

Bally PROFESSIONAL
VIDEOCADE™
 CARTRIDGE

Bally BASIC 6004

NOW WITH BUILT-IN AUDIO INTERFACE
 AND ILLUSTRATED LEARNING COURSE

There's no easier way to learn about computers than with the new Bally BASIC system. This plug-in cartridge with built-in audio tape interface converts your ARCADE into a personal computer you can program yourself. The complete self-teaching instruction book will help you learn programming while you create computer games, electronic music, and video art.

Copyright © 1981 Astrovision, Inc. All Rights Reserved.

Bally BASIC 6004
 Version 1.0 - Released Oct 19, 2000

This document has been converted to PDF format courtesy of the *Bally Alley* newsletter. For other reprints and more information visit: <http://www.ballyalley.com> Corrections? Suggestions? Email Adam Trionfo at: ballyalley@hotmail.com

Bally® **BASIC**

Bally[®] **BASIC**

By Dick Ainsworth with George Moses,
Jay Fenton and Brett Bilbrey

Bally® **BASIC**

Welcome to the world of computers. There are many versions of BASIC as well as several other computer languages. The term, BASIC, is an acronym for: **B**eginners' **A**ll-purpose **S**ymbolic **I**nstruction **C**ode. Bally BASIC is a language designed to make computers and programming easier to understand. It is an expanded version of Li-Chen Wang's Palo Alto Tiny BASIC. Written by Jay Fenton, Bally BASIC allows you to program or create pictures and sounds accompanied by a full range of 256 color choices. Bally BASIC expands your computer by letting you program your own computer games, electronic music and video art.

These programming lessons, written by Dick Ainsworth, with George Moses, Jay Fenton and Brett Bilbrey, are an introduction to understanding and using your computer. You will learn how to enter programs so that you can see many of the things your computer can do. By experiencing how BASIC works, you will soon learn the special BASIC command words such as RUN and PRINT. Follow the lessons that interest you most. Then try using the computer to express your own ideas.

The special section on computer music, written by George Moses, shows how you can use your computer as a musical instrument. With these more advanced techniques you can use the computer's built-in sound synthesizer to play music in three-part harmony using your favorite sheet music, or you can program your own original compositions.

Bally[®] **BASIC**

Copyright © 1981 Astrovision, Inc.
All Rights Reserved.

CONTENTS

Operating Instructions	6
-------------------------------	----------

Programming Course

Lesson 1	Printing, Counting and Loops	17
Lesson 2	Random Numbers, Inputs and What If?	24
Lesson 3	Subroutines	30
Lesson 4	Arrays	35
Lesson 5	Electronic Music	40
Lesson 6	Graphics	48
Lesson 7	Computer Games	54
Lesson 8	Video Art	61
Lesson 9	Three Voice Music	66

Programs

Computer Games		71
Electronic Music		
Graphs and Charts		
Video Art		
Learning Skills		

Appendix

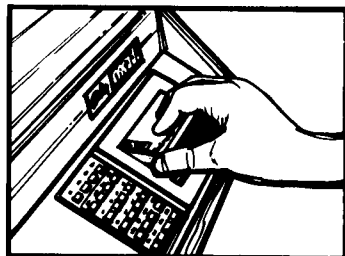
A	Character Code Table	98
B	Bus and Connector Structures	99
C	Memory Map	101
D	Input and Output Ports	102
E	Bally Basic Data Base Locations	103
F	Bally 300 Baud to 2000 Baud Tape Converter	104
G	System Block Diagram	107
H	Bally Commands and Functions Summary	108
I	General Specifications	115

Operating Instructions

If you are using your Bally Arcade for the first time, please follow the directions in the OWNER'S MANUAL packaged with your unit. Connect your Bally Arcade to a black and white or color TV and play several of the games.

After you are familiar with your Arcade and know how it operates, use Bally BASIC and discover the enjoyment of having your own personal computer.

REMOVE the keypad overlay from its envelope in the front of this manual. (This envelope is a good place to store your overlay when you're not using it.)

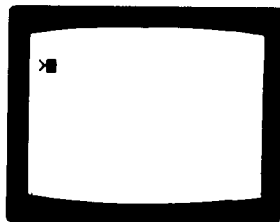


INSERT the Bally BASIC cassette in the cassette slot and press down firmly.

PLACE the keypad overlay on the keypad.



RESET your computer by pressing the RESET button next to the cassette. Your TV screen should look like this picture.



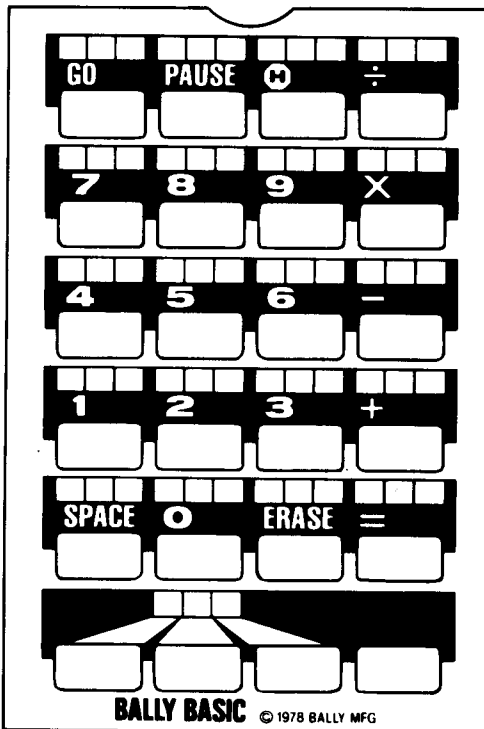
CAUTION:

RESET erases your program. If you press this button by accident, you must enter your program again from the beginning.

EJECT causes your programming cassette to pop up, so you can remove it. Pressing the EJECT button accidentally will cause your program to stop. If this happens, push the cassette back into place, press RESET, and enter your program again.

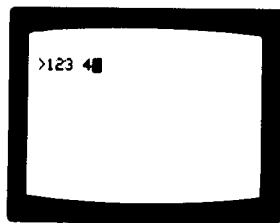
STATIC that causes dots or "snow" on your TV screen or noise in the speaker can affect your computer. If static interrupts your program and causes it to stop, press RESET and enter the program again. If your body has a static-electricity build-up from walking across a carpet in a dry environment, discharge the spark before touching your computer. This is important. One high voltage spark such as this can seriously damage sensitive integrated circuits in your computer and should be carefully guarded against.

Using the Keypad Numbers



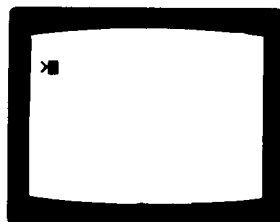
Your Bally BASIC keypad is used for three separate kinds of Information: NUMBERS, LETTERS and WORDS. The WHITE numbers and symbols on your keypad are printed on your TV screen when you press those keys.

1
2
3
SPACE
4

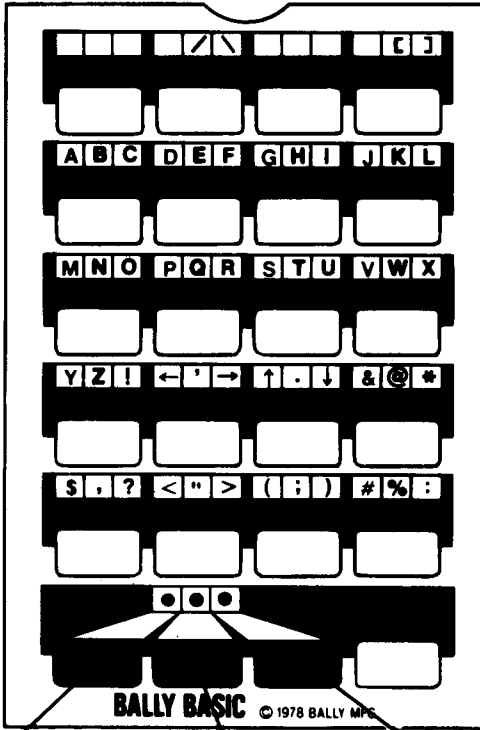


Now use the ERASE key to remove the numbers from the screen.

ERASE
ERASE
ERASE
ERASE



Letters



The GREEN shift key selects characters on the left.

The RED shift key selects characters in the center.

The BLUE shift key selects characters on the right.

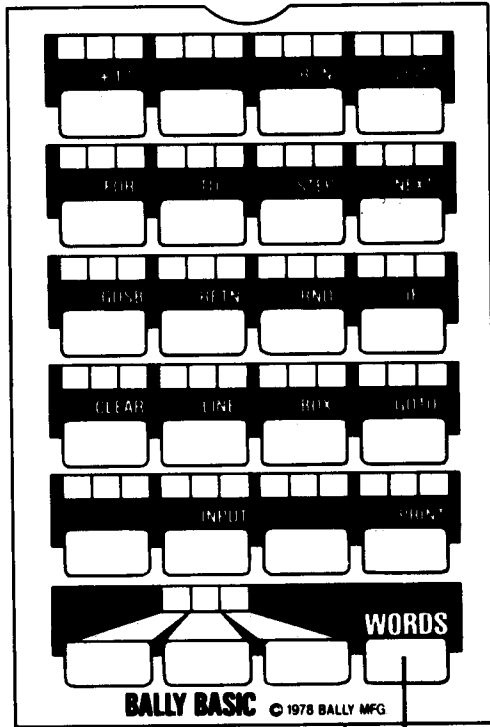
To print a letter or character on your TV screen, use the shift key in the same color. First press either the GREEN, RED or BLUE shift key to select a letter of the corresponding color. Then press the key that is under the letter you want to print.

