

INTELLIVISION[®] MATTEL ELECTRONICS[®]

STEP-BY-STEP GUIDE TO HOME COMPUTING

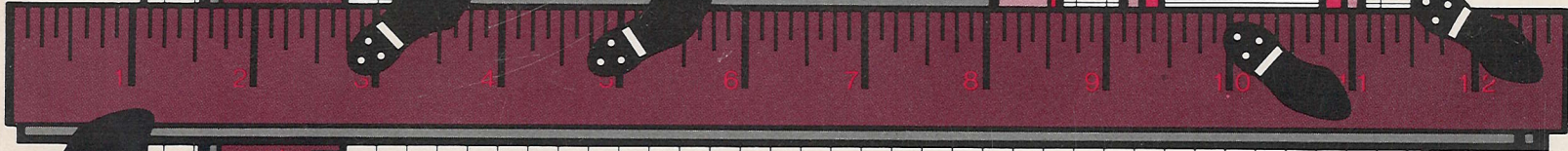
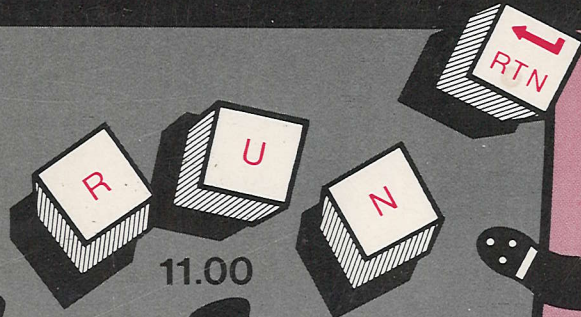


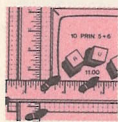
TABLE OF CONTENTS



INTRODUCTION	1
<ul style="list-style-type: none"> ■ Take A Look At Your Intellivision® Computer ■ Turning Your Computer On/Off ■ Who's Smarter, You Or The Computer? ■ What Is BASIC, Anyway? ■ What Is A Program? ■ Let's Get Started 	



CHAPTER 1: Introduction To Immediate Mode	7
<ul style="list-style-type: none"> ■ Let's Go Through A Command ■ Arrow Keys ■ Immediate Mode: As A Calculator; To Try Out Functions & Routines; To Interact With Programs (RUN, LIST, CLR, NEW, DEL, MENU) 	



CHAPTER 2: Introduction To Programmed Mode	17
<ul style="list-style-type: none"> ■ Let's Go Through A Program Step-By-Step ■ Line Numbers ■ Line Length ■ Return Key ■ Escape Key 	



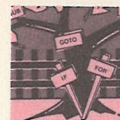
CHAPTER 3: Constants & Variables	23
<ul style="list-style-type: none"> ■ Constants: Numeric & String ■ Variables: Numeric & String 	



CHAPTER 4: Assigning Values To Variables	29
<ul style="list-style-type: none"> ■ Numeric Variables: When You Write A Program; When You Run A Program (INPU) ■ String Variables: When You Write A Program (SET & PUT); When You Run A Program (GET & PUT) ■ PRIN:Print ■ Color Coding ■ Punctuation ■ Memory 	



CHAPTER 5: Looping	39
<ul style="list-style-type: none"> ■ GOTO ■ FOR...NEXT 	



CHAPTER 6: Branching	47
<ul style="list-style-type: none"> ■ IF... ■ GOSUB ■ Nested Subroutine 	



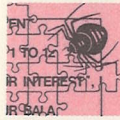
CHAPTER 7: READ...DATA	57
-------------------------------	----



CHAPTER 8: Arrays (DIM:Dimension)	61
--	----



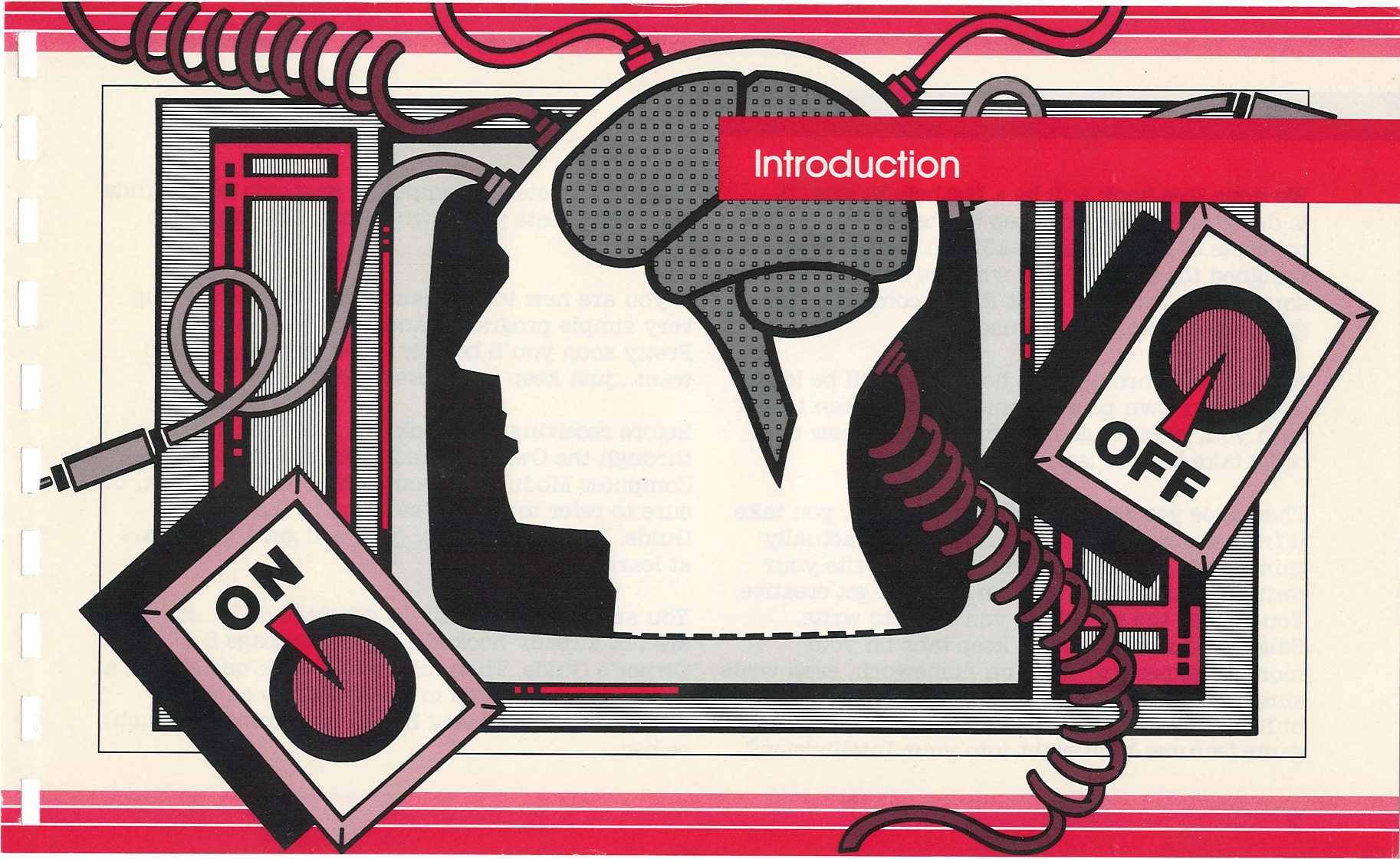
CHAPTER 9: Routines and Functions	69
--	----



CHAPTER 10: How To Write A Program	81
<ul style="list-style-type: none"> ■ Debugging A Program 	



CHAPTER 11: Sample Program	85
<ul style="list-style-type: none"> ■ Routines To Display Objects In Your Intellivision Game Cartridges 	



Introduction

Dear Owner:

We hope this book will be a big help to you. It is designed to take you step-by-step through the concepts of Intellivision BASIC. It is especially designed for those of you who are starting from square one, learning what BASIC computer programming is and how to use it!

As you read through this book, you will be learning at your own pace. If one chapter doesn't click with you, read it again. Remember, all new concepts take time to learn.

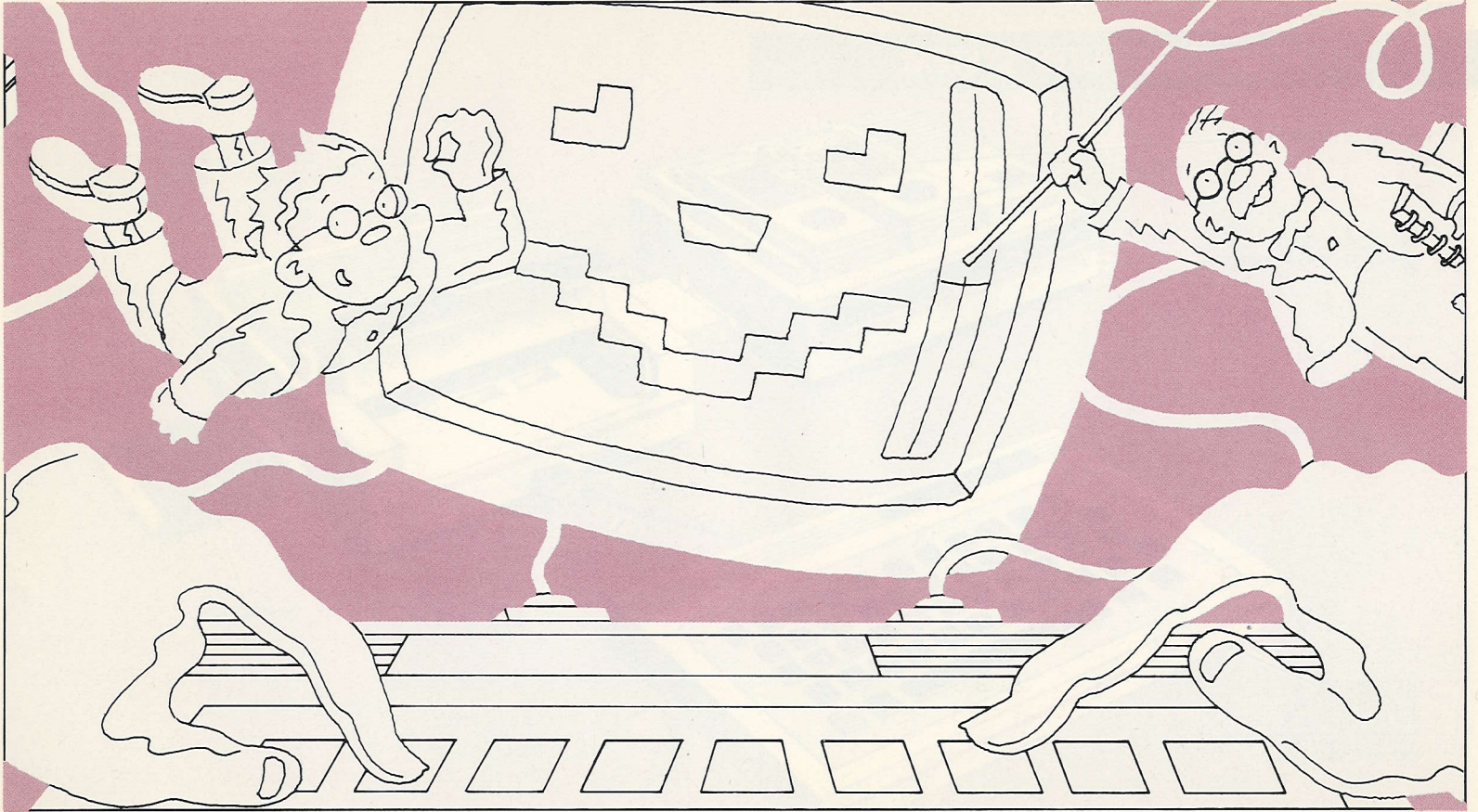
Then once you understand the concepts, you take it from there...because the next step is actually thinking in BASIC and being able to write your own programs. This is when you can get creative. You choose the programs you want to write. Balance your checkbook, keep tabs on your monthly expenses, work on homework, even write simple educational games. If you are a video game buff, you will be excited about the unique video game features built right into your Intellivision®

Computer system. Manipulate graphics and sounds and even create your own games. The sky's the limit.

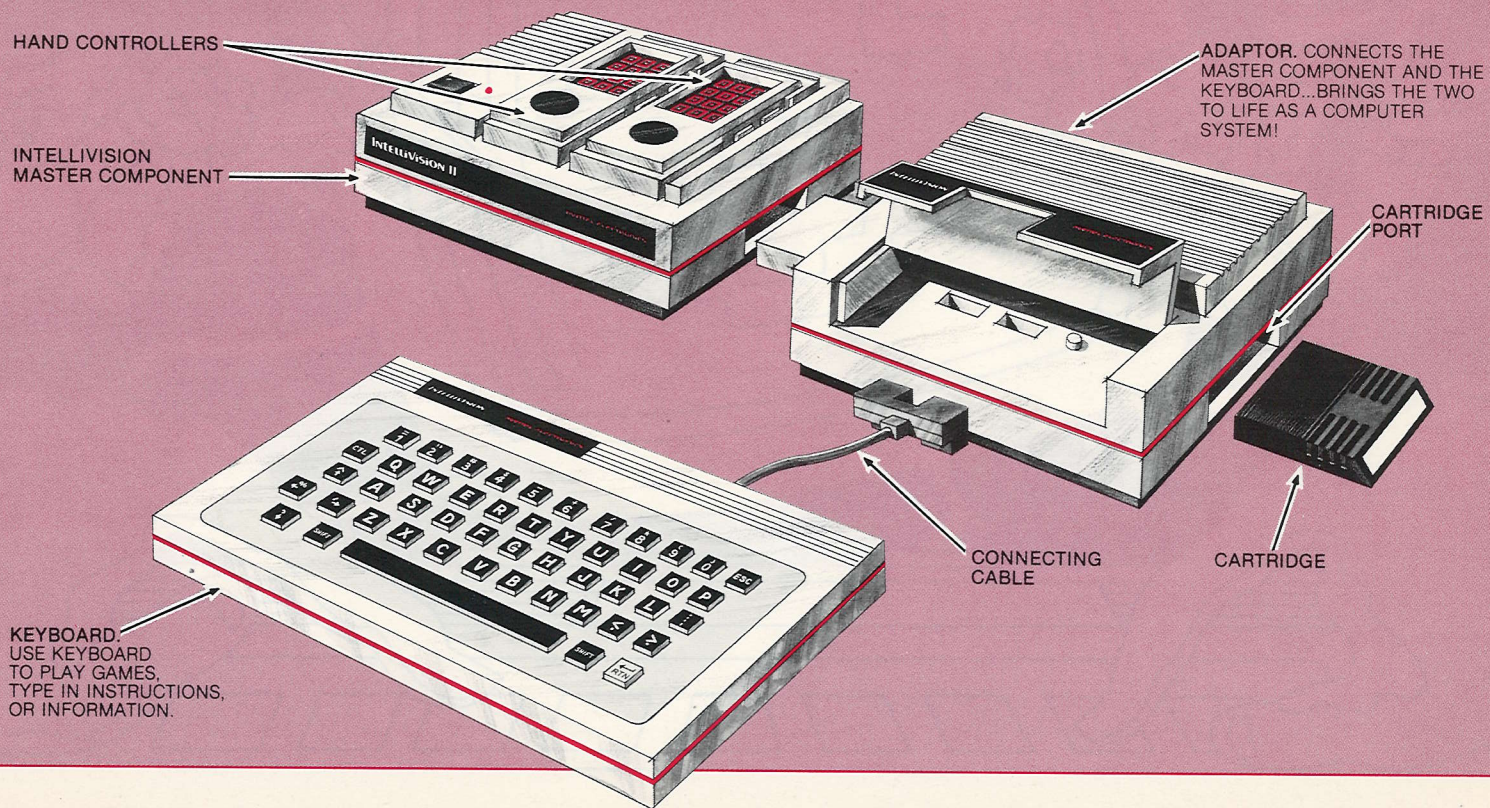
If you are new to programming, start by writing very simple programs, and work your way up. Pretty soon you'll be able to get as fancy as you want...just keep programming!

Before receiving this book, we hope you read through the Owner's Guide which came with your Computer Module. As you go through this book, be sure to refer to similar sections in the Owner's Guide. Between the two...you will get a good start at learning BASIC!

You are probably rarin' to get started. So...if you are not already hooked up, refer to page 8 in the Owner's Guide. Then follow this book one step at a time. Be sure to type in the examples as you go...it'll make learning BASIC programming much easier.



YOUR COMPUTER SYSTEM



TURNING YOUR COMPUTER ON/OFF

TURN YOUR COMPUTER ON

When your Intellivision computer system is hooked-up, it's time to get the power on.

First make sure you have inserted a cartridge then turn your Master Component ON.

Turn your TV set ON.

TURN YOUR COMPUTER OFF

To keep your system running properly, it's important to turn your computer off a certain way.

First turn off your TV set.

Then turn your Master Component off.

If for some reason you turn your Master Component off before you turn your TV off, the TV screen looks as though you are getting very poor reception. If the volume is turned up, you will hear a hissing sound. Simply turn your TV set off.

WHO'S SMARTER?



YOU OR THE COMPUTER?

A computer does not think! A computer is designed to store the information you give it, and use this information according to your instructions. It only does what you tell it to do. If you ask the computer to add $2+2$, it would do exactly that...add the two together...and that's all. If you want the answer, you have to ask for that too! You are the brains behind the machine!

WHAT IS BASIC, ANYWAY?

In order to communicate with a computer, you both need to speak the same language. There are a number of computer languages, but most are difficult for us to learn. So a special language was developed that computers can use and is easy for beginners to learn and use. This computer language is called BASIC. You will use the BASIC language to tell a computer what you want it to do.

Just like learning any new language, you have to learn a new vocabulary. Intellivision BASIC includes a few symbols and words called KEYWORDS, MONITOR COMMANDS, FUNCTIONS and ROUTINES. You will be learning about these as you read through the next 11 chapters. If you ever need a quick reference to any of these words, turn to the Owner's Guide Appendix A and look up the brief description.

These Keywords, Commands, Functions and Routines allow you to give the computer instructions...and instructions are the name of the game. When you have gotten a few instructions under your belt, you can use the computer in two different ways.

These are called MODES. You will soon learn about Immediate Mode and Programmed Modes...and how they differ.

WHAT IS A PROGRAM?

A program consists of a few specific elements. These include giving the computer information, telling it to store this information in its MEMORY, telling it to do something with this information and give you a result. Let's run through a "for instance" to see how a program works.

You could write a program that would have the computer cut a recipe in thirds and give you the new measurements. To write this program you would enter in the original recipe,

tell the computer to make exact calculations on this information, then request that it print the results. Voila, a program!

LET'S GET STARTED

Once you have hooked up your Intellivision computer system, you are ready to get started. Press **RESET** on the Master Component, then press the **Disc** on the Hand Controller and you see three main options listed on the TV screen. This is called the Main Menu.

You will only be learning about BASIC in this guidebook. To learn more about the cartridge and music options, turn to page 15 in the Owner's Guide.

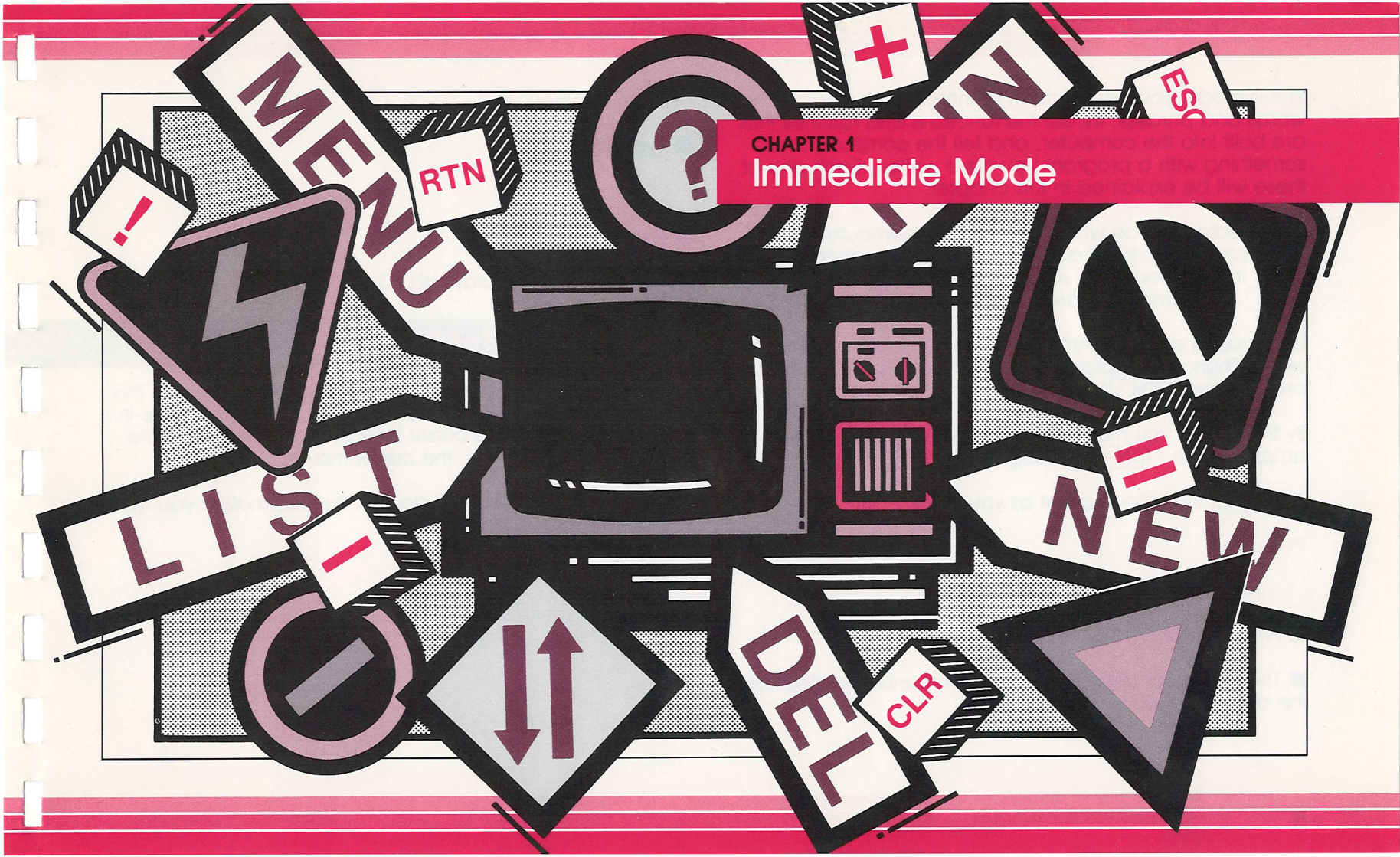
Now, in order to choose the BASIC option, press Key **1** then **ENTER** on the hand controller. What better place to start than with a blank screen!



