians because the Atari 600XL or 800XL is initially set to perform functions in radians. The **RAD** command performs just the opposite function.

**EXP:** returns the value of the natural logarithm, *e* (2.7182818), raised to a specific power.

**INT:** removes any noninteger digits in a number. For example, if you told the computer that the variable A should equal **INT**(4.3728), then A would equal 4 because the **INT** command would remove the .3728 (which is not an integer).

**LOG:** returns the logarithm of a number.

**RND:** creates a random number between 0 and 1. Telling the computer to **PRINT RND**(10) could result in .323, .542, .723, or any other number between 0 and 1.

**SGN:** returns a value of 1 if a value is positive, 0 if a value is 0, and −1 if a value is negative. **SGN** (100) would result in a 1, **SGN** (0) would make the computer give you back a 0 answer, while **SGN** (−348) would result in a value of −1.

**SIN:** gives the sine of a value.

**SQR:** returns the square root of a number. **SQR** (16) would result in a 4 being returned because 4 is the square root of 16.

You can also use special keys to perform mathematical operations on your Atari home computer. The plus sign (+) is for addition; the negative sign (−) is for subtraction; the slash (/) is for division, and the asterisk (*) is for multiplication. Other keys include the up arrow (↑) (for exponential operations); and the greater than (>), less than (<), and equal to (=) keys for comparing values. Here are some examples of each of these:

| | |
|---|---|
| A=2+2: | **LET** A EQUAL 2 PLUS 2 |
| X=8−4: | **LET** X EQUAL 8 MINUS 4 |
| C=9*6: | **LET** C EQUAL 9 TIMES 6 |
| M=4/3: | **LET** M EQUAL 4 DIVIDED BY 3 |
| T=30↑8: | **LET** T EQUAL 30 TO THE EIGHTH POWER |

The greater than, less than, and equal to signs can be used in six different ways.

| | |
|---|---|
| A < B means: | A is less than B. |
| A > B means: | A is greater than B. |
| A = B means: | A equals B. |
| A < = B means: | A is less than *or* equals B. |
| A = > B means: | A is greater than *or* equals B. |
| A < > B means: | A and B are not equal. |

You could use these expressions for statements such as **IF** A>B **THEN PRINT** "A IS GREATER THAN B". The computer would check to see if the variable A was greater than the variable B and, if it were, it would print the statement "A IS GREATER THAN B". Here's another example of how the comparison functions of the Atari can be used:

```
10 INPUT A
20 INPUT B
30 IF A>B THEN PRINT"A IS GREATER THAN B"
40 IF A<B THEN PRINT"A IS LESS THAN B"
50 IF A=B THEN PRINT"A EQUALS B"
```

In line 10, you give the computer a number for the variable A. In line 20, you give it a number for the variable B. Finally, in lines 30, 40, and 50, the 600XL or 800XL compares the variables A and B and tells you how they compare.

This completes our look at the mathematical functions of your Atari 600XL or 800XL home computer. We will now examine the significance of variables and arrays in your programs.

## Variables and Arrays

There are three kinds of variables in the BASIC language: numeric variables, string variables, and arrays.

### Numeric Variables

*Numeric variables* are letters or combinations of letters and numbers (X, Y, TU, NH) that are used to represent numbers. For example, you can tell the computer that A equals 24, W1 equals 98.234, and PI equals −2264. You would enter it into the computer as follows:

```
LET A = 24
LET W1 = 98.234
LET PI = -2264
```

### String Variables

*String variables* are used to represent letters of the alphabet as well as numbers. "Hello there", "The number is 532", "TEST", or just "532" could be a string. If you told the computer to **PRINT** any of these strings, it would not print the quotation marks, but when you assign a variable to a string, you must use quotation marks (**LET** A$ = "This is a string"). Here are some examples of strings:

```
B$ = "Here is one string"
YY$ = "Here is another string"
T1$ = "And yet another string . . . string number 3"
```

There are a couple of important things you should remember when working with strings. First of all, as I mentioned in Chapter 1, a string variable is always followed by a dollar sign. Second, before using any string, you have to use the **DIM** (for "*dim*ension") statement to tell the computer you are going to be using a particular string and that the string will contain a certain number of characters. For instance, if you wanted to use the variable B$ to represent a 10-character string, you would have to type in the following program line:

```
10 DIM B$(10)
```

If you wanted B$ to represent the maximum string length, which is 255 characters, you would have to tell the computer to **DIM** B$(255).

The computer must be given this information for all the strings in the program. Assume that your program had three string variables: A$, which had 10 characters; B3$, which had 5 characters; and YY$, which had 255 characters. You would have to enter the following information:

```
10 DIM A$(10),B3$(5),YY$(255)
```

As you see, you can list one variable after another in a **DIM** statement.

Another important thing to remember is that you cannot use a keyword as a variable because keywords have specific meanings to the computer, but you can use keywords *in conjunction with* strings. For example:

**ASC:** returns the ASCII value of a string. The ASCII value is simple a code number which corresponds to a particular character. For example, the letter A corresponds to the ASCII code number 65, so if you told the computer to **PRINT ASC**(''A''), a value of 65 would be returned.

**CHR$:** performs the opposite function of **ASC**. If you give the computer a code number along with the **CHR$** command, it will print the character that corresponds to that number. For instance, if you told the computer **PRINT CHR$**(65), it would print the letter A.

**LEN:** gives the length of a string. X = **LEN**(''test'') would give X a value of 4 because the word ''test'' has four letters.

**STR$:** assigns a value to a string variable. If you typed A$ = **STR$**(100), then A$ would equal the string ''100.''

**VAL:** does the opposite of **STR$**. For example, if you had given A$ the value 100, and then you typed **PRINT VAL**(A$), the computer would print 100 on the video monitor. On the other

hand, if the variable A$ represented the word "mountain," then the computer would print 0 because words don't have a numerical value.

### Arrays

Finally, an *array* is a special kind of numeric variable that can be used to store a great deal of information. Arrays for numeric variables may look like this: A(x), BB(y), or T1(a,b). You'll understand why in a moment.

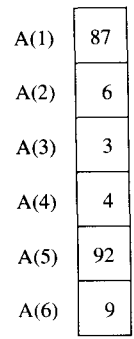


**Fig. 2-2 A sample one-dimensional array**

A one-dimensional array (Fig. 2-2) stores a list of variables which might look like a column if you wrote them down. For instance, let's assume we have a one-dimensional, three-element (part) array used to store the temperature on different days. On day 1, the temperature was 55°F; on day 2, it was 40°F; and on day 3, it was 70°F. Instead of using three separate variables to represent the three days, we could store the information in the variable T(x), as follows:

| | |
|---|---|
| T(1) | 55 |
| T(2) | 40 |
| T(3) | 70 |

It doesn't matter how big the "x" is in T(x) as long as we have enough memory and we have used the **DIM** statement to set aside three elements for T(x). In this case, therefore, we would have to enter **DIM** T(3) before we used this array. As you can see, we could put a lot of data into a one-dimensional array, but we can put even more information into a two-dimensional array (see Fig. 2-3).

Suppose we want to store a series of numbers to show how much money several persons made on certain days. We have three people: John (1), Bill (2), and Mary (3); and we have two days: Monday (1) and Tuesday (2). In a two-dimensional array, we can use the numeric array C(x,y), with x representing the number of the person and y representing the number of the day. Here, then, is the array:

C(1,1) = 50 C(1,2) = 75

C(2,1) = 65 C(2,2) = 35

C(3,1) = 68 C(3,2) = 94

The first values 1, 2, and 3 show the person, while the second values 1 and 2 show the day. Therefore, if we wanted to know

Y-element

| A(X,Y) | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 47 | 32 | 68 |
| 2 | 93 | 42 | 89 |
| 3 | 14 | 17 | 97 |
| 4 | 163 | 333 | 487 |

X-element

**Fig. 2-3 A sample two-dimensional array**

how much Bill (2) made on Monday (1), we could check the variable C(2,1) to discover he made 65 dollars on that day. Mary (3) made 94 dollars on Tuesday (2), since C(3,2) equals 94. As you can see, using an array is a very efficient way of handling several pieces of data.

There are two commands you should know when using arrays.

**DIM:** *dimensions* an array—that is, it prepares the array for your use. If you were going to be using the C(x,y) array above, you would have to enter **DIM** C(3,2) because your maximum array value is (3,2). If you were going to be using a three-dimensional array, you might enter, for example, **DIM** A(5,5,2).

**CLR:** clears all the strings, arrays, and other variables from the computer. Use this command with caution because it will wipe out everything except your program.

## Loops

A program is usually designed to do one thing at a time. To repeat the same process on another set of data, an instruction is given that sends the computer back to the first instruction of the series. This operation is called *looping* and can be accomplished by the use of **FOR** and **NEXT** statements (Fig. 2-4). A **FOR/NEXT** loop is an efficient way to get the computer to count.

In a **FOR** statement, you specify the number at which you want to begin counting, the number at which you want to end counting, and at what **STEP** you want to count. If you want to count from 1 to 10 in steps of three, for example, you would use the instruction **FOR** X = 1 **TO** 10 **STEP** 3. (The variable X is used only for this example.) The computer would then begin counting "1 . . . 4 . . . 7 . . . 10." But it won't continue counting by steps until it encounters a **NEXT** statement followed by the variable being used (**NEXT** X). This statement tells the computer to keep counting if the maximum value (in this case 10 because the computer has to count to 10) has not yet been

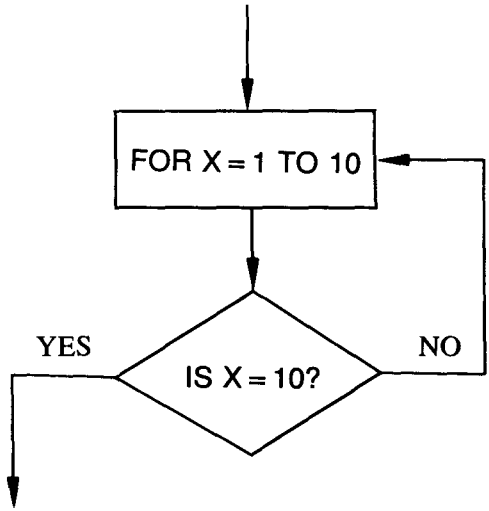


**Fig. 2-4 Diagram of a loop**

reached. To show how this works, this is a simple BASIC program for counting from 1 to 10 in steps of three, using the **FOR** and **NEXT** statements:

```
10 FOR X=1 TO 10 STEP 3
20 PRINT X
30 NEXT X
```

Here is a less efficient program that performs exactly the same function but does not use the **FOR/NEXT** statement:

```
10 X=1
20 X=X+3
30 IF X=10 THEN GOTO 60
40 PRINT X
50 GOTO 20
60 END
```

Obviously, **FOR/NEXT** loops are better for counting than are other BASIC keywords; and you can use any variables, starting and ending numbers, and steps you want.

## Branching

*Branching* is a process by which a computer transfers control to another part of a program and makes comparisons in order to "test" certain variables or values. Decisions can then be made on the basis of these comparisons. The basic branching statements are **GOTO** and **GOSUB**.

**GOTO** tells the computer to go to a specific line number in the program. **GOTO** 100 would make the computer go to line 100. **GOSUB** tells the computer to jump, temporarily, to another part of the program and access a subroutine until a **RETURN** statement is encountered. **GOSUB** 500 would make the computer go to line 500 and follow whatever instructions it finds there until it encounters a **RETURN** statement. The **RETURN** statement makes the computer return to the line immediately following the **GOSUB** 500 statement. Subroutines accessed by the **GOSUB** statement are useful when you need a function performed repeatedly throughout a program and don't want to type the instructions over and over. Subroutines are separate, helpful modules that can be plugged into a program when needed to perform a particular function.

**ON . . . GOSUB** and **ON . . . GOTO** perform the same functions as **GOSUB** and **GOTO** except that they evaluate a certain variable and act on its value. For example, if you typed in the line **ON** X **GOTO** 100, 200, 300, and if X equaled 1, the computer would go to line 100. If X equaled 2, the computer would go to line 200. If X equaled 3, the computer would go to line 300. **GOSUB** can be used in the same way. You can use any variable and any line numbers with these statements. The line numbers don't have to be in order, but each time you use the statements **ON . . . GOSUB** or **ON . . . GOTO**, the computer will check

the variable you use for the value of 1, 2, 3, and so on, and take appropriate action if the variable equals a certain one of those numbers. If it doesn't equal any of them, it will show an error.

The decision-making commands are shown below and are also easy to use:

**TRAP:** helps to find errors by directing the computer to a certain line number if an error is encountered. For example, the statement TRAP 120 would send the computer to line 120 if there was an error in that line.

**IF . . . THEN:** the most important decision-making command. Here are some examples of **IF . . . THEN** statements and their effects:

| | |
|---|---|
| **IF** X = 1 **THEN PRINT**"Hi" | prints "Hi" if the value of X is 1 |
| **IF PEEK**(1) = 34 **THEN STOP** | stops the program if the value of memory location 1 is 34 |
| **IF** A$ = "Y" **THEN PRINT**"You answered yes" | prints "You answered yes" if the value of A$ is equal to the letter Y |

As you can see, the **IF** checks to see if a certain condition is true. If it is, **THEN** the computer takes appropriate action. If it is not, the computer simply proceeds to the next line in the program.

## Input and Output

Knowing how to get information into and out of a computer system is essential to good programming. Before we explore the different commands used for input and output, we need to take a look at the devices used to send and receive data.

Sometimes you can specify what device you want to use in the **INPUT** or **OUTPUT** command. For example, if you want to

load a program called "Evade" and you want to load it from your disk drive, you would tell the computer: **LOAD** "D:EVADE". The "D:" in this command stands for "disk drive." Other device specifications that can be made in the **INPUT** and **OUTPUT** commands are:

"K:" (keyboard) used for input only

"P:" (line printer) used for output only

"C:" (cassette recorder) used for input or output

"S:" (screen) used for output only

"E:" (editor) used for input or output

Of course, not all commands can be used in conjunction with device specifications. (For example, you can't tell a computer to load a program from the video screen.) Still, the number of specifications that can be made with **INPUT** and **OUTPUT** commands on the Atari 600XL or 800XL makes these machines very versatile.

The following is a sampling of **INPUT/OUTPUT** commands:

**CLOAD:** loads a program from cassette tape.

**CSAVE:** saves a program on cassette tape.

**INPUT:** enters data via the keyboard from the user.

**LOAD:** loads a specific program from a disk (used with the disk drive).

**LPRINT:** sends the data to be printed to the printer (used with the line printer).

**PRINT:** used to print letters, numbers, or symbols with a device (used with any of the devices that accept input).

**READ:** used in conjunction with the **DATA** statement, gets data from another part of the program. For example, the program

below will read the number next to the **DATA** statement and assign it to the variable X.

```
10 READ X
20 DATA 20
```

In this case, X will be set to equal 20.

In the example below, a number and a *string* (a set of characters) will be read and assigned to the variables N and X$.

```
10 READ N,X$
20 DATA 35,TIM
```

Once this program has been run, N will equal 35 and X$ will equal "TIM."

**SAVE:** used to save a specific file on a disk (used with the disk drive).

We have now covered the most important **INPUT/OUTPUT** statements and the devices used with them. The more versatile and interactive your computer system becomes, the better your programs will be.

To become a good programmer, you must acquire as much experience as possible. Reading books is not enough. Experience is the real key to excellence in programming. The bugs, problems, accidently erased programs, electrical failures, and difficult-to-understand commands you encounter are all part of your education in programming. The more problems you face and overcome, the better your understanding of programming will be.

Chapter Three

# One Screen Is Worth a Thousand Words

The Atari 600XL or 800XL computer, like the other computers Atari makes, has some of the most powerful and versatile graphics available today on the home computer market. Vivid colors, high resolution, and easy-to-use graphics commands make the Atari an excellent machine for producing pictures, graphs, and shapes for your programs.

## Commands

There are a total of five special commands that can be used to produce graphics on your computer.

### Graphics

The first of these commands is **GRAPHICS**, which is used to select a graphics mode. One of these modes might be used for high-resolution (very fine, high-quality) graphics and no text (letters or numbers); another might be used for medium-resolution graphics with some text and a few colors; and another might be used for low-resolution graphics with more text and colors.

There are 16 modes altogether (numbered 0 to 15), and you may access any one of them by typing in **GRAPHICS** followed by the number of the mode (**GRAPHICS** 7, for example).

If you want to dedicate the entire screen to graphics, however, you must add 16 to the number of the graphics mode. For example, if you wanted to use a screen of graphics mode 13, but you wanted the entire screen elevated to graphics, you would type **GRAPHICS** 13 + 16, or **GRAPHICS** 29. Also, if you want to enter a graphics mode without first clearing your screen, you must add 32 to the number of the graphics mode. Finally, if you want to dedicate the entire screen to graphics *and* you want to enter a graphics mode without first clearing the screen, you would type **GRAPHICS** followed by the number of the graphics mode plus 16 *and* 32. For example, if you wanted to enter graphics mode 12 without first clearing the screen, and you wanted the entire monitor dedicated to graphics, you would type **GRAPHICS** 12 + 16 + 32, or **GRAPHICS** 60.