Don't hold down the shift key when you press a letter or character key. Use one discrete keystroke on each key.

RED
H
BLUE
|
SPACE
RED
T
RED
H
RED
E
BLUE
R
RED
E
BLUE
|



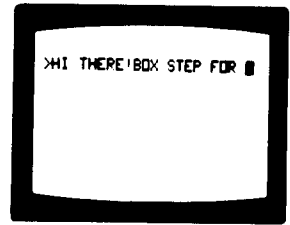
Words



The GOLD shift key selects the WORDS printed in GOLD.

Words that the computer understands are printed on the keypad in GOLD. Press the WORDS key, and then press the key under the word you want to print on your TV.

WORDS
BOX
WORDS
STEP
WORDS
FOR



Never spell out computer words like B-O-X or S-T-E-P — use the gold WORDS shift key to enter these computer command words into your program.

RUN
LIST
FOR
TO
+TO

STEP
NEXT
GOSB
RETN

RND
GOTO
INPUT
PRINT

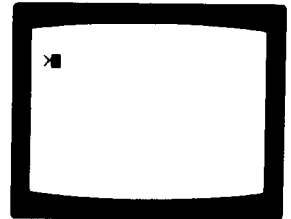
IF
CLEAR
LINE
BOX

A complete description of these computer command words will be discussed in this book.

NOTE: If your computer prints "WHAT?" on the screen, you have typed something the computer does not understand. It will point out the mistake with a question mark (?). Correct the error by typing the line again.

You can now print numbers, letters and words on the screen. Next you will learn to put programs into your computer. Reset the computer by pressing the RESET button.

RESET



The RESET button erases all instructions and programs in the computer's memory and clears the screen. Now let's enter a short program. Number the first instruction 10. Use the WORDS key to enter PRINT, and then spell out "HELLO!" The quotation marks are important because only the letters you put between them will be printed.

10PRINT "HELLO!"
GO



The GO key acts like a carriage return on a typewriter and moves you to the next line while entering the instruction you have just typed into the memory. Line numbers are used to tell the computer what to do next. NOTE: If you fill up a line with 26 characters and spaces the computer will automatically shift and start printing on the line below it. Now add the second instruction to your program and number it 20. Notice that GOTO is one word. Press the GO key to end the line.

```
20GOTO 10
GO
```



Now the program is in the computer memory. To look at the complete program, ask the computer to LIST it.

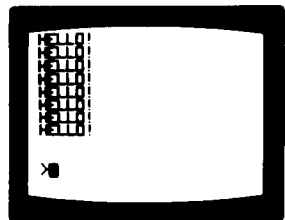
```
LIST
GO
```



Notice how line 20 tells the computer to go to line 10 and print "HELLO!" again. If your TV screen doesn't match this picture, fix the line that has the mistake by using the EDITOR feature in Bally BASIC. (The editor is described on page 16.)

Now, if your program is correct you can run it. The computer will print the word "HELLO!," go back to the beginning of your program and start over. To stop the program, press the halt key, H, and hold it down until the computer halts.

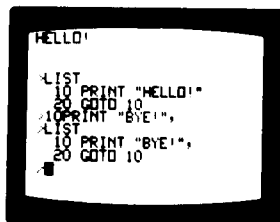
```
RUN
GO
H
```



Now LIST your program again. You need to know how to change a program so that you can fix a mistake or make your program do something different. To change an instruction enter the number of the line you want to change and then enter the new instruction. The computer will replace the old line of the same number with the new line. Now enter this new instruction to replace line 10 in your program. Don't forget the comma at the end!

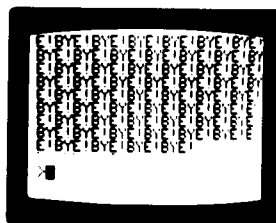
```
10PRINT "BYE!",
```

```
GO
LIST
GO
```



If your program matches the example, RUN It and see what it does. Use the HALT key **H** to stop.

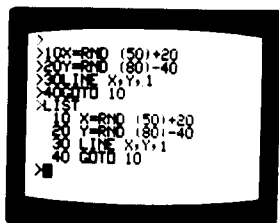
```
RUN
GO
H
```



Now enter and run this longer program. After RESET and at the end of each line, press the WORDS key before you press the GO + 10 key. Your computer will number the lines 10, 20, 30 and 40 automatically. Be careful to use the letter x, not the multiplication sign (\times).

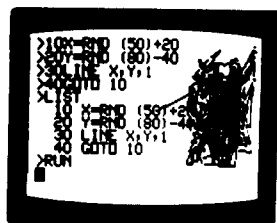
```
RESET
10X = RND(50) + 20
20Y = RND(80) - 40
30LINE X,Y,1
40GOTO 10
```

```
GO
LIST
GO
```



Check your program to make sure it matches the example, then run it.

```
RUN
GO
```



This program will draw a random line that fills the right side of your screen. You can use the PAUSE key to stop the program; press any key to start it again. You will see how to program random events and create pictures in the programming course that begins on the following pages.

In the PROGRAMS section, pages 71 - 97, are many programs you can select, enter into your computer, and run. Remember to press RESET before each program and to press GO after each line. If you want to number lines automatically, press WORDS and GO.

THE AUDIO CASSETTE INTERFACE

The audio socket on the lower-right corner of your Bally BASIC cartridge is your connection port to the interface between your computer and any cassette tape recorder. This will allow you to save any program in the computer's memory on cassette tape and to input the program from tape to memory in less than 20 seconds. The following commands relate to tape storage and retrieval of Bally BASIC programs.

:PRINT

The :PRINT command causes the stored program, the screen image, the values stored in the @ () and * () arrays, and the values of all variables to be output to tape. This process will take between 10 and 20 seconds. As only one jack is provided on the BASIC cartridge, it is necessary for the user to manually connect the audio cable to the MIC jack of the cassette recorder. Have your recorder running in the RECORD mode, type in :PRINT and press GO. When the cursor reappears, the computer is done writing your program to tape.

The recording will consist of a 3 second leader tone, then the data block, followed by a 1/2 second trailer tone.

:INPUT

To load programs, use :INPUT. This will retrieve the program, the screen image, the arrays and variables from audio tape. It is necessary to "cue up" the audio tape on the three second leader tone and switch the cable over to the EARPHONE jack on your tape recorder before loading. A light emitting diode (LED) is provided on the lower left corner of your Bally BASIC cartridge to aid in checking the playback level of your cassette recorder. The volume should be set above the level that causes the LED to glow steadily. When the tape is cued, type :INPUT GO and press PLAY on your tape recorder. When the cursor reappears your program is loaded.

:LIST

The :LIST command has been designed to perform a verify function. It scans a digital recording and checks the sum of the bits in the recorded program against the sum of the bits in the program currently in memory. The :LIST function is to be performed just after writing your program to tape with the :PRINT command, while your program is still in the computer. This allows you to check the integrity of your recording without damaging the program. If :LIST finds an error, a question mark is printed just before the cursor when it returns. If problems arise, check the playback level of your recorder, or rewrite the program to tape if necessary.

EDITOR

You can use the Bally BASIC editor to change a line in your program.

1. Type the line number of the line you wish to correct.
2. Press the PAUSE key once to recall the next character from the line in storage. Repeated pressing of PAUSE will scan across the stored line.
3. A character drawn by PAUSE looks to the computer just like a character entered directly from the keypad. This means ERASE can be used to discard unwanted characters from the stored line. Additions can also be added from the keypad, intermixed as desired.
4. If PAUSE runs off the end of a stored line, it will have the same effect as pressing the GO key. The key sequence WORDS—SPACE will also activate this feature.

THE TRACE FEATURE

Now that you have a program in the memory of your Arcade, let's try out the TRACE feature. TRACE is a self-teaching function built into Bally BASIC to let you see which statements in your program are operating while you watch the program run. A statement by statement TRACE listing may be obtained while the program is executing by holding down the WORDS key and the BLUE key at the same time. Each statement is listed completely before it is executed.

REVIEW

Check these steps to make sure you understand how to operate your computer and enter programs.

1. Insert your Bally BASIC Programming Cassette and put the keypad overlay in place.
2. Press RESET (next to cassette). This erases any old programs.
3. Enter each instruction and press GO. (If you press WORDS and GO + 10 after each line, the computer will automatically add 10 and print a new line number for you.)
4. LIST the program and check each instruction carefully. The PAUSE key lets you pause when listing long programs.
5. If there are any mistakes, correct the line using the EDITOR feature or, if you wish, enter the instruction again using the same line number. To remove an instruction completely, reenter its line number and press GO.
6. When your program matches the example, press RUN and GO.

Next Step...

Now you can continue with Lesson One and learn how to write your own programs, or you can go to the PROGRAMS section of this manual and try any of the programs you like.