1: BASIC ✚
2: CARTRIDGE
3: MUSIC

CHAPTER 1

Immediate Mode



In this mode, you can perform arithmetic operations, print words and phrases, try out the functions and routines that are built into the computer, and tell the computer to do something with a program you have written. Each one of these will be explained in this chapter.

In Immediate Mode when you give the computer instructions, you are giving it a COMMAND. It immediately executes this command as soon as you press the RETURN Key **RTN**. No commands are stored.

Let's type in a few commands and see how the computer works. Then we'll explain what you have typed in! Begin by pressing the **RTN** Key.

By the way, if you make a mistake when typing in these examples, press **RTN** and begin again.

- Type in the following just as you see it:

PRIN 5+6

PRIN stands for Print.

- Press the **RTN** Key
- The computer understood the command and displays the answer, 11.00

Let's do another:

- Type in:

PRIN "HELLO"

- Press **RTN**
- The computer displays HELLO

LET'S GO THROUGH A COMMAND

Cursor: You see a square at the left side of the screen. This is called the CURSOR. The next letter or symbol you type in will appear where the cursor is located on the screen. As you type in information, the cursor moves.

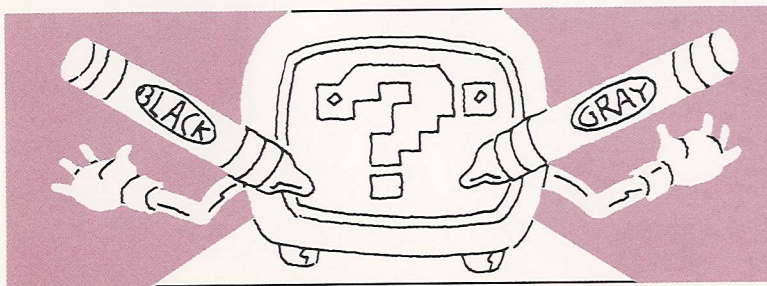
NOTE: The computer won't accept the information you type in if the cursor is flashing.

Type in information: Use the keyboard keys, space bar and shift just as in regular typing. Do not interchange the letters O and L for the numbers zero and one! Be sure that each line begins with the cursor at the far left side of the screen. Pressing **RTN** sends the cursor to this position. Now type in:

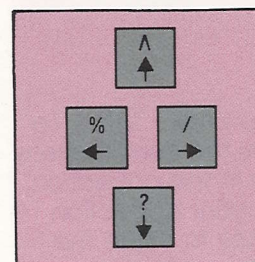
PRIN 10 + 10

RTN Key: In Immediate Mode the Return Key tells the computer to immediately execute the command you just gave. So now press **RTN**, and you will see 20.00 on the screen.

Color Coding: Did you notice the squares behind your command turned colors when you pressed **RTN**? These colors are really helpful. If the computer understands what you typed in, all the characters change color. But if some characters remain black or turn gray, this lets you know the computer doesn't understand that command. For more information about color coding, turn to page 36 in this book.



ARROW KEYS



You can correct or change something you have typed in. First position the cursor where you want to make the change. Do this by using the arrow keys. Each time you press one of these keys, the cursor moves one space.

If you want to make a correction before you press RTN, use the left arrow key. Let's try this.

Press **RTN** and type in:

PRIN "HELOO"

Now use the **left Arrow** Key and move the cursor on top of the first O. Notice that the cursor erased the right quotation mark and the two Os.

Now type in your correction:

LO''

Then press **RTN**.

You can also make a correction after you press **RTN**. To do this, use any of the arrow keys to move the cursor where you want to make the change. Now type in the correction, then either type in everything that follows your correction and press **RTN**...or use the right arrow key to move the cursor beyond the last character in that line and press **RTN**...or simply press the **Up** or **Down** Arrow Keys.

For more information about arrow keys turn to "Making Changes or Corrections" and "Special Editing Techniques" in the Owner's Guide.

USE AS A CALCULATOR

ARITHMETIC KEYS

You will use these keys in BASIC to do arithmetic operations.

+ to add Example: $10 + 5$

- to subtract Example: $3 - 2$

***** to multiply Example: $4 * 5$

/ to divide Example: $10 / 2$

The **+** **-** ***** **/** symbols are called OPERATORS.

ARITHMETIC OPERATIONS

You can do simple or really complex arithmetic operations. You can even combine operations in one expression. The tricky part is knowing how to type in the expression so that the computer calculates the answer in the order you want.

For example, if you type in $6 + 6 * 3$, does the computer calculate $6 + 6$ first then multiply the result times 3 to get 36? Or does it multiply 6 times 3 and then add 6 to the result, getting 24? The order in which the computer calculates the expression makes a difference in the answer you get.

The computer calculates arithmetic operations in the following order:

1. Any arithmetic operation within parentheses
2. Negation (a number with a negative value, like -3)
3. Multiplication and division
4. Addition and subtraction

For example, if you want the computer to add $2 + 2$ then multiply this result by 3, here's how you would type it in:

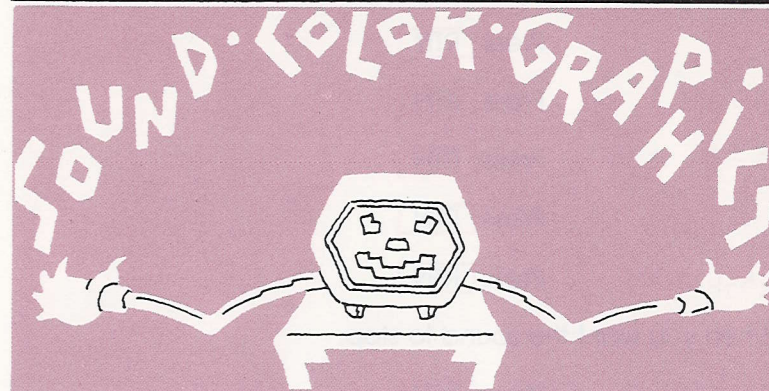
PRIN (2 + 2)*3

Try this!

You will see that the computer rounds off numbers to 2 decimal points.

You cannot divide by zero!

FUNCTIONS AND ROUTINES



There are mini programs called ROUTINES, and special tools called FUNCTIONS built right into the computer. These allow you to play with the objects in any Intellivision cartridge. They also allow you to create sound effects and play with color or graphics in the games you develop.

Following are just a few examples of functions and routines so you can see how they work in Immediate Mode. We won't be explaining what you are typing in. This comes later. We only want you to see what's possible. You will get the complete look at functions and routines in Chapter 9 starting on page 69, and in the Owner's Guide on page 36.

Sound Effects: The following routine is an example of one sound effect you can use. Type in:

E = 8 Press

P = 10 Press

C = 0 Press

L = 10000 Press

CALL ENVN Press

When you want the sound to stop:

CALL HUSH Press

Colors: The following function will give you a color display on your screen. Type in:

BK(20) = 0 Press

A black square appears on the second line of the screen. The cursor is at the top left of the screen. Now type in this next line right over the last:

BK(21) = 1 Press

Continue typing one line over the other. Each will bring a different color onto the screen.

BK(22) = 2 Press

BK(23) = 3 Press

BK(24) = 4 Press

Graphics: In order to work with the graphics, you must first bring an object onto the TV screen. Once the object is displayed, you can do all sorts of things with it. Following are the routines that will display an object for 2 different Intellivision game cartridges: BURGERTIME™* and NIGHT STALKER™. (If you do not have any of these cartridges, turn to page 37 in the Owner's Guide and learn how to display an object for a wide variety of Intellivision game cartridges.)

DISPLAY AN OBJECT:

BurgerTime™*

Night Stalker™

M = 11

M = 5

N = 80

N = 22

O = 1

O = 1

D = 2

D = 1

CALL GRAB

CALL GRAB

You will see the pickle.

You will see a bat flying.

CHANGE COLOR:

Co(1) = 7

Co(1) = 13

Co(0) = 6

ANIMATES OBJECT:

XV(1) = 20

SQ(1) = 25

SQ(1) = 20



REMEMBER, this is just a preview of the many functions and routines built into your computer!

INTERACT WITH PROGRAMS

There are several commands which you will use in Immediate Mode that do something to the screen or to a program you have written. These commands are RUN, LIST, CLR (Clear), NEW, DEL (Delete) and MENU.



Important, Until you have learned about the Programmed Mode, skip this section and turn to Chapter 2 on page 17. We will tell you when to come back to this section.

Each of these commands must begin at the far left side of the screen. This lets the computer know you are in Immediate Mode. Also, these commands do not change color when you press **RTN**.

RUN When you have finished writing a program, type in **RUN** and press **RTN**. This sets the program in action.

Example; type in:

10 PRIN "HAPPY" Press **RTN**

20 PRIN "BIRTHDAY" Press **RTN**

30 PRIN "TO" Press **RTN**

40 PRIN "YOU" Press **RTN**

RUN Press **RTN**

You see HAPPY BIRTHDAY TO YOU on the screen.

LIST When you want to look at the program currently in the computer's memory, type in **LIST** and press **RTN**. The first 12 lines of the program show on the screen. If there are more than 12 lines in the program, the computer begins to scroll the program...and the lines at the top move off the screen and the remaining lines come on one at a time.

Example; type in:

LIST Press **RTN**

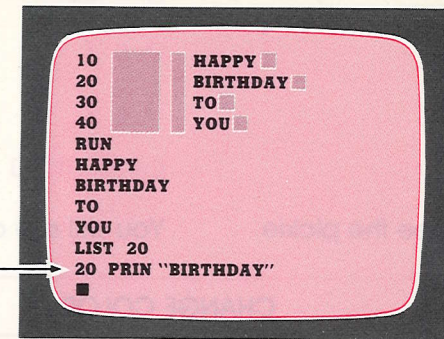
You see the program you entered above.

You can also look at a certain line in a program. To do this, type in **LIST** followed by the line you want to see.

Example: Let's look at line 20 of our example program. To do this, type in:

LIST 20

Press **RTN**
and you see:



You can also look at a certain portion of a program. To do this, type in **LIST**, followed by the line number of the first line you want to see, a comma, and the line number of the last line you want to see. In our example, to look at the first three lines, you would type in:

LIST 10,30

Press **RTN** and you see:

10 PRIN "HAPPY"

20 PRIN "BIRTHDAY"

30 PRIN "TO"

You cannot type in LIST while a program is in the process of being run. Press the **ESC** (Escape) Key and this stops the program. Then you can LIST.

You can make changes to a program once it is listed. Make a change as you did earlier, see arrow keys page 9, and line numbers page 20.

CLEAR: When you want to clear the screen, type in CLR and press **RTN**. Whatever was on the screen is now gone. The cursor appears at the top left of the screen. Type in:

PRIN 2 + 2 Press **RTN**

PRIN 5 + 3 Press **RTN**

PRIN "HOW ARE YOU" Press **RTN**

CLR Press **RTN**

The screen is now clear, everything you typed is gone.

If you have entered anything in Programmed Mode and you use CLR, the screen is cleared, but the program is not erased from the computer's memory. CLR does not erase the memory. It only erases what's on the screen. Type in:

10 PRIN "HELLO" Press **RTN**

20 PRIN "HOW ARE YOU" Press **RTN**

RUN Press **RTN**

CLR Press **RTN**

LIST Press **RTN**

You see that your last program is still in the computer's memory.

Use the CLR command in a program: You can also use the CLR command in a program. When the program is run, the screen will go blank when it gets the CLR command.

Type in:

10 PRIN "2 + 3 =" Press **RTN**

20 PRIN 2 + 3 Press **RTN**

30 CLR Press **RTN**

40 PRIN "5 + 5 =" Press

50 PRIN 5 + 5 Press

RUN

You see 2+3=5, the screen clears and you see 5+5=10 on the screen.

NEW When you type in NEW, the program in the computer's memory is erased. Oftentimes, when you are done with a program, you will want to begin the next program with a clean slate. To do this, type in NEW and press .

Example; type in:

10 PRIN 2 + 2 Press

RUN Press

LIST Press

You see the program you just wrote. Now type in:

NEW Press

LIST Press

The program is no longer in the computer's memory. When you use the NEW command, the computer automatically sets all variables equal to zero.

DELETE: There may be times that you want to delete certain lines from a program. To delete one line, type in DEL followed by the line number you want deleted.

To delete more than one line, type in DEL followed by the first line you want deleted, a comma and the last line you want deleted. When you press RTN, the computer deletes the lines you listed and those inbetween.

Example: Let's say your program covers lines 10 — 100, numbering in increments of ten. Let's also say that you would like to delete lines 70, 80 and 90. Here's what you would type in:

DEL 70,90 Press

MENU: Look at page 57 in the Owner's Guide to learn about Menu.

CHAPTER 2

Programmed Mode

10 PRIN 5+6

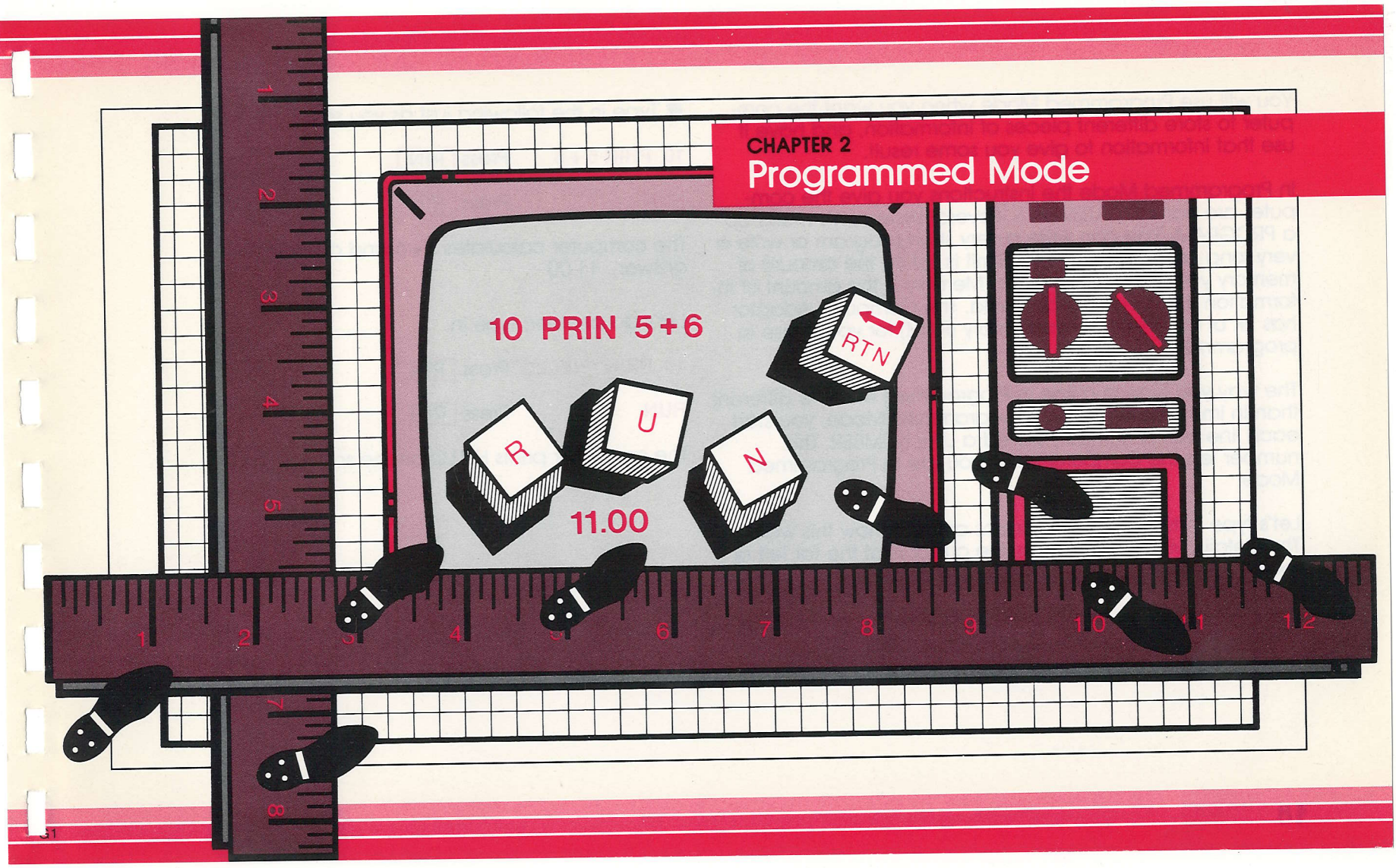
R

U

N

11.00

RTN



You will use Programmed Mode when you want the computer to store different pieces of information, and have it use that information to give you some result.

In Programmed Mode the instructions you give the computer are called STATEMENTS. Several statements make up a PROGRAM. You can write a very short program or write a very long program. Your only limit is set by the amount of memory you have to work with. Memory is the amount of information the computer can hold. The Computer Adaptor has 2K of random access memory (RAM) for you to use in programming.

The way you type in a line in Programmed Mode is different than in Immediate Mode. In Programmed Mode, you start each line with a number, called a LINE NUMBER. This number lets the computer know you are in Programmed Mode.

Let's type in two simple programs and see how this works. The details will follow. Be sure the cursor is at the far left of the screen. Press **RTN**.

■ Type in the following just as you see it:

10 PRIN 5+6 Press **RTN**

RUN Press **RTN**

The computer calculates 5+6 and displays the answer...11.00

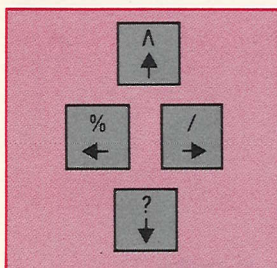
Let's do another; type in:

10 PRIN "HELLO" Press **RTN**

RUN Press **RTN**

The computer prints HELLO on the screen

LET'S GO THROUGH A PROGRAM



Cursor & Arrow Keys: You use the cursor and arrow keys the same here as you did in Immediate Mode.

Line Numbers: Each statement in Programmed Mode must begin with a line number. This is the computer's cue that you are in Programmed Mode. (All the information about line numbers is on page 20) Be sure that the cursor is at the far left side of the screen. Now begin a statement by typing in the line number. For example, type in:

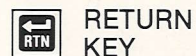
10

Type In A Statement: Type in the statement right after the line number. For example, type in:

PRIN "HELLO"

Your statement will look like this:

10 PRIN "HELLO"



When you press the RTN Key in Programmed Mode you are telling the computer to store the information just typed in. This is different than in Immediate Mode. (Remember, when you press RTN in Immediate Mode, the computer acts on the command immediately.) In Programmed Mode, pressing RTN puts this information in a holding pattern until you tell the computer to do something with it. You will press RTN after you type in each line! For example, press **RTN** now.

Check The Color Coding: Remember to look at the colors. If every letter and symbol turned color, the computer understands your statement. If there are still black or gray letters or symbols on the screen, use the Arrow Keys to position the cursor, then make your correction. Then press **RTN** again. For more information on color coding, turn to page 36 in this book.

RUN: When you have written all the statements, and the program is completed, it's time to tell the computer to do something with this program. To do this, you will type in a

command in Immediate Mode. Remember this command has no line number! In our example, type in:

RUN

RTN

You now see HELLO on the screen.

LINE NUMBERS

Additional information on line numbers is in Chapter 9 in the Owner's Guide.

Numbers You Can Use: The highest number you can use as a line number is 31999.

Here are the particulars about line numbers:

- Whole numbers only (Example: 2,3,10,100)
- Positive numbers only (Example: never use -10)
- Numbers only (Example: not 10B)

Adding Line Numbers: A good rule of thumb is to use line numbers in increments of 10. This gives you the space needed if you decide to insert a line later. Let's see how this works...

For example, type in:

NEW

10 PRIN "HELLO"

30 PRIN "GOOD-BYE"

Let's add a line, type in:

15 PRIN "HOW ARE YOU"

LIST

You see that your line numbers are now in order.

Replace A Line Number: The computer will only save one statement for each line number. So if you type in the same line number more than once, the last statement you typed in replaces the first. Let's try this in another example. Type in:

NEW

10 PRIN "HELLO"

20 PRIN "HOW ARE YOU"

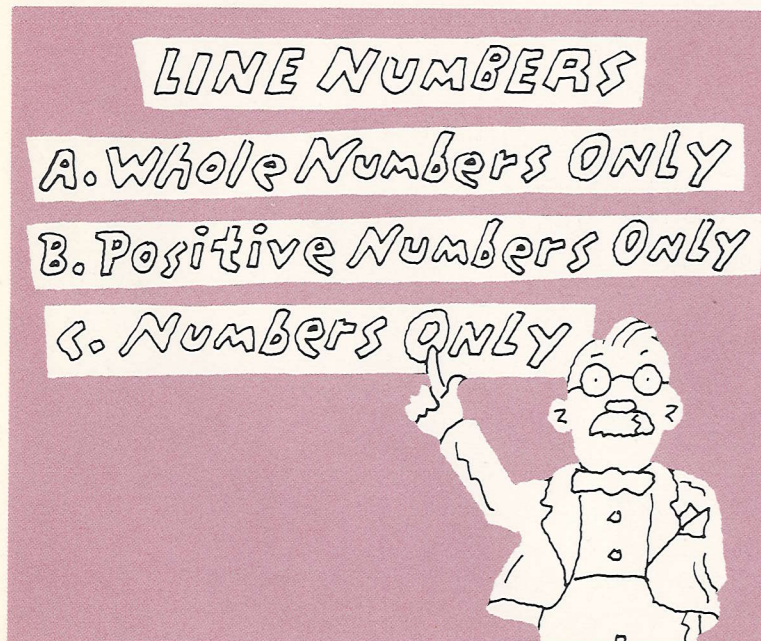
20 PRIN "WHAT IS YOUR NAME"

30 PRIN "SEE YOU LATER"

LIST

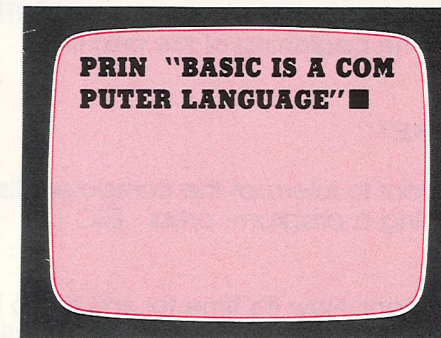
Here's what you'll see when you list this program:

```
10 PRIN "HELLO"
20 PRIN "WHAT IS YOUR NAME"
30 PRIN "SEE YOU LATER"
```



LINE LENGTH

Each letter, number, symbol or space is one character. In Immediate and Programmed Modes, you can type in a maximum of 39 characters per line. The computer doesn't allow you to type a line longer than that. When you reach this maximum line length, you must press RTN to continue typing.