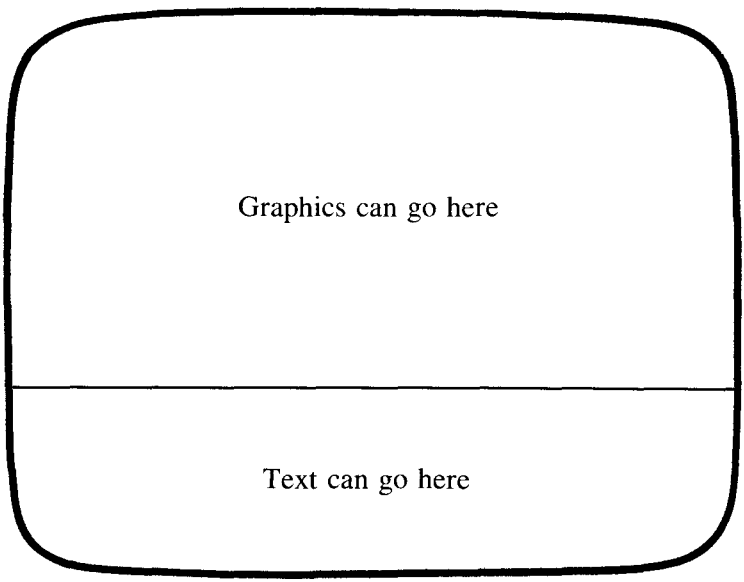


**Fig. 3-1 The split screen in graphics mode**

Table 3-1. **GRAPHICS MODES**

| Mode | Type | Columns | Rows | Colors |
|---|---|---|---|---|
| 0 | Text | 40 | 24 | 1 |
| 1 | Text | 20 | 24 | 5 |
| 2 | Text | 20 | 12 | 4 |
| 3 | Graphics | 40 | 24 | 4 |
| 4 | Graphics | 80 | 48 | 2 |
| 5 | Graphics | 80 | 48 | 4 |
| 6 | Graphics | 160 | 96 | 2 |
| 7 | Graphics | 160 | 96 | 4 |
| 8 | Graphics | 320 | 192 | 1 |
| 9 | Graphics | 80 | 192 | 16 |
| 10 | Graphics | 80 | 192 | 9 |
| 11 | Graphics | 80 | 192 | 16 |
| 12 | Text | 40 | 24 | 5 |
| 13 | Text | 40 | 12 | 5 |
| 14 | Graphics | 160 | 192 | 2 |
| 15 | Graphics | 160 | 192 | 4 |

In addition, each of the 16 graphics modes can be divided into two different mode types: (1) **TEXT**, which supports letters and numbers, and (2) **GRAPHICS**, which is used primarily to draw graphics, although text can still be printed at the bottom of the split screen which the **GRAPHICS** command creates (Fig. 3-1). There are also a different number of *columns* (horizontal *x*-coordinate values) and *rows* (vertical *y*-coordinate values) for each mode, and only a certain number of colors may be used in each graphics mode. For your reference, Table 3-1 shows the 16 graphics modes classified by type, column and row values, and the number of colors they use.

### SETCOLOR

Another command used to produce graphics on the Atari 600XL or 800XL is **SETCOLOR**. There are 256 colors available, and these are generated by using three numbers, one for color regis-

ter, one for hue, and one for luminance. To set a color, therefore, you start with the **SETCOLOR** command followed by a register number, a hue number, and a luminance number. If you wanted to set register 2 with hue 3 and luminance 6, for example, you would type **SETCOLOR** 2,3,6. You can use any number from 0 to 2 for register; 0 to 15 for hue; and 0 to 14, even numbers only, for luminance. *Hues* are simply basic colors you can use to draw your pictures while *luminance* numbers control the brightness from 0 (very dark) to 14 (almost white).

The register number determines which part of the picture is going to be affected. (Register 0 controls the graphics color, register 1 controls the character color, and register 2 controls the background color.) Therefore, if you have drawn some graphics and you would like to change the color of the graphics, you would use register 0. To change the color of the text on the screen, you would use register 1, and to alter the background color, you would use register 2. Table 3-2 shows the different hue colors and their numbers.

Table 3-2. **HUE COLORS**

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

### COLOR

Another command used to enhance graphics is **COLOR**. The **COLOR** command selects a color register to use with the **PLOT** statement, which draws graphics. In other words, the **COLOR** command gives your graphics a color so they will be visible. If you do not change the **SETCOLOR** command as explained above, here are the results of the **COLOR** command:

| | |
|---|---|
| **COLOR** 0: | changes the graphics to black (invisible) |
| **COLOR** 1: | changes the graphics to orange |
| **COLOR** 2: | changes the graphics to green |
| **COLOR** 3: | changes the graphics to dark blue |

Because monitors vary somewhat, it's difficult to say exactly what colors will appear as a result of certain hue/luminance/color combinations. Nevertheless, the hue chart in Table 3-2 should give you a good idea of what type of color you'll encounter when the **COLOR** statement is used; the luminance value changes the degree of brightness of that color. The best advice I can give is: *experiment* to find your desired color, and use the hue chart as a guideline.

### PLOT

You now know how to enter a graphics mode and how to determine which color will be used when drawing graphics, but what are the commands for actually producing graphics? The first of these commands is **PLOT**, which will produce a single point on the screen at a certain place. The format for this command is **PLOT** X,Y, with X and Y being the coordinates on the screen. If you wanted to put a point at position 20,25, for example, you

would type **PLOT** 20,25. To draw a line from position 20,25 to position 40,25, you would type the following:

```
FOR X=20 TO 40:PLOT X,25:NEXT
```

### DRAWTO

A more advanced graphics command is **DRAWTO**. This command draws a line from the last position of the cursor (or the last point drawn) to the X,Y location you specify. After you use the **DRAWTO** statement, the cursor will be repositioned at the X,Y location you specified. For example, if the cursor is at 5,10 and you type **DRAWTO** 40,50, the computer will draw a line from 5,10 to 40,50 and the cursor will move to position 40,50.

## Sample Programs

Now that we know the important graphics commands of the 600XL or 800XL, I will demonstrate their use in some sample programs. (Remember that imagination comes into play when using graphics, so don't feel restricted in the various pictures, graphs, and effects you create on your computer's monitor.)

My first sample program uses the **PLOT** and **DRAWTO** commands to draw lines from the upper-left corner (position 0,0) to random points on the screen. This program creates many lines, all originating at position 0,0. To begin the program, I set the color to register 1 (orange), go into graphics mode 15, and add 16 to my **GRAPHICS** command so that the full screen will be used for graphics:

```
10 COLOR 1
15 GRAPHICS 15+16
```

I then position the cursor at the point of origin:

```
20 PLOT 0,0
```

After this, I program the computer to choose two random numbers: X, ranging from 0 to 159; and Y, ranging from 0 to 191. The random-number generator in the Atari computer will

select a random number between 0 and 1, such as .23523523, .67452, or .82724. You can get a random number with the **RND** command by typing **PRINT RND**(1) or **LET** A = **RND**(1) or by using **RND** any way you want. To get a whole number, you have to use a technique that allows you to give a variable a random number between 1 and another "high limit." The formula is as follows:

variable = **INT(RND**(1)*high limit) + 1

The variable is any variable you want to use, and the "high limit" is the highest random number you want the computer to select. For example, if you want the variable A to equal a number between 1 and 150, you would type in the following:

A = **INT(RND**(1)*150) + 1

With this in mind, you'll see that with the two statements below, the computer will give the variable X a random number between 0 and 159 and the variable Y a random number between 1 and 191. The random numbers will be one digit lower than in our previous example because we eliminated the " + 1" in the formula. The " + 1" adds 1 to the random number selected.

```
30 X = INT(RND(1)*160)
40 Y = INT(RND(1)*192)
```

I then tell the computer to draw a line from the point of origin to the position X,Y and then repeat the process by going to line 20.

```
50 DRAWTO X,Y
60 GOTO 20
```

The effect produced with this program is shown in Fig. 3-2.

The second sample program is almost exactly like the first except that the cursor does not return to the point of origin. By repeatedly using the **DRAWTO** statement so that a line is drawn to

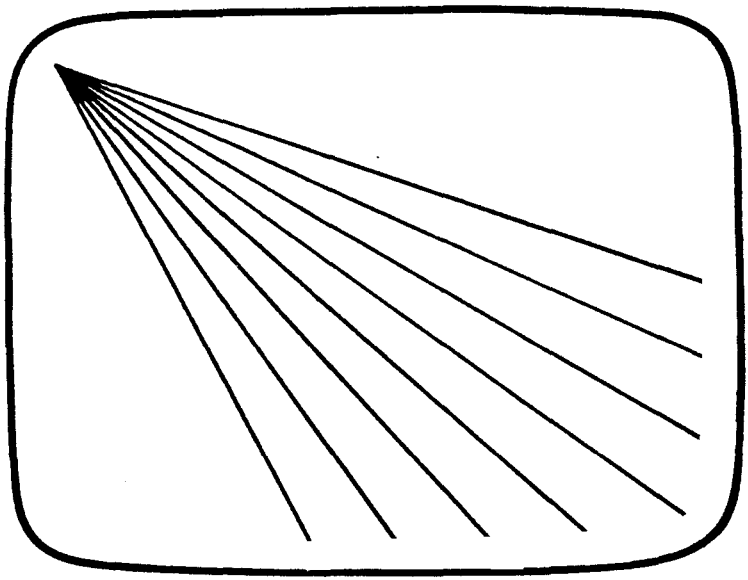

**Fig. 3-2 Graphics Program One**

X,Y from the point where it just ended (Fig. 3-3), the program creates a long, continuous line with many angles. The line will not stop until I BREAK the program. As you will see, the only change in the formula is that line 20 has been replaced by a **REM** statement. A **REM** is simply a *rem*ark for your own purposes—the computer ignores the remark. If there is a **REM** in a program line, everything in that line will be ignored by the computer. The computer won't read it, use it, print it, or do anything with it, so you can type anything in it from "This is where the **FOR/NEXT** loop begins" to "Togoman's coming!".

```
10 COLOR 1
15 GRAPHICS 15+16
20 REM This line has been removed
30 X=INT(RND(1)*160)
40 Y=INT(RND(1)*192)
50 DRAWTO X,Y
60 GOTO 20
```

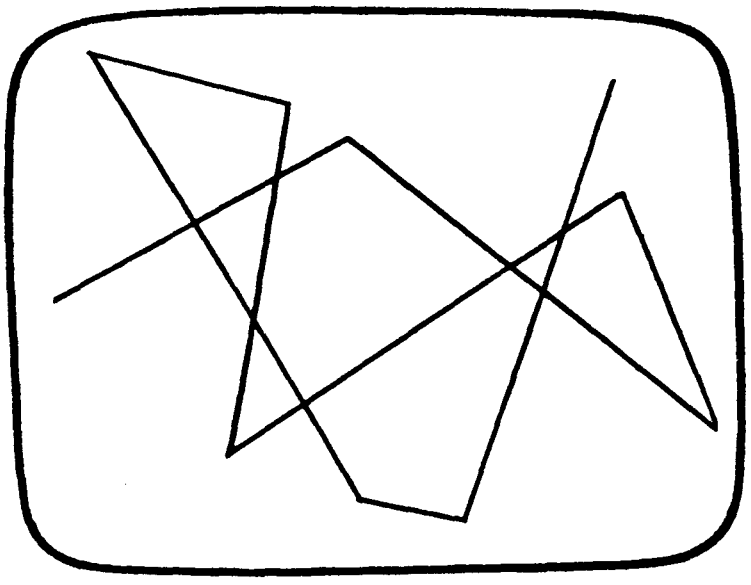

**Fig. 3-3 Graphics Program Two**

There is another small modification that can make the program do something entirely different. If you change line 50 from a **DRAWTO** statement to a **PLOT** statement, a "starry-sky" effect can be achieved. Simply make line 50:

```
50 PLOT X,Y
```

These are three simple programs that merely demonstrate how commands can be used. We shall now take a look at some more complex programs, programs you might want to incorporate into your own creations. The first program, called "Perspective," produces a "three-dimensional" graphics effect on your computer screen by drawing a number of lines from the bottom of the screen to a point near the middle of the screen. These lines, coupled with a horizontal line at the bottom of the monitor, create graphics which look like many long roads leading deep into space (Fig. 3-4).

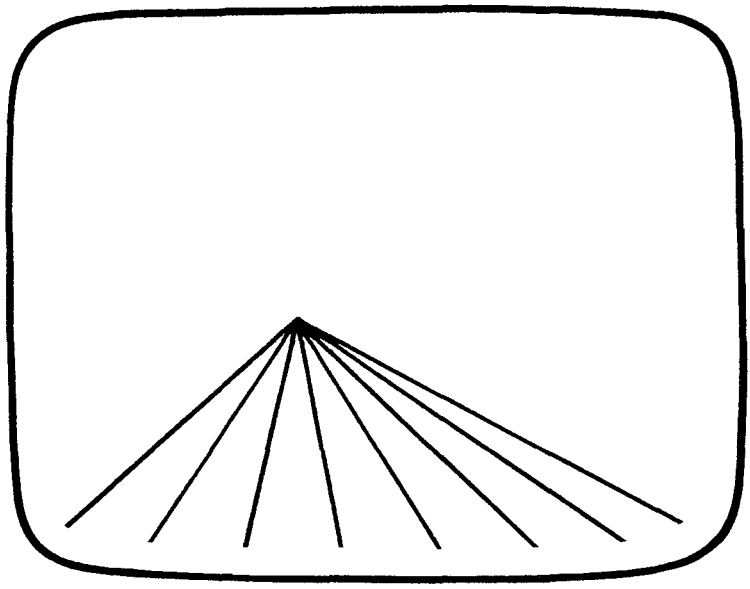

**Fig. 3-4 Perspective program**

```
10 GRAPHICS 14+16
15 COLOR 1:SETCOLOR 1,2,4
20 PLOT 0,191:DRAWTO 159,191
30 FOR X=0 TO 159 STEP 10
40 PLOT X,191
50 DRAWTO 80,96
60 NEXT X
65 PLOT 159,191:DRAWTO 80,96
70 GOTO 70
```

You may have noticed that lines 15, 20, and 65 each give two separate commands to the computer. This is possible because there is a colon (:) separating the commands. You can put several different commands to the computer in one program line so long as the commands are separated by colons. This program will continue running until you hit the BREAK key.

Another "three-dimensional" program is shown below. This program draws a box (not exactly a cube) on the screen. The **PLOT** statements position the cursor, and the **DRAWTO** statements draw the lines which make up the box.

```
10 COLOR 1:GRAPHICS 15+16
20 PLOT 10,10:DRAWTO 100,10:DRAWTO 100,100:DRAWTO
   10,100:DRAWTO 10,10
30 PLOT 30,30:DRAWTO 120,30:DRAWTO 120,120:DRAWTO
   30,120:DRAWTO 30,30
40 PLOT 10,10:DRAWTO 30,30
50 PLOT 100,10:DRAWTO 120,30
60 PLOT 10,100:DRAWTO 30,120
70 PLOT 100,100:DRAWTO 120,120
80 REM Box finished
90 GOTO 90
```

The rectangles were drawn by lines 20 and 30 in this program, while lines 40 through 70 drew the "connecting lines" between the two rectangles, thus forming the complete box.

## Graphs

If you want to use graphics for plotting or business purposes, you need to know how to create graphs with your graphics. This is not difficult, because you only have to draw the *x*-axis, the *y*-axis, and the data (in whatever graph form you want) onto the screen. Below is an example of a graph that uses thin vertical lines to plot random data. The axes are drawn, random vertical points are selected, and lines are drawn from the bottom of the *y*-axis to the random vertical point. Together, all these lines form an interesting random graph effect. Another command, **SOUND**, is used to create tones while the graph is being drawn. (The **SOUND** commands will be explained in Chapter 4.)

```
10 COLOR 2:GRAPHICS 15+16
20 PLOT 10,10:DRAWTO 10,191
```

```
 30 DRAWTO 159,191
 35 FOR X=11 TO 149
 40 Y=INT(RND(1)*192)
 50 PLOT X,191:DRAWTO X,Y
 60 SOUND 0,Y,10,15
 70 NEXT X
 80 SOUND 0,0,0,0
100 GOTO 100
```

Note that sounds are produced corresponding to the random Y variable, so the higher the graph goes, the higher the pitch of the sound will be.

A few small modifications in this program will produce bars instead of lines for the random graph. This is done by changing the single **DRAWTO** statements (which produces a vertical line) into a number of **DRAWTO** statements (which make a bar). In addition, the **STEP** of the **FOR/NEXT** loop must be increased because bars have to be spaced farther apart than do lines, which require no spacing. To make a bar graph, therefore, change line 35 to:

```
35 FOR X=11 to 159 STEP 5
```

Then add this line:

```
55 DRAWTO X+3,Y:DRAWTO X+3,191
```

Although these graphs are completely random, you will find a useful and interesting program for forming your own graph in Chapter 8 of this book.