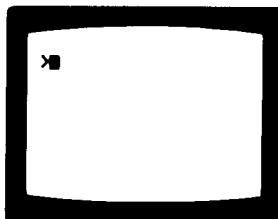
LESSON 1

PRINTING, COUNTING AND LOOPS

Before you begin these lessons, please read and understand the OPERATING INSTRUCTIONS. They begin on page 6 and show you how to enter, list and run programs on your computer.

Learning how to write your own programs is not hard at all. Soon you will be able to have your computer play your own games, music and video art. Begin by writing a short program. First clear the computer's memory with the RESET button:

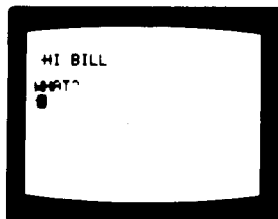
RESET



Now spell out HI and your name. Use the color shift keys as described on page 8. To type the letter H, for example, press the red shift key and then the key under the red letter H. After you have finished typing, press GO.

HI BILL

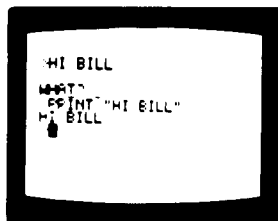
GO



The computer prints "WHAT?" on the screen because it doesn't know what you said. The words "HI" and "BILL" are not words your computer understands. Instead, use the WORDS shift key to enter the word PRINT. Then spell out "HI BILL." Don't forget the quotation marks.

PRINT "HI BILL"

GO



PRINT is one of the special words your computer understands. When you pressed GO, the computer followed your instruction and printed the words between the quotation marks. Now press GO again and see what happens:

GO

```
>HI BILL!  
WHAT?  
>PRINT "HI BILL"  
HI BILL  
>  
>  
>
```

You can't print these words a second time because the computer doesn't remember what to do. To have your computer remember your instruction, just give it a line number. Number your instruction 10 and enter it again.

10PRINT "HI BILL"

GO

```
>HI BILL!  
WHAT?  
>PRINT "HI BILL"  
HI BILL  
>  
>10PRINT "HI BILL"  
>  
>
```

Now you have a one-line program in the computer memory. You can run this program as many times as you like. To run your program, use the WORD shift key and enter RUN.

RUN

GO

```
>HI BILL!  
WHAT?  
>PRINT "HI BILL"  
HI BILL  
>  
>10PRINT "HI BILL"  
>RUN  
HI BILL  
>  
>
```

Add a second instruction to your program and number it 20. Then list your program.

20GOTO 10

GO

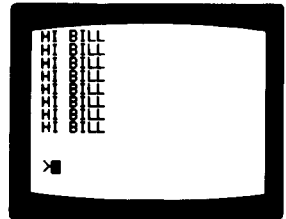
LIST

GO

```
>PRINT "HI BILL"  
HI BILL  
>  
>10PRINT "HI BILL"  
>RUN  
HI BILL  
>20GOTO 10  
>LIST  
10 PRINT "HI BILL"  
20 GOTO 10  
>  
>
```

Here's what your program will do. The computer will print HI BILL, go back to the beginning again, print HI BILL, and continue until you press the HALT key.

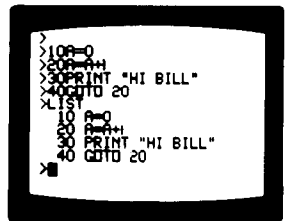
RUN
GO
H



Press and hold H until the computer stops.

How many times did you run your program? There is an easy way to find out. Make a counter to keep track of the number of times it ran. Reset your computer, then enter and list this new program.

RESET
10A = 0
20A = A + 1
30PRINT "HI BILL"
40GOTO 20
GO
LIST
GO



This program uses the letter A as a counting variable. Here's what happens when you run the program.

In line 10 the computer sets A equal to 0.

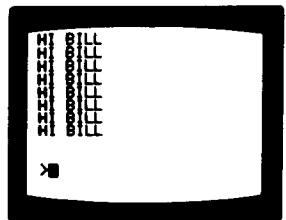
In line 20 the computer adds 1 to the number stored in A.

In line 30 the computer prints whatever is between the quotation marks.

In line 40 the computer goes back to line 20, adds one more to the number stored in A, and repeats.

Run your program and print "HI BILL" about a dozen times. Then press and hold the halt (H) key.

RUN
GO
H



The computer sets the variable A to zero in line 10. In line 20, 1 is added to A. Next, in line 30, the computer loops back to line 20 and repeats.

COUNTING LOOP

```
10A = 0
20A = A + 1
30PRINT A
40GOTO 20
```

WHAT THE COMPUTER DOES

```
10A = 0
20A = 1
30PRINT 1
40GOTO 20

20A = 2
30PRINT 2
40GOTO 20

20A = 3
30PRINT 3
40GOTO 20
```

Until you press halt **[H]**

Programs that repeat are called loops. Another way to program a loop is with the words FOR and NEXT. FOR, TO and NEXT are computer words. Use the WORDS shift key to enter them just as you enter RUN, PRINT, LIST and all other Bally BASIC computer words.

RESET your computer to erase the counting loop and enter this program.

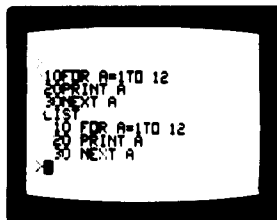
[RESET]

```
10FOR A = 1TO 12
20PRINT A
30NEXT A
```

[GO]

[LIST]

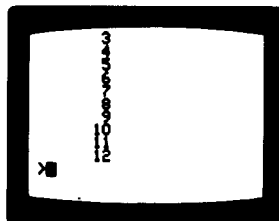
[GO]



In line 10 the computer sets the variable A to 1. In line 20, A is printed. The word NEXT in line 30 means add 1 to A and loop back to the word FOR. NEXT A replaces A = A + 1 and GOTO 20 which were used in the last program. Run your program and print the number in A as the A goes from 1 to 12.

[RUN]

[GO]



This time the program loop stopped automatically at 12. List your program again.

LIST

GO

```
7
9
10
11
12
XLIST
10 FOR A=1 TO 12
20 PRINT A
30 NEXT A
>
```

FOR/NEXT LOOP

```
10FOR A =1 TO 12
20PRINT A
30NEXT A
```

WHAT THE COMPUTER DOES

```
10A = 1
20PRINT 1
30A = 2;GOTO 20

20PRINT 2
30A = 3;GOTO 20

20PRINT 3
30A = 4;GOTO 20

Until A = 12
```

The FOR/NEXT loop adds 1 to the variable A. You can also add 2, 3, or any other number to a variable. Change line 10 to count by 2's.

```
10FOR A = 1 TO 12STEP 2
GO
LIST
GO
```

```
12
XLIST
10 FOR A=1 TO 12
20 PRINT A
30 NEXT A
>10 FOR A=1 TO 12STEP 2
XLIST
10 FOR A=1 TO 12STEP 2
20 PRINT A
30 NEXT A
>
```

Now run your program and see if it prints all the odd numbers between 1 and 12.

RUN

GO

```
10 FOR A=1 TO 12STEP 2
20 PRINT A
30 NEXT A
>RUN
1
3
5
7
9
11
>
```


You could also change line 10 and print all the tens from one to one hundred or all the leap years since your birthday. You can even step backwards by using negative numbers. Erase the program now in the computer with RESET, and enter this new program.

RESET

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

GO

LIST

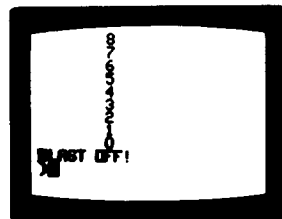
GO



Now run your program. You're at 10 seconds and counting!

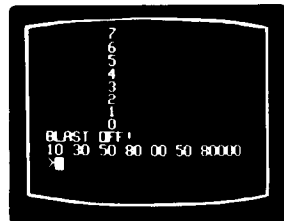
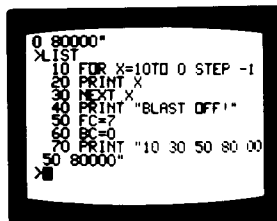
RUN

GO



Now for some fun to end your first lesson. Add the following three lines to your program. Lines 50 and 60 change the foreground color (controlled by the two-letter variable FC) and the background color (controlled by the two-letter variable BC) and line 70 plays a tune at the end. Notice that line 70 is now longer than the screen. Just continue entering numbers. The computer will scroll the entire screen up one line to make room for the new bottom line automatically whenever the maximum of 11 lines of printing is already on the screen. Enter a multiplication sign (on the key marked "NEXT") as shown below with (x).

```
50FC = 7
60BC = 0
70PRINT "100300500 x 100000
0500 x 100000000"
```



The improved program should play a tune at the end. Try it!

You will learn all about colors in Lesson 8, and music is explained in Lesson 5. The remaining lessons are no more difficult than the one you have just completed, so feel free to skip ahead to whatever subject is most interesting.

LESSON 2

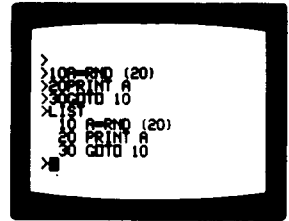
RANDOM NUMBERS, INPUTS AND WHAT IF?

It's often handy to have your computer pick out numbers at random. Here's a program that selects random numbers between one and twenty.