You can type in 20 characters across your TV screen, then the cursor automatically goes to the next line on the screen and you can continue typing up to a total of 39 characters. For example, type in the following to see how this works:

```
PRIN "BASIC IS A COMPUTER LANGUAGE"
```

Turn to page 23 in the Owner's Guide for more information on line length.



RETURN KEY

Let's quickly recap the RTN Key! Press RTN in Immediate Mode and the computer takes immediate action. In Programmed Mode, you press RTN after you have completed each statement. This puts the information into the computer's memory. In both modes, pressing RTN sends the cursor down to the beginning of the next line.



ESCAPE KEY

Any time you want to interrupt the computer's listing or printing or running a program, press **ESC**.



Important: Now it's time for you to go back to page 13 and learn about the Immediate Mode commands you will use with a program.



CHAPTER 3

Constants & Variables

SUSAN

ANN

JEFF

WHAT IS YOUR NAME?

Up to now we have talked about giving the computer information. This information is called DATA. Now it's time to learn the different kinds of data you can use. There are constants and variables, and these are the nitty gritty of BASIC.

When you write a program, you will want some data to remain the same any time you run the program. This is constant data. You will also want to use data that can change. This is variable data.

As a simple example, you might write a program that asks for a person's name, then prints that name. In this case, you always want the computer to ask "What is your name?". This question is constant data. Then you want the computer to accept any name you type in and print it out. Because the name you type in can change each time you run the program, the name is variable data.

Constants and variables are further broken down. You can use two kinds of constants and two kinds of variables:

- Numeric constants
- String constants
- Numeric variables
- String variables

Numeric data is numbers only. String data is a combination of numeric characters, letters and symbols.

If this information about constants and variables is brand new to you, read this chapter carefully and the one that follows. Be prepared to go back to re-learn this information until it becomes second nature to you.

CONSTANTS

Whenever you want to use data over and over and never have it change, you will be using constants. Constants do just what the name implies...they do not change! There are two kinds of constants you can use: numeric constants and string constants.

WHAT ARE NUMERIC CONSTANTS?

The numbers you use in an arithmetic operation are called numeric constants. A numeric constant is always a number.

A numeric constant can be any of the following:

- Whole number (Example: 3 45 500)
- Negative number (Example: -6 -23)

- Decimal number, negative or positive (Example: 3.4 -5.6)

- A number larger than 999999, expressed in scientific notation† (3.2E06 is scientific notation for 3200000)

- A number smaller than .000001 expressed in scientific notation† (9E-07 is scientific notation for .0000009)

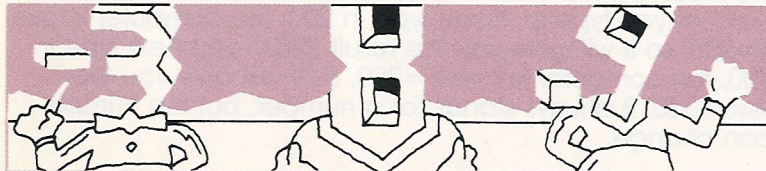
How To Use Numeric Constants: Numeric constants are always used in an arithmetic operation. You can add, subtract, multiply and divide numeric constants.

For example:

PRIN 3+7

The 3 and 7 are numeric constants. Their numeric values always stay the same...3 and 7.

†Intellivision scientific notation numbers can have 3 numeric digits left of the E (3 significant figures), and the exponent goes from +72 to -70.



WHAT ARE STRING CONSTANTS?

Whatever letters, numbers or symbols you enclose in quotation marks is a string constant. This is also called a literal string. Whatever you type within quotes is printed that way each time you run the program.

A string constant can be any of the following:

- Numbers only (Example: "50" "4")

- Letters only (Example: "HI" "WHAT")

- Combination of letters, numbers and symbols (Example: "2+2 EQUALS")

How To Use String Constants: Whenever you want something to appear exactly as you type it in, you will use string constants. String constants can never be used in an arithmetic operation.

For example:

PRIN "HELLO"

Since HELLO is typed within quotation marks, it is a string constant.

Combine Numeric & String Constants In The Same Program

```
10 PRIN "WHAT IS 2 + 2?"  
20 PRIN 2 + 2  
RUN
```

STRING CONSTANTS
NUMERIC CONSTANTS

When you run this program, the computer will print out the string constant WHAT IS 2+2?, then prints the calculation of the numeric constants 2+2 and displays the answer, 4.

VARIABLES

Variables identify data that may change. There are two kinds of variables you can use: numeric variables and string variables.

WHAT ARE NUMERIC VARIABLES?

A numeric variable stands for a number that may change.

As an example of this, let's say you want a program to add 10% to a certain number and give you the result. This program can be written using constants or variables. Let's see how variables make this a much more versatile program.

If you use a numeric constant, the program would look like this:

```
10 PRIN 100 + (100*.1)
```

Here the computer would calculate 10% of 100, add this amount to 100 and give you the result of 110. This program would only add 10% to the constant you use. If you want to know the result of adding 10% to ten different numbers, you would have to write this program a total of ten times...once for each number.

But if you use a numeric variable in this program, you can use this same program over and over! The program would look like this:

```
10 INPU B  
20 PRIN B + (B*.1)
```

B is the numeric variable that stands for any number you want. Here the computer is prepared to calculate 10% of any number, add this amount to the number and give you the result. INPU B means to type in a number. B equals the number you type in. If you type in 200, the computer knows B=200, so it will give you the result 220. If you next type in 300, the computer knows B=300, so it will give you the result 330. B always stands for a number, but the number can change.

A numeric variable can be any of the following:

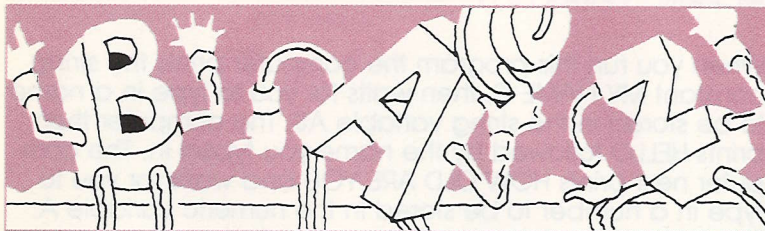
- Any single letter from A — Z

How To Use Numeric Variables: A numeric variable stands for a number. Therefore you will always use a numeric variable in arithmetic operations. You can add, subtract, multiply and divide numeric variables. The computer always uses the current numeric value for the variable.

For example:

```
PRIN B*.1
```

Here the computer would multiply the current value for the numeric variable B times the numeric constant .10.



WHAT ARE STRING VARIABLES?

A string variable stands for a string that may change.

As an example of this, let's say you want a program to print HELLO followed by a name. Let's compare a program using a constant and a program using a variable.

If you use a string constant, the program would look like this:

```
10 PRIN "HELLO MARY"
```

When you use constants, the computer will only print the string constant you use. Here the computer would print HELLO MARY. If you want to print HELLO JOHN or HELLO ANN, you would have to write this program again for each name you want.

But if you use a string variable in this program, you can use this same program over and over. The program would look like this:

```
10 GET A$
```

```
20 PRIN "HELLO"
```

```
30 PUT A$
```

A\$ is the string variable that stands for any string constant you want (up to 20 characters). Here A\$ stands for any name you want. GET A\$ tells you to type in a name. The name you type in is stored in the string variable A\$. Next the computer prints HELLO. Then PUT A\$ tells the computer to print the name stored in A\$.

After you type in the program, and type RUN, you can type in a name. If you type in ANN, then A\$=ANN, and the computer would print out HELLO ANN. If you next type in FRANK, the computer would print out HELLO FRANK. Here the string variable A\$ always stands for a name, but the name can change.

Important: GET and PUT will be more fully explained in the next chapter.

A string variable can only be:

■ A\$ B\$ C\$

How To Use String Variables: You will use string variables just like string constants. String variables cannot be used in arithmetic operations. The computer always uses the current value for the variable.

For example:

PUT A\$

PUT means "print" in string variables. Here the computer would print the current value for the string variable A\$.

Numeric & String Variables Are Combined In The Same Program:

```
10 PRIN "MY NAME IS"
20 GET A$
30 PRIN "HELLO"
40 PUT A$
50 PRIN "HOW OLD ARE YOU?"
60 INPU A
70 PRIN "I AM",A
80 PRIN "YEARS OLD"
```

STRING CONSTANT

STRING VARIABLE

NUMERIC VARIABLE

When you run this program the computer prints the string constant MY NAME IS, then waits for you to type in a name to be stored in the string variable A\$. The computer then prints HELLO followed by the name you typed in. The computer next prints HOW OLD ARE YOU and waits for you to type in a number to be stored in the numeric variable A. The computer prints I AM, followed by the number you typed in, followed by YEARS OLD.

Important: GET, PUT and INPU will be more fully explained in the next chapter.

CHAPTER 4

Assigning Values To Variables



You have learned what numeric variables and string variables are...now it's time to learn how to assign values to these variables.

Assigning means to give something a value. For example, if you assign the numeric variable A with the value of 2, you would write it as: $A=2$. When you assign a value to a variable, the computer stores this information.

You can assign a value to a numeric variable or a string variable. You can assign a value to a variable when you write a program or when you run a program.

NUMERIC VARIABLES



Remember, a numeric variable must always stand for a number!

ASSIGN A VALUE WHEN YOU WRITE A PROGRAM

You can assign a value to a numeric variable when you write a program.

For example:

NEW

10 A = 5

20 B = 10

30 PRIN A + B

RUN

This program has two numeric variables, and the values for each are written right into the program. A is equal to 5 and B is equal to 10.



Note: You can also assign a value to a numeric variable in Immediate Mode. This saves memory space as you don't have to store this information with a line number.

For example:

NEW

A = 5

B = 10

10 PRIN A + B

RUN

USE INPU (INPUT) TO ASSIGN A VALUE WHEN YOU RUN A PROGRAM

You can also assign a value to a numeric variable when you run a program. Do this by using the Input Statement, INPU. This tells the computer to stop running the program so that you can enter a number, and assigns the number you enter to the variable.

For example:

NEW

10 PRIN "HOW OLD ARE YOU?"

20 INPU A

30 PRIN "YOU ARE", A

RUN

Here's how the computer will run this program:

- 10** The Keyword PRIN tells the computer to print the string constant, "HOW OLD ARE YOU?" on the screen.
- 20** INPU tells the computer to STOP and waits for you to enter a number. Here A stands for your age. The cursor is at the far left of the screen and the computer is waiting for you to type in your age. Type in your age and press [RTN]. The number you type in is stored in the numeric variable A.

- 30** Then the computer prints the string constant "YOU ARE" and the value for the numeric variable, A. (That's the number you just typed in.)

YOU CAN USE INPU WITHOUT USING THE PRINT STATEMENT

Let's see how this is done:

Type in this same example another way:

NEW

10 INPU "HOW OLD ARE YOU?", A

20 PRIN "YOU ARE", A

RUN

Here's how the computer will run this program:

- 10** The computer prints HOW OLD ARE YOU?, then waits for you to input your age.
- 20** The computer prints YOU ARE, then prints the value of A.

Here's another example of input:

NEW

10 INPU "ENTER BALANCE", B

20 I = B * .1

30 PRIN "YOUR INTEREST IS", I

10 The computer prints ENTER BALANCE and waits for you to input a number for the variable B.

20 The variable I stands for interest. Here interest equals balance times 10%.

30 The computer prints YOUR INTEREST IS and prints the value for I.



Important:

■ A numeric variable can be set equal to a numeric constant, (Example: A=5)

■ A numeric variable can be set equal to another numeric variable
(Example: A=B)

■ A numeric variable can be set equal to another numeric constant plus, minus, times, or divided by a numeric variable
(Example: A=5+B)

INPU may be followed by from 1 to 7 variables.

For example: **INPU A,B,C**

INPU may be followed by one string constant, up to 20 characters long, with 1 or more numeric variables.
For example: **INPU "THE NUMBERS ARE", A,B**

STRING VARIABLES

USE SET TO ASSIGN A VALUE WHEN YOU WRITE A PROGRAM

You must use a SET statement to assign a value to a string variable. AND you must use the PUT statement to print a string variable. SET assigns a value...PUT prints the value of the string variable. For example:

NEW

10 SET A\$ = "HELLO"

20 PUT A\$

RUN

10 SET A\$ assigns the string constant "HELLO" to the string variable A\$.

20 PUT A\$ tells the computer to print the value for A\$. Here it will print HELLO.



Note: You can also assign a value to a string variable in Immediate Mode. This saves memory space.

For example:

```
SET A$ = "INTEREST"
```

USE GET TO ASSIGN A VALUE WHEN YOU RUN A PROGRAM

You will use the GET statement to assign a value to a string variable. Again, the PUT statement is used to print a string variable. When the computer runs your program it stops at your GET statement and waits for you to assign a value to a string variable. (This is just like INPU for a string variable.)

For example:

```
NEW
```

```
10 PRIN "WHAT IS YOUR NAME?"
```

```
20 GET A$
```

```
30 PRIN "HELLO"
```

```
40 PUT A$
```

```
RUN
```

Here's how the computer will run this program:

- 10 The Keyword *PRIN* tells the computer to print the string constant "WHAT IS YOUR NAME?".
- 20 *GET* tells the computer to get the value for the string variable *A\$*. It waits until you type in your name (a string constant). Type in your name and press **RTN**. The computer now stores your name as the string variable, *A\$*.
- 30 Then the computer prints the string constant "YOUR NAME IS"
- 40 and prints the value for the string variable *A\$* (that's the name you just typed in.)



Note: GET and PUT statements may be followed by one string variable. A string variable can be assigned a value up to 20 characters long.



Important:

- A string variable can be set equal to a string constant.

Example:

```
SET A$ = "HELLO"
```

- One character of a string variable may be set equal to one character of another string variable.

Example:

```
SET A$ = "ABC"  
SET B$ = "123"
```

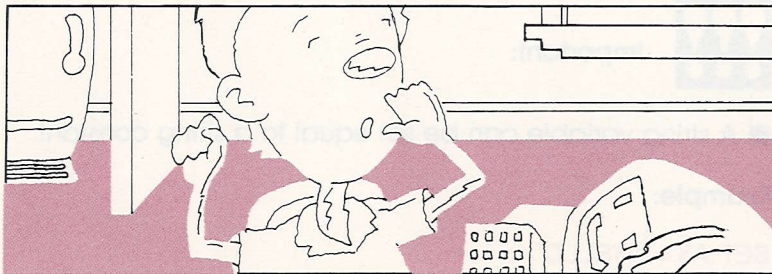
```
A$(1) = B$(2)
```

- One character of a string variable may be set equal to a numeric constant, using numbers from 0-63.

Example:

```
A$(1) = 20
```

It's Break Time! Time to get up and stretch, fix yourself a snack...take time out to play an Intellivision game. Don't try to learn too much all at once. Pace this out so you don't get overwhelmed!



PRINT (PRIN)

Let's get down to the nuts and bolts of the keyword PRIN. You have now used PRIN in Immediate Mode and Programmed Modes. In both, you can use the keyword PRIN in three different ways.

Use PRIN With String Constants:

You can only have 1 string constant per PRIN statement.

Example:

```
PRIN "2 APPLES"
```

Use PRIN With Numeric Constants:

You can use from 0-6 operators with a PRIN statement (+ - * / () ,).

Example:

```
PRIN 6 + 6, 3 - 2, 4 + 5
```

Example:

PRIN A,B,C

You can also use PRIN with a combination of a string constant and a numeric variable...and a string constant and an arithmetic operation.

Example:

10 PRIN "THE ANSWER IS", 6 + 7

10 PRIN "THE ANSWER IS", A,B

WATCH YOUR PUNCTUATION

If you want to print a string constant and forget to enclose it in quotation marks, the computer interprets your string as an arithmetic expression, and will print whatever value it has. For example, if you meant to type PRIN "HELLO", but you typed in PRIN HELLO, the computer understands PRIN H and prints out the value of H. If you haven't assigned a value to H, it will print out 0.00.

In this example, the color coding would be:

PRIN, black on pink background (for a BASIC Keyword)

H, black on yellow background (for a numeric variable)

ELLO, black on green (for information not understood)

If you want to print an equation and mistakenly enclose it in quotations, it will be treated as a string constant. For example, if you wanted the computer to add $2+2$, but you mistakenly typed in PRIN "2+2", the computer prints 2+2.

PRIN & THE RETURN KEY

In Immediate Mode, the constant or equation following PRIN is immediately printed on the screen once the RTN Key is pressed.

Example:

PRIN "WHAT IS YOUR NAME"

PRIN 30 - 11

In Programmed Mode, the constant or equation following PRIN is stored information until the program is run.

Example:

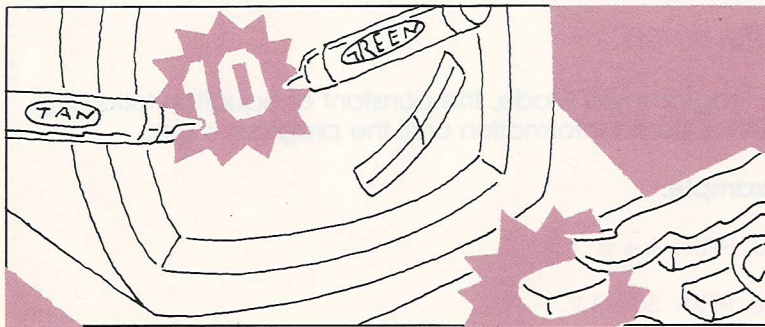
10 PRIN A + B

20 PRIN 30 - 11

COLOR CODING

Color coding is helpful in both Immediate and Programmed Modes. Each element in a line is designed to turn a specific color once you press RTN. For example, a line number turns to tan characters on a green background. To see which colors all the elements should turn, turn to page 85 in the Owner's Guide.

Color Coding lets you see whether the computer has understood what you typed in or not. After you press **RTN**, look to see if every character in the line has turned color. If it did, this means that the computer understood what you typed. For example, type in PRIN 1 + 1. Press **RTN** and everything turned color.



If you press RTN and only some of the characters are colored, the computer didn't understand some things, but made sense out of your line and proceeds. For example, type in PRIN 1 + + 1. Press **RTN** and everything turned color except the second + symbol. It's still black.



Important: Color coding is not fool proof! If you type a statement or command incorrectly, the computer may recognize enough to execute it in a different way than you intended. For example, if you type PRIN A\$, the computer will color PRIN A. It will then look for a numeric value of A. If it does not find one, it will assume a value of zero and print 0.00. It will not print the string value of A\$, since you must use the PUT keyword to print a string variable.

When the computer can't make any sense out of your line, the characters remain black or some may turn gray. For example, type in PRI 1 + 1. Press **RTN** and the P turned gray. Everything else is still black. The computer doesn't understand PRI. It tried to figure it out and notes this by turning P gray. You must correct this line. It is not correct until the characters turn color.

When you LIST or RUN a program, any lines that the computer cannot make sense of, and are non-executable, turn to white characters instead of black. Retype these lines or delete them using DEL.

PUNCTUATION

Intellivision BASIC uses commas, quotation marks and parentheses.

Commas are used to separate elements in a command or statement.

For example:

```
PRIN 2 + 4,3 + 6
```

Quotation marks enclose string constants.

For example:

```
PRIN "BASIC IS A COMPUTER LANGUAGE"
```

Parentheses enclose the argument of a function or the condition of an IF statement.

For example:


```
10 BK(20) = 3
10 IF (A = 20) GOTO 30
```

Turn to page 77 in the Owner's Guide for more information about punctuation.


MEMORY

In general, if you use the same constant more than three times, it will save memory to assign it to a variable. Turn to page 84 in the Owner's Guide for more information on memory.

*If you use the **SAME** **CONSTANT** more than **3 times**, It will save **MEMORY** to assign it to a **VARIABLE**.*



CONSTANT more than
MEMORY to
VARIABLE.



It will save

If you use the SAME

In general, if you use the same constant more than three times, it will save memory to assign it to a variable. Turn to page 64 in the Owner's Guide for more information on memory.

Quotation Marks

In addition, BASIC uses commas, quotation marks and parentheses.

Commas are used to separate elements in a command or statement.

For example:
`PRINT 2+4, 3+6`

Quotation marks enclose string constants.

For example:
`PRINT "BASIC IS A COMPUTER LANGUAGE."`

Parentheses enclose the argument of a function or the condition of an IF statement.

For example:
`IF (A=2) GOTO 30
 IN (A=2) GOTO 30`

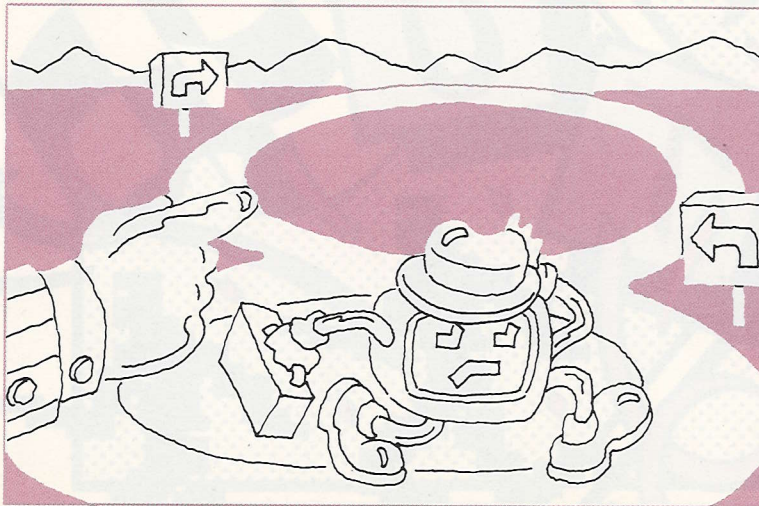
Turn to page 77 in the Owner's Guide for more information about punctuation.