As you explore the graphics of the Atari 600XL or 800XL, you should get some ideas of your own on how the "picture power" of this home computer can be used. You might want to put graphics into games, business programs, or utility programs; but whatever your reason for using graphics of the Atari 600XL or 800XL, you should get full use of its 256 vibrant colors and its ability to create nearly any shape in fine detail.

# The Sounds and Music of the 600XL or 800XL

One of the most powerful and useful functions of the Atari 600XL or 800XL is its ability to create sounds and music. *Sounds* are the tones you create to represent things such as rocket blasts or laser guns going off, or "user prompting" signals to tell a person to do something while using the program, such as press a certain key. *Music* is the combination of notes to form songs. The tones used to produce notes are much longer than the tones used to generate sounds because sounds are really a great many very short notes compressed so closely together that they make a particular sound effect.

There is only one special command used to produce sounds and music on your Atari 600XL or 800XL. This command is, appropriately enough, **SOUND**, and its format is:

**SOUND** (voice, pitch, distortion, volume)

This sound command can support all four *voices* (sound channels) on the Atari, and you can create virtually any sound effect or song with this powerful command and the four voices it supports. The elements in the **SOUND** command are as follows:

**Voice:** a number, ranging from 0 to 3, which specifies which voice will be used. Each voice requires a separate **SOUND** command.

**Pitch:** a number ranging from 0 to 255, which specifies the frequency of your sound. The larger the number, the lower the pitch. A frequency value of 255 would be a very low bass tone; a frequency value of 0 would be a very high-pitched tone.

**Distortion:** a number, ranging from 0 to 14, which specifies the distortion value. A value of 10, for example, will give you music, while the number 12 will create a buzzing sound useful in sound effects. Try each of the numbers, along with different pitches, to see what sound effects you can create.

**Volume:** a number ranging from 1 to 15, which specifies the volume of the sound. The number 1 will create a barely audible tone; the number 15 will create a loud sound.

Table 4-1 shows musical notes and their corresponding pitch values so that you can create music accurately with the **SOUND** command.

## Sample Programs

Now that we know how to use the **SOUND** command I will demonstrate its use in sample programs. You can type them into your Atari exactly as they are written here, **RUN** them, and listen to what is played. I think you'll be impressed by the power your computer possesses.

My first sample program uses the pitch variable because it moves rapidly down the pitch scale to create lower and lower sounds. You might use this for a special sound effect, such as a bomb being dropped. The first line sets up the **FOR/NEXT** loop that will take the pitch value from high (0) to low (255).

```
10 FOR X=0 TO 255
```

Then the **SOUND** command is used to create a noise in voice 2 with a pitch of X, a distortion value of 10, and a volume of 12:

```
20 SOUND 2,X,10,12
```

Table 4-1. **MUSICAL CHART**

| | Note | Pitch Value |
|---|---|---|
| High notes | C | 29 |
| | B | 31 |
| | A sharp | 33 |
| | A | 35 |
| | G sharp | 37 |
| | G | 40 |
| | F sharp | 42 |
| | F | 45 |
| | E | 47 |
| | D sharp | 50 |
| | D | 53 |
| | C sharp | 57 |
| | C | 60 |
| | B | 64 |
| | A sharp | 68 |
| | A | 72 |
| | G sharp | 76 |
| | G | 81 |
| | F sharp | 85 |
| | F | 91 |
| | E | 96 |
| | D sharp | 102 |
| | D | 108 |
| | C sharp | 114 |
| Middle C | C | 121 |
| | B | 128 |
| | A sharp | 136 |
| | A | 144 |
| | G sharp | 153 |
| | G | 162 |
| | F sharp | 173 |
| | F | 182 |
| Low notes | D | 193 |
| | D sharp | 204 |
| | D | 217 |
| | C sharp | 230 |
| | C | 243 |

Finally, the **FOR/NEXT** loop is completed with the **NEXT** statement. Remember to specify the variable used in the **FOR/NEXT** loop whenever the **NEXT** statement is used because the Atari will give you an error code if you don't.

```
30 NEXT X
```

You can **RUN** the program now to hear the sound.

You might want to try some other effects with this program. For example, you can type in a new line specifying a sound in voice 1 with a distortion value of 10, a volume of 12, and a pitch value which is exactly the opposite of voice 2's (for example, when voice 2's pitch value is 255, voice 1's is 0; when 2's is 254, 1's is 1; when 3's is 253, 1's is 2, and so on). This creates an interesting effect which makes the two voices sound like they are approaching one another and then moving apart.

```
25 SOUND 1,255-X,10,12
```

Or you can add a line specifying a sound in voice 3 with a pitch value slightly higher than voice 2's. This will distort the total sound even more.

```
15 SOUND 3,X+5,10,12
```

The second sample program, "Random Noises," takes advantage of the **RND** statement to make a rapidly changing noise. Here is the formula. It first creates a random number between 0 and 255; makes a sound on voice 1 with a pitch of X, a distortion value of 10, and a volume of 15; and then repeats the process by going back to line 10.

```
10 X=INT(RND(1)*255)+1
20 SOUND 1,X,10,15
30 GOTO 10
```

To make this program sound even more horrendous, add the following line 15 to the program. (It will create a value for R, the volume of the sound.) Then, add line 17 to create a value for V to determine which voice shall be used. After you have added these two lines, change line 20 to look like the one below, so that the variables R and V can be used in the **SOUND** command.

```
15 R=INT(RND(1)*15)+1
17 V=INT(RND(1)*3)+1
20 SOUND V,X,10,R
```

As you might guess before you even run this program, the sound created will be pretty horrible. The whole program basically creates a random sound for a random voice with a random volume; and it continues to do this until there are three voices working with one another to make a menagerie of sounds. Remember that a sound does not stop playing until the **END** command is given, another sound command is used for that voice, or the SYSTEM RESET key is pressed.

The third sample program uses the distortion element to show you the different sounds possible with the Atari. After you type in this program and **RUN** it, simply enter a distortion value (from 0 to 14) and listen to the sound produced. It should give you a good idea of what distortion value to use to create a particular sound.

```
10 PRINT"Please enter a number for distortion ";
20 INPUT D
30 SOUND 1,200,D,15
40 GOTO 10
```

Instead of hearing a single tone distorted, you might want to hear a great many pitches in distorted mode. If so, add the following lines 25 and 35 to the program and modify line 30 as shown to make the computer play pitches from 1 to 200 while distorting the sound.

```
25 FOR F=1 TO 200
30 SOUND 1,F,D,15
35 NEXT
```

You have now heard the different effects which are produced when you change each of the four variables for the **SOUND** command; but you haven't really heard any genuine sound effects yet. I have therefore prepared three sounds for the program below. Simply type in the program, **RUN** it, and select a sound from the choices given.

```
 5 PRINT"(1) SIREN"
 6 PRINT"(2) LASER"
 7 PRINT"(3) TANK"
10 PRINT"Please select a sound from the choices given ";
20 INPUT C
30 ON C GOSUB 50,60,70
40 PRINT:SOUND 1,100,1,0:GOTO 5
50 FOR N=1 TO 5:FOR C=1 TO 10:SOUND 1,C,10,15:
   NEXT C:NEXT N:RETURN
60 FOR N=1 TO 20:SOUND 1,N,12,15:NEXT N:RETURN
70 SOUND 1,200,8,15:FOR C=1 TO 500:NEXT C:RETURN
```

You'll notice how few commands were needed to produce these rather complex sounds because the Atari has such a powerful and simple command to create sounds and music.

Up to now, we've been programming different tones and sounds, but the Atari is also capable of producing music. In fact, you can create complex four-piece musical renditions on your inexpensive but powerful home computer.

To give you a start, here's a program which will play part of a well-known song, using **READ/DATA** statements to determine the notes (variable N) and the length of the notes (variable L), the **SOUND** command to produce the notes, and a **FOR/NEXT** loop to hold the notes for the time specified by the variable L. Note that the **SOUND** command in line 40 is used to stop the sound being played, much like the command used in line 40 in the previous program.

```
10 READ N,L
20 IF N=256 THEN END
30 SOUND 1,N,10,15
40 FOR D=1 TO C:NEXT D
45 SOUND 1,N,10,0
50 GOTO 10
60 DATA 40,250
70 DATA 40,125
80 DATA 35,125
90 DATA 42,250
100 DATA 0,250
110 DATA 45,250
120 DATA 45,250
130 DATA 40,125
140 DATA 47,250
150 DATA 0,500
160 DATA 40,125
170 DATA 35,125
180 DATA 45,250
190 DATA 0,250
200 DATA 45,125
210 DATA 45,125
220 DATA 40,125
230 DATA 47,200
240 DATA 256,256
```

The last **DATA** statement is meant to signal the computer to stop reading other **DATA** statements because there are no others. Note that I used Table 4-1, "Musical Chart," to find the data for these musical notes, and I used a delay value of 125 for eighth notes, 250 for quarter notes, and so on. Lastly, the values of 0 that you see in the data statements for the variable N are meant for rests in a piece of music.

## Game Controllers

Often graphics and sounds are used in games that make use of a joystick or paddle. These *Game controllers*, in lieu of the arrow

keys, make entertainment programs much more enjoyable, and so you might want to use them in some of your programs. The paddle (or dial) controls only two directions (left and right), while the joystick controls eight directions, as shown in Fig. 4-1. The four commands for these controllers are:

**PADDLE:** gives the condition of a particular paddle. For example, if you tell the computer to **PRINT PADDLE**(0), the value of the first paddle will be given. You can put any number from 0 to 3 (for the four paddles) within the parentheses to specify a certain paddle. The value returned by the computer can range from 1 to 228, with 1 representing a condition in which the paddle is turned

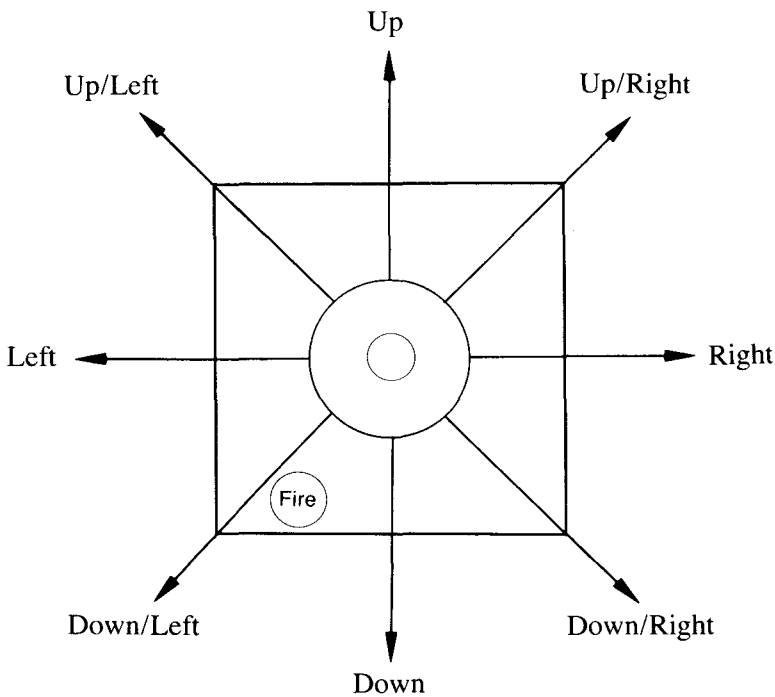


**Fig. 4-1 The eight directions of the Atari joystick**

completely to the right and 228 representing a condition in which the paddle is turned completely to the left. In other words, the farther counterclockwise you turn the paddle, the higher its value will be.

**PTRIG:** tells you whether a paddle trigger is being pressed. A value of 0 indicates that the paddle trigger *is* being pressed; 1 indicates it is *not* being pressed. For instance, you might tell the computer **IF PTRIG**(0) = 0 **THEN PRINT** "You pressed the button on the first paddle." Again, a number from 0 to 3 is acceptable within the parentheses.

**STICK:** like the **PADDLE** command, gives you the status of joystick controllers. However, you can only use a number from 0 to 1 within the parentheses because only two joysticks may be used on the Atari 600XL or 800XL. The numbers returned for certain conditions are:

| | |
|---|---|
| 15 means: | joystick is centered |
| 14 means: | joystick is up |
| 13 means: | joystick is down |
| 11 means: | joystick is left |
| 7 means: | joystick is right |
| 10 means: | joystick is up and left |
| 6 means: | joystick is up and right |
| 9 means: | joystick is down and left |
| 5 means: | joystick is down and right |

Therefore, if a value of 14 were returned from **PRINT STICK**(0), you would know that the first joystick was being pushed up.

**STRIG:** does the same for the two joysticks as **PTRIG** does for the four paddles.

To give you an idea of how joysticks can be used, I have created a couple of sample programs.

The first one, "Joystick," monitors the joystick for up and down conditions, the centered position, and whether it is being fired or not. The computer merely uses the **STICK** and **STRIG** commands to find the condition of the joystick and prints the status on the screen.

```
10 A=STICK(0)
20 B=STRIG(0)
25 IF B=0 THEN PRINT"FIRE!";
30 ON 16-A GOSUB 40,50,60
35 GOTO 10
40 PRINT"CENTERED"
45 RETURN
50 PRINT"UP"
55 RETURN
60 PRINT"DOWN"
65 RETURN
```

Start this program with the joystick in the first joystick port, and move it up and down, pressing the fire button occasionally to see the program in action.

The second sample program uses the **SOUND** and **STICK** (joystick) commands to create a variety of noises. Plug the joystick into port one. Then move it up to make the sound go higher and down to make the sound go lower. Press the fire button to "toggle" between clear musical sounds and sound effects.

```
 5 D=10
 6 F=100
10 A=STICK(0)
20 B=STRIG(0)
25 IF B=0 THEN GOSUB 40
30 ON 15-A GOSUB 50,60
31 IF F>255 THEN F=255
32 IF F<0 THEN F=0
```

```
33 SOUND 1,F,D,15
35 GOTO 10
40 IF D=10 THEN D=8:GOTO 42
41 D=10
42 RETURN
50 F=F-1
55 RETURN
60 F=F+1
65 RETURN
```

As you have probably already figured out, F is the pitch of the sounds generated and D is the distortion value. This is a simple application of the joysticks, but it shows you how helpful "extras" like game controllers can be.

Now that you have been introduced to the sound and game-controller commands of the Atari 600XL or 800XL, you should have some ideas of your own on how you can use these commands. You might want to program your favorite song or add exciting sounds to a game you have created. Whatever your application of these commands, you should make full use of the Atari's four voices and versatile game controllers because you can learn most about your computer by using all its power.

Chapter Five

# The Atari 600XL or 800XL at Home

The usefulness of the Atari 600XL or 800XL computer is most apparent in the home. There are a great number of purposes which the 600XL or 800XL may serve in your house, and we'll be exploring the most important of these in this chapter.

Briefly, here are the different uses we'll be covering in this chapter:

**Entertainment:** both "serious" and "light," including strategy games, adventure and fantasy games, and arcade games

**Education:** including education for preschoolers, education for elementary and secondary students, and education for adults interested in learning more about math, languages, or other subjects

**Finances and record keeping:** to help you keep your files organized and your tax planning and home banking up to date

**Word processing:** one of the most important uses for a home computer; discussed in detail in this chapter as well as the next

**Communications:** use of modem to call up bulletin board services, large networks, and even your friends for a friendly chat or an exchange of programs

Of course, there are many more uses for computers in the household, and you might think of some others while reading this chapter. But the five shown here are the ones best represented by software already on the market and the ones most in demand by people who buy the Atari 600XL or 800XL home computer.

## Entertainment

By far the best selling and most popular type of software is in the entertainment category. There are several different kinds of entertainment software, some more popular than others, and each type appeals to nearly every age level. Many of these entertainment programs cost around $30, but some are so well-made and exciting that they are well worth the money. Others, of course, aren't worth the diskette on which they're recorded.

Games may not seem useful to you, but they *are* fun, and so they serve the purpose of entertainment. More important, the Atari 600XL or 800XL computer has such powerful graphics and sound capabilities that its most obvious use is for games. The different types of game categories made for the Atari include:

**Arcade:** the fast-action games, with flashy graphics, excellent sound, and a lot of entertainment value because they appeal to a wide assortment of people, especially young people. These are the most popular games, and you'll find a tremendous assortment of them for the Atari computer.

**Adventure and fantasy:** take a more intellectual approach to entertainment. You might want to try these occasionally mind-boggling programs, which usually involve guiding someone through a mysterious place such as outer space or a haunted mansion. These "puzzle-solving" games involve strategy, memory, and luck, but not the hand-eye coordination needed for arcade games. This is the second most popular category of software, and although these games usually lack graphics and sound, they are excellent for entertainment and will occupy hours of your time as you try to solve a mystery, find a treasure, or just stay alive.