RESET

```
10A = RND(20)
20PRINT A
30GOTO 10
```

GO
LIST
GO

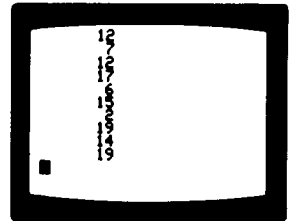


```
>
>10 A=RND (20)
>20 PRINT A
>30 GOTO 10
>LIST
10 A=RND (20)
20 PRINT A
30 GOTO 10
>
```

In line 10 the computer will set the variable A equal to a random number between one and twenty. In line 20 the computer prints the value of A. Line 30 sends the computer back to line 10. The computer continues picking a random number, printing it, and looping back to the beginning of the program.

The numbers this program selects are different each time, so don't expect your numbers to match the example.

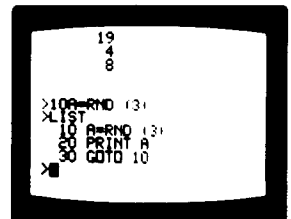
RUN
GO



```
19
4
8
```

Now change line 10 to put random numbers from one to three into the variable A.

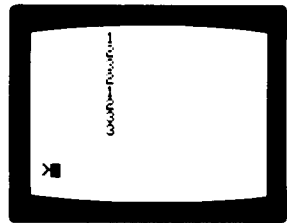
H
10A = RND (3)
GO
LIST
GO



```
19
4
8
>10 A=RND (3)
>LIST
10 A=RND (3)
20 PRINT A
30 GOTO 10
>
```

Now run your program and let it list a few numbers.

RUN
H

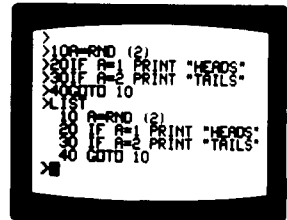


IF is a computer word that lets you check and see whether or not an expression is true. Enter this program:

RESET

```
10A = RND (2)
20IF A = 1PRINT "HEADS"
30IF A = 2PRINT "TAILS"
40GOTO 10
```

GO
LIST
GO



First, the computer picks a random number that is either 1 or 2 and sets the variable A equal to the number picked. If A = 1, the computer prints "HEADS," and if A = 2, the computer prints "TAILS." Then the computer goes back to line 10 again, and the loop continues.

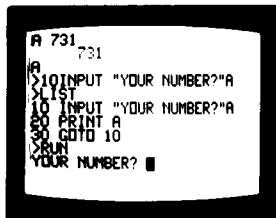
The IF command tests the value of the expressions IF A = 1 and IF A = 2. If either one of these expressions are true, the value of the expression is 1. If the expression is false, its value is 0. When an IF statement is true the rest of the statement will be executed. So IF A is equal to 1 in line 20, the computer will execute the command to print "HEADS." IF A is equal to any number except 1, the expression is false with a value of 0. The rest of the statement will be skipped and execution continues at the next statement. The value of any expression (1 or 0, true or false) can be tested with a simple immediate command:

PRINT A = 3

GO

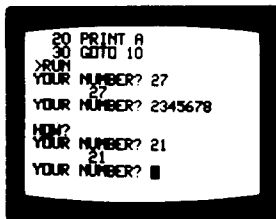
The computer prints "A" to remind you that your input will be stored in the variable A. You can have the computer remind you in other ways, too. Try this change in the program:

```
H
10INPUT "YOUR NUMBER?"A
GO
LIST
GO
RUN
GO
```



Now enter these numbers:

```
27
GO
2345678
GO
21
GO
```



The largest number your computer's memory can hold is 32767. You just saw what happens when you input a number larger than that. The computer will ask WHAT? when it doesn't understand you. It will ask HOW? when it understands but can't do what you requested.

You have been using INPUT to set a variable equal to the number you type. This next program uses two variables and inputs numbers for A and B.

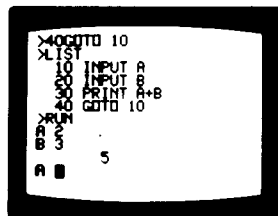
RESET

```
10INPUT A
20INPUT B
30PRINT A + B
40GOTO 10
GO
LIST
GO
```



The variable A is set equal to the first number you enter and the variable B is set equal to the second number. These two numbers are stored in the variables A and B. The computer prints their sum (A + B) and loops back to the beginning of your program. Try adding these numbers together then try some of your own.

```
RUN
GO
2
GO
3
GO
```

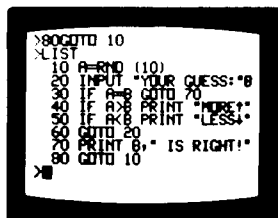


Input lets you put numbers into the computer and RND has the computer pick numbers at random. Now you can combine these and build a guessing game.

```
RESET
```

```
10A = RND(10)
20INPUT "YOUR GUESS:"B
30IF A = BGOTO 70
40IF A >BPRINT "MORE"
50IF A <BPRINT "LESS"
60GOTO 20
70PRINT B,"IS RIGHT!"
80GOTO 10
```

```
GO
LIST
GO
```



This program is longer than your others so we'll look at it step-by-step. First the computer picks a random number between one and ten and stores it in A. When you try to guess the number, your input is stored in B.

Now there are three things that can be true. If $A = B$ then your guess is right. The computer goes to line 70 and prints your answer and the words "IS RIGHT!" If A is larger than B, $A > B$, then your guess is too small and the computer prints "MORE." If A is smaller than B, $A < B$, then your guess is too big and the computer prints "LESS."

There are two loops in this program. If $A = B$ the computer goes to line 70, prints the number you picked and the words "IS RIGHT!" and then loops back to the beginning to start a new game. If you didn't get the right answer the computer loops back to line 20 so you can try again.

HERE'S HOW A SAMPLE RUN MIGHT LOOK:

```
RUN
GO
2
GO
4
GO
5
GO
7
GO
6
GO
```

```
YOUR GUESS: 2
MORE↑
YOUR GUESS: 4
MORE↑
YOUR GUESS: 5
MORE↑
YOUR GUESS: 7
LESS↓
YOUR GUESS: 6
6 IS RIGHT!
YOUR GUESS: █
```

You can change line 10 to `A=RND (100)` and make the game harder, or add a counter to keep track of the number of guesses it took. Any of the words inside the quotation marks, like "MORE" can be changed to say whatever you want. Before you try your game on your friends, learn how to win every time. When the computer asks for your guess, just enter the letter A.

Here's a program add-on that you will like. Enter the line numbers as shown and the computer will add them to the program in the right order.

```
N
80NT = 20
90PRINT "60605 - 5 - 504 - 5 - 50"
100NT = 2
110GOTO 10
GO
LIST
```

```
30 IF A=B GOTO 70
40 IF A>B PRINT "MORE!"
50 IF A<B PRINT "LESS!"
60 GOTO 20
70 PRINT "B," IS RIGHT!"
80 NT=20
90 PRINT "60605-5-504-5-
50
100 NT=3
110 GOTO 10
>
```

Now try the guessing game again and be ready for a surprise when you get the answer right!

```
RUN
GO
```

```
MORE↑
YOUR GUESS: 10
LESS↓
YOUR GUESS: 5
MORE↑
YOUR GUESS: 8
LESS↓
YOUR GUESS: 6
6 IS RIGHT!
60605-5-504-5-50
YOUR GUESS: █
```

LESSON 3

SUBROUTINES

It can be very helpful to create a subroutine whenever you wish to repeat an action several times in a program. Any instruction or group of instructions can be used. These instructions are "called" in the program with the computer word GOSUB. When the computer reads the instruction GOSUB 1000, for example, it transfers to line 1000 and follows the instructions listed there. When the computer reads the word RETURN, it returns to the instruction following the GOSUB command.

This program prints the words ROCK, SHEARS and PAPER several times. To avoid having to write these same print instructions over and over, we will use GOSUB and RETURN to create a subroutine.

These lines form the first part of the program.

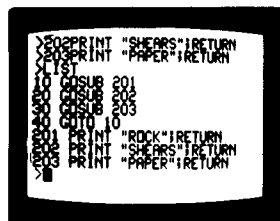
RESET

```
10GOSUB 201
20GOSUB 202
30GOSUB 203
40GOTO 10
201PRINT "ROCK";RETURN
202PRINT "SHEARS";RETURN
203PRINT "PAPER";RETURN
```

GO

LIST

GO



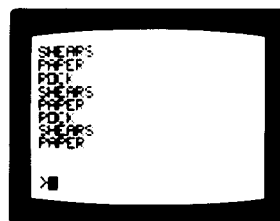
Here is what will happen. When the computer reads line 10, it will jump to the subroutine at line 201 and continue until it reaches the word RETURN. Then the computer will return to line 20 and continue. The computer will also go to subroutines as directed in lines 20 and 30, returning when it reads the word RETURN.

In line 40, the GOTO instruction tells the computer to go back to line 10 and start the program over again. Now RUN this part of your program.