CHAPTER 5
Looping



So far, programs have been pretty straight forward. The computer marches straight ahead, reading your statements one at a time. Once it has completed one line, it automatically goes to the next, unless told to go somewhere else.

Now you can break away from that straight ahead pattern...flip forward or backward in a program. When you send the computer somewhere to repeat a group of statements, you have created a LOOP. Let's take a look at GOTO and learn about loops.



GOTO

GOTO does exactly what it says. It tells the computer to go to some other place. The thing to remember is that it doesn't tell the computer to come back. Since the computer needs to know the line to go to, the GOTO command is followed by a line number.

Example:

GOTO 30

Let's try a GOTO statement. This example is the same as the one you learned on page 33. Now by using GOTO, the program can continue indefinitely.

NEW

10 PRIN "WHAT IS YOUR NAME?"

20 GET A\$

30 PRIN "HELLO"

40 PUT A\$

50 GOTO 10

RUN

- 10 When you run this program, the computer prints the string constant `WHAT IS YOUR NAME`
- 20 It waits for you to type in your name. Then press `RTN`. Your name is stored in the string variable `A$`.
- 30 Next it prints `HELLO`
- 40 It follows `HELLO` by printing the value of `A$` (your name)
- 50 Then the computer goes back to the beginning...over and over again.

This is called a PERPETUAL LOOP. To get out of a perpetual loop, press the `ESC` (Escape) Key.

Here's another example of GOTO:

NEW

- 10 `INPU "ENTER BALANCE", B`
- 20 `I = B * .1`
- 30 `PRIN "YOUR INTEREST IS", I`
- 40 `GOTO 10`
- RUN**

Here's how the computer will run this program:

- 10 The computer prints `ENTER BALANCE` and waits for you to enter a number. This number is stored under the numeric variable, `B`. For example, type in 30 and press `RTN`.
- 20 The computer calculates your interest (the balance times 10%) and stores this in the numeric variable `I`.
- 30 The computer prints `YOUR INTEREST IS` followed by the value of `I`.
- 40 The program begins again. Enter another number for `B` and the interest figure will change.



Important: GOTO is not always used in a perpetual loop. You can use GOTO to send the computer to another part of a program. This is called Branching. You will learn about the different types of branching starting on page 47.

FOR...NEXT

The FOR...NEXT statement allows you to repeat something a certain number of times. In other words, it allows the computer to count. Once the computer makes the loop the specified number of times, it then goes on with the rest of the program.

The FOR and NEXT statements work together. The FOR statement begins a loop and sets a starting value for the numeric variable. We will use the numeric variable N. It also sets the maximum value N can have. The starting and ending values can be numeric constants or variables.

Once the loop is begun, the FOR statement checks to see if the value of N is greater than the maximum value. If the value of N is less than the maximum value, it continues the loop. When the value of N is more than the maximum value, it goes on to the statement in the program following the NEXT statement.

The NEXT statement does two things. It adds 1 to the current value of N and sends the computer back to the FOR statement.

The statements that fall within the For...Next will repeat each time the loop is executed.

In its simplest form, the FOR...NEXT is used only to count. There are no statements in between FOR and NEXT. This is called a TIME DELAY LOOP.

For example:

```
10 FOR N = 2 TO 5
```

```
20 NEXT N
```

- 10 *In the first loop, FOR sets the starting value of N at 2 and the ending value at 5. FOR looks to see if N (now 2) is greater than 5. It is less, so the loop continues.*
- 20 *The NEXT statement adds 1 to N (now N=3) and sends it back to FOR in line 10.*
- 10 *FOR sees if N is greater than the end value. It is not, so the loop continues.*

This loop will repeat 4 times. It will count 2,3,4,5. Once the NEXT statement makes N equal 6 the FOR statement sees that this is greater than 5 and the loop ends. It uses the values of N to count. This is useful in a program where you want to create a delay before continuing with the program.

In our next example, the FOR...NEXT statement is used to count and print out the numbers that it's counting too.

For example:

NEW

10 FOR N = 1 TO 5

20 PRIN N

30 NEXT N

40 END

- 10** FOR gives N the values of 1,2,3,4,5. It will carry out 5 loops. It looks to see if the current value of N is greater than 5. If it is less, it goes to the next line.
- 20** The computer prints the current value of N. During the first loop it will print 1.
- 30** NEXT adds 1 to the value of N (now N = 2) and sends it back to the FOR statement.
- 10** The computer looks to see if the current value of N is greater than 5. If it is less, it goes to the next line.
- 20** The computer prints 2.
- 30** NEXT adds 1 to N (Now N = 3). Since it is less than 5, the loop continues.

The computer will print 1,2,3,4,5 in a column on the screen. Once the value of N reaches 6, the FOR...NEXT loop ends and the computer goes to the line following NEXT.

In this next example, the computer counts the number of times other statements are made.

For example:

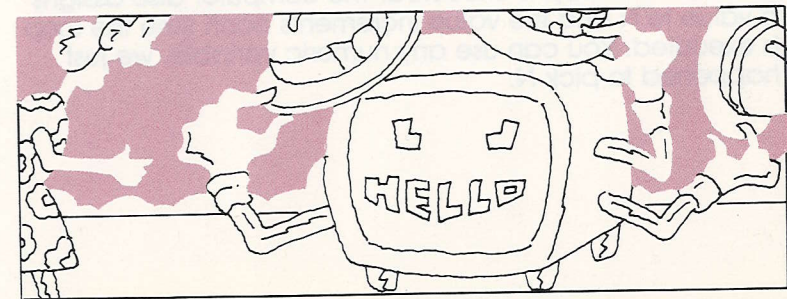
NEW

10 FOR N = 1 TO 5

20 PRIN "HELLO"

30 NEXT N

Here the FOR statement is used to count the number of times the computer will print out HELLO.



In this example, the computer counts the number of times you assign a value to the numeric variable A.

For example:

```
10 FOR N = 1 TO 5
```

```
20 INPU A
```

```
30 PRIN A
```

```
40 NEXT N
```

This program lets you input a number then it prints it. This will repeat five times.



Remember: In a FOR...NEXT statement the computer always keeps track of the number of times the loop is executed. The computer also assigns a value to N and the value increments each time the loop is executed. You can use any numeric variable, we just happened to pick N.

Here we'll take the simple program we used in GOTO on page 41 and change it...so that it becomes a more useful program using the FOR...NEXT statement. Here the FOR...NEXT is used only to count. Now you can enter in your original balance, the computer computes the interest, then it adds the interest to give you a new balance. It will do this 12 times. This is interest for a year, compounded monthly at 10% interest. Type in:

```
NEW
```

```
10 INPU "ENTER BALANCE", B
```

```
20 FOR N = 1 TO 12
```

```
30 I = B * .1
```

```
40 B = B + I
```

```
50 PRIN "YOUR INTEREST IS", I
```

```
60 PRIN "YOUR BALANCE IS", B
```

```
70 NEXT N
```

```
RUN
```

- 10 The computer prints out ENTER BALANCE and waits for you to input a number for the variable B.
- 20 In this program the For statement is used as a counter...it will repeat the loop 12 times.
- 30 You tell the computer that I (interest) equals the balance times 10%.
- 40 You tell the computer that B (balance) equals the balance plus the interest. You will continually come up with a new value for B because the balance changes each loop.
- 50 The computer prints out YOUR INTEREST and calculates the interest for the current value of B and prints it out.
- 60 The computer prints YOUR BALANCE and calculates the balance by adding the interest to the current value of B.
- 70 The computer takes the latest value of B and goes to the next FOR statement and begins the loop again.

NESTED FOR...NEXT

When there is a FOR...NEXT statement within another FOR...NEXT, you have a nested FOR...NEXT statement.



Here is an example of a nested FOR...NEXT.

```

10 FOR A = 1 TO 5
20 PRIN "HELLO"
30 FOR B = 1 TO 3
40 PRIN "BASIC"
50 NEXT B
60 NEXT A
RUN

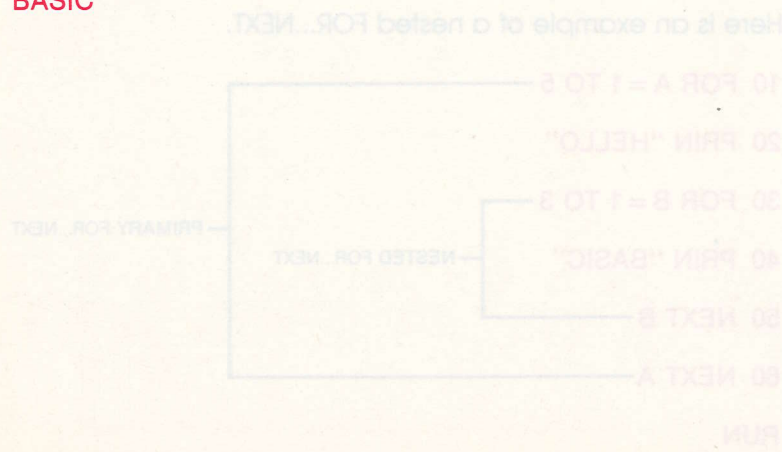
```

Diagram illustrating the flow of execution for the nested FOR...NEXT loops:

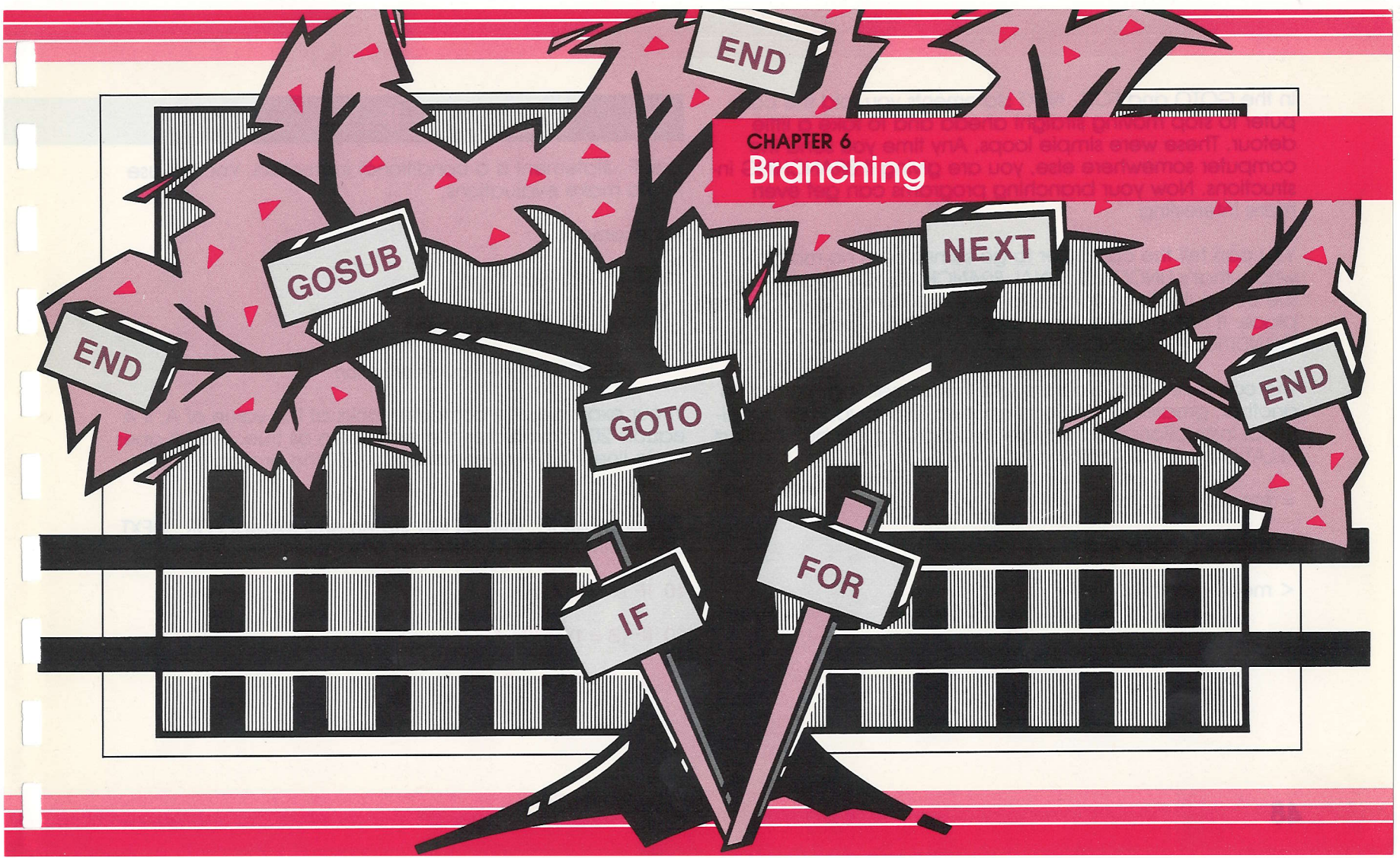
- The outer loop (FOR A = 1 TO 5) is labeled as the **PRIMARY FOR...NEXT**.
- The inner loop (FOR B = 1 TO 3) is labeled as the **NESTED FOR...NEXT**.
- The flow starts at line 10, goes to line 20, then to line 30, and loops back to line 40 until line 50 is reached.
- From line 50, the flow goes to line 60, then loops back to line 10.

The first FOR statement sets up a loop to repeat 5 times. It will print HELLO. Before it gets to its companion NEXT statement, the computer encounters a nested FOR...NEXT on lines 30-50. This will print BASIC three times. Once this is completed, the computer goes to line 60 and reads the NEXT statement for the original FOR...NEXT. The loop continues. You will see the following printed 5 times:

HELLO
 BASIC
 BASIC
 BASIC



CHAPTER 6
Branching



In the GOTO and FOR...NEXT statements you told the computer to stop moving straight ahead and to take a little detour. These were simple loops. Any time you send the computer somewhere else, you are giving it BRANCHING instructions. Now your branching programs can get even more interesting.

You can tell the computer to go somewhere no matter what...that's UNCONDITIONAL BRANCHING. OR you can tell the computer to check the facts, if one thing is true, go one place. If not, go to another place. This is called CONDITIONAL BRANCHING.

The computer can evaluate whether something is equal to another, greater or less than another. When you use conditional instructions, you will use the following symbols, called RELATIONAL OPERATORS:

= means equal to

> means greater than

< means less than

IF...

The IF...statement is a conditional instruction. You will use this to make evaluations.

For example:

```
10 IF (A = 20) GOTO 70
```

(A = 20) is an expression

GOTO 70 is the course of action

In our example, the computer looks at the value of A. If A equals 20, it evaluates the expression as true. Then it goes on to line 70. If A does not equal 20, the program continues.

You can use IF...with all key statements except FOR...NEXT, DATA, DIM and REM. Here are other examples of IF...

```
10 IF (A = 0) END
```

```
10 IF (B = 1) PRIN A
```

```
10 IF (A = 16) A = 0
```



Note: You can only use one IF...statement per line.

Let's try the IF...statement in a sample program:

NEW

10 PRIN "WHAT IS 5 + 6?"

20 INPU A

30 IF (A = 11) GOTO 60

40 PRIN "TRY AGAIN"

50 GOTO 10

60 PRIN "THAT'S RIGHT"

70 END

Let's go through this program one step at a time:

10 *First the computer prints out WHAT IS 5 + 6?*

20 *The cursor is on the next line waiting for you to type in a number for the numeric variable A (Here A = 5 + 6)*

- 30 *If your answer is correct (A = 11), the computer skips to line 60.*
- 40 *If your answer is more than or less than 11, the computer automatically goes to line 40 and prints out TRY AGAIN.*
- 50 *When your answer is incorrect, the program goes to line 10 so you can do it over again.*
- 60 *Once you give the correct answer, the computer prints out THAT'S RIGHT!*
- 70 *Then the program ends.*



Remember: The computer reads your program line for line. Once it has completed one line it automatically goes to the next, unless told to go somewhere else.

Let's take the last example program a step further. There are now two arithmetic operations in this program.

NEW

10 PRIN "WHAT IS 5 + 6?"

20 INPU A

30 IF (A = 11) GOTO 60

40 PRIN "TRY AGAIN"

50 GOTO 10

60 PRIN "THAT'S RIGHT!"

70 PRIN "WHAT IS 20 + 11?"

80 INPU B

90 IF (B=31) GOTO 120

100 PRIN "TRY AGAIN"

110 GOTO 70

120 PRIN "THAT'S RIGHT!"

130 END

Here's what's happening:

- 10 The computer prints out WHAT IS 5 + 6?
- 20 Again, the computer waits for you to enter a number to store in the numeric variable A.
- 30 If your answer is correct, the computer goes to line 60 and continues.

40 If your answer is not correct, the computer automatically goes to this next line and prints TRY AGAIN.

50 The computer goes back to line 10 for you to try again...

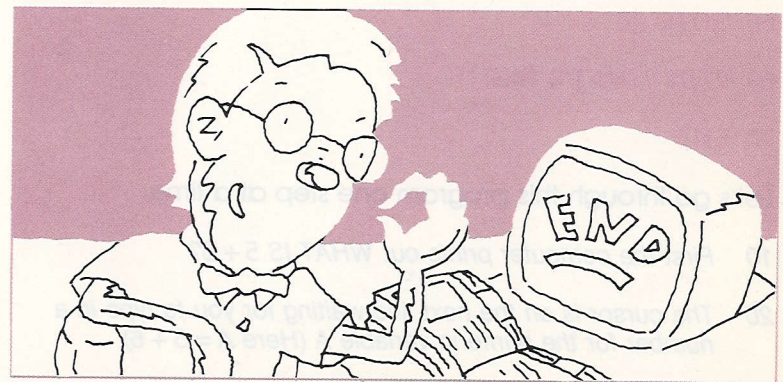
60 The computer prints THAT'S RIGHT.

70 You see the next arithmetic problem on the screen: WHAT IS 20 + 11?

80 Now you input your answer and it's stored in the variable B.

90-120 Continues as 30-60 above.

130 The program ends once you have answered both problems correctly.



Here's another sample program that is a sevens multiplication quiz. The FOR...NEXT here counts and uses the value of N as the multiplier.

NEW

10 PRIN "SEVENS TIMES TABLES"

20 FOR N = 1 TO 12

30 PRIN "WHAT IS 7 TIMES"

40 PRIN N

50 INPU Y

60 Z = N * 7

70 IF (Y = Z) GOTO 100

80 PRIN "TRY AGAIN"

90 GOTO 30

100 PRIN "RIGHT"

110 NEXT N

Let's go through this step-by-step:

- 10 The computer types out the title of the program.
- 20 The computer knows this loop will repeat 12 times.
- 30 The computer asks the question, WHAT IS 7 TIMES
- 40 The first value of N is printed on the next line...1
- 50 The computer stops at Y and waits for you to type in the answer to $1 * 7$. (Y now is the numeric variable that stands for $1 * 7$. Here Y should equal 7.)
- 60 The answer will always equal Z.
- 70 If you are correct you GOTO 100. If you are incorrect you automatically go to 80.
- 80 The computer prints TRY AGAIN.
- 90 GOTO 30 and do this one again.
- 100 The computer tells you the answer you typed in was correct.
- 110 Now the computer gives N the next value, this time it's 2, and goes back to line 20.

40 The current value for N is printed out...2.

50 Now you must input a number for Y (Here Y is the numeric variable that stands for 2*7. Here Y should equal 14.)

...and so on until you have completed 12*7 correctly.

GOSUB

Once you start to write rather lengthy programs, you need all the short cuts you can find. Sometimes you will use the same instructions more than once in a program. Instead of typing them in over and over, you can type them in once and tell the computer to read these instructions whenever you want. This is called a SUBROUTINE. Once the computer has completed a subroutine, it goes back to the program where it left off.

To get to a subroutine, you will use the Gosub (GSUB) statement, followed by the line number of the first line of the subroutine. The last line in a subroutine must always be RET (Return). This sends the computer back to the line following the GOSUB statement from which it just left.

For example, let's build onto the program on page 51 and create a program that has two multiplication quizzes. This program first goes through the sevens tables, then through the nines. Both of these quizzes will use the subroutine from lines 130 through 180. But you only have to write this subroutine once in the program.

NEW

10 PRIN "TIMES TABLE"

20 T=7

30 FOR N=1 TO 12

40 PRIN "WHAT IS 7 TIMES"

50 GSUB 130

60 NEXT N

70 T=9

80 FOR N=1 TO 12

90 PRIN "WHAT IS 9 TIMES"

100 GSUB 130

110 NEXT N

```

120 END
125 REM STARTING THE SUBROUTINE
130 PRIN N
140 INPU Y
150 Z = N * T
160 IF (Y = Z) GOTO 190
170 PRIN "TRY AGAIN"
180 GOTO 130
190 PRIN "RIGHT"
200 RET

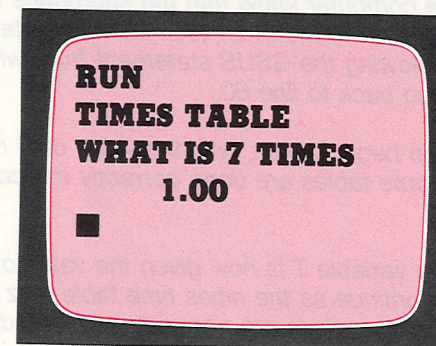
```

Let's go through this program step-by-step:

- 10 The computer prints the title *TIMES TABLE*.
- 20 This gives the computer information. $T = 7$ (T is the multiplier).
- 30 (Here's the beginning of the first *For...Next* statement.) N will be used as a counter. But look down to line 130. N will also use its values, 1 — 12 as the multiplicands.

40 The computer prints out *WHAT IS 7 TIMES*.

50 The computer is sent to line 130.



125 It's always a good practice to begin a subroutine with a *REMARK* statement. For details on *REM*, turn to page 58.

130 The subroutine begins here. Here the computer prints the first value of N1

140 You type in a number and it's stored in the variable Y .

150 The computer checks to see if Z (the answer) equals the value of $N * 7$.

- 160 If the number you typed in for Y equals Z, then go to line 190.
- 200 The last statement in a subroutine is always RET for Return. This lets the computer know that the subroutine has ended and sends it back to the main routine...to the statement immediately following the GSUB statement from which it left. Here it will go back to line 60.
- 60 Now the loop begins again, with the value of N now 2. When the sevens time tables are done correctly the computer goes to line 70.
- 70 The numeric variable T is now given the value of 9. The steps now continue as the nines time table quiz gets into action. When all the nines are correctly completed, the program ends.