**Strategy:** the least popular games because they are usually adaptations of board games onto the computer, such as a computerized version of Monopoly, Othello, or chess. If you enjoy board games, you might want to try the computer version, but it just seems to lose something in the transition.

There are hundreds of games that will work on the 600XL or 800XL, the majority being the arcade variety. Here are some of the most popular pieces of entertainment software currently available, along with their prices.

**CHOPLIFTER:** (Broderbund Software, Inc., 1938 Fourth St., San Rafael, CA 94901)—This game involves flying a helicopter to a rescue site where you save hostages being guarded by enemy tanks, aerial mines, and jets. This is an exciting game with a lot of coordination required, but it will cost you $35.95 to get the diskette with the program on it.

**DEADLINE:** (Infocom, Inc., 55 Wheeler St., Cambridge, MA 02138)—This is not an arcade game, but it is still tremendously popular with Atari owners. In this game, you are a private eye who must solve a murder mystery within a certain amount of time. The manufacturer of this program also manufactured the popular Zork adventure series (which you might want to consider buying) as well as Suspended. Deadline costs $49.95, but if you like adventures, you won't stop playing this game.

**DEMON ATTACK:** (Imagic, 981 University Ave., Los Gatos, CA 95030)—This was one of the first games released by Imagic, and its inventor made $1 million from it. If Imagic didn't have a royalty ceiling of $1 million, the inventor would still be collecting money. Its popularity is a good indicator of the quality of this fast-paced arcade game, which has you controlling a laser cannon and shooting flapping demons in space. The Atari version of this program costs $34.95.

**FLIGHT SIMULATOR:** (SubLogic, 713 Edgebrook Drive, Champagne, IL 61820)—SubLogic's simulation of controlling an

airplane from the cockpit is nothing short of incredible. Controlling the many functions of this simulation will keep you very busy, but if you want to know some of what's involved in flying a simple prop plane, this program will show you. If you want to try your hand at an air battle, there's an option in the program that will make you a World War I ace. SubLogic sells the program to Atari owners for $49.95—a bargain for the experience of flying an airplane.

**PINBALL CONSTRUCTION SET:** (Electronic Arts, Inc., 2755 Campus Drive, San Mateo, CA 94403)—Bill Budge, the famed maker of Raster Blaster, teamed up with Electronic Arts to make this program, which allows you to make and play your own pinball machines. You can position bumpers, flippers, or gates and change the laws of gravity if you want to. The $40 price is well worth it.

**STAR RAIDERS:** (Atari, Inc., 1265 Borregas Ave., Sunnyvale, CA 94086)—One of the most highly praised arcade games ever made, Star Raiders puts you in the middle of space, trying to destroy enemy ships attacking your star bases. On the video screen, you "see into space" and can observe the stars, star bases, asteroids, enemy ships, and anything else that might be in space. As you "move," the stars, along with other objects in space, seem to pass by you. It is an exciting game because the images on the screen are so lifelike, right down to the explosion of the enemy ship when you hit it. I highly recommend Star Raiders because each of its games is somewhat different, and the graphics and sounds are absolutely amazing. The price for this classic is about $40.

**ZAXXON:** (Created by Sega, manufactured by Datasoft, 19519 Business Center Drive, Northridge, CA 91324)—Datasoft put the arcade experience of this three-dimensional game onto the screens of Atari computers everywhere, and they did a fine job. You steer your spaceship through outer space and over the surface of an asteroid—destroying planes, mobots, fuel tanks,

and other objects as you go—until you face the powerful Zaxxon, which is just about impossible to destroy (for me, at least). The cost for this program is $39.95.

It is wise to try game software out at the store or to read a review or two about it before actually purchasing it. If you are going to pay $40 for an entertainment program, it should be entertaining.

## Education

The teaching potential of computers has been recognized ever since personal computers first appeared, and there is still a great deal of educational software on the market. But only a few pieces of this software are any good. Programming a computer to teach a human is no easy task, and only a few companies have been successful in their attempts to create high-quality educational software.

A computer is especially good at teaching math, languages, and simple concepts such as the alphabet and numbers because each of these can be represented easily on a video screen. For this reason, it is best suited to teaching very young children.

Good educational software takes advantage of a computer's graphics (to display letters, numbers, or shapes) sound (to prompt, congratulate, or inform a person he or she was incorrect), and memory (to keep track of scores, to store tests which can be given to the user, and to retain information about the problems the user is having with some parts of the instruction).

It is more difficult to program software for older children. Only Control Data Institute has had any success so far in producing educational programs for high school children.

Some of the most popular educational programs include the following:

**PLATO:** (Control Data Corp., 8100 34th Ave. South, Minneapolis, MN 55440)—This is actually a series of programs, not a single program. In the series are programs designed for children through the eighth grade who need help in reading, writing, or

arithmetic. There are also some programs for teaching high-school science. Each program within the series costs about $40 to $70, so education with an Atari is not an inexpensive proposition.

**CHATTERBEE:** (Don't Ask Computer Software, 2265 Westwood Blvd., Suite B-150, Los Angeles, CA 90064)—This is a game which teaches you how to spell. The gimmick here is that the game *talks*. The computer will use one of the 2500 words in its memory within a spoken sentence so that you can hear the word in context. If you misspell the word, the computer will teach you the correct spelling. There are multiple learning levels in this excellent game, and excellent graphics and music accompany the program ($39.95). This is how an educational program should be—both entertaining and good at teaching useful subjects.

**FRACTION FEVER:** (Spinnaker Software, 215 First St., Cambridge, MA 02142)—With this program, a young child learns fractions by matching them with pictures that appear on the screen. Children aged 7 to 13 should enjoy this program; and at $34.95, it's quite a value.

**LEARNING WITH LEEPER:** (Sierra On-Line, 3657 Mudge Range Rd., Coarsgold, CA 93614)—This program is strictly for the very young (ages 3 to 6). It teaches colors, numbers, and the alphabet. This $34.95 program will help preschoolers develop basic skills while it familiarizes them with the computer.

**PAINT WIZARD:** (Datamost, 8943 Fullbright Ave., Chatsworth, CA 91311)—Because the Atari has such outstanding graphics, Paint Wizard is an excellent educational program for this machine. Children aged 7 to 11 learn about shapes and graphics by making a design and then shrinking it, expanding it, or filling it with bright patterns. The program costs $39.95.

**TYPE ATTACK:** (Sirius Software, 10364 Rockingham Dr., Sacramento, CA 95827)—Type Attack might be mistaken for an arcade game because of its explosions and other features, but it is

really a program designed to familiarize children with the computer keyboard. A letter appears on the screen, and children must press the correct letter to shoot it out of the sky. This is an excellent program for teaching typing skills to children aged 7 to 12. It costs $39.95.

Educational programs not only serve the purpose for which they were made but also help children learn how to use computers. Educational software still has a long way to go, but if you select a program from a reliable, well-known company, you should acquire something which a child will both enjoy and can use in his or her learning. At the same time, you might look for some software for yourself because there are educational programs on the market designed to teach foreign languages, science, and even computer skills to adults.

## Finances and Record Keeping

Programs designed for home finances are primarily for tax-planning purposes and home banking. Home banking programs will establish the budget for your home, balance your checkbook, keep track of your interest payments and charge accounts, keep files of your tax-deductible spending, plot trends in your home banking, and so on. You might not think that a program to watch your own finances is necessary, but you probably could use a tax-planning program (the cost is tax-deductible) to help you once a year in calculating your taxes, deductions, and so forth.

Electronic record keeping is not a necessity in most homes. Many people think that a computer would be useful for storing recipes or other information usually stored in files. Indeed, you can put recipes into the computer's memory, but it will probably take you longer to retrieve a recipe from a computer than from a cookbook, so there isn't any real advantage to having recipes on a diskette. Anyway, who wants a computer crowding up the kitchen?

An electronic filing program can be useful, however, if you work at home even part of the time. You can file data for what-

ever needs the convenience of mass storage with speed, cross-indexing, and fast-search capabilities. If you are self-employed, a computer will certainly increase your productivity and efficiency.

As a rule, buy a software package for storing information or helping you manage your money only if you truly need it. It always surprises me how many people "create a need" for software and buy it only because a computer is supposed to make everything more efficient. In some cases, computers can slow down work, so don't create needs for yourself—simply recognize them. This rule will save you money, time, and a lot of frustration with programs you might not need.

## Word Processing

Word processing is the best use for a home computer. If you do any writing at all (even just writing to friends), using a word processor will double or triple your efficiency. You don't have to worry about rough drafts, correction fluid, incorrect margins, retyping, or any of the other frustrations that accompany the use of a typewriter.

The basic idea behind a word processor is that what you type into the computer is displayed only on the video screen. It is not physically printed anywhere, so you don't have to worry about physically erasing it. A word processor can do the following:

1. Delete words you have typed into the computer
2. Insert words, letters, or any amount of text anywhere within a document
3. Search for text
4. Search for text and replace it, all with one command
5. Move a whole block of text (such as a paragraph or several sentences) to a different part of a document

While you are typing in a document, the computer will automatically "wrap" the words on the screen. That is, instead of splitting a word by printing part of it at the end of a line and the

rest of it at the beginning of the next line, the computer will automatically bring the whole word onto the next line. This makes your text look neat. In addition, when you finally print your document onto paper (with a printer), the right margin will usually be *justified*; that is, it will be lined up perfectly with the margin so that both margins are straight. This gives the final printed document a very neat appearance. Also, the pages will be numbered, headers and footers you type will be printed, and characters will be printed in italics or boldface as you specify.

Besides the advantages of being able to modify documents and print them in a large variety of ways, there is an added convenience of word processing—you have a permanent copy of every letter, chapter, or essay you create and save on a diskette. If you ever need to refer to a document or print it again, you can do so in a matter of seconds. An excellent example of this is the time I wrote a lengthy essay for school, and the teacher accidently lost it. Instead of having to write the whole paper over, it took me about 20 seconds to recall the document and print it for the teacher. You must take care, though, to treat your diskettes well and to make backups of each one.

There are a number of word processors on the market which can be used with the Atari, but here are a few I would personally recommend:

**ATARIWRITER:** (Atari, Inc. 1265 Borregas Ave., Sunnyvale, CA 94086)—This is a simple-to-use yet sophisticated word processor with easy editing capabilities, special print styles, and safety features to make sure you don't accidently erase a document you want to keep. We will be examing AtariWriter closely in Chapter 6. It retails for $99.00.

**BANK STREET WRITER:** (Broderbund Software, Inc., 1938 Fourth St., San Rafael, CA 94901)—This excellent word processor, at $69.95, has many of the same features as AtariWriter, but it is even easier to use. Each command is fully explained and shown on the screen, a special tutorial (instruction document) is included to help you get started, and the program has been heav-

ily tested to ensure ease of use and bug-free operation. The only disadvantage is that it requires 48K, which eliminates its use with the Atari 600XL.

**LETTER PERFECT AND SPELL PERFECT:** (LJK Enterprises, 7852 Big Bend Blvd., St. Louis, MO 63119)—Although the Letter Perfect word processor doesn't have all the features of the above two word processors, it does have some nice extras, such as its ability to support *80-column boards* (modifications to the computer which will allow it to display 80 characters per line rather than 40) and load-in data from LJK's Data Perfect data base. It retails for $99. And, the Spell Perfect spelling checker will tell you how many words there are in a document and will check all the words for proper spelling. If any words are misspelled, you can correct them. It retails for $79.

## Communications

### Modems

Computer communications are accomplished with a device called a *modem*, which can be hooked up to your Atari 600XL or 800XL computer. The modem is basically a hardware device that connects your home computer to the "outside world" via the telephone. You can call up other computers via the telephone and exchange information with them. With this power, you can exchange programs, receive information, send information, talk with friends, or even post advertisements in computer memories which other people can read.

Some of the modems I would recommend for computer communications are the following:

**ATARI 1030 DIRECT CONNECT MODEM:** (Atari, Inc., 1265 Borregas Ave., Sunnyvale, CA 94086)—This is the standard Atari modem ($149). It comes equipped with the ModemLink program so that you can get involved in the "world connection" immediately.

**1080 VERSAMODEM:** (Bizcomp, P. O. Box 7498, Menlo Park, CA 94025)—This is a standard 300-*baud* (bits-per-second) modem which connects directly to the Atari and retails for $199.

**VOLKSMODEM:** (Anchor Automations, 6624 Valjean Ave., Van Nuys, CA 91406)—This is probably the most incredible value in a modem yet. For less than $100, you get a direct-connect modem for your Atari, a required $12.95 cable, and, even more amazing, a $100 subscription to The Source. The subscription almost makes this a "free" modem!

Sending information, programs, and mail at nearly the speed of light is pretty exciting, and so computer communication has received a lot of publicity. A modem is a smart investment because it is an extremely useful tool and an entertaining addition to your Atari 600XL or 800XL home computer.

### Connections

There are several types of computers you can call with your computer and your modem. The most simple connection is with another home computer. You might call a friend and ask for a program, which he or she can then transfer through the telephone line. Or you might simply chat with your friend through the computer. As you type on your keyboard, the words will appear on your screen and be sent to his or her screen as well. As he or she types, you will see the words on your screen.

On a larger scale, you can call a *bulletin board service* (BBS), a small computer that can store and retrieve messages, store whole programs, and do basically anything a home computer can do. The bulletin board service takes care of itself, and one person at a time can call the board to leave or retrieve messages, chat with the system operator who owns the board, get a program from the computer's memory, or do some electronic shopping by browsing through a catalog (shown on the video monitor) and selecting items he or she would like to purchase.

The largest type of computer a computer owner can call is a *data base*, such as CompuServe or The Source. These can sup-

port thousands of people at the same time, and you can perform a wide variety of functions on these huge computers, such as send and receive electronic mail, play games, chat (via your keyboard) with hundreds of other people, get financial information, or even read a newspaper article. This "world connection" of computer owners is extremely useful because it gives you instant access to a huge variety of information. Furthermore, it is an excellent way to meet people.

Here are some of the most popular data bases in the United States. You might want to write to one of them to find out more about their services:

**COMPUSERVE INFORMATION SYSTEM:** (CIS) (5000 Arlington Centre Blvd., Columbus, OH 43220)—CompuServe is a powerful network with access to major newspapers, CB radio, electronic mail, a bulletin board, and dozens of other services. You can get the current stock prices from Quick Quote, historical data from MicroQuote, and business data from Standard & Poor's and Value Line. The service costs $5 an hour in non-prime time and $22.50 during business hours. The purchase of a start-up package will give you the software you need as well as a free hour of time on the system.

**THE SOURCE:** (1616 Anderson Rd., McLean, VA 22102)—Like CompuServe, The Source is an information system from which you can get a daily news summary, financial information, or movie reviews. It also offers electronic shopping and a games section. The $100 hookup charge is high, and it costs from $7.75 to $20.75 an hour to use this service.

**DOW JONES RETRIEVAL:** (P.O. Box 300, Princeton, NJ 08540)—This is a service for serious investors and businesspeople. It gives up-to-date information on stocks, bonds, options, and corporate finances, and also offers general news, sports, weather, and articles from *The Wall Street Journal* and *Barron's*. There is a $50 hookup fee, and it costs from 15 cents to $1.20 a minute to use the service.

If you don't want to spend the money to hook up to a data base, you can still send and retrieve information with your computer via the telephone lines. You can also use a free bulletin board, of which there are many (there are about 1000 bulletin boards in the United States currently owned and operated by individuals). You might even become a bulletin board operator yourself one day, once you learn more about this "world connection." Or you might be happy just communicating and exchanging programs with your friends on a periodic basis. The choice is yours.

### Modem Software

**TELETARI:** (Don't Ask Software, 2265 Westwood Blvd., Suite B-150, Los Angeles, CA 90064)—Here is an advanced communications program, for $39.95, which features a *buffer* (storage area) of 20 kilobytes, a simple menu-driven format, and the power to support fast modems with 1200-baud speed.

## The Useful Computer

As you can see from this chapter, your Atari 600XL or 800XL is an extraordinarily useful machine in the home. You have probably already thought of other ways in which you can use it, but the uses I have discussed are the ones for which a tremendous amount of software is available.

I urge you to read at least two reviews of a program and to try it out in the store before you actually purchase it. The Atari is indeed an excellent machine, and there is a lot of useful software for it, but there are also a lot of low-quality programs which will prove to be useless, frustrating, and a waste of time and money. If you go to a reputable software store and try out the program, you can find out if it has the features you want, the ease of use you need, and the specifications to fit your computer and your needs.

## Chapter Six

# Using AtariWriter

*Word processing* is a system for writing something with the aid of a computer. The system can be used to write an essay, a letter to a friend, or a chapter of a book. I discussed this "magic typewriter" earlier, but it would be worthwhile to take another look at the word processor, its uses, and its advantages.

Word processors can be used to create, edit, and print documents. Because they can be used to alter, delete, move, or insert copy anywhere in a document before the document is actually printed out, they eliminate the need for "white-out" fluid, the retyping of rough drafts, and other hassles which accompany the more traditional methods of writing (by hand or on a typewriter). You can transform your Atari into a word processor by using it with a word-processing program. There are a number of these programs on the market, some better than others, but we'll be examining only one of the better ones in this chapter: the AtariWriter.