RUN

GO

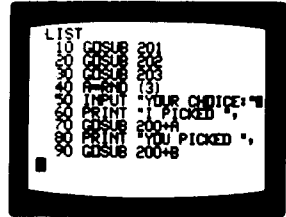
H



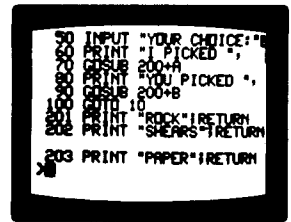
When you add the following lines to your program it will be too long to fit on the screen, but it will scroll up after the maximum of 11 lines of 26 characters occupy the screen, allowing you to keep typing on the bottom line.

```
40A = RND (30)
50INPUT "YOUR CHOICE:"B
60PRINT "I PICKED",
70GOSUB 200 + A
80PRINT "YOU PICKED",
90GOSUB 200 + B
100GOTO 10
```

```
GO
LIST
GO
```



After you have entered these additional instructions, list the program. While it is being listed on the screen, press PAUSE to stop the listing so you can read it. To continue the listing, press any key. A feature called SCROLL MODE CONTROL has been built into your Bally BASIC to allow more user control of the scrolling process which occurs when printing reaches the bottom of the screen.



SCROLL MODE CONTROL

The two-letter SCROLL MODE variable is SM, which can be set to any one of five values:

- SM = 0** Scrolls conventionally.
- SM = 1** Suppresses scrolling. Cursor stays at bottom.
- SM = 2** Holds cursor at screen bottom, clears after each carriage return.
- SM = 3** Clears the screen and resets cursor to screen top.
- SM = 4** AUTO-PAUSE. Allows the listing to fill the screen to the bottom line,

at which time the computer will go into the pause mode until you touch any key. Then the screen will clear and start printing from the top line down, automatically pausing again when the screen is full. This is the ideal mode for viewing your program listings "one page at a time."

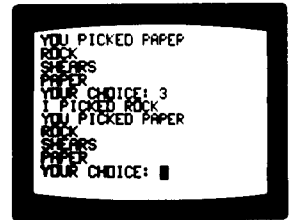
Going back to our program, here's what you've added. In line 40 the computer will select 1, 2, or 3 at random and set the variable A to this value. In line 50 the computer will ask for your choice (1, 2, or 3) and set B equal to the number you input.

Line 60 prints "I PICKED" and line 70 goes to a subroutine at line number 200 + A. If A = 1 the computer will GOSUB to line 201. If A = 2, it will GOSUB to line 202. And if A = 3, it will GOSUB to line 203. Depending on the value of A, "ROCK," "SHEARS," or "PAPER" will be printed after the words "I PICKED." And the comma ending lines 60 and 80 tells the computer to keep printing on the same line.

Lines 80 and 90 use the same GOSUB feature to print your selection. Line 100 loops the program back to the beginning.

RUN your program and INPUT 1, 2 or 3 to select ROCK, SHEARS, or PAPER.

```
RUN
GO
1
GO
2
GO
3
GO
```

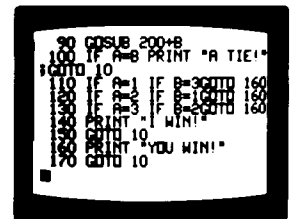


Now you can play ROCK, SHEARS, PAPER with your computer. The rules are:

ROCK breaks SHEARS
SHEARS cut PAPER
PAPER wraps ROCK

You can add more instructions to the program so that the computer will tell you who won. Halt the program and add that feature with these lines:

```
H
100IF A = BPRINT "A TIE!";GOTO 10
110IF A = 1IF B = 3GOTO 160
120IF A = 2IF B = 1GOTO 160
130IF A = 3IF B = 2GOTO 160
140PRINT "I WIN!"
150GOTO 10
160PRINT "YOU WIN!"
170GOTO 10
```



```
GO
LIST
GO
```

Use the PAUSE button to stop the list so you can check it.

```
GO
```



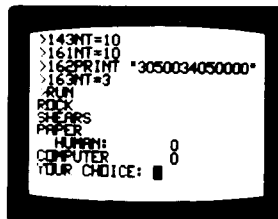
If you would like your computer to keep score, just add these lines. The computer will place them in your program automatically.

```
R
6H = 0
8C = 0
32PRINT "HUMAN:",H
34PRINT "COMPUTER:",C
145C = C + 1
165H = H + 1
```

If you want to add music, these instructions will do it.

```
141NT = 10
142PRINT "135 x 105 x 10000"
143NT = 2
161NT = 10
162PRINT "3050034050000"
163NT = 2
```

Now RUN your program and see if you can beat your computer.



```
RUN
GO
```

Here is a complete listing of your ROCK, SHEARS AND PAPER game.

```
1 .ROCK/SHEARS/PAPER
2 .BY DICK AINSWORTH
10 H = 0
20 C = 0
30 GOSUB 301
40 GOSUB 302
50 GOSUB 303
60 PRINT " HUMAN:",H
70 PRINT "COMPUTER:",C
80 A = RND (3)
90 INPUT "YOUR CHOICE:"B
100 PRINT "I PICKED ",
110 GOSUB 300 + A
120 PRINT "YOU PICKED ",
130 GOSUB 300 + B
140 IF A = BPRINT "A TIE!";GOTO 30
150 IF A = 1IF B = 3GOTO 240
160 IF A = 2IF B = 1GOTO 240
170 IF A = 3IF B = 2GOTO 240
180 PRINT "I WIN!"
190 NT = 10
200 PRINT "135 x 105 x 10000"
210 NT = 3
220 C = C + 1
230 GOTO 30
240 PRINT "YOU WIN!"
250 NT = 10
260 PRINT "3050034050000"
270 NT = 3
280 H = H + 1
290 GOTO 30
301 PRINT "ROCK";RETURN
302 PRINT "SHEARS";RETURN
303 PRINT "PAPER";RETURN
```

```

6 H=0
  C=0
10 GOSUB 201
20 GOSUB 202
30 GOSUB 203
32 PRINT "HUMAN!";H
34 PRINT "COMPUTER";C
40 R=INT (3)
50 INPUT "YOUR CHOICE:";B
60 PRINT "I PICKED ";

```

```

143 NT=3
  C=C+1
  GOTO 10
  PRINT "YOU WIN!"
  NT=10
  PRINT "3050034050000"
  NT=3
  H=H+1
  GOTO 10
201 PRINT "ROCK";RETURN

```

```

70 GOSUB 200;A
80 PRINT "YOU PICKED ";
90 GOSUB 200;B
100 IF A=B GOTO 10
110 IF A=1 IF B=3 GOTO 120
120 IF A=3 IF B=1 GOTO 130
130 IF A=2 IF B=2 GOTO 140
140 PRINT "I WIN!"
141 NT=10
142 PRINT "135*105*10000"

```

```

202 PRINT "SCISSORS";RETURN
203 PRINT "PAPER";RETURN

```

Here is another program that uses subroutines to play each verse. There is no advantage to using subroutines in a small program like this, but it shows how subroutines work. To enter this program into your computer, press RESET to erase the previous program and enter these instructions:

```

10GOSUB 100
20GOSUB 100
30GOSUB 200
40GOSUB 300
50GOTO 10
100PRINT "1456";RETURN
200PRINT "6 + 665";RETURN
300PRINT "4654";RETURN

```

When you run this program, you will hear a song with a separate subroutine for each verse. Try typing NT = 15 before running the program to slow the music down.

LESSON 4 ARRAYS

It's often handy to be able to work with a sequence of numbers or letters known as arrays. Arrays can be made up of numbers, letters or musical notes. The numbers stored in an array can, in fact, stand for anything you choose. Arrays are a common method of storing names and addresses, inventory information, and similar data. The advantages of using arrays are that they save memory space and are organized numerically for simple retrieval of stored information.

The concept of an array can be understood by using the Post Office as an analogy. Your Post Office serves many mailboxes, each with a different number, or address. Each mailbox can contain a different message, or, some may contain nothing. Similarly, an array has many addresses, or elements, each of which can contain a separate number. You can command your computer to change the number in a given address of an array, or to fetch the number for use in calculations.

Every character on your keypad occupies one byte of memory when you enter it into a statement. It follows that a number such as -25000 would require six bytes of memory to store as a number, including the minus sign. However, if you store that same number in an array it occupies only two bytes of memory and is easily retrieved by using the array number you stored it in, just as you would use a letter-variable.

Here's how arrays work in your computer. The @ and * characters are your computer's symbols for arrays. There are 1800 bytes of usable programmable memory in Bally BASIC. With every byte of program you store, that number decreases by one byte. After you finish typing in a program, ask the computer to tell you how many bytes of storage you have left over.

PRINT SZ

GO

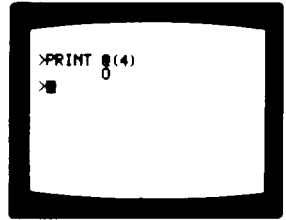
The number that prints on the screen will be the remaining memory size, or "SZ." The @ array begins where this leftover memory space starts at the end of the stored program. The first item is at address 0, or @(0). The second item in the array is at address 1, or @(1). The third item is at address 2, or @(2) and so on. As mentioned before, each item in an array occupies two bytes of memory, so if SZ = 100, for example, you have enough room to store SZ + 2, or an array up to 50 items long. Since the @ array begins where the program ends in the memory, it's important to remember that if you modify your program and make it longer, you will cover up the data you had stored in the memory area your program expanded into. To avoid this occurrence there is a second array symbol, the * array, which refers to the free memory area in reverse order with *(0) starting at the end of the memory and working backwards towards the end of the stored program. The same rules apply as with the @ array with two bytes used per item. So an SZ of 100 would indicate a maximum storage capability of 50 items numbered *(0) thru *(49). Don't neglect to count the zero as a number when using arrays. It will always be your first useable array address.