NESTED SUBROUTINE

You have just learned that a subroutine branches off from the main routine. You can also branch off from other subroutines. This is called NESTING. Here's an example:

```

10 PRIN "STARTING"
20 PRIN "GOSUB 100"
30 GSUB 100
40 PRIN "ALL DONE"
50 END
100 PRIN "AT 100"
110 PRIN "GOSUB 200"
120 GSUB 200
130 PRIN "RETURN AT 140"
140 RET
200 PRIN "AT 200"
210 PRIN "RETURN AT 220"

```

NESTED SUBROUTINE

220 RET

Follow along and see how the computer will read this program:

- 10 It begins by printing out STARTING
- 20 The computer prints out GOSUB 100
- 30 The computer goes to the subroutine on line 100.
- 100 The computer prints out AT 100.
- 110 The computer prints out GOSUB 200.
- 120 This tells the computer to go to the nested subroutine on line 200.
- 200 The computer prints out AT 200
- 210 The computer prints out RETURN TO 220
- 220 The RET command sends the computer back to the next statement following the GSUB statement it last came from...in this case it goes to line 130.
- 130 The computer prints out RETURN AT 140
- 140 This sends the computer to the statement following the first GSUB statement...in this case it goes to line 40.

40 The computer prints ALL DONE

50 The program ends.

Here's another example of a nested subroutine.

- The major program covers lines 10-50
- The subroutine covers lines 100-130
- The nested subroutine covers lines 200 and 210

Follow line for line and see how this works.

10 FOR A = 2 TO 4

20 REM COMPUTE A*2+3 AND PRINT

30 GSUB 100

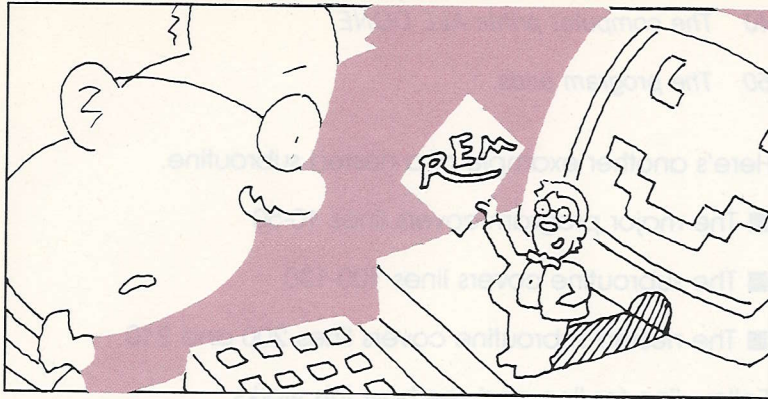
40 NEXT A

50 END

100 REM SET C TO A*2+3

110 GSUB 200

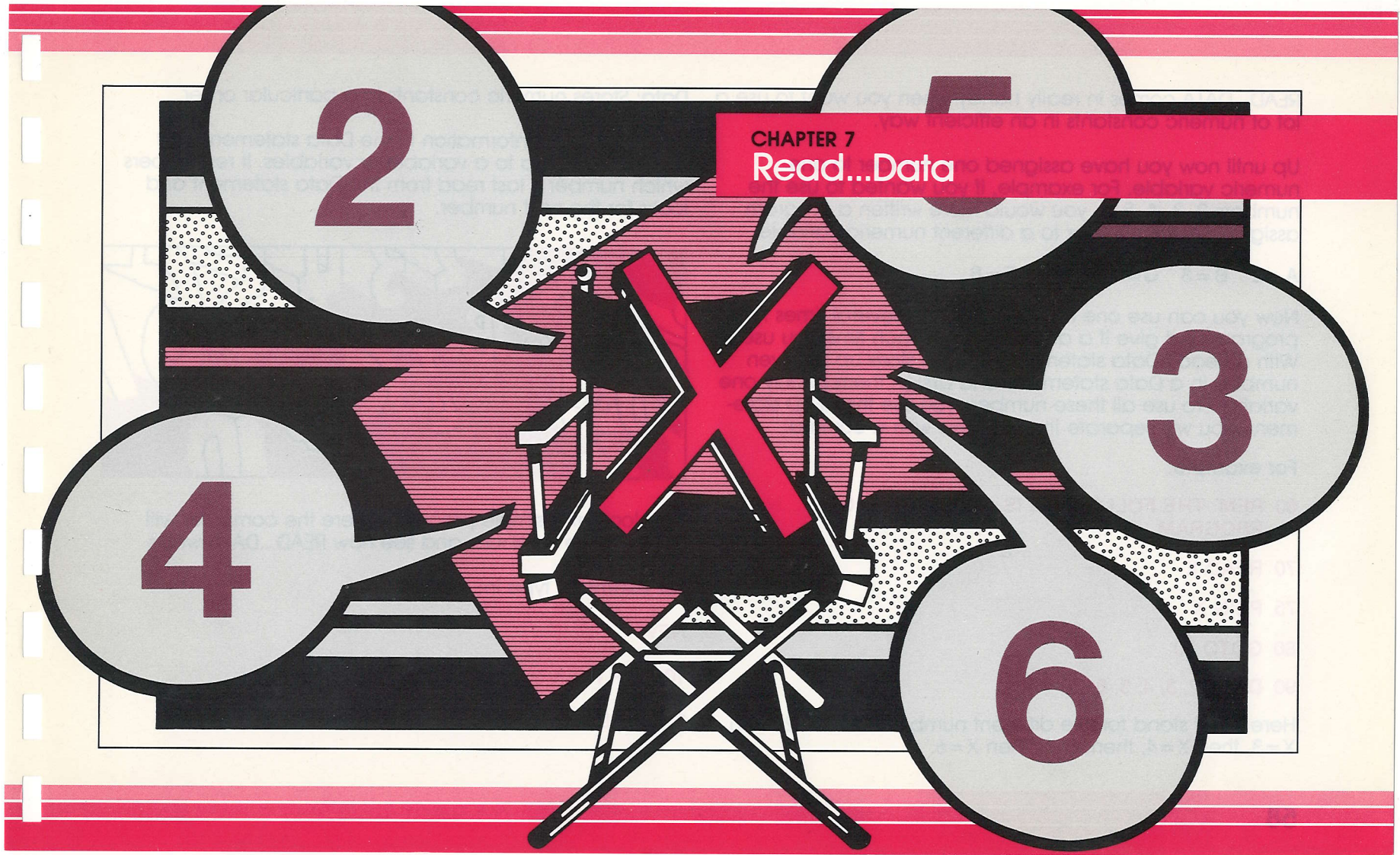
120 PRIN A,C



```
130 RET
200 C = A*2 + 3
210 RET
```

*For more information about REM (Remark), turn to page 55 in the Owner's Guide.

- 10 This sets up a loop that will repeat 3 times and gives the values 2, 3 and 4 to A.
- 20 REM stands for Remark*. This is used as a note to yourself. Here it tells you what GSUB 100 is going to do. A Remark statement prints out when you LIST, it does not print when you run a program.
- 30 The computer goes to the subroutine on line 100.
- 100 Remark reminds you that this subroutine will compute $A*2 + 3$.
- 110 This tells the computer to go to the nested subroutine on line 200.
- 200 The computer makes C equal to $A*2 + 3$.
- 210 The computer returns to line 120.
- 120 The computer prints the values for A and C. (In the first loop you will see 2 and 7.)
- 130 This sends the computer to line 40.
- 40 The loop increments and continues.
- 50 Once the 3 loops are completed the program ends.



CHAPTER 7
Read...Data

READ...DATA comes in really handy when you want to use a lot of numeric constants in an efficient way.

Up until now you have assigned one number to one numeric variable. For example, if you wanted to use the numbers 2, 3, 4, 5, 6, you would have written a program assigning each number to a different numeric variable.

A=2 B=3 C=4 D=5 E=6

Now you can use one numeric variable several times in a program and give it a different value each time you use it. With a Read...Data statement you can store up to seven numbers in a Data statement, and assign them to only one variable. To use all these numbers you use the READ statement. You will separate the numbers with a comma.

For example:

```
60 REM THE FOLLOWING IS AN EXAMPLE ONLY...NOT A PROGRAM.
```

```
70 READ X
```

```
75 PRIN X
```

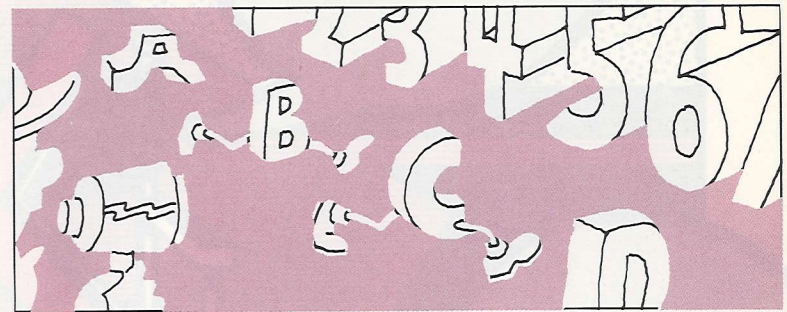
```
80 GOTO 70
```

```
90 DATA 2, 3, 4, 5, 6
```

Here X will stand for five different numbers. First X=2, then X=3, then X=4, then X=5, then X=6.

Data: Stores numeric constants in a particular order.

Read: Looks for information in the Data statement and assigns that data to a variable or variables. It remembers which number it last read from the Data statement and looks for the next number.



Let's look at a simple program where the computer will print out five numbers, and see how READ...DATA works.

For example; type in:

```
NEW
```

```
10 FOR N=1 TO 5
```

```
20 READ X
```

30 PRIN X

40 NEXT N

50 DATA 2,9,777,51,45

10 This sets up a loop that will repeat five times.

20 The numeric variable X is assigned to the numbers in the Data statement. When the computer sees READ, it looks for information in the Data statement.

30 Prin X tells the computer to print the current value of X. In the first loop, X = 2; in the second loop X = 9 and so on.

40 Next N tells the computer to begin the loop again...read and print the next value of X. Each time the Read statement is executed, the next number in the data statement is read and assigned to the numeric variable X.

50 The numbers are used in the exact order that they are listed in the DATA statement.

You can set up a program with more loops than numbers available in DATA. When this happens, the computer will use the numbers again. To see this, type in a change in the last program.

LIST

10 FOR N = 1 TO 10

RUN

Now you will see the numbers printed twice. When the fifth loop completes, the computer begins at the first number in DATA and begins printing the numbers out again.

See what happens when you have fewer loops than data. To see this, change the program:

LIST

10 FOR N = 1 TO 3

RUN

Now let's look at a program that will read 5 numbers and add them together before printing the total. Type in:

```
10 T=0
20 FOR M = 1 TO 5
30 READ X
40 T=T + X
50 NEXT M
60 PRIN T
70 DATA 1, 2, 3, 4, 5
```

- 10 This starts the total (T) out as zero. Each time you run this program it will start at zero.
- 20 This sets up a loop that will repeat five times.
- 30 The numbers in the Data statement are assigned to the numeric variable X.
- 40 This tells the computer to calculate the total by adding the value of X to the current total.
- 50 The computer goes back and reads the next value of X and calculates the total. It continues by reading each number

and adding it onto the last.

- 60 When the 5 loops are completed, the computer prints the total.
- 70 These are the numeric constants being used.

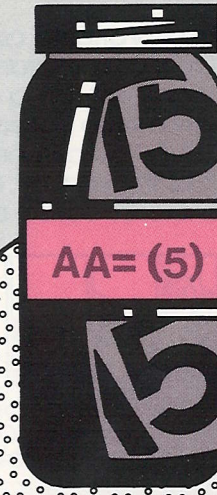
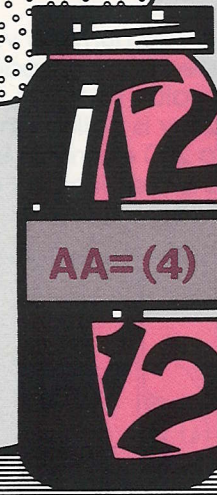
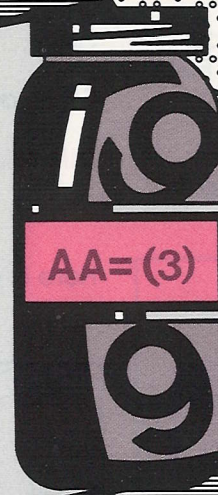
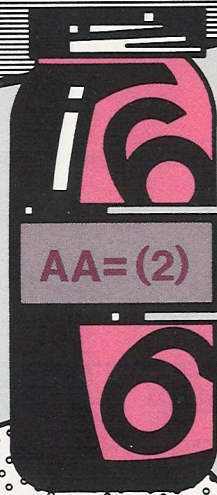
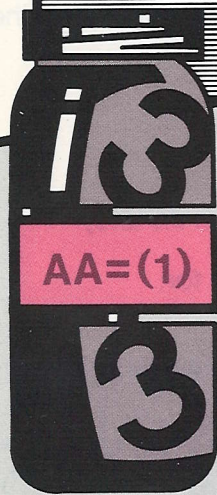
You can also use READ...DATA with multiple variables in READ.

For example:

```
NEW
10 DATA 3, 6, 9, 12
20 DATA 15, 18, 21, 24
30 FOR A = 1 TO 4
40 READ B, C
50 PRIN B, C
60 NEXT A
```

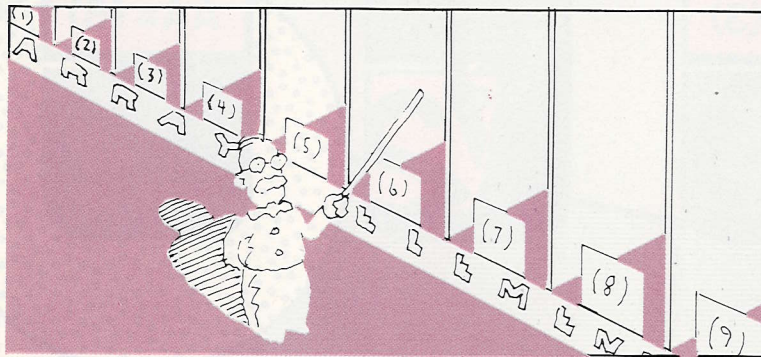
The variable B will get the numbers 3, 9, 15, 21 from the DATA statements. The variable C will get the numbers 6, 12, 18, 24.

CHAPTER 8
Arrays



You learned that Read...Data is useful when you want to store and use several numbers in a particular order. Now you can take this one step further and store a group of numeric values (constants and variables) and use them in any order you want...any time you want! To do this you will store these values in an ARRAY VARIABLE.

An Array Variable can store many numeric values. You can picture it as a rectangle divided into compartments. Each compartment has a numeric value stored in it. Each numeric value stored in an array is called an ARRAY ELEMENT. You can have up to 250 elements in an array. Each of these array elements is identified by a number in parentheses called a SUBSCRIPT.



You can use a total of three arrays in a program. They are identified by the names AA, AB, AC.

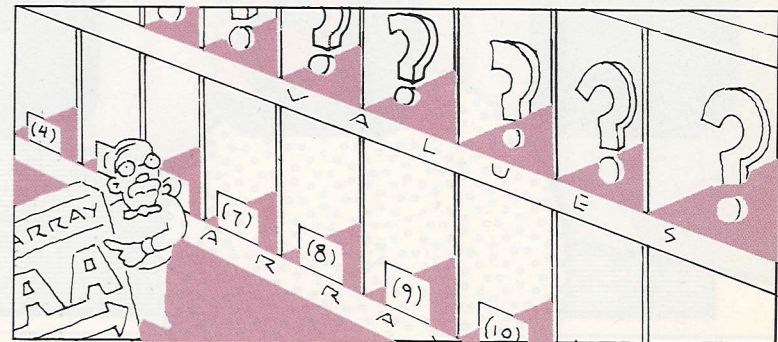
How to set up an array: In order to set up an array variable, you must first decide how many elements you need. If you want to use 5 different values, then you will need an array with 5 elements. To give the computer this information, you will use the DIM (Dimension) statement. The DIM statement gives the computer two pieces of information: the name of the array and the number of elements in the array.

For example:

DIM AA(5) AA is the name of the array

(5) is the number of elements in the array

Here is what the array would look like so far:



This array has five elements. These elements are identified by the subscripts (1), (2), (3), (4), (5). The name of each element is AA(1), AA(2), AA(3), AA(4), AA(5). The values have not yet been assigned to each element.

Assign numeric values to the elements: Now you need to assign a value to each element. Let's say the values we want to use are 3, 6, 9, 12, 15. You must assign one value to one element. Where you assign the values is completely arbitrary. Just be sure that you have the same number of elements as values.

For example:

```
10 DIM AA(5)
```

```
20 AA(1) = 3
```

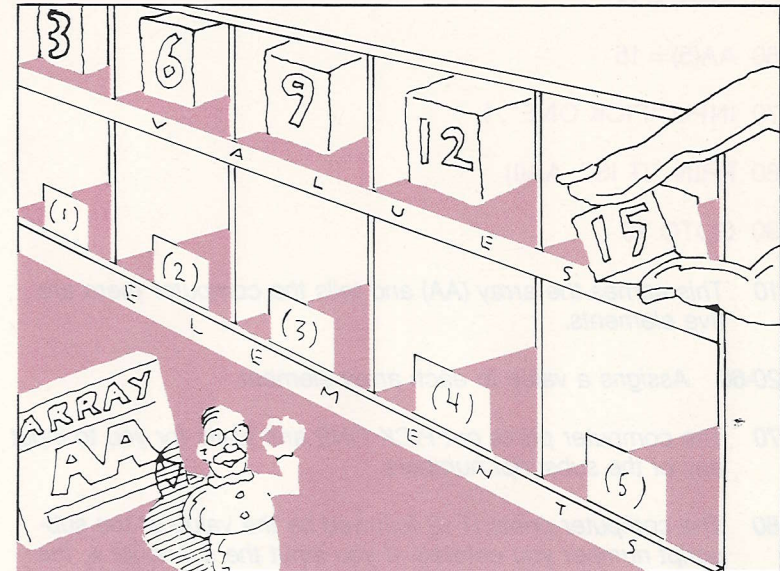
```
30 AA(2) = 6
```

```
40 AA(3) = 9
```

```
50 AA(4) = 12
```

```
60 AA(5) = 15
```

Here's how the array looks now that you have stored these values:



Let's go through this simple program and see how it works:

```
10 DIM AA(5)
```

```
20 AA(1) = 3
```

```
30 AA(2) = 6
```

```

40 AA(3) = 9
50 AA(4) = 12
60 AA(5) = 15
70 INPU "PICK ONE", I
80 PRIN "IT IS", AA(I)
90 GOTO 70

```

10 *This names the array (AA) and tells the computer there are five elements.*

20-60 *Assigns a value to each array element.*

70 *The computer prints out PICK ONE and waits for you to input any of the subscript numbers.*

80 *The computer prints IT IS followed by the value of the subscript number you entered. If you input the subscript 4, the computer prints out IT IS 12. (12 is the value of AA(4).)*

90 *The loop begins again.*

The computer has stored five values and you can access any one at any time. Try this again and type in another subscript number for I.

Let's go a step further and use READ...DATA to assign values to the elements in an array.

```

10 DIM AA(5)
20 DATA 10,20,30,40,50
30 FOR A = 1 TO 5
40 READ B
50 AA(A) = B
60 NEXT A
70 INPU "PICK ONE", I
80 PRIN "IT IS", AA(I)
90 GOTO 70

```

10 *The Dimension statement names the array (AA) and tells the computer there are five elements.*

20 *The values 10, 20, 30, 40 and 50 are stored in this Data statement.*

- 30 This sets the value of A equal to 1, 2, 3, 4, 5 in that order through five loops. These will be the subscript numbers.
- 40 B is the variable that will get the values in the Data statement.
- 50 This assigns a subscript number and a value to each array element. In the first loop, AA(A) = AA(1) and it equals 10...AA(1) = 10.
- 60 The computer goes to the next value of A and continues until all 5 array elements are given subscript numbers and values.

```
AA(1) = 10
AA(2) = 20
AA(3) = 30
AA(4) = 40
AA(5) = 50
```

- 70 The computer prints out PICK ONE and waits for you to input one of the five subscript numbers.
- 80 The computer prints out IT IS, and prints the value of the array you selected. If you input 4 for I, the computer will print out 40.
- 90 This lets you input another subscript number and see it's value printed.

Now let's use INPU statements to assign subscript numbers and values for the elements.

```
10 DIM AA(5)
20 FOR A = 1 TO 5
30 INPU "NUMBER", N
40 AA(A) = N
50 NEXT A
60 INPU "PICK ONE", I
70 PRIN "IT WAS", AA(I)
80 GOTO 60
```

- 10 Again, this names the array and sets up five elements.
- 20 This sets up a loop that will repeat five times and assigns the subscript number to each element (AA(1), AA(2)...)
- 30 You see NUMBER on the screen and you type in a value for the first element.
- 40 AA(A) is equal to the value that you give N in a particular loop. If you input 10 in line 30, it is assigned to element AA(1). AA(1) = 10.

50 The loop continues, if you enter 11 for the next N, it is assigned to AA(2). AA(2)=11. This continues until you have input all five values.

60 Now the computer prints out PICK ONE and you must enter a subscript number for the variable I.

70 The computer prints IT WAS and prints the value of the array for the subscript you picked. If you had entered values of 10, 11, 12, 13, 14, 15, and you had input 1 for the first I, then on line 70 you would see IT WAS 10.

80 The loop takes you back to the input statement. Try another!

Let's try a program that will choose two of five array elements and add their two numeric values together.

10 DIM AA(5)

20 DATA 2,7,1,3,4

30 FOR A = 1 TO 5

40 READ B

50 AA(A) = B

60 NEXT A

70 INPU "PICK TWO", L,M

80 T= AA(L) + AA(M)

90 PRIN "THE TOTAL IS", T

100 GOTO 70

10-60 These lines set up the array with five elements and assign a value to each element.