AtariWriter is not the best word processor on the market; there are probably one or two which might suit your needs even better. But it is the most *popular* word processor, and it contains most of the features commonly found on a word processor. For these reasons, it is an excellent example for our purposes. By examin-

ing AtariWriter's various commands and the means by which these commands are implemented, you will get a good idea of how to work with any word processor.

Becoming familiar with the AtariWriter will allow you to create, edit, and print word-processing files (term papers, letters, or even an article you want to write for a magazine). A *file* is simply any document you create.

Word processing is an excellent application for the Atari 600XL or 800XL computer, but there are some terms you need to know before you can fully understand the powers of the AtariWriter. Here are some of the most important terms:

**CENTERING:** putting a line of text in the middle of the page. This is handy for titles, which are usually centered.

**CHAINING:** used when you have a document that is so long that you have to split it into two files on your diskette. The computer will print the first file, load in the second, and then print the second.

**CREATE FILE:** allows you to start a new document. Before you begin creating a document, the Atari will make sure there is no other document in memory which you might accidentally erase. If there is, the Atari will make sure you want to create a new document and erase the old one.

**DELETE FILE:** removes a file from your diskette. Be careful when using this command because you don't want to erase a file which you might need later.

**EDIT FILE:** allows you to revise a document. This includes deleting, inserting, moving, and generally altering the contents of a file.

**FILE:** a document on a diskette or cassette.

**FONTS:** print *styles*; that is, different types of printing (pica, elite, or italic print, for example).

**FORMATTING:** allocating space for the future insertion of information. If you are going to be storing files on a new disk-

ette, you must format it so that it will be ready to accept files from AtariWriter.

**HEADERS AND FOOTERS:** text at the top (*header*) or bottom (*footer*) of each page printed. This is handy for printing titles on each page.

**JUSTIFICATION:** setting of type so as to fill a full line. When a printout is *justified*, the left and right margins are aligned vertically.

**PRINT PREVIEW:** lets you see how your document will look before it is actually printed. When you see your "printed" document on the video screen, you can make any needed adjustments in the document before you print it.

**PROGRAMMING CAPABILITY:** a feature that allows you to create files in programming language, such as BASIC or assembly language, for later use. [After you create a file and save it on a disk or tape, you can load it into the computer with BASIC language or using the Macro Assembler that's in memory. The advantage to this is that editing a file is especially easy with AtariWriter.]

**SEARCH AND REPLACE:** used to find a string of characters, such as "look for this", "he went home", or any string of up to 25 characters and replace them with something else.

**SUBSCRIPTS AND SUPERSCRIPTS:** used in scientific papers. A *subscript* is like the $_2$ in $H_2O$; a *superscript* is like the $^2$ in $E = MC^2$.

**TABS:** the series of arrows which appear every five spaces on the screen. Use the TAB key when you are starting a new paragraph or when you need to move the cursor to the right to print something like the date or your name and address. Pressing the TAB key advances the cursor to the next column.

This completes the terms you need to know before you use a word processor. If you have an AtariWriter cartridge now, you can begin using it.

## The Menu

After you have inserted the Atari 810 Master diskette into your disk drive and have put the AtariWriter cartridge into the slot of your 600XL or 800XL, turn the computer on and look at the MENU. The options included on this MENU are:

Create File
Delete File
Edit File
Format Disk
Index of Disk Files
Load File
Print File
Save File

Pressing the first letter of any of these options will make the computer access the routine for performing that option. In the next section, we'll see what features are available with each of these options (except D, F, or I).

## Writing and Editing

### Create File

If you press C to "Create File," the computer will clear the screen and display several things. At the top of the screen will be a *print formatting block*, an area that shows the current *parameters* (conditions) of the word processor. It will list such things as margins, page spacing, and other variables which you can change to alter your final printout. Also on the screen is the cursor, which will show you where you are on the screen. When you begin typing, the copy will appear on the video display wherever the cursor is located. The message window, near the bottom of the screen, displays questions, prompts, and other information. The tab arrows show where the tabs are located, while the *screen line* and *character space* show the vertical and horizontal locations of your cursor.

To type documents into the computer, simply use the keyboard. If you want to skip to the next line, press the RETURN key. You do *not* need to press RETURN when you come to the

end of a line. The computer will automatically go to the next line for you. For paragraph indentation, press the CONTROL and P keys at the same time. The following shows how you might type a note to a friend:

April 9, 1985 (RETURN)
(RETURN)
(RETURN)
Dear Yen-Chi,(RETURN)
(CONTROL)(P)I was accepted into the University today! Isn't that great! Please come over and show me your acceptance also, and we can celebrate.(CONTROL)(P)
By the way, the dean said we could go to Palo Alto right now and thank him personally. See you soon.(RETURN)
(RETURN)
(TAB) (TAB) (TAB) (TAB) Tim Knight(RETURN)

As you can see, AtariWriter is easy to use. The *real* beauty of a word processor, however—its ability to edit documents so that they will be perfect before you print them—is not yet apparent.

First, you must know how to move your cursor around the screen to get to the places you would like to edit. Here are the cursor commands:

| | |
|---|---|
| CONTROL ↑ means: | move up |
| CONTROL ↓ means: | move down |
| CONTROL ← means: | move left |
| CONTROL → means: | move right |
| SELECT T means: | move to top of file |
| OPTION ↑ means: | move up one screen |
| OPTION ↓ means: | move down one screen |
| CONTROL A means: | move to beginning of line |
| CONTROL Z means: | move to end of line |
| TAB means: | move to next tab stop |

There are also some special options while you are creating a file. For example, you can use the key with the Atari logo to "toggle" between "underline" and "no-underline" modes. For example, if you press the Atari key, everything you type will be underlined until you press the Atari logo again. If you want to change nonunderlined to underlined text or underlined to nonunderlined text, however, press the CONTROL and U keys together as many times as you need to in order to move your cursor across the letters or words you want reversed.

There are some other handy features available with the Create File option in addition to those already mentioned. Pressing SHIFT and CAPS together will make the computer go to an all-uppercase mode, while pressing CAPS alone will switch the computer back to a lowercase mode. If you want to reverse your uppercase to lowercase or lowercase to uppercase (just as you did with the underlining), you can use the CONTROL and CAPS keys as many times as you need to, keeping your cursor at the location you want changed.

To get out of the Create File option (or any option, for that matter), press the ESCape key.

Before pressing the ESC key, however, edit your document. You can move your cursor around the screen and insert any new text you want. If you would like to *delete* (remove) or insert any new letters, words, sentences, or even paragraphs, however, use the following commands:

| | |
|---|---|
| DELETE BACK SPACE means: | delete character left of cursor (seems to "eat" text) |
| CONTROL, DELETE BACK SPACE means: | delete character above cursor |
| SHIFT, DELETE BACK SPACE means: | delete everything to the end of the line the cursor is on |
| SELECT, DELETE BACK SPACE means: | delete everything to the end of the document |
| START, INSERT means: | replace text last deleted ("undo") |

### Print File

Printing a file is as simple as creating one. Once you are back to the AtariWriter MENU, press P to "Print a file;" the computer will ask you what printer you are using, if the whole document should be printed, and how many copies you would like. After you have answered these questions, the computer will begin printing your letter, stopping only after it has finished or when you press the BREAK key. Incidentally, if you do not want to print the whole document, you can tell AtariWriter what pages you would like printed.

A few other commands are available with AtariWriter which may be helpful to you when you are writing. OPTION F will display how much memory is left in the Atari 600XL or 800XL, which is a good indicator of how much more you can type into the computer. OPTION P will give you a print preview, displaying your document as it will appear on paper. You can use OPTION ← and OPTION → while in print-preview mode to move left and right while reading your document because the entire "page" cannot be printed on the screen at one time.

### Save File

Saving and loading text files is also very simple. To save a file, press the S option and type "C:filename" if you have a cassette and "D:filename" if you have a disk drive. "Filename" should be the name you want to call your document, such as "Sample ltr" or "Essay." Remember to use only letters and digits in your file name because something like "TIM$!!#" will not be accepted by AtariWriter as a file name.

### Load File

In order to load a file back into memory, merely press the L key while in menu mode, answer the question "Erase file in memory?" with a positive Y answer, then type "C:filename" for cassette and "D:filename" for a disk system.

This completes the basic commands of AtariWriter. Now we'll explore some of the more complex features of the system.

## Advanced Features

### Delete File

To delete a large block of text, position the cursor under the first character in the block that you want to delete and press the CONTROL and X keys. Then move to the last character in the block that you want to delete and press (CONTROL)(X) again. Finally, press OPTION, DELETE BACKSPACE (press all three keys at the same time). To make sure you are not deleting something you want to retain, AtariWriter will ask, "Block delete—Are you sure, Y/N?". If you want to delete the entire block of text, answer with an affirmative Y.

### Edit File

You can also *move* large blocks of text with AtariWriter. To move a block of text, type (CONTROL)(X) at the beginning and end of the block you would like to move. Then press OPTION M where you want that block of text to be moved. AtariWriter will delete the block at its old location and insert it at the location you specify.

### Index File

If you are looking for something in your document so that you can replace it or delete it, use the Search and Replace command. First, press SELECT S, then enter the string you are looking for. Once the computer has found the string, it will ask REPLACE STRING, Y/N? If you respond with a Y, you can type in anything you want, and the computer will replace one string of characters with another. AtariWriter will then ask, "REPLACE GLOBAL, Y/N?" This means, "Do you want to replace the old string with the new string everywhere the old string appears in the document?" This would be useful if, for example, you used the name "John" throughout your text when you wanted to use the name "Jack." If you answer Y to this question, you will not have to search for every occurrence of the name "John." On the

other hand, you might not want to replace the original string globally, so you would respond with an N answer to the "Replace global" question. At this point, the computer would ask, "Continue searching, Y/N?" A Y response would result in the computer searching for the next occurrence of the old string, while an N response would stop AtariWriter from searching. Remember to press RETURN after every command you enter.

The print formatting block, found at the top of the screen, can be quickly accessed by pressing SELECT T. Once you are there, you can move to the different parts of the line, delete the numbers which set up the format for your printout, and replace them with new numbers. This allows you to change the format of your printed document.

The variables used to format the printed page are listed below, each with a number indicating the *default* (current) value of the variable.

**B (12):** indicates the bottom margin. The number (12) means that the bottom margin is 12 half-lines, or about an inch.

**D (4):** indicates the spacing between paragraphs.

**G (1):** indicates the print style. Print style (1) is 10 *CPI* (characters per inch). Other print styles include condensed print (2) and proportional spacing (3).

**I (5):** indicates paragraph indentation. Used when the (CONTROL)(P) command is issued. The number (5) means that each paragraph is indented five spaces.

**J (0):** indicates whether the right margin is justified. The number (0) means that it is not; the number (1) would mean that it is.

**L (10):** indicates the left margin. The number (10) means the document will start 10 spaces from the left edge of the page.

**R (70):** indicates the right margin. The number (70) means the type will extend 70 spaces from the left edge of the page, resulting in a 60-character-per-line printout if the left margin shown above is used.

**S (2):** indicates line spacing. The number (2) means the line is set to two half-lines, or single-spaced. The number (4) would mean that the printout would be double-spaced.

**T (12):** indicates the top margin. The number (12) means 12 half-lines, or 1 inch.

**Y (132):** indicates the total length of the page in half-lines. The number (132) means the page is 11 inches long. Change this only if you change the paper on which your document is being printed.

You can also change print types within the document itself by inserting special formatting commands within your text. These commands will appear in inverse video, and they will not be printed on your document. However, they will show up on the video screen so you will know where your print formatting is being altered.

To specify these variations from the overall formatting values, enter these commands, followed by the appropriate new number for the command:

| | |
|---|---|
| (CONTROL)(B) means: | bottom margin |
| (CONTROL)(M) means: | second column left margin (used for double-column printing) |
| (CONTROL)(J) means: | justified right margin |
| (CONTROL)(L) means: | left margin |
| (CONTROL)(S) means: | line spacing |
| (CONTROL)(I) means: | paragraph indentation |
| (CONTROL)(D) means: | paragraph spacing |
| (CONTROL)(G) means: | print style (font) |
| (CONTROL)(R) means: | right margin |
| (CONTROL)(T) means: | top margin |

You can also elongate print if you have a printer that allows you to print twice-normal width text. To print something in large type, press SELECT E at the beginning of the string of characters you wish to be in elongated mode, and press SELECT E again at the end of the string. One other special command similar to this can be used in AtariWriter in the same manner: entering CONTROL (C) at the beginning of a line will center that line so long as you press RETURN at the end of that line.

If you are typing a document which needs a title or a page number at the beginning or the end of each page, you will have to know how to create a header or a footer.

To create a header, press H followed by the text of the header ("Report on My New Computer" for example), and then press RETURN. Your header will be printed at the top of each page. To create a footer, follow the instructions above, but use (CONTROL)(F) instead of (CONTROL)(H).

You can center a header with (CONTROL)(C) or print it on the right side of the page by pressing (CONTROL)(C) twice. Finally, to print page numbers, insert "@" within the header or footer at the point where you want the page number to appear. For instance, if you want the footer "My Book Report, Page 1" to appear centered at the bottom of every sheet (with different page numbers, naturally), you would press (CONTROL)(C)(F), let go and type My Book Report, Page @(RETURN).

If you have a disk drive, you can use the D and I options for "Delete File" and "Index of Disk Files."

To find out what files are on your disk, use the I option. After printing the files on the screen, the computer will ask "**PRINT** DIRECTORY, Y/N?" A Y answer will give you a printed copy of the directory; an N answer will cause the computer to return to the MENU.

Deleting a file from the diskette is accomplished by pressing D and typing "D:filename". AtariWriter will make sure you want to delete the file by asking "ARE YOU SURE?" to which you would respond with a Y. The computer would then erase the specified file from the diskette. Finally, use the F OPTION to format only new disks which you will be using for storing files.

Another feature of the AtariWriter, mentioned in the glossary of terms at the beginning of this Chapter, is "chaining." This is done by pressing (CONTROL)(V) at the end of the first file, typing in "D:filename" (the file name being the name of the second file you want chained to the first), and then printing the file as you normally would.

Not all of the features of AtariWriter have been covered in this chapter because some of them are used only for the most sophisticated applications of a word processor. Whatever word processing program you use, you will find that its commands are very similar to those of the AtariWriter.

## Chapter Seven

# The Money Machine

You have probably heard a great many stories about individuals getting rich from their involvement in the computer industry. Most common are the stories about teenagers making hundreds of thousands of dollars by writing game programs.

The most spectacular success stories are not as common as they used to be, but people are still making money from computer software. There are still many opportunities for a person to create a piece of software, market it, and see a substantial profit from it. Bill Budge, the creator of Raster Blaster and Pinball Construction Set made over half a million dollars from his work before he was 28 years old; an 18-year-old friend of mine, Tom McWilliams, earns a near six-figure income from his games; and I have done well as both an author of computer books and a creator of software. You, too, can make money from your Atari 600XL or 800XL. In this chapter, we will explore how you can turn your Atari into a "money machine" as well as ways to manage (or spend) the money you earn.

### Creating Software

Creating software for the 600XL or 800XL (or any computer, for that matter) can be an enjoyable, profitable activity. If your

program is well-written and well-marketed, you can make $30,000 to $100,000 in royalties over a fairly short period of time. The most popular programs, naturally, are games; but you might want to make a business program or a program for computer communications.

The money in software comes in the form of royalties. To illustrate this, assume that you create a game for the Atari 600XL or 800XL computer system and send it to a software publisher for consideration. About a month later, you find that your program has been accepted and will be marketed nationwide within a month. The program will retail for $20 a copy. Your contract states that you will receive 25 percent of the *net price* (wholesale price), which is usually half the retail price; or $10. Therefore, you will receive 25 percent of $10, or $2.50, for each program sold. Assume that your program sells fairly well—10,000 copies. That amounts to $25,000. If the program took you only a couple of months to write, that's an excellent profit!

This scenario has occurred many times, but of course you can't be positive that your program will sell well. I therefore have some advice on how you can better your chances of producing a successful program. First, take full advantage of the power of your computer. The Atari has a lot of graphics and sound potential, so you'll want to exploit these to the fullest. But the trade-off for its great graphics and sound is that it does not have too much memory or "business computer power," so you may want to concentrate on entertainment programs at first.

Second, make your program as original as possible. Originality is essential to the acceptance of a program because no publisher wants to market two similar programs. Also, go further than just making your program original—make it useful or entertaining to the greatest number of people.

Finally, keep the user in mind; in other words, make the program *user-friendly*, or as simple to use as you can. Also, make sure it is so well documented and planned that unexpected errors won't occur. Remember that most computer owners are inexperienced, so the simpler your program is to use, the more popular it will be among computer software buyers.