To find the number stored at address 4 in an array, you can tell the computer to print the value of @ (4) like this:

```

RESET
PRINT @ (4)
GO

```

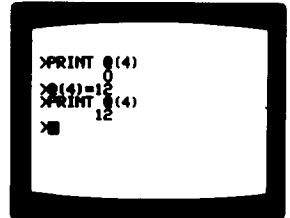


When you RESET the computer, all 1800 memory addresses are cleared and filled with zeroes. So address 4 in the array contains a zero. Store the number 12 at address 4 like this, then check it.

```

@ (4) = 12
GO
PRINT @ (4)
GO

```

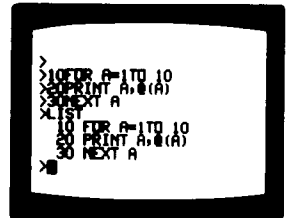


The following program uses a FOR/NEXT loop to list the numbers stored at the first ten addresses in the @ array.

```

RESET
10FOR A = 0 TO 9
20PRINT A, @ (A)
30NEXT A
GO
LIST
GO

```

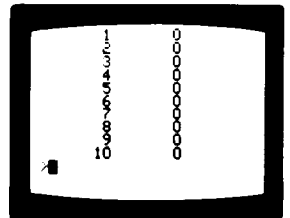


As the variable A advances from 0 to 9, the computer prints 0, and the number stored at address 0, 1 and the number stored at address 1 and so on up to address 9, which contains the last value stored in the array.

```

RUN
GO

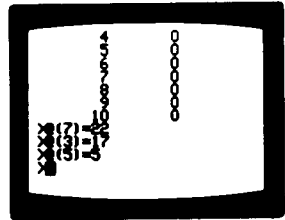
```



Now enter these instructions. Each time you press GO, the computer follows your instruction, storing number 22 at array address 7, number 17 at array address 3, and number 5 at array address 5.

```

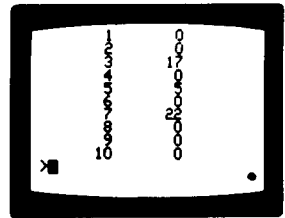
@ (7) = 22
GO
@ (3) = 7
GO
@ (5) = 5
GO
    
```



Now run your program and see what numbers are stored at the first 10 array addresses.

```

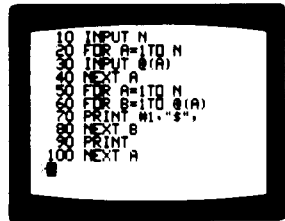
RUN
GO
    
```



This next program prints a simple graph, using the array to store the numbers to be plotted.

```

RESET
10 INPUT N
20 FOR A = 1 TO N
30 INPUT @ (A)
40 NEXT A
50 FOR A = 1 TO N
60 FOR B = 1 TO @ (A)
70 PRINT #1, "$",
80 NEXT B
90 PRINT
100 NEXT A
GO
LIST
GO
    
```



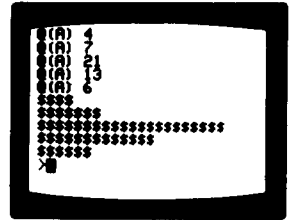
What was the meaning of the # symbol in line 70? It is a formatting symbol that tells your computer to leave a certain number of spaces before printing a character. For a complete explanation see Appendix H under PRINT #A,B.

In line 10, you will set N equal to the number of items in the graph. The loop in lines 20, 30, and 40 stores the value of each item. Lines 50 through 100 are a FOR/NEXT loop using the variable A. This loop prints each item in the graph.

Lines 60, 70 and 80 are a smaller loop that "counts" each item. For example, if @(3) is a 6, this loop will cycle six times and print \$\$\$\$\$\$ on the screen.

Run your program and draw a bar graph. Enter the number of items, then the value of each item.

```
RUN
GO
5
GO
4
GO
7
GO
21
GO
13
GO
6
GO
```



STORING TEXT IN ARRAYS

At the beginning of this lesson you learned that one of the most useful applications of arrays was to store letters and numbers, or, to use a well-known computer expression, "alpha-numeric characters." You know now that your computer doesn't understand words and letters except the keypad command words. To store words in arrays we have to store them as a sequence of numbers that your computer can translate back into letters. Every character and command word on your keypad has a numeric equivalent stored in your computer. These numbers conform to a standard agreed upon by most computer manufacturers called *American Standard Code for Information Interchange*, or ASCII.

To show you which numbers are assigned to which characters we have included an ASCII Character Code Table in APPENDIX A for easy reference. But there is a way to ask your computer to display these codes with two functions built-in to your Bally BASIC.

THE KP AND TV FUNCTIONS

"KP" stands for KEYPAD, and is an expression of the ASCII numerical equivalent of every key on your keypad. "TV" is a function that prints the character the ASCII number stands for when you tell the computer where to look for it, whether in a variable, an array or straight from the keypad when a key is depressed. The following short program will demonstrate how KP and TV can work together to give you ASCII code information about any character.

```
RESET
```

```
10A = KP:PRINT A:RUN
```

KP works like an input statement because it tells the computer to stop running until it receives a number from you. The difference is that KP takes the first digit, stores it in a variable (A in this program), and begins running the program again without a GO command. The number stored in the variable A is the ASCII code for the key you pressed.

Here is what your computer does:

The program tells the computer to set A equal to the ASCII code for the key you pressed. The next instruction tells the computer to PRINT the number stored in A, and the next instruction tells the computer to RUN, or to go back to the first line in the program, the only line in this case, and begin executing instructions there. GOTO 10 would have done the same thing, but would have used 3 bytes. RUN uses only 1 byte, as do all the other keypad command words. As you may have noticed, this short program has three commands packed into one line. You can do this by using the semicolon to separate commands. This saves memory because each line number uses two bytes and each carriage return (GO) uses one byte. The semicolon uses only one byte and does the same thing. You are limited to 104 bytes per line, including two bytes for the line number and one byte for the carriage return. Remember, when an IF statement is false, all commands following it on the same numbered line are ignored by the computer, which goes on to execute the next line in the program.

To retrieve the information stored in ASCII code and display it as the character it stands for, just tell the computer to set the TV function equal to A. Change the program to print the ASCII code for the key you press plus the character it stands for:

```
10A = KP;TV = A;PRINT A;RUN
```

This program will display the code and the character for any number, letter or computer command word. Then it will wait for your next KP input and do it all over again.

You are now probably saying to yourself, "All I have to do to store text in arrays is to tell the computer to store the ASCII code numbers in consecutive array addresses with the KP function. Then to retrieve the information, I'll just tell the computer to set TV equal to the numbers stored in each address, using a loop that counts up to the number of characters stored."

Okay, if you're ready let's give it a try. Let's use a short loop that will store up to 10 characters or command words as you might use in a word game.

RESET

```
10PRINT "INPUT YOUR MESSAGE";FOR A = 0 TO 9
20@(A) = KP
30TV = @(A)
40NEXT A
50CLEAR ;PRINT "TO READ MESSAGE PRESS GO";IF KP#13GOTO 50
60FOR A = 0 TO 9
70TV = @(A)
80NEXT A;PRINT ;RUN
```

Now, run the program and put in any 10 letter word. If the word is less than ten letters, fill the rest with spaces. To see your message retrieved and printed on the screen just press GO. After reading your message you can type in a new one because line 80 tells the computer to RUN the program again automatically.

This program uses two FOR/NEXT loops. The first one stores the ASCII codes in consecutive array addresses. The second loop looks into the addresses and sets TV equal to each of them to reprint the message on the screen. Notice the "#" in line 50. It is the sign meaning "not equal to," which told the computer that if you press any key except GO, or ASCII number 13, go back to line 50 and wait for another input.

In the next lesson you will see how arrays can be used to store and play back musical notes.

LESSON 5

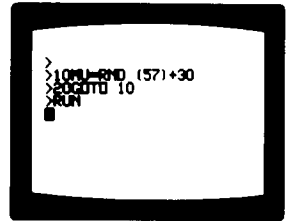
ELECTRONIC MUSIC

There are four ways you can play music on your computer. The PRINT instruction can be used to play any tune you like. You can also use the MU instruction to convert numbers directly into notes without printing on the screen. The third method involves programming the music synthesizer ports &(16) thru &(23), either directly or through the new Bally BASIC sound synthesizer device variables, and is described on page 114.

The following program sets MU equal to a random number between 31 and 87. Numbers in this range produce musical notes in your TV speaker.