AA(1) = 2

AA(2) = 7

AA(3) = 1

AA(4) = 3

AA(5) = 4

70 The computer prints out PICK TWO and waits for you to input two subscript numbers. The first number will be stored as the numeric variable L...the second as the numeric variable M.

80 The computer adds the two numbers. If you input the subscripts 1 and 3, 1 is stored in the variable L and 3 is stored in the variable M. The computer gets the values of AA(1) and AA(3) and adds the two values together. Here, the total would be 3.

90 The computer prints the total.

100 You can input two other subscript numbers.

SOME RULES ABOUT ARRAYS

■ You can use an array exactly as you would any other numeric variable. For example:

```
PRIN AA(3) * AA(5)
```

■ In Intellivision BASIC, you can only dimension numeric arrays.

■ You can use any element any time.

■ There are three arrays you can use in any program: AA, AB, AC.

■ Each array can hold a maximum of 250 values.

You can use string variables as arrays. Here is an example of manipulating and comparing string variables:

```
10 SET B$ = "Z"
```

```
20 SET A$ = "A"
```

```
30 PUT A$
```

```
40 A$(1) = A$(1) + 1
```

```
50 IF (A$(1) > B$(1)) END
```

```
60 GOTO 30
```

10 The first element of B\$ string is "Z".

20 The first element of A\$ string is "A".

30 This prints the current value of A\$ (now it's "A").

40 This increments the value of A\$ (now it's "B").

50 This checks to see if the value of A\$ has reached "Z" yet.

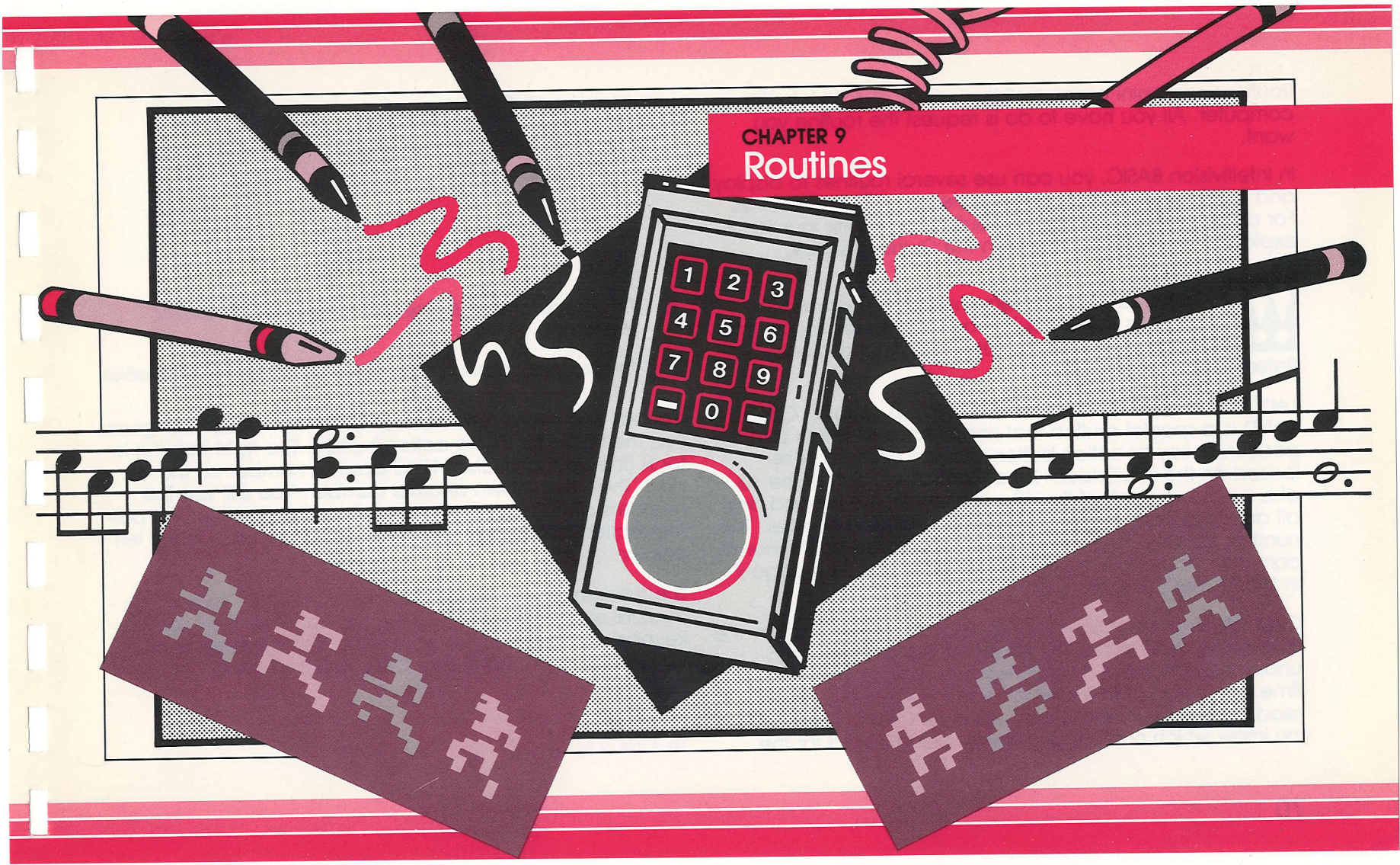
60 If the value is less than or equal to B\$(1), the program goes back to line 30.

30 This prints the current value of A\$ (now it's "B").

40 This increments the value of A\$ (now it's "C").

50 This program continues until A\$ reaches the value of "Z". Once A\$ is greater than "Z", the program ends.

CHAPTER 9
Routines



Routines are "mini programs" that are built right into the computer. All you have to do is request the routine you want.

In Intellivision BASIC, you can use several routines to display and manipulate objects and create and change sounds. For a complete look at all the routines and their individual explanations, turn to page 68 in the Owner's Guide.



Important: Before reading any further about routines in this book, learn how to use routines to display and manipulate objects and sounds.

Refer to the Owner's Guide.

Let's take a closer look at the CALL HAND routine. CALL HAND is a special routine that keeps track of which disc position or key or action button was last pressed or released on the Intellivision Hand Controller. Each of the 16 disc positions, each of the 12 numbers on the keypad and all action buttons are given its own identifying number. This number identifies the specific control pressed on the hand controller. These identifying numbers are listed on page 71 in the Owner's Guide.

CALL HAND stores values for the disc and keypad under the variable H...and stores the values for the action button under the variable A (H only reads one hand controller at a time. The value of R tells which hand controller is being read). When used with a print statement, CALL HAND let's you know which number is currently being stored for the

variables H and A.

Try this simple program and see how this works.

```
10 CALL HAND
```

```
20 PRIN A, H
```

```
30 GOTO 10
```

10 *The computer finds the current values for the variables H and A.*

20 *The computer prints out the current value for these variables.*

You will see two columns of numbers. The left column gives you the value for A (the action button). The right column is for H (the keys and disc). Hold down a number on the keypad. Then hold down another number. You will see the numbers change on the screen. Press the top then the bottom action button and see the numbers change in the left column.

The following program allows you to set up a "major" musical scale. You will store 12 musical notes and use the keypad numbers to play each note. This program uses CALL NOTE, CALL TONE and CALL HAND routines.

```
5 V = 5
```

```
10 DIM AA(12)
```


20 DATA 47, 30, 32, 34, 35, 37

30 DATA 39, 41, 42, 44, 46, 49

40 FOR B = 1 TO 12

50 READ M

60 AA(B) = M

70 NEXT B

80 CALL HAND

90 IF (H < 0) GOTO 80

100 IF (H > 99) GOTO 80

110 H = H + 1

120 N = AA(H)

130 CALL NOTE

140 CALL TONE

150 CALL HAND

160 IF (H < 99) GOTO 150

170 GOTO 80

5 *This sets the volume. (See page 10 in the Owner's Guide to learn more about volume.)*

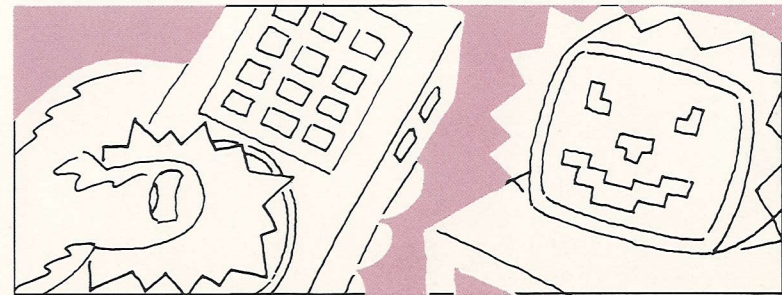
10 *This names an array with 12 elements.*

20 & 30 *These values in DATA represent the note number of each tone that will be played.*

40-70 *This gives each array element a subscript number and assigns each a value.*

80-100 *Once you run the program, hold down any numeric key on the hand controller. The computer is looking to see if a key is being pressed.*

110 *This takes the value of the key you pressed (0-11) and adds 1 to it. So we have a number from 1 to 12.*



- 120 N is the variable used with CALL NOTE. This is the note number for the tone you will generate.*
- 130 This sets the pitch (P) for the note (N).*
- 140 This generates the tone.*
- 150 & 160 This looks at the keypad again. You must release a key in order to stop the 150-160 loop.*
- 170 This sends the computer back to 80 so you can now press another key for another tone.*



FUNCTIONS

There are many functions you can use in a program. Generally, most are used to manipulate and control moving objects in Intellivision cartridges. Functions can be used just as variables.

Each of the functions has been given an identifying name. The name begins with two letters. This is the function label and is the general heading for the function. It is followed by a number in parentheses called the argument. The argument identifies what object or value the function relates to.

For example, let's look at the function that deals with the color of an object. CO(3) is the color of object 3.

CO is the label.
(3) is the argument.

The label for a function is always the same, but you can change the argument by putting in another value. For example CO(4) is now the color of the object 4.

In order to tell the computer to do exactly what you want, you must set the function equal to a numeric value. This can be a numeric constant or a numeric variable. This number tells the computer specifically what to do to that object (the argument).

Let's continue using the example of the color function. There are 16 possible colors available in the color function. They number from 0-15. (The colors and their corresponding numbers are shown on page 56.) If you want a specific object to be a specific color, here's how to do it.

$CO(3) = 6$

This will turn object 3 yellow.

To recap: The function tells you three things. The information on the left of the equal sign tells you what the function is (the label) and what will be affected (the argument). The information on the right of the equal sign tells what it will do to the argument.

Try an example by first displaying an object on your screen, using the CALL GRAB routine. See page 87 to find the CALL GRAB routine for the cartridge you want!

Next assign number 1 to your object and type in these commands. Watch the object change color. Here you are setting the function equal to a numeric constant. This constant represents a color. Each numeric constant affects the argument. It sets the color of the argument.

$CO(1) = 7$

$CO(1) = 3$

$CO(1) = 5$

Here you are setting the function equal to a numeric variable.



Important: Turn to page 58 in the Owner's Guide and learn about all the functions.

LET'S TAKE A CLOSER LOOK AT THE FORMAT FUNCTION

The Format Function allows you to display columns of numbers in different ways. The argument for the format function can be a number from 0 to 4. See page 65 in the Owner's Guide for the details on the argument numbers.

In the following examples, the numeric value affects the argument. It assigns how many numbers will be to the right of the decimal point.

$FM(0) = 0$

PRIN 1,2

You will see: 1 2

```
FM(0) = 1
```

```
PRIN 1,2
```

You will see: 1.0 2.0

```
FM(0) = 2
```

```
PRIN 1,2
```

You will see: 1.00 2.00

In this example, the numeric values affect the character color for the print statement by giving it a different color each loop.

```
10 FOR A = 0 TO 7
```

```
20 FM(4) = A
```

```
30 PRIN A
```

```
40 NEXT A
```

In the first loop the color will be black (0), the second loop will change the color to blue (1)...and so on.



NOTE: Color 5 is the same color green as the background color on the TV screen...therefore you don't see it!

Here's a program that uses the Format Function to set up a column for income and a column for expenses and totals each. This is a useful program for household budgeting.

```
5 FM(0) = 2
```

```
10 DIM AA(5)
```

```
20 DIM AB(5)
```

```
30 T = 0
```

```
40 S = 0
```

```
50 FOR A = 1 TO 5
```

```
60 INPU "INCOME",I
```

```
70 S = S + I
```

```
80 INPU "EXPENSE",E
```

```
90 T = T + E
```

100 AA(A) = I

110 AB(A) = E

120 NEXT A

130 FOR A = 1 TO 5

140 PRIN AA(A), AB(A)

150 NEXT A

160 PRIN "##TOTAL#####TOTAL"

170 PRIN "##INCOME####EXPENSE"

180 PRIN S,T

- 5 This sets the format to two numbers per line with two digits right of the decimal point.
- 10 This income array has five elements.
- 20 This expenses array has five elements.
- 30 T = total expenses. It begins at zero.
- 40 S = total income. It begins at zero.
- 50 This sets a loop that will repeat 5 times.

USE THE SPACE BAR
WHEREVER YOU SEE #.

60 Input an income number and it's stored in the variable I.

70 This tells the computer to keep a running total for S, by adding each I to S.

80 Now you input an expense number and it's stored in the variable E.

90 This tells the computer to keep a running total for T.

100 This creates an income array.

110 This creates an expense array.

120 The loop continues. You input 5 income numbers and 5 expense numbers and the computer assigns them to arrays.

130-150 A new loop is started which prints out 2 columns of numbers. The left column of 5 numbers is for income, the right column of numbers is for expenses.

160 You see TOTAL at the bottom of each column.

170 You see INCOME and EXPENSE.

180 The computer prints the totals for income and expenses.

To print this out (using a printer), type:

D = -1

CALL OUTP

LIST

To return to the screen, type:

D = 1

CALL OUTP

THE BACKGROUND FUNCTION

BK stands for Background Color. This sets the color on a certain portion of the TV screen. The screen is divided into a grid of 20x12 squares. Each square is called a card. Each card is numbered from 0 to 239.

0-19	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
20-39	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
40-59	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
60-79	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
80-99	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
100-119	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
120-139	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
140-159	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
160-179	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
180-199	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
200-219	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
220-239	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

The argument of the background function is the card number. The function can be set equal to any of the 16 color numbers (from 0 — 15).

0 = Black	8 = Gray
1 = Blue	9 = Cyan
2 = Red	10 = Orange
3 = Tan	11 = Brown
4 = Dark Green	12 = Pink
5 = Green	13 = Light Blue
6 = Yellow	14 = Yellow-Green
7 = White	15 = Purple

For example:

BK(33) = 10

Means card 33 will be orange.

BK(20) = 1

Means card 20 will be blue.

Here's how you can set a specific color to a specified card.

10 INPU A,B

20 BK(A) = B

30 GOTO 10

10 Tells you to input two numbers. The variable A represents a card number from 0-239. B represents a color number from 0-15.

20 This tells the computer to make the card you specify turn the color you specify.

To make card 233 red, input 233 and 2.

To make card 150 yellow, input 150 and 6. You type this over your last command.

Just for the fun of it, run the following program and you will see the screen fill with alternating black and blue cards.

10 X=0

20 FOR A=0 TO 239

30 BK(A)=X

40 X=X + 1

50 IF (X=2) X=0

60 NEXT A

70 END

- 10 This starts the color at black.
- 20 This sets a loop to change all cards on the screen.
- 30 This sets the color.
- 40 & 50 This switches the color back and forth from black to blue.
- 60 This sends the computer back to begin the next loop.
- 70 This ends the program.

List this program and change line 50.

50 IF (X = 3) X = 0

Now run it again and see what happens.

THE RANDOM NUMBER FUNCTION

The Random Number Function tells the computer to choose a number from 0 to 99. There is no limit on the amount of random numbers the computer will generate. The argument for this function is a "dummy". This means it can be any number. Try this to see how it works.

A = RN(0)

PRIN A

A number from 0-99 is printed on the screen.

The following program generates two random numbers and adds them together.

10 A = RN(0)

20 B = RN(0)

30 PRIN "ADD", A,B

40 C = A + B

50 INPU D

60 IF (C = D) GOTO 90

70 PRIN "WRONG", C

80 GOTO 10

90 PRIN "RIGHT"

100 GOTO 10

10 & 20 *This selects two random numbers.*

30 *This prints ADD followed by two numbers.*

40 *This adds the two numbers together.*

50 *Input a number here, the sum of the two random numbers.*

60 *If your answer is correct, the computer goes to line 90.*

70 *If your answer is incorrect, the computer prints WRONG followed by the correct number.*

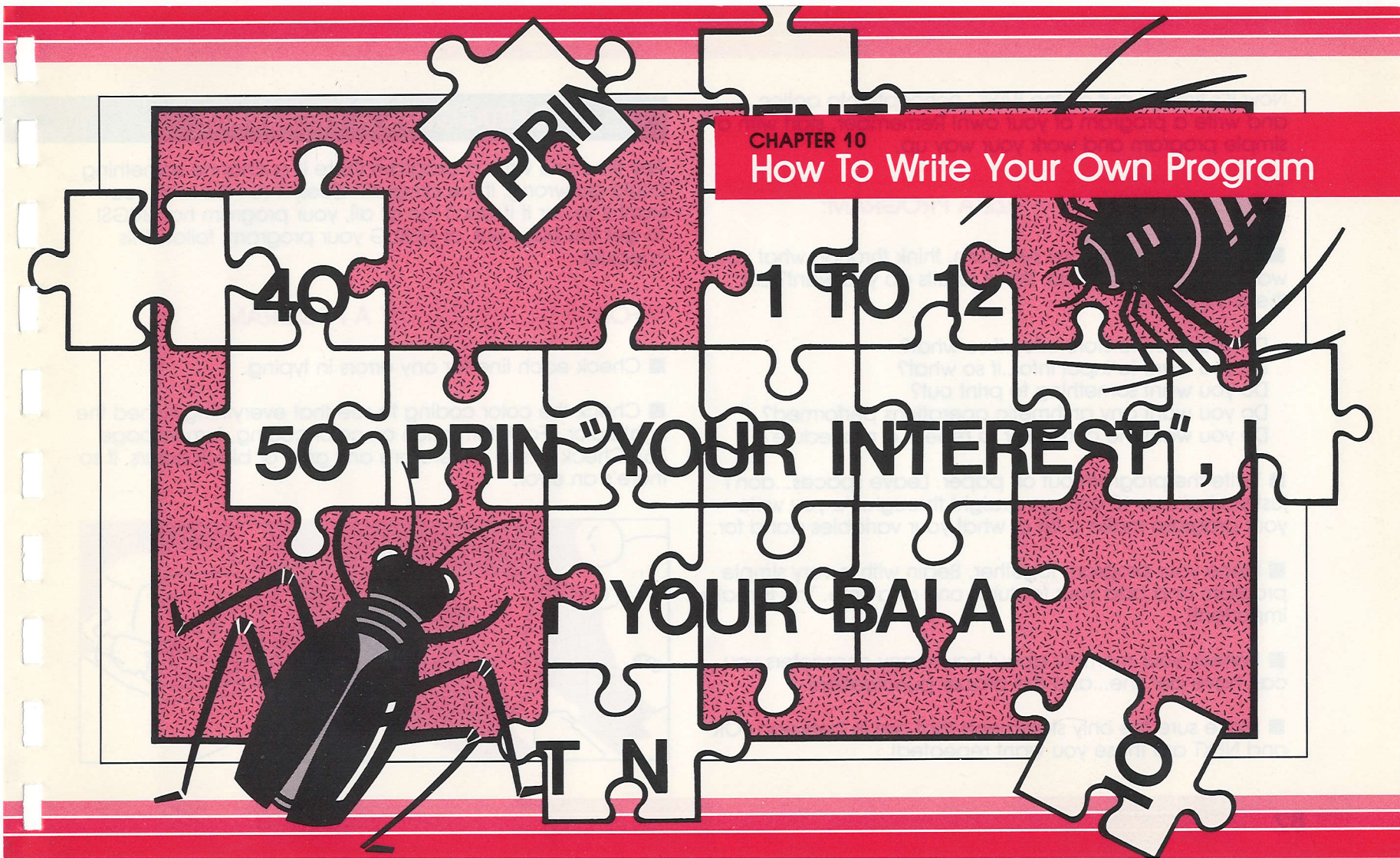
80 *The computer begins another addition problem.*

90 *If your answer at line 60 is correct, the computer prints RIGHT.*

100 *The computer begins again.*

CHAPTER 10

How To Write Your Own Program



Now it's time to put all the BASIC concepts into action... and write a program of your own! Remember, start with a simple program and work your way up.

HERE'S HOW TO ORGANIZE A PROGRAM:

■ Before you begin any program, think through what you want this program to do. What results do you want? List these things out:

- Do you want to store info...if so what?
- Do you want to input info...if so what?
- Do you want something to print out?
- Do you want any arithmetic operations performed?
- Do you want the computer to repeat a procedure?

■ Write the program out on paper. Leave spaces...don't just start at line 10 and go straight through. As you write your program make a list of what your variables stand for.

■ Piece your program together. Begin with a very simple program and add your features one at a time. This is really important!

■ Refresh your memory about how many characters you can have per line...and the proper punctuation!

■ Make sure the only statements that are in between FOR and NEXT are those you want repeated!

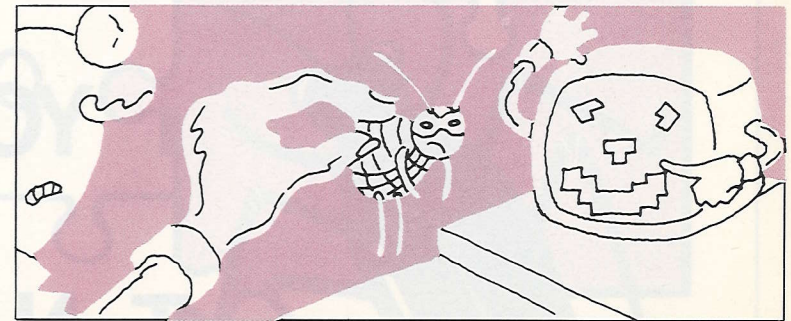
DEBUGGING A PROGRAM

Any time you write a program there is a chance something could go wrong. If the program doesn't run the way you want it to...or if it won't run at all, your program has BUGS! To get the bugs out, or DEBUG your program, follow this checklist:

BEFORE YOU RUN OR LIST A PROGRAM

■ Check each line for any errors in typing.