You may not think of yourself as the software programming "type," but that may be due to the stereotype of the program designer as a "nerd." Actually, most successful software designers are not "nerds" at all, but simply imaginative, intelligent, and knowledgeable people. I find that most people who write programs for computers need these five important qualities:

**Patience:** because writing and selling a program takes time. The slow process of building a program from scratch requires patience. In addition, it takes about six weeks for a program to be evaluated by a software publisher and another month or so before the first royalty check arrives. In fact, it took a year for one of my game programs to be marketed by a software publisher, so patience was essential.

**Logical minds:** because a person who works with a machine must think logically at least some of the time. Therefore, a logical mind is important to the successful production of a software package.

**Imagination:** in order to come up with the idea for a program as well as create the graphics, design the sound, and develop the program.

**Knowledge:** because a program designer must know his computer, the people for whom he is writing the program, and what the computer hardware and software markets are like at any one time.

**Business know-how:** because a person who creates programs must know about contracts and royalties and must be aggressive enough to drive a good bargain. I know many programmers who have all the qualities needed to create a program but cannot manage to sell one. They may be shy or simply indifferent to the idea of making money.

If you think you have the qualities of a programmer and would like to create and market a software package, there are certain steps you must follow in the process of going from an idea for a program to the receipt of a royalty check. The procedure I have outlined is not rigid, but it should help you in some ways.

The first step, naturally, is to create the program. You need a good idea which you can develop into a working product. After you know all the details of your program, begin writing it on a regular schedule so you can finish it by a certain date. It's important to stick to your schedule, and it's also important that you try to remove any *bugs* (problems) in the program while you are writing it so you won't have to search for the bugs after you've completed the entire program.

Once you have finished the program, send it to a reputable software publisher along with a descriptive letter. Make sure you have a completely debugged program with thorough documentation so that it won't be sent back to you for further development. Plan to wait four to six weeks for a "yes," "no," or "work on it some more" answer. For your reference, the following software publishers might be interested in programs you develop for the Atari 600XL or 800XL:

*Adventure International* (Box 3435, Longwood, FL 32750)

*Atari, Inc.* (1265 Borregas Ave., Sunnyvale, CA 94086)

*Broderbund Software, Inc.* (1938 Fourth St., San Rafael, CA 94901)

*The Software Guild* (23214 Clawiter Rd., Hayward, CA 94545)

*Sirius Software* (10364 Rockingham Dr., Sacramento, CA 95827)

There are many other publishers, of course. To find their names and addresses, just check the advertisements in any computer magazine that supports the Atari line of home computers.

If your program is turned down, you can try sending it to another publisher and waiting some more. If it is accepted, however, you will be offered a contract specifying the royalties you will receive for each program sold, what your rights are to the program, and what the publisher's rights are regarding your work. After you sign the contract, you probably won't have to do anything else except wait for the royalty checks to arrive. Your

part—the creation of the program—is over; and now the software publisher will go to work selling your program, hopefully making both you and the company money.

## Reviewing Programs

If you are interested in software but don't want to write your own programs right now, you might try becoming a software reviewer—a "critic" of what is currently being published. Writing reviews has some excellent advantages. First, you learn a great deal about the software market: what makes good and bad software, how programs are being packaged, and who is making the best programs. Second, you can make some excellent contacts in the software industry. Last, reviewing can generate a small income ($1000 a year or so), useful for the purchase of things like diskettes, printers, or other "computer tools." On top of this, the programs you review are usually yours to keep.

The first step in reviewing a program is acquiring it. You can do this by either buying it, asking a publisher to send it to you for review, or asking a magazine if they have any programs for you to review. If you buy your own or ask a publisher for a specific program, make sure it is a relatively new one. Magazines are not interested in reviews of programs which have been on the market for six months or more.

Next, you must write the review. To do this, you must be able to identify both the good and the bad points of a program; make suggestions as to how the program could be improved; compare the program to other, similar programs; and, finally, tell the reader of your review whether to buy the program and for what purpose. In your review, comment on the product's features, anything that is new or exciting about it, and how well the documentation is written. Conclude your review by summarizing your thoughts in such a way that the reader will be left with a good or bad feeling about the program, whichever it deserves.

If a magazine assigns you to do a review, you simply send the review to the magazine, wait for it to be published, and then wait

for your check. However, if you write an *unsolicited review* (a review not assigned to you), you may have to send it to several magazines before it is accepted. In such a case, you must know which magazines are likely to buy it. The following is a list of magazines that support the Atari:

*Infoworld* (530 Lytton Ave., Palo Alto, CA 94301) pays about $75 on publication. Takes solicited reviews only.

*Personal Computing* (50 Essex St., Rochelle Park, NJ 07662) pays about $50 on publication.

*Computer Shopper* (407 S. Washington Ave., Box F, Titusville, FL 32780) pays about $50 on publication.

*Softside Magazine* (6 South St., Milford, NH 03055) pays about $75 on publication.

*Antic* (600 18th St., San Francisco, CA 94107) pays about $50 on publication.

*Creative Computing* (39 E. Hanover Ave., Morris Plains, NJ 07950) pays about $75 on acceptance.

It is a good idea to write each of these magazines for an author's guideline before you start your review. These guidelines tell you how the particular magazine likes reviews submitted, how much it pays, and what special considerations should go into your writing and submission to them.

You aren't going to become rich writing reviews, but the advantages mentioned earlier make your efforts worthwhile. Try to become "regular" with at least one magazine. Once a magazine has printed a few of your reviews, it will start sending programs to you for review automatically.

## Technical Writing

After you have written reviews for a while and gained experience and made some contacts in the publishing world, you will proba-

bly want to expand into another area of work: technical writing. Writing technical articles requires more work than does writing reviews, and the manuscripts are longer. Still, the extra money makes the effort worthwhile. Also, you will receive more recognition from writing articles ("tutorials," programming articles, and news stories, for example) than you will from writing reviews.

Some of the techniques used to write articles are the same as those used to write reviews. You must come up with an idea for the article, write it, and then send it to one of the magazines listed. Again, it is a good idea to have some kind of an author's guideline before you start writing. But you will usually be paid twice as much for an article as you would for a review; you can make up to $500 for a six-page article in a major computer magazine.

The first category of article, the "tutorial," is an article designed to educate the reader about some technical aspect of the computer. For example, you could write an article explaining how to use the graphics of the Atari 600XL or 800XL. You could begin the article by introducing the concept of graphics and then detail the places in the computer's memory used when forming graphics. You could describe the different shapes and colors that can be created with the 600XL or 800XL and discuss some of the special powers the 600XL or 800XL has with respect to graphics. Finally, you could conclude the article by suggesting ways in which the computer's graphics can be used. As you can see, a tutorial is simply a guided tour through some technique. Tutorials should be easy to read, understandable to new computer owners, and complete enough to be useful.

Programming articles are similar to tutorials in that they help explain something to the reader, but they are different in that they deal entirely with a program that the reader can actually type into the computer and use. For example, if you made a game for the 600XL or 800XL in BASIC language, you might write an article about how you came up with the idea for the game, how you created it, what programming techniques you used to finish

it, and how the reader can play the game.. The programming article is meant to give the reader a usable piece of software while it teaches him or her something about the computer itself. These articles are not difficult to write so long as you are able to create an interesting, useful program and explain it thoroughly.

The third category of article, the news story, could be one of several things. You might write a story on some new "peripheral" that has been introduced for the Atari line of computers—the Topo robot, for example. Or you could do a story on a celebrity in the world of computers, such as the owner of Adventure International, Scott Adams, or the young, successful programmer Bill Budge. Or you might consider writing a news story on trends in the computer hardware or software business. You might even try writing an article on your own experiences with the Atari 600XL or 800XL home computer. It should be kept in mind, however, that some computer magazines accept news stories only from their own staffs.

If you decide to write an article, it is a good idea to write a query letter before you put your time into the article itself. A *query letter* is basically a note to the editor of a magazine to see if he or she would be interested in your article. If the answer is yes, your chances for acceptance are better than if you simply send the article. Always keep articles in clear, simple language; and keep beginning and intermediate computerists in mind when you write because they constitute the largest segment of computer magazine readers.

## Computer Software and Supplies

If you're not going to save all the money you make from your involvement in the computer industry, what better way to spend it than on more computer hardware and software? Building your computer system will increase your opportunities both for making money with your computer and for enjoying it.

Hardware is usually more expensive than software because it

consists of the electronic components of a machine. Items such as printers, disk drives, and modems are defined as *hardware*, and they can range in price from about $150 to $1500. When buying hardware, keep these three rules in mind:

1. Try to get the lowest possible price without sacrificing service. If a mail-order company and a store offer the same printer under the same conditions of warranty and service, but the mail-order company offers it for $100 less, you should probably buy it from the mail-order company. Remember, however, that you will have to wait a few days for the printer to arrive; and if something goes wrong with it, you will have to send it to a repair center and wait for it to be repaired and sent back to your home. Computer equipment doesn't break down very often, believe it or not, and so the $100 you save could make it worthwhile to buy the printer from the mail-order company.
2. Make sure that the hardware you are buying will benefit you in some way. Don't pay $200 for a modem unless you are going to be using computer communications. In other words, don't buy anything you will use for a few weeks and then disregard. I have bought many things, even computers, which I didn't use later, and I regretted wasting my money on them.
3. Check to make sure that the hardware you are buying is compatible with your Atari 600XL or 800XL. A lot of equipment advertised as being made for "Atari computers" will not work on the 600XL or 800XL.

Buying software can be even more difficult than buying hardware. There are thousands of programs on the market, and it is sometimes difficult to know which of them will best suit your needs. The rules I've outlined for buying hardware should also be applied to buying software, and you should also test software before you buy it. Most software stores will let you sample packages before purchase if you request it. This will give you a chance to see what the software can do, how easy it is to use, and if you can put it to good use.

Making money with computers is admittedly more difficult than spending the money you make, but making the money can also be enjoyable and educational. If you find something that you feel can fill a need in the computer or upcoming personal-robotics industry, give it a try. Or try your hand at writing software or articles for others like yourself. Many people want to find out more about their computers so that they can use them to their full potential.

Chapter Eight

# A Sampling of Programs

You have already seen a number of sample programs in this book which you might have already typed into your computer, but these programs are neither long nor particularly useful. This chapter contains three programs that are longer than the ones earlier in this book and might prove useful to you in your programming efforts.

In addition, I will give some more basic information on the Atari 600XL or 800XL, information on how to use some of the special function keys on the computer and how to produce graphics and international characters with your keyboard.

Finally, I will give information on the hardware products Atari offers for its machine so that you can decide how to expand your computer system in the future (see Fig. 8-1).

## Three Programs

### Graphics Program

The first program is involved with graphics, and it will make it easy for you to draw pictures on your computer screen. Basically, you use the keyboard to move up, down, left, right, or diagonally. In order to detect which keys are being pressed, the

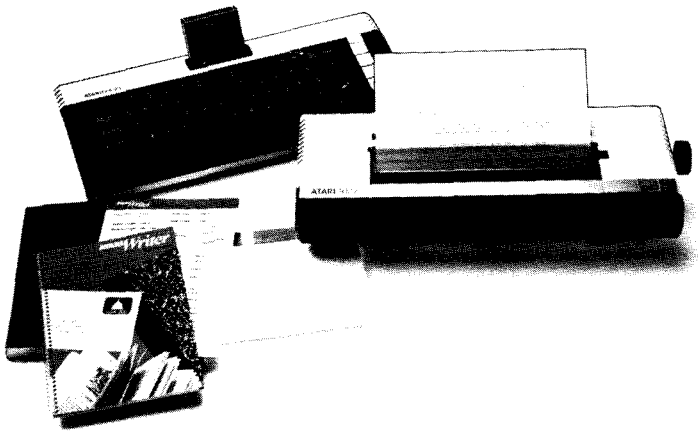

**Fig. 8-1 The Atari 600XL working in conjunction with the 1027 Printer** ***(Courtesy of Atari, Inc.)***

computer stores the value of the key being pressed in memory location 764. You can find out what is in byte (memory location) 764 with the command **PEEK**. Therefore, **LET** X = **PEEK**(764) would assign the value in byte 764 to the variable X. The computer is simply taking a look (*peeking*) into byte 764, and whatever value it finds there is assigned to the variable X. Each key on the keyboard will result in a distinct number being put into byte 764. The numbers resulting from those directional keys, along with the control keys and their corresponding numbers, are shown in Table 8-1.

The program is basically made to detect the values in location 764 and to act on those values. In addition, the program makes it possible to increase the *step*, or rate, at which you draw. Instead of drawing a line across the screen at one point for each key pressed, you can increase the step to 10 points or 15 points or however many you want, thus speeding up the process of drawing while, at the same time, reducing the fineness of your picture

if it involves shapes other than boxes. The variable which controls the step is S, and the keys to use to bring S up and down are 1 (up), and 2 (down). The values at location 764 for these are 31 and 30, respectively.

The first few lines of the program put the Atari in high-resolution graphics mode, make sure the variables are set to their proper values, and assign a color to the screen.

```
 1 REM DRAWING PROGRAM
 2 GRAPHICS 0
 3 PRINT"Drawing Program"
 4 PRINT:PRINT"Use the Y, B, G, and H keys to
   move":PRINT
 5 PRINT"Up, down, left, and right. Use the ":PRINT
 6 PRINT"T, V,Y, AND N keys to move diagonal.":PRINT
 7 PRINT"Lastly, use the 1 and 2 keys to make":PRINT
 8 PRINT"the rate go up or down, C to clear the":PRINT
   "screen, and E to toggle."
 9 IF PEEK(764)<>33 THEN 9
10 GRAPHICS 30
20 COLOR 1
30 X=0:Y=0:S=1
```

Table 8-1. **KEYBOARD COMMANDS**

| Direction/Command | Control Key | Number |
|---|---|---|
| Up | Y | 43 |
| Down | B | 21 |
| Left | G | 61 |
| Right | H | 57 |
| Up/right | U | 11 |
| Down/right | N | 35 |
| Up/left | T | 45 |
| Down/left | V | 16 |
| Increase value | 1 | 31 |
| Decrease value | 2 | 30 |
| Clear screen | C | 18 |
| Toggle | E | 42 |

I then position the drawing cursor at its initial location (0,0) and make sure that location 764 has the value 255 in it. The reason for putting 255 in location 764 is that 255 will be the number for a "No key pressed" signal. If a key is pressed, location 764 will hold the value of that key until another key is pressed. Therefore, a **POKE** (insertion of data into a location) of 255 must be continually made to 764 to assure that the computer does not mistake a value in 764 for a key to be held down constantly.

```
40 PLOT 0,0
45 POKE 764,255
```

The next line is the most important one in the program because it checks location 764 for a value. If the value is still 255, it goes back to line 50.

```
47 REM CHECK KEYBOARD
50 I=PEEK(764):IF I=255 THEN GOTO 50
```

The next lines check for keys being pressed to indicate a direction. Corresponding changes in the variables X and Y will be made if a directional key is pressed, and the computer will go to the routine at line 100.

```
55 IF I=35 THEN Y=Y+S:X=X+S:GOTO 100
60 IF I=43 THEN Y=Y-S:GOTO 100
65 IF I=45 THEN Y=Y-S:X=X-S:GOTO 100
70 IF I=61 THEN X=X-S:GOTO 100
75 IF I=11 THEN Y=Y-S:X=X+S:GOTO 100
80 IF I=57 THEN X=X+S:GOTO 100
85 IF I=16 THEN Y=Y+S:X=X+S:GOTO 100
90 IF I=21 THEN Y=Y+S:GOTO 100
```

The next two lines, also for key detection, change the drawing step value.

```
91 IF I=31 THEN S=S+1:GOTO 100
92 IF I=30 THEN S=S-1:GOTO 100
```

These next two lines check to see if the C or E key has been pressed. If C is pressed, the computer will clear the screen. If E is pressed, the computer goes to a subroutine at line 1000 (shown below) which makes the computer go into a nondrawing mode if it is in draw mode or a draw mode if it is in a nondrawing mode. This toggling is accomplished by checking the value of the variable **FLAG** and changing the color appropriately so that what you draw will either appear or not appear (whichever is chosen) on the screen, though your drawing cursor will still move.

```
 93 IF I=18 THEN GRAPHICS 0:GRAPHICS 30
 94 IF I=42 THEN GOSUB 1000
999 REM ROUTINE FOR DRAW/NON-DRAW TOGGLE
1000 IF FLAG=0 THEN FLAG=1:COLOR 0:RETURN
1010 FLAG=0:COLOR 1:RETURN
```

If no keys have been pressed, the computer will wind up at line 95, which repeats the whole process by going back to line 45.

```
95 GOTO 45
```

If a key has been pressed, the computer will go to line 100, which checks to see if the values of X and Y are *legal* (within the boundaries of the screen). If they are not, then the computer will make the correction(s) to the illegal variable(s).

```
 98 REM CHECK FOR ERRORS
100 IF Y<0 THEN Y=0
110 IF Y>191 THEN Y=191
120 IF X<0 THEN X=0
130 IF X>159 THEN X=159
```

Also, to make sure that the step was neither too low nor too high, the computer will check the value of S:

```
135 IF S<1 THEN S=1
137 IF S>30 THEN S=30
```

Finally, the computer will draw the graphics to the current location and return to line 45.

```
140 DRAWTO X,Y
150 GOTO 45
```

The program is now ready to run with the **RUN** statement, and you can use it to draw whatever you want. You might want to save this and the other programs in this book for later use so that you won't have to type them into the computer again.