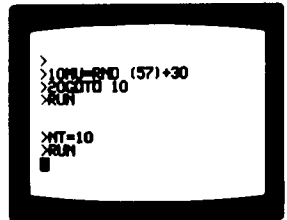
Enter and RUN this random music generator.

```
RESET
10MU = RND (57) + 30
20GOTO 10
GO
RUN
GO
```



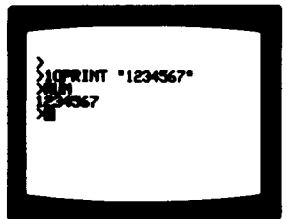
To change the speed of the notes, adjust the built-in note timer controlled by the two-letter variable, NT. HALT your program and set the note time to 10.

```
R
NT = 10
GO
RUN
GO
```

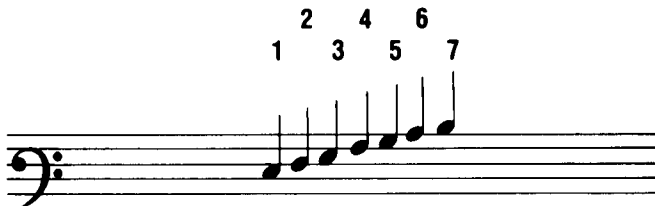


With PRINT and the numbers 1 through 7 you can play a musical scale. The note timer automatically returns to 2 whenever you press RESET.

```
RESET
10PRINT "1234567"
GO
RUN
GO
```



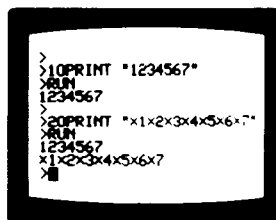
Here are the notes you just played:



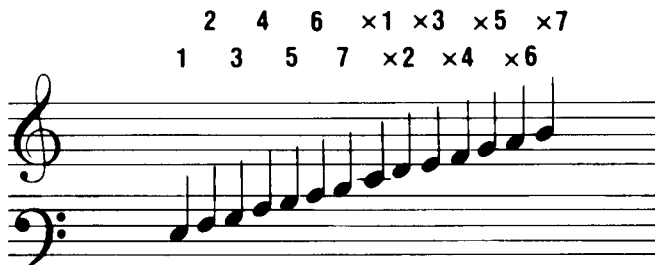
To expand this scale one octave higher, just put a multiplication sign in front of each number.

```
20PRINT "x1x2x3x4x5x6x7"
```

```
GO  
RUN  
GO
```



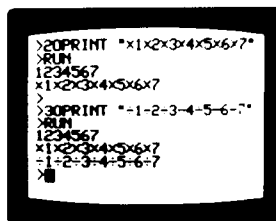
Your program now plays these notes:



Now add the lowest octave and play your computer's full musical scale. Put the division sign in front of the numbers 1 through 7.

```
30PRINT "+1+2+3+4+5+6+7"
```

```
GO  
RUN  
GO
```

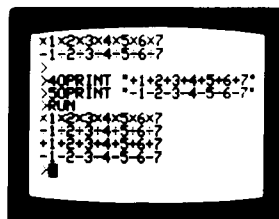


Your computer's complete musical scale is now:

÷2	÷4	÷6	1	3	5	7	×2	×4	×6	
÷1	÷3	÷5	÷7	2	4	6	×1	×3	×5	×7

Sharps are selected by using an addition (plus) sign in front of the notes, and flats are shown with a subtraction (minus) sign.

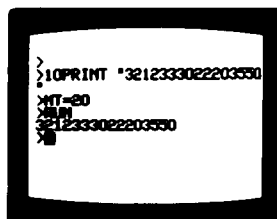
```
40PRINT "+1+2+3+4+5+6+7"
50PRINT "-1-2-3-4-5-6-7"
GO
RUN
GO
```



Always put the sharp or flat sign in front of the octave sign, like this: - +2 or + ×4.

Now RESET the computer and play this tune. Slow the music down by making the note time equal to 20.

```
RESET
10NT =20;PRINT "3212333022203550"
20NT =0
GO
RUN
GO
```



Notice we have set NT equal to zero in the last command in this program. This is to eliminate the long discordant note resulting when the cursor appears as the program stops.

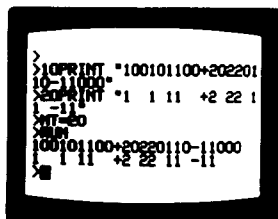
Rhythm can be added two ways. You can space between notes, or add a 0, depending on the sound you want. Try these examples and hear the difference.

We will use the ■ symbol to indicate a SPACE.

```

RESET
10NT = 20;PRINT "100101100 + 20220110 - 11000"
20PRINT "1 ■ 1 ■ 11 ■ ■ + 2 ■ 22 ■ 11 ■ - 11 ■ ■ ■"
30NT = 0
GO
RUN
GO

```



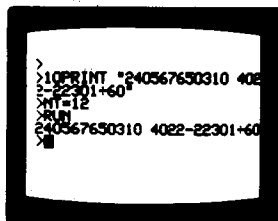
Notice that the notes hold or continue when you use a 0. The space key makes a rest. RUN this program again if you want to listen to the difference.

This next program combines everything you have learned. Notice how the space and the 0's set the rhythm.

```

RESET
10NT = 12;PRINT "240567650310 ■ 4022 - 22300 + 60"
20NT = 0
GO
RUN
GO

```

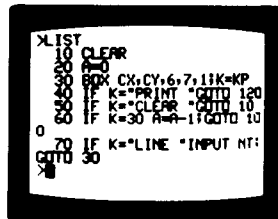


Now build a player piano that stores an entire song and then plays it back. You will enter this program in two sections so it will be easier to check.

```

RESET
10CLEAR
20A = 0
30BOX CX,CY,6,7,1;K = KP
40IF K = "PRINT "GOTO 120
50IF K = "CLEAR "GOTO 10
60IF K = 31A = A - 1;GOTO 100
70IF K = "LINE "INPUT NT;GOTO 30
GO
LIST
GO

```

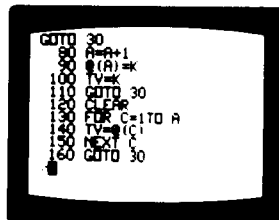


NOTE: See Appendix H for explanation of CX and CY functions shown in line 30.

Compare your program with the example, correct any errors, and then enter the second section.

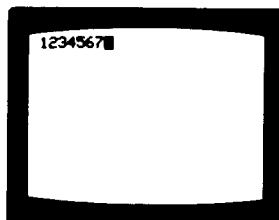
```
80A = A + 1
90@(A) = K
100TV = K
110GOTO 30
120CLEAR
130FOR C = 1TO A
140TV = @(C)
150NEXT C
160GOTO 30
```

```
GO
LIST
GO
```



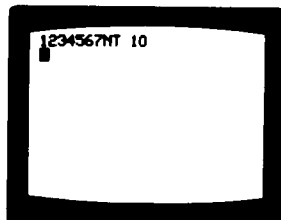
Check your program carefully. When you run it, the screen will go blank. Enter a scale and play it back with the word PRINT.

```
RUN
GO
1234567
PRINT
PRINT
```



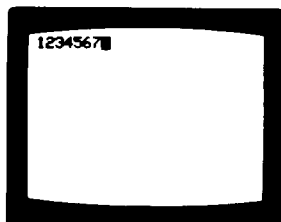
To change the note time, use the word LINE, enter the new note time and press GO.

```
LINE
10
GO
```



With this program the GO key is ONLY used after you enter a new note time. Play back at the new note time, using PRINT as before.

```
PRINT
```



The word CLEAR is used to clear the memory so you can enter a new song. With ERASE you can back up and change any or all of the notes. Now enter this song. The numbers are shown here in groups of four because there are four beats to a measure. Enter the numbers in a continuous line. Do not press GO at the end of each line.

CLEAR

100 + 5
1 + 512
3000
1000
4004
1020
3000
000■

I've been
working on the
rail-
road.
All the
live-long
day.

■ = Space Key

2002
+ 1232
1000
+ 5000
4044
1122
3000
000■

Can't you
hear the whistle
blow-
ing?
Rise up so
early in the
morn.

100 + 5
1 + 512
3000
1033
3020
2030
2000
000■

I've been
working on the
rail-
road. Just to
pass the
time a-
way.

+ 600 + 7
11 + 71 + 6
+ 5000
1000
3040
3020
1000
000■

Can't you
hear the captain
shout-
ing.
Di-na
blow your
horn.

LINE

10

GO

PRINT

If you would like to know more about the Player Piano Program, list it and read the following section.

The variable A keeps track of how many notes are stored in the @ array. After clearing the screen and setting A to 0, the computer draws the cursor box and waits for you to enter a number on the keypad. The variable K is set to this number.

Next, the computer checks to see if any words have been entered. If you enter PRINT, the program goes to line 120 to play back the notes. If you enter CLEAR, the computer goes back to the beginning of the program and sets A to 0. Key 31 is the erase key, and if this is pressed, A is reduced by one. The word LINE is used in this program to input a new number for NT, the note time.

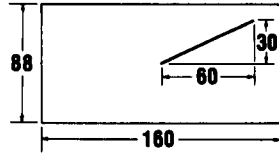
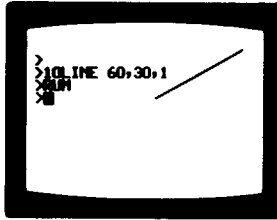
After checking to see if you have entered any special words, the computer adds one to the number stored in A. The new note is added to the @ array (line 90), and is shown on the TV (line 100). GOTO 30 sends the computer back to wait for the next input from the keypad (Line 30).