■ Check the color coding to see that everything turned the right color. (For information on color coding, turn to page 36). Check to see if there are any gray or black colors. If so there's an error.

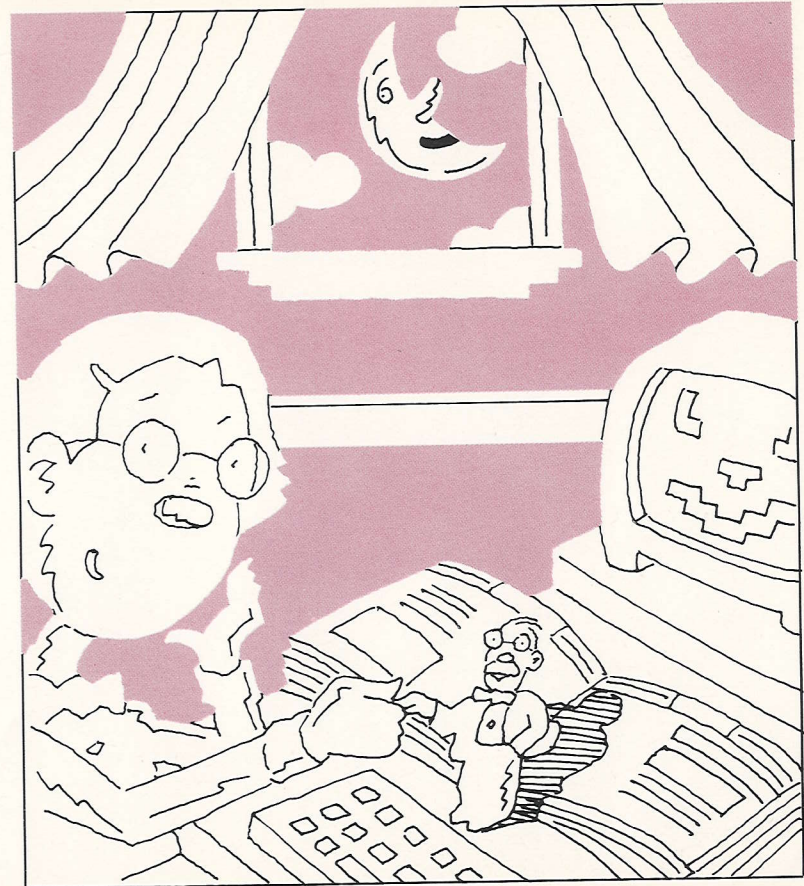


EVERYTHING TURNED COLOR BUT THE PROGRAM STILL DOESN'T RUN!

- List the program and check the values of what you are entering. Did you use your constants and variables correctly? Check IMPORTANT on page 32 & 33 to see that you have used the correct values for numeric and string variables.
- Check your Print statements. Did you assign your string variables correctly? Did you use string constants with quotes?
- Check the punctuation and use the arrow keys to make any corrections.

IF YOU JUST CAN'T FIND ANYTHING WRONG

The disconcerting moment may arise when you just can't find a blasted thing wrong...but the program still won't run. When this happens, it could be blamed on a logical error. The mistake could be in your thinking. Go through your program and make sure you told it to do what you think you did! Good luck!





EVERYTHING TURNED COLOR BUT THE PROGRAM STILL DOESN'T RUN!

■ List the program and check the values of what you are entering. Did you use your constants and variables correctly? Check IMPORTANT on page 32 & 33 to see that you have used the correct values for numbers and string variables.

■ Check your Print statements. Did you design your string variables correctly? Did you use string constants with quotes?

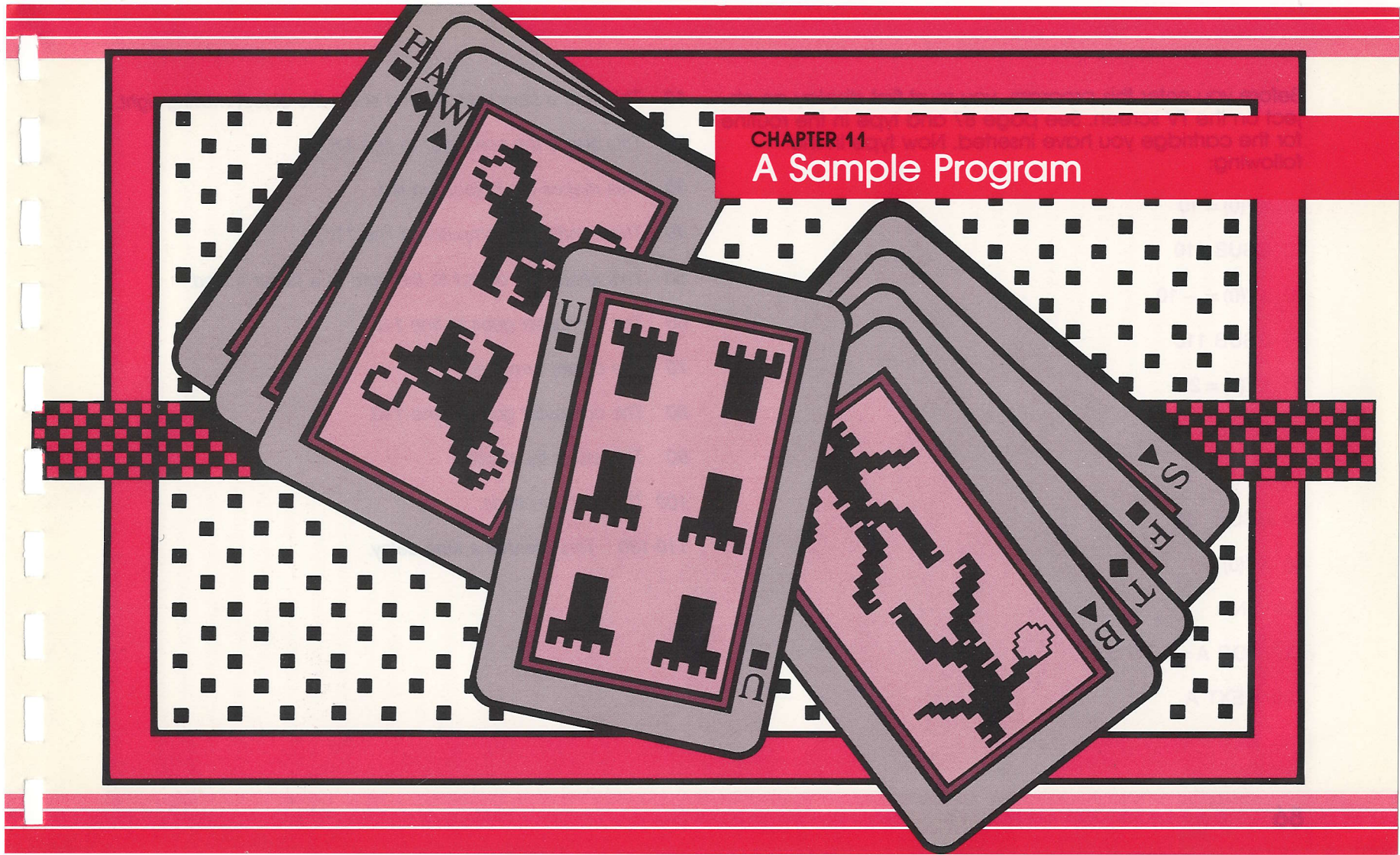
■ Check the punctuation and use the arrow keys to make any corrections.

IF YOU JUST CAN'T FIND ANYTHING WRONG

The disappointing moment may arise when you just can't find a blasted thing wrong...but the program still won't run! When this happens, it could be blamed on a logical error. The mistake could be in your thinking. Go through your program and make sure you told it to do what you think you did! Good luck!

CHAPTER 11

A Sample Program



Before you enter this program, you must first display an object on the TV screen. See page 87 and type in the routine for the cartridge you have inserted. Now type in the following:

10 XV(0) = 10

20 GSUB 110

30 XV(0) = - 10

40 GSUB 110

50 XV(0) = 20

60 GSUB 110

70 CO(0) = 2

80 GSUB 110

90 XV(0) = 0

100 END

110 FOR A = 1 TO 30

120 NEXT A

130 RET

10 This sets a slow velocity and starts the object moving right.

20 This sends the computer to line 110.

30 This makes the object run left.

40 This sends the computer to line 110.

50 This sends the object to the right at a faster speed.

60 The computer goes to line 110.

70 The object turns red.

80 The computer goes to line 110.

90 This stops the object.

100 The program ends.

110-130 This creates a time delay.

ROUTINES TO DISPLAY OBJECTS IN INTELLIVISION GAME CARTRIDGES

Every Intellivision cartridge has several moving objects. Once you insert a cartridge into the Computer Adaptor, you can display any of its objects on the TV screen. First find the routine for the cartridge you are using. Next find the object you want to display. Now type in the following routine and fill in the numbers for M, N AND D. Press **RTN** after each line.

M =
N =
O =
D =
CALL GRAB

For example, look at Advanced Dungeons and Dragons below. To display the bow, you could type in:

M = 0 Press **RTN**
N = 6 Press **RTN**
O = 1 Press **RTN**
D = 1 Press **RTN**
CALL GRAB Press **RTN**

If an object is made up of two halves, you will need to type in the following:

M =
N =
O = 1
D =
CALL GRAB

M =
N =
O = 2
D =
CALL GRAB

CALL LINK
XV(2) = 8

For example, to link the left and right of the Bear in Math Fun, you would type in:

M = 0
 N = 50
 O = 1
 D = 2
 CALL GRAB

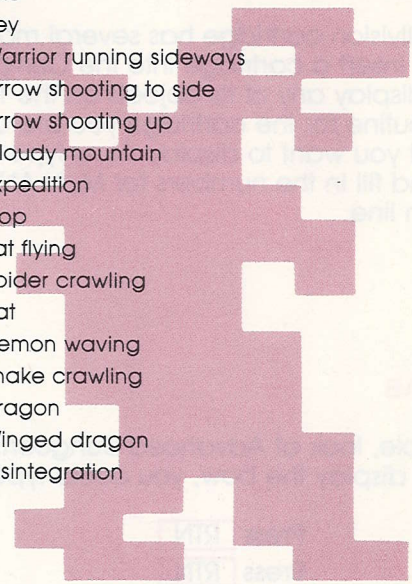
M = 0
 N = 52
 O = 2
 D = 2
 CALL GRAB

CALL LINK
 XV(2) = 8

**ADVANCED DUNGEONS & DRAGONS™ *
 TREASURE OF TARMIN™ * CARTRIDGE**

N	D	M	DESCRIPTION
0	1	0	Crown
1	1	0	Food trail
3	1	0	Exit staircase
4	1	0	Demon sign

5	1	0	Arrows container
6	1	0	Bow only, to shoot with
7	1	0	Boat
8	1	0	Axe
9	1	0	Key
22	2	3	Warrior running sideways
30	1	0	Arrow shooting to side
32	1	0	Arrow shooting up
39	2	2	Cloudy mountain
40	1	2	Expedition
43	1	1	Blop
45	1	1	Bat flying
47	1	1	Spider crawling
49	1	1	Rat
51	2	1	Demon waving
55	2	1	Snake crawling
59	2	1	Dragon
63	2	1	Winged dragon
67	2	1	Disintegration



*ADVANCED DUNGEONS & DRAGONS and TREASURE OF TARMIN are trademarks owned by and used under license from TSR, Inc. This Cartridge is approved by TSR, Inc., the publisher of the Fantasy Role Playing Games sold under the trademark ADVANCED DUNGEONS & DRAGONS® © 1982 TSR, Inc. All Rights Reserved.

ARMOR BATTLE®

N	D	M	DESCRIPTION
0	2	0	Tank in reserve
2	2	8	Right half of tank in action going counterclockwise 10th position to 1st position.
20	2	8	Left half of tank in action going counterclockwise 10th position to 1st position.
38	2	0	Bullet
39	2	6	Explosion
53	2	0	Tank destroyed

ASTROSMASH®

N	D	M	DESCRIPTION
0	1	0	Laser gun
1	1	4	Bullet moving
6	1	0	Small rocks
6	2	3	Large rocks (4 types)
14	1	3	Small spinner
18	1	5	Explosion
24	1	3	Guided missile

AUTO RACING

N	D	M	DESCRIPTION
			Car direction: going straight right or straight left
0	2	0	Left half of car
2	2	0	Right half of car
			Car direction: going at 15 degrees angle apart.
4	2	0	Left half of car
6	2	0	Right half of car
8	2	0	Left half of car
10	2	0	Right half of car
12	2	0	Left half of car
14	2	0	Right half of car
16	2	0	Left half of car
18	2	0	Right half of car
20	2	0	Left half of car
22	2	0	Right half of car
24	2	0	Left half of car
26	2	0	Right half of car
			Car direction: going upward
28	2	0	Left half of car
30	2	0	Right half of car
			Car direction: going downward
32	2	0	Left half of car
34	2	0	Right half of car

BACKGAMMON

N	D	M	DESCRIPTION
Only use			
O=3			
0	1	0	Cursor

BASEBALL

N	D	M	DESCRIPTION
0	1	0	Ball
1	2	0	Player in sitting position
3	2	3	Player running up and down
11	2	7	Player running sideways
27	2	0	Pitcher (non-throw position)
27	2	7	Pitcher throws ball
37	2	2	Player
41	2	0	Player
43	2	7	Batter hits ball

BASKETBALL

N	D	M	DESCRIPTION
At O=1			
0	2	7	Player running
16	2	0	Player standing
18	1	3	Ball bouncing
22	2	1	Movement of player blocking with hands raised
26	2	3	Controlled player bouncing ball
34	2	4	Controlled player sets shot
44	2	6	Controlled player jumps shot
58	2	3	Movement of player jumping up to get ball
66	2	1	Player throwing ball from corner
At O=0			
D=1			Ball only

BURGERTIME™†

N	D	M	DESCRIPTION
0	2	4	Chef
10	2	4	Details for chef
36	2	4	Chef side-ways
44	2	1	Chef's demise
48	2	1	Details for chef's demise
52	2	1	Explosion
56	2	3	Hot Dog
64	2	3	Hot Dog, back side
72	2	3	Hot Dog
80	2	11	Pickle
104	2	3	Egg

†BURGERTIME is a trademark of DATA EAST USA, Inc. used under license.
© 1982 DATA EAST USA, Inc.

BOWLING

N	D	M	DESCRIPTION
0	2	0	Player in still position
2	2	6	Upper part of player in throwing stance
16	2	11	Lower part of player in throwing stance
46	2	1	Spotter ball lowering
50	1	3	Bowling ball rolling
54	1	0	Standing pins
55	1	0	Pin
56	1	0	Pin
57	1	2	Pin knocked down

CHECKERS

N	D	M	DESCRIPTION
0	1	0	Unmoved checker
1	1	0	King
2	1	0	Checker as cursor
3	1	0	Arrow

TRON DEADLY DISCS*

N	D	M	DESCRIPTION
0	2	7	Tron or warrior men running
16	2	0	Tron blocks
18	2	0	Tron gets hit
20	2	0	Tron ducks
22	2	2	Tron running up and down
28	2	3	Tron disintegrates
36	1	0	Disc when harmless
38	1	3	Paralyzer stick
41	1	2	Disc movement
48	2	1	Recognizer
52	1	3	Paralyzer beam
56	2	0	Paralyzed shock

* © 1982 Walt Disney Productions.

FOOTBALL

N	D	M	DESCRIPTION
0	2	7	Player running sideways
16	2	0	Player in non-play (standing) position
18	2	0	Player in grouping position
20	2	0	Player in grouping position

22	2	0	Player tackled
24	2	3	Tacked player gets up

FROG BOG™

N	D	M	DESCRIPTION
2	2	10	Left side of frog jumping into water
32	2	0	Right half of still frog
34	2	10	Right side of frog jumping into water
56	2	1	Frog climbing up from water onto bank
62	2	0	Right half of sleeping frog
65	1	1	Tongue curling out (right side)
67	1	1	Tongue (left side)
69	1	1	Tongue curling in (right side)
74	2	3	(5 pts) bug
82	2	2	(15 pts) bug flying
88	2	2	(10 pts) bug flying
94	1	1	(35 pts) bug flying

GOLF

N	D	M	DESCRIPTION
10	2	3	When golfer putters
37	2	4	Bar movement

HOCKEY

N	D	M	DESCRIPTION
0	2	0	Goalie
8	2	4	Player running
18	2	6	Goalie gets hit and falls down and gets back up again
32	2	4	Player trips and sits inactive
42	2	0	Penalized player



HORSE RACING

N	D	M	DESCRIPTION
0	2	5	Right half of running horse
12	2	5	Left half of running horse

LOCK 'N' CHASE™†

N	D	M	DESCRIPTION
0	2	1	Eye of thief moving
4	2	1	Top and feet of thief
8	1	0	Middle part of thief
10	2	7	Sideways movement of policeman
24	2	8	Up and down movement of policeman
42	2	8	Policeman disappears and reappears again



†LOCK 'N' CHASE is a trademark of DATA EAST USA, Inc. used under license.
© 1981, 1982 DATA EAST USA, Inc.

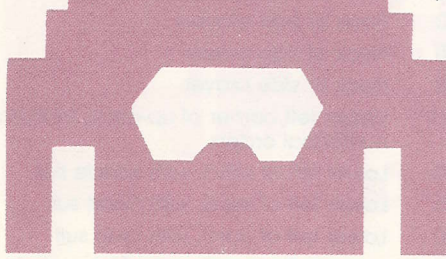
MATH FUN™ †

N	D	M	DESCRIPTION
0	2	8	Left half of monkey running, at initial start Right half is mirror image only
18	2	0	Left of lion
20	2	0	Right of lion
22	2	0	Left of whale, creature in water
24	2	0	Right of whale, creature in water
26	2	0	Left of deer
28	2	0	Right of deer
30	2	0	Left of kangaroo
32	2	0	Right of kangaroo
34	2	0	Left of hippopotamus, creature in water
36	2	0	Right of hippo, creature in water
38	2	0	Left of flamingo
40	2	0	Right of flamingo
42	2	0	Left of lion
44	2	0	Right of lion
46	2	0	Left of elephant
48	2	0	Right of elephant
50	2	0	Left of bear
52	2	0	Right of bear
54	2	0	Left of crocodile
56	2	0	Right of crocodile
58	2	1	Monkey jumping up and down, at end of game
62	2	2	Right half of monkey jumping into water
68	2	2	Left half of monkey jumping into water
74	2	1	Monkey in water
78	2	1	Monkey reappears in full form on land
98	2	0	Left of sheep
100	2	0	Right of sheep
102	1	1	Water splashing

† © 1981 Children's Television Workshop, Inc.

TRON MAZE-A-TRON™

N	D	M	DESCRIPTION
0	2	4	Flynn running
11	2	0	Flynn stands immobile or when is moved from one circuit board to another
12	1	4	Force field and rom chips
27	2	2	Electronic impulse
41	1	9	Mouth of MCP
51	2	0	Bit-gun sight
53	1	1	Bullet
55	2	2	Bit-gun sight firing
59	2	5	Flynn running toward you
71	2	6	Movement of recognizer
85	2	8	Explosion as MCP shoots back at Flynn



*© 1982 Walt Disney Productions.