### Extension of a Graphing Program

The next program is an extension of the graphing programs I made earlier because it plots data which was input from the terminal by a person. This means that you can INPUT data about something (such as sales data, grades, or any other information which can be represented by a bar graph) and the computer will plot it. There are some limitations on this simple program, though. First of all, you can INPUT a maximum of 20 pieces of data. Also, the data cannot be less than 0 or greater than 191.

The first few lines for this program describe its purpose: **DI**-Mension the D(x) array which will hold the data input from the keyboard and the ANSWER$(x) array for use later in the program, and clear the screen with the **GRAPHICS** 0 statement. Also, the statement at line 5 tells the Atari that if an error occurs in the program, it should go to line 300, where the program will be stopped.

```
  5 TRAP 300
 10 REM GRAPHING PROGRAM
 20 DIM D(20),ANSWER$(1)
 25 GRAPHICS 0
```

The computer then asks how many pieces of data are going to be entered. The computer receives the number from the keyboard and then checks to see if the number is legal or not. If it is, the computer clears the screen once again and proceeds to the rest of the program. If it is not, the computer returns to the question, asking how many pieces of data should be INPUT. Also in this group of lines (45 to 48) is an **INPUT** routine that lets you tell the computer if you want the graph to be drawn with solid block bars or simple line bars.

```
30 PRINT"How many pieces of data";
40 INPUT N
45 PRINT"Would you like these in block Graphics";
46 INPUT ANSWER$
47 IF ANSWER$ = "Y" THEN B = 1:GOTO 50
48 B = 0
50 IF N<1 OR N>20 THEN GOTO 30
55 GRAPHICS 0
```

The program then allows a person to INPUT all the data, with each number being assigned to the D(x) array.

```
60 FOR X = 1 TO N
70 INPUT TM
80 D(X) = TM
85 IF D(X)<1 THEN D(X) = 1
86 IF D(X)>191 THEN D(X) = 191
90 NEXT X
```

Once the data has been INPUT into the computer, the computer goes into high-resolution graphics mode, sets the color of the screen, and draws the two lines which make the grid for the graph.

```
100 GRAPHICS 30
150 COLOR 1
200 PLOT 1,10:DRAW 1,191
```

```
207 DRAWTO 159,191
208 FOR XX=10 TO 190 STEP 10:PLOT 0,XX:DRAWTO
    159,XX:NEXT XX
```

Drawing bars for this graph is not hard for the Atari 600XL or 800XL because it only has to draw a line up to each of the points determined by the D(x) array, go to the right a few points, and then draw down to the bottom of the graph again. This forms each of the bars in the D(x) array until the **FOR/NEXT** loop is finished.

```
210 FOR X=1 TO N
220 PLOT 3+X*7,191
230 DRAWTO 3+X*7,D(X)
240 DRAWTO (X+1)*7,D(X)
250 DRAWTO (X+1)*7,191
255 IF B=1 THEN FOR TT=3+X*7 TO (X+1)*7:PLOT
    TT,191:DRAWTO TT,191-D(X):NEXT TT
260 NEXT X
```

The last part of this program checks to see if the ESCAPE key has been pressed. If it is pressed, the computer runs the program again. If not, the computer continues to wait. Finally, line 300 stops the program if the computer encounters an error.

```
270 IF PEEK(764)<>28 THEN 270
280 RUN
300 GRAPHICS 0:END
```

### Sound Program

The last program of the three I created for this chapter involves sound—that is, the four voices of the Atari 600XL or 800XL. This program allows you to change the pitch of any of the four voices and to observe the pitch of each voice as you make the change. All four voices start at pitch 100, but you can change the pitch (from 0 to 255) by pressing U (for up) or D (for down) followed by the number of the voice you want to change. For example, if you want to increase the pitch of voice 3, you press U and then 3. Any number you press after 3 will result in that voice being increased in pitch. Also, if you press 1 several times, voice 1 will increase in pitch substantially. If you want to switch from

INCREASE FREQUENCY to DECREASE FREQUENCY, you merely press D (for down). This may seem contradictory to you, since the **SOUND** command usually creates lower pitches for higher numbers and higher pitches for lower numbers. But in this program, I "reversed" the addition and subtraction so that it would seem easier and more logical to the end user: "UP" to increase the number (and hence the frequency) and "DOWN" to decrease the number (or the frequency). Each time you change the pitch of a voice, you will see the change registered on the video screen. By altering the tone of each voice, you can create some very interesting tone effects for music or sounds in your programs.

I use the "keyboard detect" sequence, involving byte 764, again in this program, and I use the keys and numbers shown in Table 8-2 for the commands.

I will start the program by identifying it with a **REM**ark statement, clearing the screen, and setting the pitch and noise/tone variable values for each voice (at lines 5 and 15):

```
 5 T0=10:T1=10:T2=10:T3=10
10 REM Sound program for four voices
11 GRAPHICS 0
12 COLOR 2
15 F0=100:F1=100:F2=100:F3=100
```

Table 8-2. **SOUND COMMANDS**

| Command | Key | Value of Byte 764 |
|---|---|---|
| Frequency up | U | 11 |
| Frequency down | D | 58 |
| Frequency up fast | Shift U | 75 |
| Frequency down fast | Shift D | 122 |
| Voice 0 | 1 | 31 |
| Voice 1 | 2 | 30 |
| Voice 2 | 3 | 26 |
| Voice 3 | 4 | 24 |
| Change to noise | N | 35 |
| Change to tone | T | 45 |
| End program | E | 42 |

I then **POKE** location 764 with 255 and set up the statement which monitors that location in the computer's memory.

```
20 POKE 764,255
30 I = PEEK(764)
```

At this point, I program the computer to position the cursor at location 0,22 so that it will print the pitch values at the top of the screen. I also program the **GRAPHICS** statements so that while I am altering the pitches of the voices I can see, graphically, how high or low each of the voices is at any given time.

```
31 REM DRAW GRAPHICS
32 GRAPHICS 3
33 PLOT 3,19:DRAWTO 3,F0/10:PLOT 13,19:DRAWTO
   13,F1/10
34 PLOT 23,19:DRAWTO 23,F2/10:PLOT 33,19:DRAWTO
   33,F3/10
35 POSITION 0,22:PRINT F0,F1,F2,F3
```

I then program the Atari to recognize the keyboard input and to take appropriate action on it. If the value of the input is 255, no key is pressed, so the computer returns to line 30. If, however, U or D (or the SHIFTed version of those keys) is pressed, the value of A (which determines whether the pitch will go up or down) will be altered.

```
39 REM CHECK KEY INPUT
40 IF I = 255 THEN GOTO 30
50 IF I = 11 THEN A = -1:GOTO 20
55 IF I = 75 THEN A = -5:GOTO 20
57 IF I = 58 THEN A = 1:GOTO 20
60 IF I = 122 THEN A = 5:GOTO 20
```

If the keyboard input is a number key, then the computer will make the appropriate changes in the pitch that is "named" from the keyboard.

```
61 IF I=31 THEN F0=F0+A
62 IF I=30 THEN F1=F1+A
63 IF I=26 THEN F2=F2+A
64 IF I=24 THEN F3=F3+A
```

Then the Atari will check the values of F0, F1, F2, and F3 to make sure that none of the pitch values is "out of range." If any of them are, the Atari will make the proper corrections.

```
69 REM CHECK VALIDITY
70 IF F0>230 THEN F0=230
71 IF F1>230 THEN F1=230
72 IF F2>230 THEN F2=230
73 IF F3>230 THEN F3=230
75 IF F0<0 THEN F0=0
76 IF F1<0 THEN F1=0
77 IF F2<0 THEN F2=0
78 IF F3<0 THEN F3=0
```

Then the computer will again test the value of I (byte 764's value) to see if the T, N, or E key was pressed. If one of them was pressed, the Atari will take appropriate action, such as going to a subroutine or ending the program.

```
80 IF I=45 THEN GOSUB 250
81 IF I=35 THEN GOSUB 200
82 IF I=42 THEN GRAPHICS 0:POKE 764,172:END
```

As the pitches are being altered, the sound of the four voices is continuously being played. I used a "pure" tone with a fairly low volume of 7 so that the sound wouldn't be too annoying as it was being played. However, if you alter the value of T1 by pressing the N ("noise") key for any of the voices, then the sound of that voice will be altered.

```
 99 REM PLAY SOUNDS
100 SOUND 0,F0,T1,7
```

```
110 SOUND 1,F1,T1,7
120 SOUND 2,F2,T1,7
130 SOUND 3,F3,T1,7
```

The computer then returns to line 20 so that you can change any more sounds you want to alter.

```
140 GOTO 20
```

Finally, the subroutine below is used at lines 80 and 81 so that you can change a voice from "noise" to "tone" or vice versa.

```
199 REM CHANGE TO NOISE
200 I=PEEK(764)
210 IF I=31 THEN T0=8:RETURN
220 IF I=30 THEN T1=8:RETURN
225 IF I=26 THEN T2=8:RETURN
230 IF I=24 THEN T3=8:RETURN
240 GOTO 200
249 REM CHANGE TO TONE
250 IF I=31 THEN T0=10:RETURN
260 IF I=30 THEN T1=10:RETURN
270 IF I=26 THEN T2=10:RETURN
280 IF I=24 THEN T3=10:RETURN
290 GOTO 250
```

To use this program, just press U or D to change the pitch of a voice. After you press U or D, press the number of the voice you want to change (0 to 3). If you want to change the pitch more rapidly, press SHIFT U or SHIFT D. If you want to change a voice from a noise to a tone, press T and then the number of the voice you want to change. If you want to change a tone to a noise, press N and then the number of the voice. Finally, if you want to end the program altogether, press E.

You might have some ideas of your own for programs, or you might want to use parts of these three programs in your own program.

## Information on the Atari 600XL or 800XL

You have already learned a great deal about the Atari 600XL or 800XL personal computer, but you still need to know something about its special keys and key functions. Remember that these special-function keys do not make the computer do anything unless the software in the computer was designed to recognize them.

**SELECT:** used in some programs to select an item from a MENU of options. In the self-test program that comes with the Atari 600XL or 800XL computer, you can select a particular test with this key.

**RESET:** stops the computer, erases the screen, and prevents the computer from communicating with outside "peripherals." However, it *does not erase your program*, so don't be afraid to use it as a "panic button."

**START:** starts a program. If a program is designed to recognize this key, pressing it will start the program running.

**OPTION:** allows you to select among several variables within a program, if the program so allows.

**HELP:** provides extra help in some programs. Once you have received this help, you can return to the program.

**BREAK:** interrupts the function of a program, making the computer go back to BASIC language. Use this key rather than the RESET key as the standard method to stop a program.

**CONTROL:** functions somewhat like the SHIFT key in that it alters the function of a key. For example, if you press CONTROL along with an alphabet key you can produce graphics on the video screen (Fig. 8-2).

**CAPS:** switches between lowercase and uppercase modes.

**CONTROL CAPS:** switches between CONTROL LOCK and regular modes. If you are going to be typing a lot of graphics or

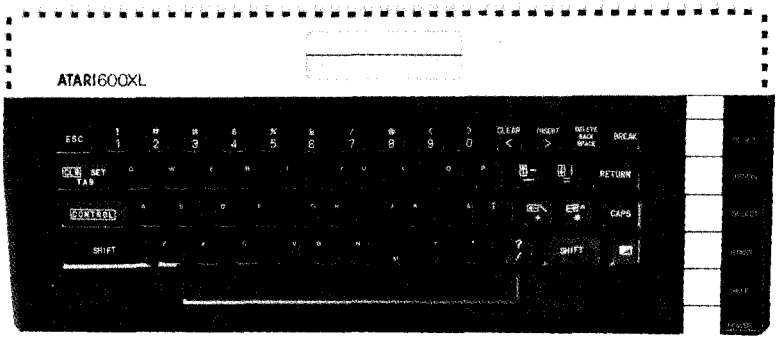


**Fig. 8-2 The graphics characters** ***(Courtesy of Atari, Inc.)***

international characters, you might want to use this key combination (Fig. 8-3).

**REVERSE VIDEO:** switches between regular video (white on black) and inverted video (black on white). Use this only for special printing effects.

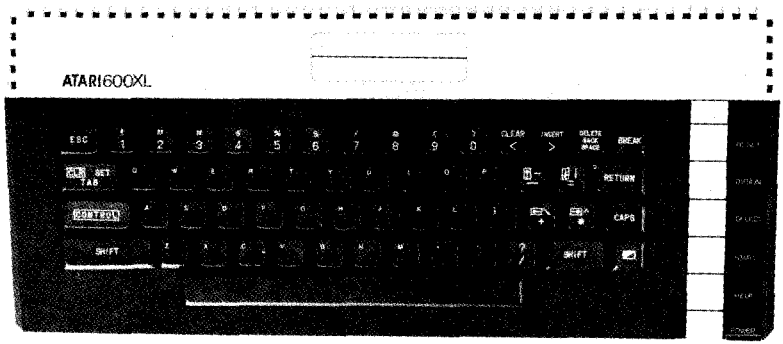


**Fig. 8-3 The International Character Set** ***(Courtesy of Atari, Inc.)***

There are a number of graphics and international characters available with the 600XL or 800XL. To print a graphics character, simply hold the CONTROL key down and press an alphabet key (A–Z) or the comma, period, or semicolon key. The above procedure also creates international characters, but only if you first type in the command **POKE** 756,224. To get back into graphic-characters mode, give the computer the command **POKE** 756,204.

## Buying Guide/Peripherals

There are a great many products available for the Atari 600XL or 800XL computer. Some are from Atari (Atari, Inc., 1265 Borregas Ave., Sunnyvale, CA 94086), but most are from "third-party" companies. Some products from Atari aren't the best, but because you need items which will be compatible with your computer and will be supported by Atari, I will name only the Atari products. Use this list as a guideline in deciding what "peripherals" may be added to your computer system.

### Input/Output Media

Probably the most important peripheral regarding your programming is the input/output medium because you'll want to save your programs and data for later retrieval. (Besides, you'll want to run commercial software as well.) You can use either a cassette recorder or a disk drive for this function.

**ATARI 1010 PROGRAM RECORDER:** This is very much like a cassette recorder in that it uses cassette tapes for storage and retrieval (Fig. 8-4). In addition, you can listen to the tapes play for instructional courses in Italian, Spanish, German, and French or audio instructions in other programs. However, this inexpensive input/output unit is slow, and if you want a program on the end of a cassette tape, you'll have to fast-forward the tape all the way to the end before you can retrieve it. Buy a program recorder only if you don't have the money for a disk drive.

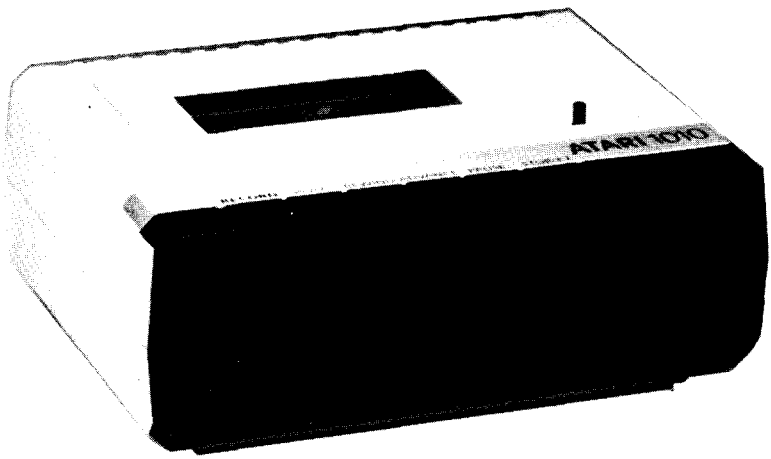


**Fig. 8-4 The Atari 1010 Program Recorder** *(Courtesy of Atari, Inc.)*

**ATARI 1050 DISK DRIVE:** Most good programs come on diskettes, 5¼-inch objects that hold programs and data. Disk drives are fast and reliable. For these reasons, they are the best medium for storing and retrieving your programs and data. You can store about a hundred pages of information on this device, and the time you save, along with the availability of programs on diskette, make the disk drive an excellent investment in your work with the computer (Fig. 8-5).

### Printers

One of the most popular peripherals is the Atari printer. There are actually three different printers available, as shown in Fig. 8-6.