NIGHT STALKER™

N	D	M	DESCRIPTION
0	2	10	Action of man running
0	2	7	Action of man running sideways
16	2	2	Action of man running forward
22	1	5	Bat flying
28	1	0	Bat hanging
29	1	5	Invisible robot
35	1	3	Robot explosion (1st half)
38	2	0	Robot explosion (2nd half)
40	1	1	Gun
41	2	2	Action of spider crawling horizontally
46	2	2	Action of spider crawling vertically
50	1	2	Bullet moving only
50	1	5	Bullet moves, hits, then explosion
57	2	1	Movement of blue robot
61	2	2	Movement of white robot
67	2	1	Movement of black robot
71	2	2	Movement of gray robot
75	2	1	Explosion
79	2	1	Man gets shot
81	2	2	Man gets paralyzed by bat or spider

POKER AND BLACKJACK

N	D	M	DESCRIPTION
0	2	3	Dealer's face looking side to side
8	2	0	Dealer's happy face
10	2	0	Dealer's frowning face
12	2	3	Card's movement
20	2	3	Dealer's hand deals card to players
28	1	11	Upper left half of cards
49	1	0	Upper right half of cards
41	1	0	Left half of heart card
42	1	0	Right half of heart card
43	1	0	Left half of spade card
44	1	0	Right half of spade card
45	1	0	Left half of diamond card
46	1	0	Right half of diamond card
47	1	0	Left half of club card
48	1	0	Right half of club card
50	1	0	Upper half (right side) of card 10
51	1	0	Upper half of card face down Mirror image to make one whole card
53	1	0	Flashing arrow

REVERSI®

N	D	M	DESCRIPTION
At O=0			
0	2	0	Cursor

ROYAL DEALER™

N	D	M	DESCRIPTION
At O=1			
2	2	0	Left half of card facing down
4	2	0	Right half of card facing down
1	1	0	Face of middle player
6	1	0	Face of side players
7	1	0	Neck of side player
9	1	0	Neck of side player
10	1	11	Upper left corner of up-card, featuring card numerical orders
23	1	0	Lower left of card, with spade suit
24	1	0	Lower left of card, with heart suit
25	1	0	Lower left of card, with club suit
26	1	0	Lower left of card, with diamond suit
27	1	0	Card face down

28	1	0	Left hairpiece of middle player
29	1	0	Right hairpiece of middle player
30	1	0	Left neck and shoulder of middle player
31	1	0	Right neck and shoulder of middle player
32	1	0	Bodice of middle player
33	1	0	Left hairpiece of side player
34	1	0	Right hairpiece of side player
35	1	0	Left neck and shoulders of side player
36	1	0	Right neck and shoulders of side player
37	1	0	Left half at arm level
38	1	0	Right half at arm level
39	1	0	Left half at chair level
40	1	0	Right half at chair level
			More descriptions for side player with plaid skirt
41	1	0	Left shoulder
42	1	0	Right shoulder
43	1	0	Left bodice
44	1	0	Right bodice
45	1	0	Left half of plaid skirt
46	1	0	Right half of plaid skirt

SEA BATTLE™

N	D	M	DESCRIPTION
0	1	0	One ship in fleet of 3
1	1	0	Two ships in fleet of 3
2	1	0	Three ships in fleet of 3
3	1	0	Aircraft carrier
4	1	0	Troop transport
5	1	0	Battle ship
6	1	0	Submarine
7	1	0	Destroyer
8	1	0	PT boat
9	1	0	Mine sweeper
9	1	0	Mine layer
10	1	0	Mine sweeper
11	1	1	Bullet
13	1	4	Multiple gunshots (5 variations)
18	2	5	Explosion of ship
30	1	0	Cross hair
31	1	2	Destruction of ship into pieces

SHARK! SHARK!™

N	D	M	DESCRIPTION
0	2	1	Player fish size 1
4	2	1	Player fish size 2
8	2	1	Player fish size 3
12	2	1	Player fish size 4
16	2	1	Player fish size 5
20	2	3	Front head of swimming shark
28	2	3	End tail of swimming shark
36	2	1	Front head of shark killed
40	2	1	End tail of shark killed
44	2	2	Other (blue) swimming fish, size 0
50	2	2	Blue swimming fish, size 1
56	2	2	Blue swimming fish, size 2
62	2	2	Blue swimming fish, size 3
68	2	1	Tan swimming fish, size 0
72	2	1	Tan swimming fish, size 1
76	2	1	Tan swimming fish, size 2
80	2	1	Tan swimming fish, size 3
84	2	6	Movement of sea horse
98	2	2	Purple fish, size 3
104	2	1	Pink fish, size 4

108	2	1	Orange fish, size 4
112	2	2	Movement of crab
118	2	2	Movement of lobster
124	2	1	Movement of bubbles

SHARP SHOT™

N	D	M	DESCRIPTION
Only use O=5			
0	2	2	Movement of spaceship (SPACEGUNNER)
6	1	3	Submarines (4 variations)
10	1	2	Submarine sinks
13	1	2	Bullet (SPACEGUNNER)
16	2	3	Explosion at gunsight (SPACEGUNNER and SUBMARINE)
23	1	0	Bullet (SUBMARINE)
25	2	7	Player running (PASS GAME)
41	2	0	Offending player passing ball
43	2	0	Receiving player getting ball
45	2	0	Tackled player (pass defender)
47	2	0	Tackled player (pass offender)
49	1	0	Ball on ground
50	1	0	Ball on player's hand

51	2	0	Player kneeling with both hands and feet on ground
53	2	0	Player half bending
55	2	0	Player in blocking position
57	2	1	Movement of blob-like target monsters
61	2	1	Movement of devil-like target monster
65	2	1	Disintegration of monster
69	1	1	Movement of spider-like monsters
71	1	0	Gunsight pointing right
72	1	0	Gunsight pointing up
73	1	0	Gunsight pointing left
74	1	0	Gunsight pointing down
75	1	0	Island (SUBMARINE)
76	1	0	Celestial bodies (SPACEGUNNER)
77	1	0	Celestial bodies (SPACEGUNNER)
78	1	0	Arrow pointing left (MAZE SHOOT)
79	1	0	Arrow pointing up (MAZE SHOOT)
80	1	0	Arrow pointing right (MAZE SHOOT)
81	1	0	Arrow pointing down (MAZE SHOOT)
82	1	0	Player's treasure
83	1	0	Left of pyramid
84	1	0	Right of pyramid

SKIING

N	D	M	DESCRIPTION
0	2	6	Skis circling a one/half circle as skier turns
14	2	1	Skis tumble as skier falls
16	2	3	Skier falls and gets up
24	2	12	Skier in different positions from start to finish

SNAFU®

N	D	M	DESCRIPTION
Only use O=1			
0	1	0	Head of serpent going left
1	1	0	Head of serpent going diagonally up to the right
2	1	0	Head of serpent going up
3	1	0	Head of serpent going diagonally up to the left
4	1	0	Head of serpent going right
5	1	0	Head of serpent going diagonally down to the left
6	1	0	Head of serpent going down
7	1	0	Head of serpent going diagonally down to the right
8	1	7	Twisting action of ending link (BITE GAME)
16	1	5	Obstacles (6 variations) Also explosion

SOCCER

N	D	M	DESCRIPTION
0	1	0	Ball
1	2	0	Receiving player
3	2	3	Player running up and down
11	2	7	Player running sideways
27	2	0	Throw-in: Player returns the ball inbounds by throwing it over side line
29	2	0	Player sidekicks
31	2	2	Offensive controlled player kicks ball to start of

SPACE ARMADA®

N	D	M	DESCRIPTION
0	2	0	Laser gun
0	2	4	Laser gun disintegrates
10	2	2	Explosion when laser shot hits space creatures
16	2	3	Movement of white wiggling bombs
22	2	1	Explosion of white bomb as it lands on ground
26	2	5	Movement of laser shots
38	1	0	Left half of first row creatures (orange)
46	1	0	Left half of first row targets (orange)

50	1	0	Left half of first row targets (orange)
39	1	0	Left half of 2nd row targets (green)
40	1	0	Right half of 2nd row targets (green)
41	1	0	Left half of 3rd row yellow targets
42	1	0	Right half of 3rd row yellow targets
43	1	0	Left half of 4th row purple targets
44	1	0	Right half of 4th row purple targets
45	1	0	Laser gun
51	2	4	Red spaceship movement and explosion
61	2	1	Guided missile movement
65	1	0	Bunker
66	1	0	Laser gun
67	2	1	Explosion of guided missile
71	2	2	Explosion
77	2	1	Guided missile

SPACE BATTLE™

N	D	M	DESCRIPTION
0	2	1	Alien squadron in battle scene (front view)
2	2	1	Alien squadron flying to left upper side
6	2	15	Different faces of alien squadron
38	2	0	Gun sight
40	2	0	Gun missiles
42	2	0	Explosion of gun missile
44	2	3	Explosion of alien squadron
50	1	3	Laser beam from being small and harmless to large and harmful
54	1	0	Alien squadron of two fighters
55	1	0	Alien squadron of four fighters
56	1	0	Alien squadron of six fighters
57	1	0	Alien squadron of eight fighters
58	1	0	Alien squadron of ten fighters
59	1	0	Alien squadron of twelve fighters
60	1	0	Alien squadron of fourteen fighters
61	1	0	Alien squadron of sixteen fighters
62	1	2	Mothership squadron changing from one to three units

SPACE HAWK™

N	D	M	DESCRIPTION
0	1	1	Movement of celestial bodies
2	1	0	Shooting star
3	1	0	Hunter in reserve
4	2	1	Double bubbles (2 variations)
8	2	1	Single bubbles (2 variations)
12	2	5	Explosion of bubble
24	2	5	Single bubble breakage
36	2	2	Comet
42	2	7	Hunter's movements in circle
58	2	3	Hunter's bullet, fired and exploded
66	1	2	Trail left in air as hunter thrusts in air
69	1	7	Hawk flying, gets hit and disintegrates

STAR STRIKE™

N	D	M	DESCRIPTION
0	1	0	Air-to-air laser trigger
1	1	1	Red target
2	1	0	Alien spaceship in small size (farther view)
3	1	0	Alien spaceship in large size (closer view)
4	1	1	Explosion
6	1	1	Flash of flame when alien spaceship fires mother-ship
8	1	0	Bullet released by alien spaceship
11	1	0	Explosion of red target
12	1	0	Your spaceship

SUB HUNT™

N	D	M	DESCRIPTION
0	1	0	One ship in enemy convoy
1	1	0	Enemy convoy
2	1	0	Your submarine selected
3	1	0	Single destroyer
4	1	4	Destroyer appears from far to near
9	1	3	Troopship from far to near
13	1	1	Destroyer from far to near
15	1	0	Left of a troopship at close view
16	1	0	Right of a troopship at close view
17	1	0	Upper left of a destroyer
18	1	0	Upper right of a destroyer
19	1	0	Lower left of a destroyer
20	1	0	Lower right of a destroyer
21	1	2	Bullet shot in periscope
24	1	0	Gauge in measuring meters
25	1	3	Explosion of ship
29	1	1	Explosion in water
31	1	0	Your submarine flashing in attacking scene
32	1	1	Explosion
34	1	0	Enemy convoy fleet

TENNIS

N	D	M	DESCRIPTION
Only at O=1			
1	2	1	Ball and shadow
6	2	5	Player running
18	2	2	Player serving ball
24	2	1	Player running horizontally from inner to outer court when serving ball
28	2	6	Movement of racket

TRIPLE ACTION™

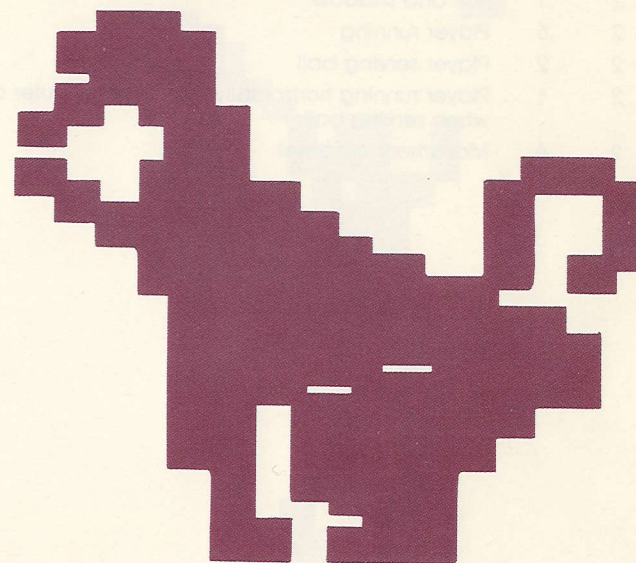
N	D	M	DESCRIPTION
0	1	0	Bullet in BATTLE TANK
1	2	4	Movement of tank turning one quarter of circle
11	1	4	Explosion
16	2	7	Plane circling one half of circle
32	2	0	Car in normal shape
34	2	0	Car in accident
36	2	0	Elevated structures in BIPLANES
38	1	0	Balloon
40	1	0	Bullets of BIPLANES
41	1	1	Lane divider in CAR RACING
43	1	1	Cloud formation in BIPLANES (2 variations)
45	1	0	Plane ruined in accident

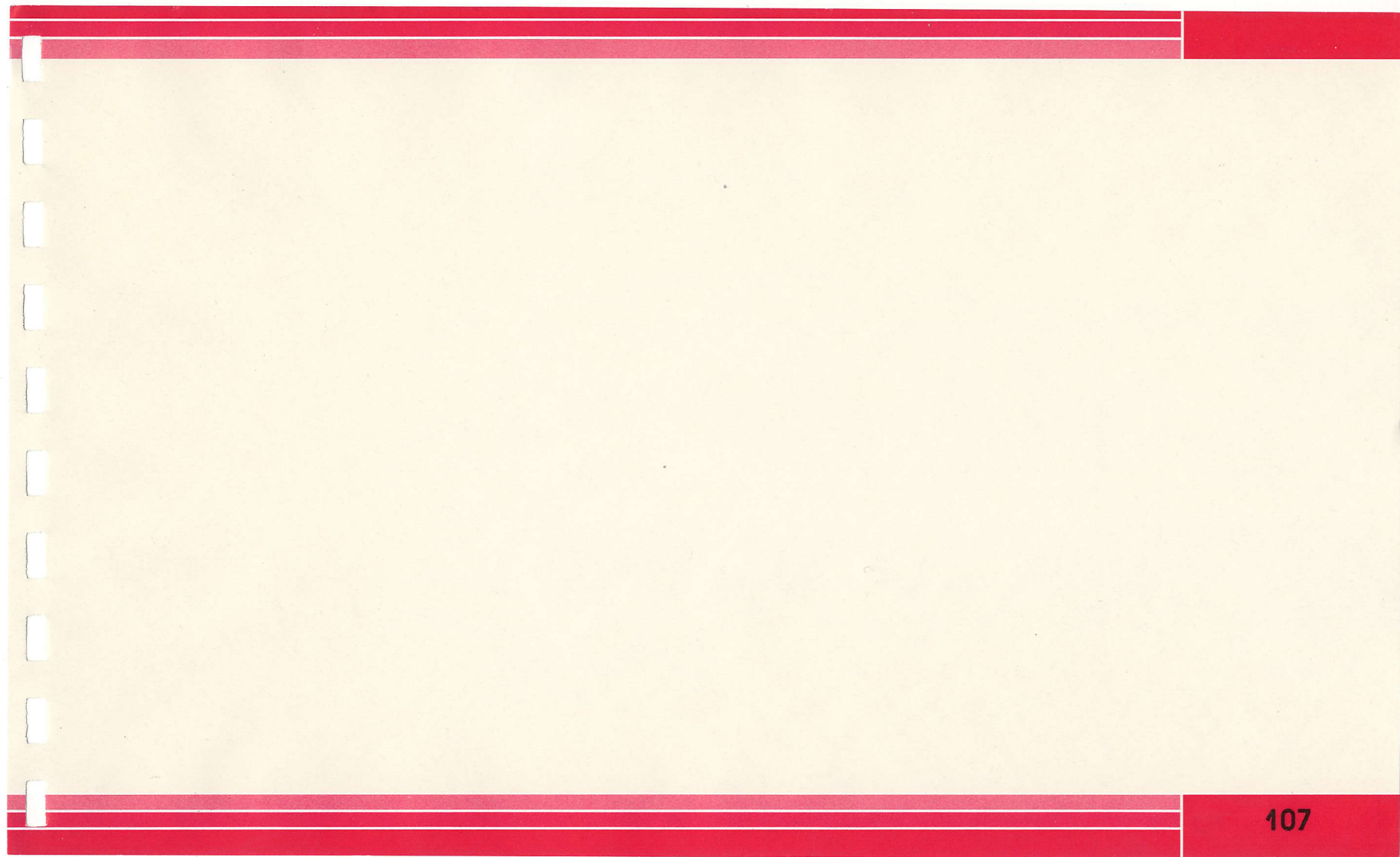
UTOPIA[®]

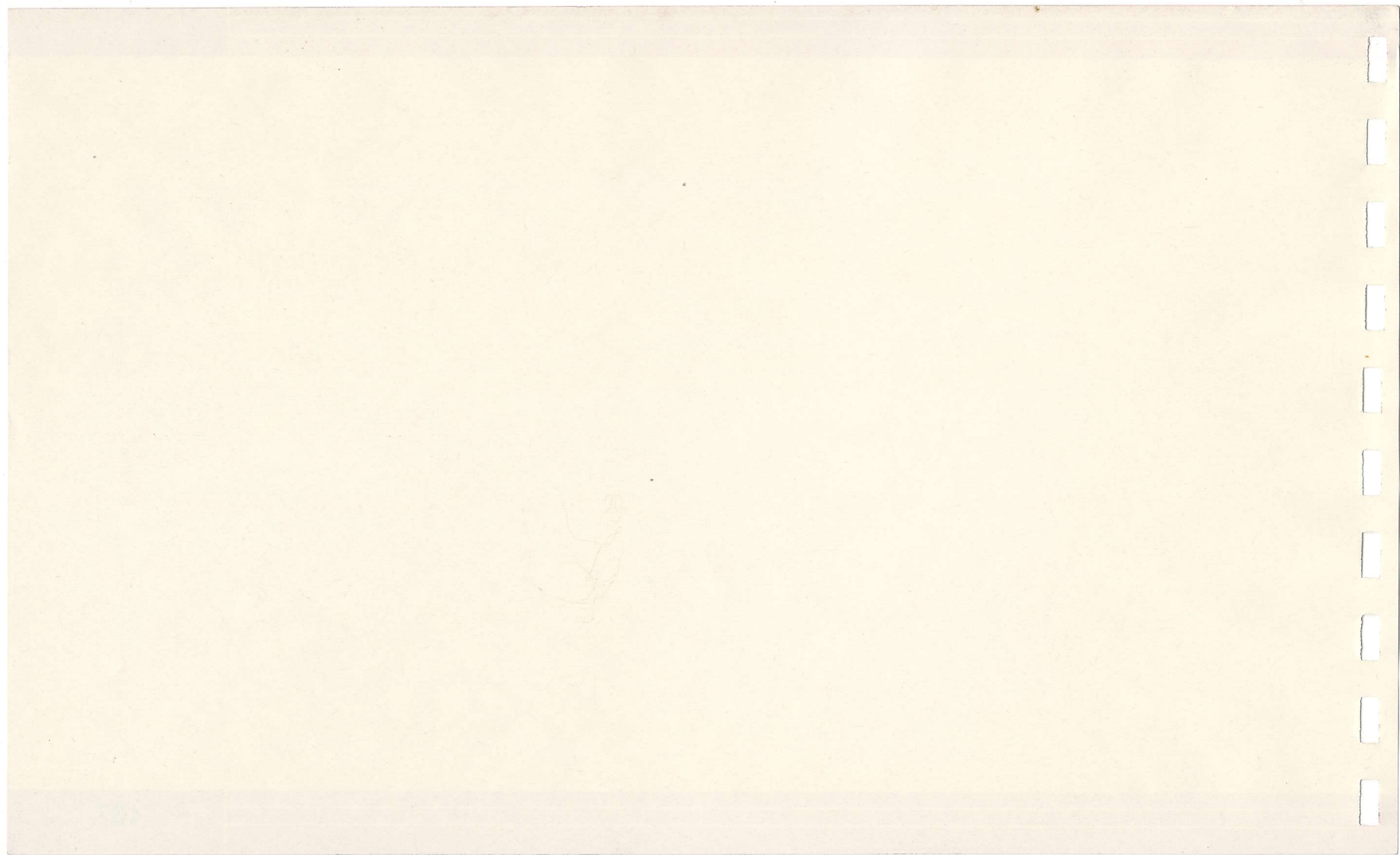
N	D	M	DESCRIPTION
0	1	0	Cursor
1	1	0	Fishing boat
2	1	0	PT boat
3	1	7	Sinking of fishing boat
11	1	3	School of fishes moving
15	1	0	Pirate ships
16	2	4	Movement of rain storms or tropical storms
26	2	5	Movement of hurricanes
39	1	0	Fort
40	1	0	Factory
41	1	0	Crops
42	1	0	School
43	1	0	Hospital
44	1	0	Housing project
45	1	0	Rebel soldiers
46	1	0	PT boat
47	1	0	Fishing boat
48	1	8	Sinking of fishing boat

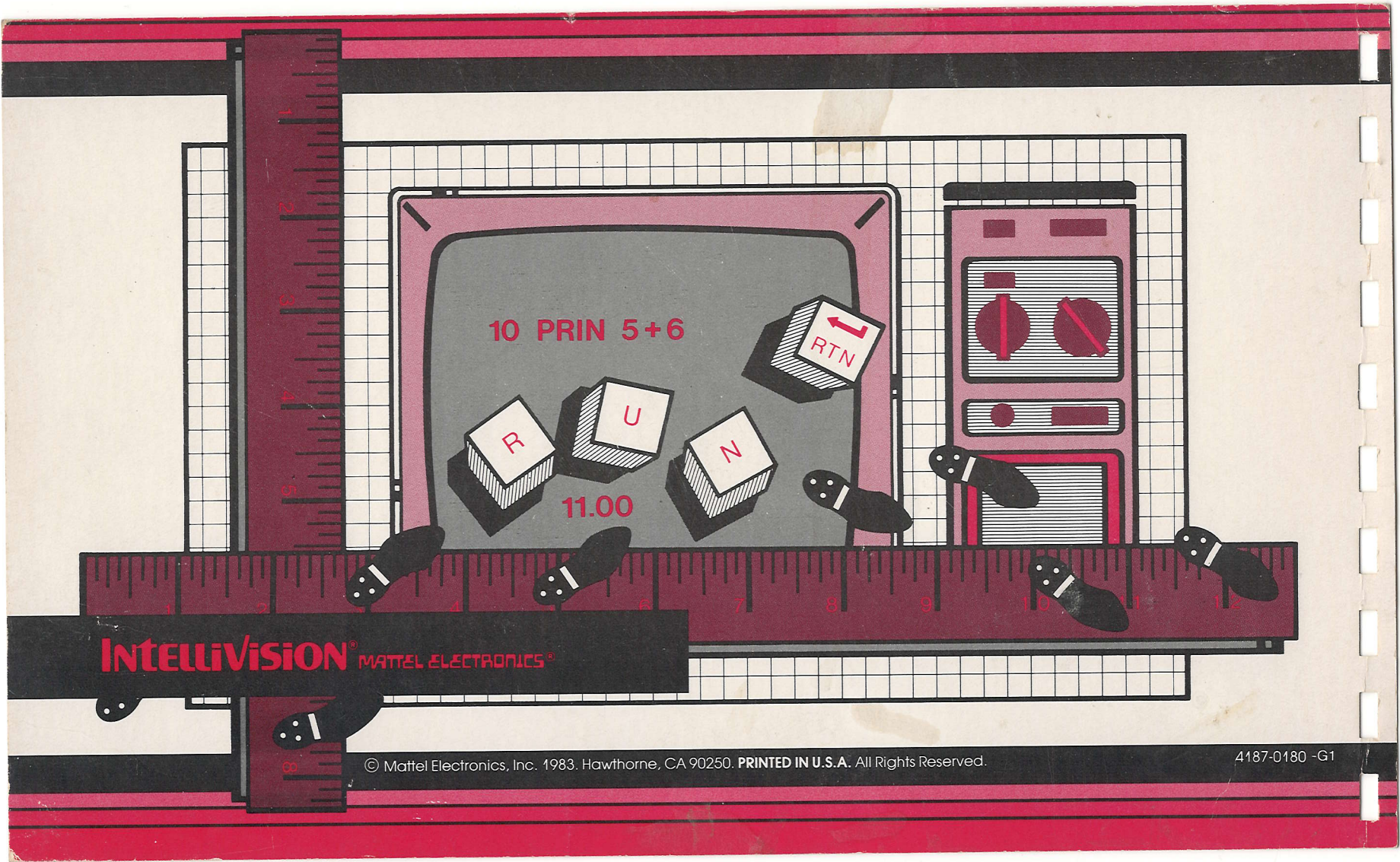
WORDFUN[™]

N	D	M	DESCRIPTION
8	2	1	Front head of monkey
12	2	1	End tail of monkey









Intellivision® MATTTEL ELECTRONICS®

© Mattel Electronics, Inc. 1983. Hawthorne, CA 90250. PRINTED IN U.S.A. All Rights Reserved.

4187-0180 -G1