**ATARI 1020:** If you are going to be doing a lot of graphics and would like them on paper, this four-color printer can produce

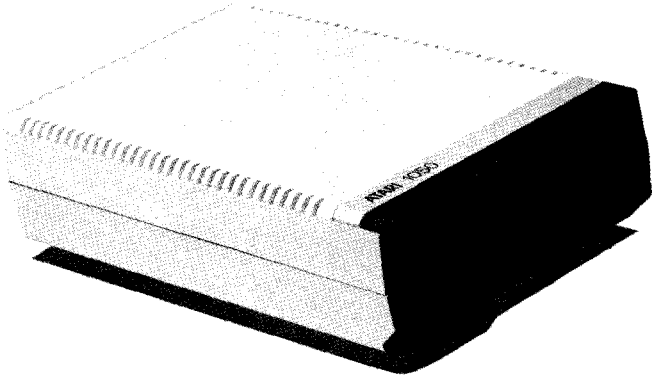

Fig. 8-5 The Atari 1050 Disk Drive *(Courtesy of Atari, Inc.)*

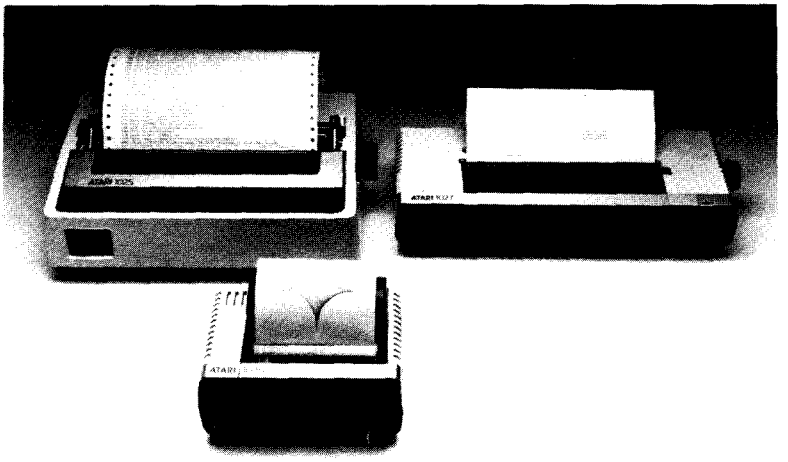

Fig. 8-6 The Atari line of printers: 1020, 1025, and 1027 *(Courtesy of Atari, Inc.)*

pictures and text, though at a much lower speed (10 character per second) than the 1025. The printer comes with software on cassette for you to use with the printer, along with the four pens it will use in printing text and graphics. Buy this printer only if you need graphs, pictures, and other visual information on paper.

**ATARI 1025:** This 80-column printer is *dot-matrix quality*, meaning it is excellent for home use, use with a word processor, and use in listing programs; but you shouldn't write formal letters with it because the fineness of the print isn't the best.

Although not perfect in letter quality, this printer is relatively fast (40 cps) and offers a choice of print styles (enlarged, condensed, and regular). This 80-column printer, standard among computer owners, is probably your best choice.

**ATARI 1027:** If you want your printout to look its finest, buy a *letter-quality printer*, which will give you hard copy resembling that of a typewriter. At 20 cps, this printer's speed is respectable, but it doesn't have the versatility or speed of the 1025. If you are going to be using a word processor mainly for letters, this would be your best choice; but listings and rough documents are better suited for less expensive dot-matrix printers.

### Some Helpful Add-ons

Atari makes some lesser known products that could prove to be quite useful to you (see Fig. 8-7).

**ATARI CX85 KEYPAD:** If you are going to be entering a lot of numbers into your computer, you will find a keypad more convenient than the keyboard for entering data. The keypad is designed specifically for programmers who are going to be entering a lot of data, those who want the computer for financial or business purposes. The keypad has buttons for all the digits, an enter/plus key, a minus key, and buttons for ESCAPE, DELETE, and "Yes/No" answers.

**ATARI 1030 MODEM:** Computer communications is a rapidly expanding field that offers you the conveniences of electronic

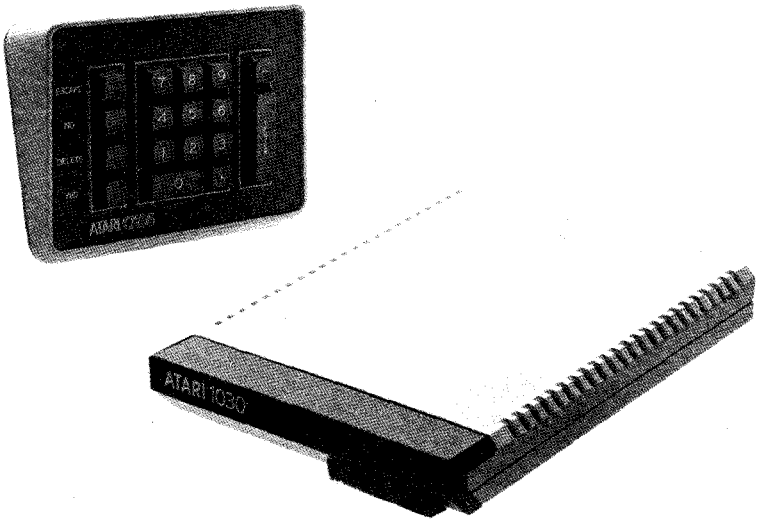


**Fig. 8-7 The Atari CX85 Keypad and the 1030 Modem** ***(Courtesy of Atari, Inc.)***

mail, on-line games, communications with hundreds of other people at nearly the speed of light, bulletin boards, and instant information. This modem enables you to hook up the Atari 600XL or 800XL with your telephone. Read a book on computer communications before you invest in this device, and if you feel you would like to get into this field, find a modem to purchase and get involved in the "world connection" which is happening now among thousands of home computer owners.

### Entertaining Add-ons

Some of the more frivolous, yet still useful, items you can purchase for your 600XL or 800XL (Fig. 8-8) include the following:

**ATARI JOYSTICKS:** These are among the least expensive but most desirable products you can get for your computer because they are essential to full enjoyment of many games you might

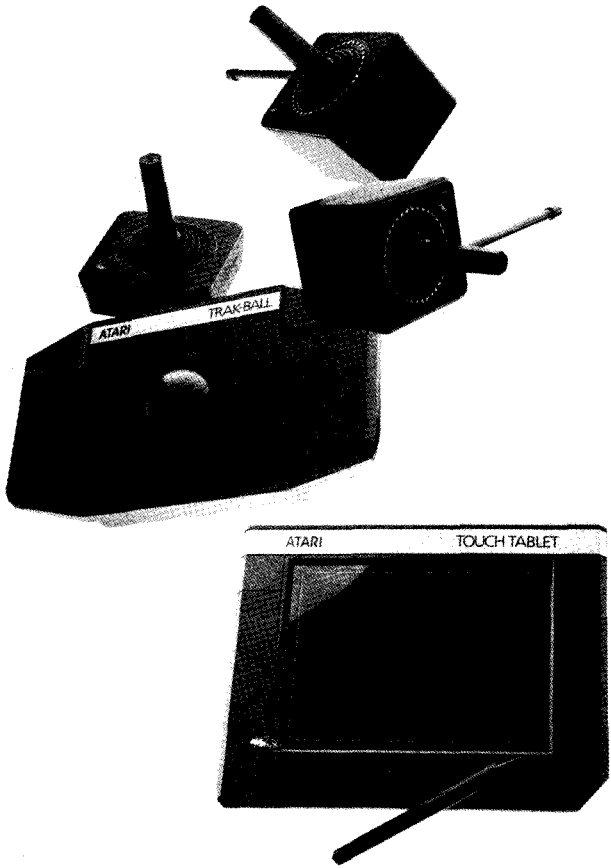


**Fig. 8-8 The Atari joysticks, Trak-Ball, and Touch Tablet** ***(Courtesy of Atari, Inc.)***

acquire for your system. You can buy a plug-in joystick or a remote-control joystick (which requires no wire connection).

**ATARI TRAK-BALL:** This add-on is needed for games that require quick directional changes such as the fast-action games, Centipede and Missile Command.

**ATARI TOUCH TABLET:** With this tablet, you don't even have to use your keyboard to draw pictures; you can "draw" on this tablet, and the graphics will appear on the screen. The graphics are actually produced by the Graphics Tablet Cassette Program accompanying this "peripheral." If you have a color printer, you will particularly enjoy this program because the graphics can be sent directly to the printer. Even if you don't own a color printer, you might want to try your artistic talent out with the Touch Tablet.

I have not covered *all* of the "peripherals" available for your computer. There is hardware available from other companies which can be used with the 600XL or 800XL, and there is a great deal of software from Atari and others that will work on these machines. But I *have* covered the major components you can add to your system.

In Chapter 9, we will explore the future of computers, personal robots, and your relationship with the "home computer revolution."

## Chapter Nine

# The Exciting XL

As useful and powerful as the Atari 600XL or 800XL is right now, many new "peripherals," including both software packages and hardware enhancements, will become available for these computers over the next couple of years. Some of the "peripherals" in the future of the 600XL or 800XL are a light pen, higher-quality printers, and personal robots which can be controlled by the computer.

Some of these products will come from Atari itself, but many will be made by companies other than Atari, such as Androbot, Inc., which will be manufacturing the Topo personal robot for the 600XL or 800XL home computer. You will be seeing advertisements for these products in computer magazines and newspapers.

As the software and hardware markets become more competitive, you will also see better quality in products for your computer. Although prices will not fall much below what they are now, you will be getting more for your money because only the best hardware and software will be offered for sale at the major stores.

## The Future of the Computer

Because home computers like the Atari 600XL or 800XL have been selling at such a brisk pace, some people believe that the home computer will soon be as commonplace as the television set. These people also feel that the computer will become the center of activity in the home and that people will begin to act and think like computers themselves.

I feel that these predictions are groundless. True, a large number of computers have been sold, but still, 95 percent of the population in the United States doesn't own a computer. Furthermore, not everyone *needs* a computer, so it's unlikely that everyone will purchase one. Television is popular because it entertains and brings information into the home. The computer is capable of doing both these things, but in an unusual way, and a full computer system costs at least $700, a sum not everyone in the country has to spare.

I do not believe that computers will reach as many homes as the "experts" would have us believe, but the people who *do* own computers will be at a definite advantage over those who don't because they will have access to the greatest amount of information in the shortest possible time; and they will have a source of entertainment, file management, and word processing all in one machine. Computers can greatly simplify life for those who own them, and many computer owners will come to wonder how they ever got along without them.

The one function of computers that will undergo the greatest change is computer communications.

Because many people will be using their home computers to "log on" to large data bases for electronic mail, playing games, or simply chatting with one another, a "human network" will form for those people who use and program computers (Fig. 9-1). The home computer will unify people who share common interests. Friendships will develop through computers, and computer communications will bring people closer together just as

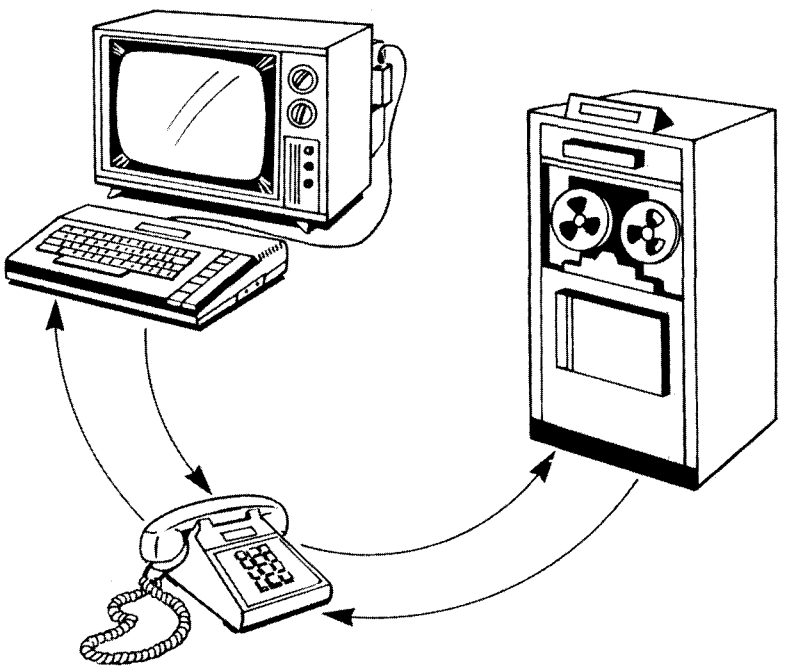

Fig. 9-1 The path of computer communications

the invention of the telephone brought us closer together. As computer technology develops, our planet will thus begin to seem smaller.

## Personal Robots

An important extension to the computer will be the personal robot. Right now, your Atari 600XL or 800XL is capable of controlling a robot such as the Androbot Topo machine. This squatty little creature can move about on command, detect objects, and even talk. A *personal robot* is simply a mobile extension of a computer with the power to manipulate physical objects. The computer will be the "brain" of the "probot" (my word for "personal robot"), while the robot itself will be the "brawn."

"Probots" are so useful that soon people will begin to think of computers as simply "handicapped" personal robots. Some of the possible uses for a "probot" in the home include:

1. Guarding the home, and alarming the occupants with a loud "Burglar! Burglar!" if it detects someone who shouldn't be present at that time
2. Passing out drinks at parties, and entertaining the guests with jokes
3. Walking the dog; baby-sitting; and waking you up at a specific time with a request to "Wake up, please" and a nudge

Actually, some "probots" today have computers already built into their "bodies." But if you already own the "brain" of a robot (the Atari 600XL or 800XL), you might want to check into the "brawn" (Topo from Androbot, Inc.) to see if *you* could use a personal robot.

## The Future of You and Your Atari 600XL or 800XL

The Atari 600XL or 800XL home computer is an inexpensive, versatile, and helpful machine. The computer has a lot of excel-

lent uses and is not, as many science-fiction movies would have us believe, a machine that we need to fear. Computers are built by humans, controlled by humans, and can be destroyed by humans. They are nothing more than tools with which we can work to make our lives more organized and enjoyable.

What positive effects will the home computer revolution have on us and our society? One positive change it will bring about is the better organization of the household. The entire maintainence, files, and security of the household can be put under the supervision of a fast computer with a large memory. This would result in greater safety within the house itself because anything constituting a threat to the home or family would be detected through the computer's security system. It would also result in closer unity of the household because all time schedules, documents, and other important information would be located within the computer's memory rather than on papers scattered throughout the home.

Computers can also improve our thinking habits. To work with a computer, a person must think in the way a computer "thinks"—that is, logically. After using a computer for an extended period of time, a person tends to think more logically.

If a problem is encountered with a computer, the owner must solve the problem by logical means because computers are based on logic. A computer programmer needs to work with logic constantly, and so his or her thinking is even more affected. It is almost inevitable that anyone who works with computers will begin to think in more logical patterns, meaning that he or she will begin to solve problems more efficiently by more systematic means. In addition, that person will seek further organization within his or her own life because of the way a home computer shapes the mind to think more logically, to work more efficiently, and to be more organized.

This does not mean that anyone who works with a computer becomes a robotic machine. In fact, an appreciation for the efficiency and organization of a logical computer actually increases

one's appreciation for the aspects of life that are *not* computerized but are still pleasing. Those who work with cold, hard logic all day tend to seek out things which are warm and alive in their spare time.

Your Atari 600XL or 800XL home computer may not change your personality drastically, but you will probably encounter some important positive changes in your life once you have worked with the machine for a while. The computer's efficiency, organization, and wide number of applications will some day make you, too, ask "How did I ever manage without it?"

# Index



## A

## B



## C

## D



## E



## F

## G



## H



## I



## J



## K



## L

## M



## N



## O



## P

## Q



## R



## S

## M



## N



## O



## P

## T



## U



## V

## W



## Z

Catalog

If you are interested in a list of fine Paperback books, covering a wide range of subjects and interests, send your name and address, requesting your free catalog, to:

McGraw-Hill Paperbacks
1221 Avenue of Americas
New York, N.Y. 10020

# HOW TO EXCEL ON YOUR ATARI 600XL AND 800XL

Timothy O. Knight

Get the most out of your new Atari 600XL and 800XL home computers. This book offers you all the information you'll need to know to make your 600XL or 800XL work for you. All key terms are defined and many are accompanied by illustrations.

HOW TO EXCEL ON YOUR ATARI 600XL AND 800XL has chapters on programming, graphics, sound and music, and they are written in simple, easy-to-understand terms. There's no need to worry about technical jargon. Instead, you'll read about how you can use your new home computer for business and fun.

This is the ideal, indispensable companion to the new and exciting Atari 600XL and 800XL machines that offer you:

- 16K RAM memory, expandable to 64K, for the 600XL; 64K memory for the 800XL
- a built-in help key
- an International character set
- 256 colors
- great sound, with a 3½ octave range with 4 individual voice channels
- The expanded 64K of memory of the 600XL and the 64K of the 800XL make each machine compatible with 2,000 high-powered Atari software programs and the extra memory makes word processing—still the most popular "utility" program—easier.

Let this book help you excel on your Atari 600XL or 800XL.

Timothy O. Knight is a young programmer. He is currently involved with his own company, Probotech, Inc.

ISBN 0-07-035104-X