If PRINT is entered, the computer goes to line 120 and starts the playback process. The screen is cleared, and a FOR/NEXT loop is started. Remember that A keeps track of how many notes there are. This part of the program (lines 130, 140 and 150) loops once for each note until all the notes have been written on the TV and played.

LESSON 6 GRAPHICS

With only the words LINE and BOX you can draw an endless variety of pictures, graphs, and designs on your TV. Here's how LINE works.

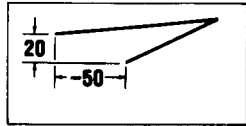
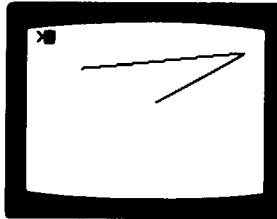
```
RESET  
10LINE 60,30,1  
GO  
RUN  
GO
```



The smallest dot your computer can produce on the screen is called a "pixel," short for "picture element." Your TV screen is 160 pixels wide, (X direction), and 88 pixels high, (Y direction.) Zero is in the center. When you run this program, the computer starts in the center of your screen and draws a line to a point that's 60 pixels to the right of the center (60) and 30 pixels up from the center (30).

After you have run the program, add these instructions to clear the screen and draw the second line.

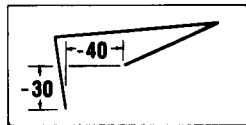
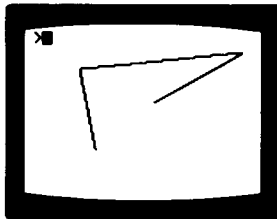
```
5CLEAR  
20LINE -50,20,1  
GO  
RUN  
GO
```



Run the new program and see the computer draw two lines. The computer moved to a point 50 pixels to the left of center (-50) and 20 pixels up from center (20) to draw the second line.

Now add this instruction.

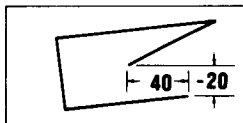
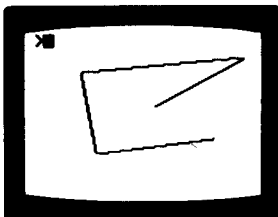
```
30LINE -40,-30,1  
GO  
RUN  
GO
```



The computer moves to a point that is 40 pixels to the left of center (-40) and 30 pixels down from center (-30). Continue drawing in the lower right section of your screen with this instruction that means 40 to the right (40) and 20 down (-20).

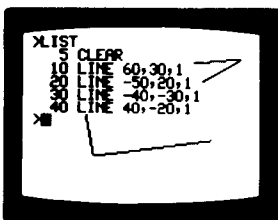
40LINE 40, -20,1

GO
RUN
GO



List your program and check to see that you have all the instructions properly entered.

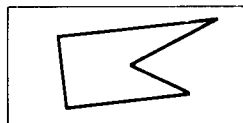
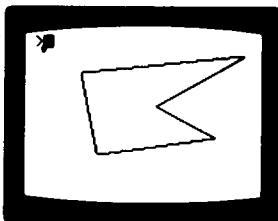
LIST
GO



Finally, draw a line back to the center (0,0) to complete your first graphic design.

50LINE 0,0,1

GO
RUN
GO

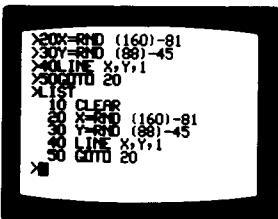


Run this program several times and see how the LINE instruction is used. Erase the program with RESET, and enter the following program that fills the screen with random lines.

RESET

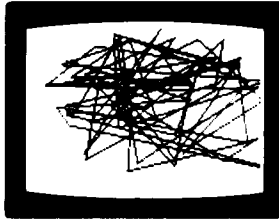
10CLEAR
20X=RND (160) - 81
30Y=RND (88) - 45
40LINE X,Y,1
50GOTO 20

GO
LIST
GO



The computer selects random numbers for X and Y. Then it draws a line to the point on the TV screen that is X pixels right or left of center and Y pixels up or down. It loops back and picks a new X and Y position and then continues drawing.

```
RUN  
GO
```

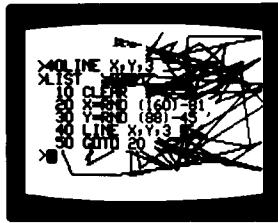


The number 1 after LINE means draw a black line. There are four kinds of lines you can make.

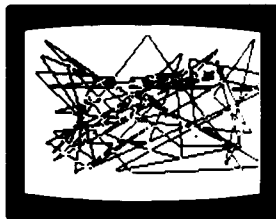
```
LINE X,Y,1 = Black  
LINE X,Y,2 = White  
LINE X,Y,3 = Reverse  
LINE X,Y,4 = None
```

Change line 40 and find out what "reverse" lines are.

```
H  
40LINE X,Y,3  
GO  
LIST  
GO
```

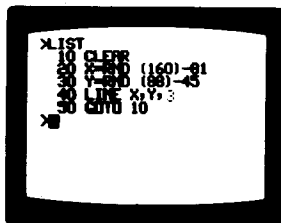


```
RUN  
GO
```

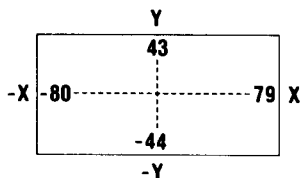


One trick to use: If you draw anything on the screen with color option 3, (as in LINE X,Y,3) you can erase it simply by drawing it again the same way. Halt your program, clear the screen, and list it.

```
H
CLEAR
GO
LIST
GO
```

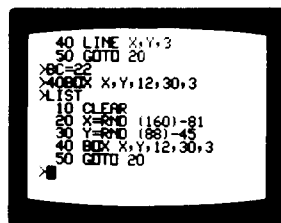


Here's how the computer draws lines that match the size of your TV screen.
 In Line 20, the computer picks a number for X between - 80 (on the far left edge of the screen) and 79 (on the right edge of the screen.)
 In Line 30, the computer selects a random number for Y that is between - 44 (on the bottom edge of your screen) and 43 (on the top edge of your screen.) Bally BASIC will accept line coordinates that go beyond the screen but will print only the portion of the line that falls within the 160x88 pixel area on the screen.



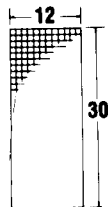
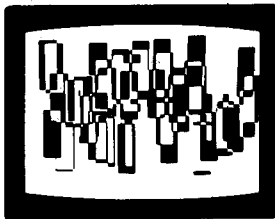
Now change your program and create "reverse" boxes all over your screen. Also change the background color (BC).

```
BC = 22
40BOX X,Y,12,30,3
GO
LIST
GO
```



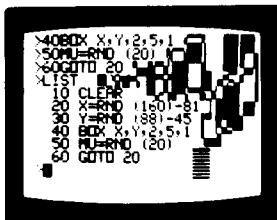
The random numbers X and Y, position the box on the screen. The next two numbers, 12 and 30, tell the computer how many dots wide and tall to make the box. The last number, 3, reverses as before.

RUN
GO

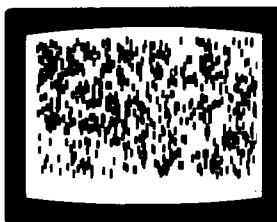


Now make something different. Change the size of the boxes to look like the holes in an IBM card. Change the last number in line 40 to a 1, which will make all the boxes look black. Add some computer music with line 50.

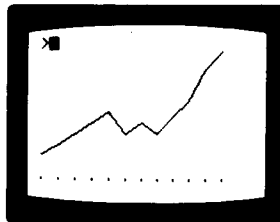
H
40BOX X,Y,2,5,1
50MU = RND (20)
60GOTO 20
LIST
GO
LIST
GO



RUN
GO



This next program draws a graph. First, it asks how many numbers you have, then it asks for each number. Finally, it draws a graph that might look like this.



Enter and LIST this part of the program.

```

RESET
10CLEAR
20INPUT "←-A-▶"A
30FOR N=1 TO A
40PRINT N,
50INPUT "?"@ (N)
60IF @ (N)▶87GOTO 40
70NEXT N
GO
LIST
GO
    
```

```

>60IF @ (N)▶87 GOTO 40
>70NEXT N
XLIST
10 CLEAR
20 INPUT "+A+"A
30 FOR N=1TO A
40 PRINT N,
50 INPUT "?"@ (N)
60 IF @ (N)▶87 GOTO 40
70 NEXT N
    
```

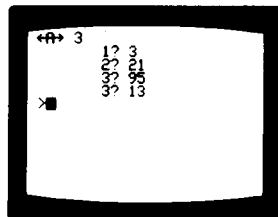
In Line 20, the computer asks how many items the graph will have, then stores the answer in A.

The FOR/NEXT loop prints the number of each item, stores the value in the array @ (N), and checks to see if the value is over 87. If it is over 87, it will not fit on the TV screen and the computer goes back to line 40 for a new input.

Run this portion of the program.

```

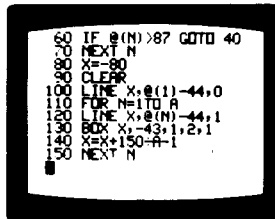
RUN
3
GO
3
GO
21
GO
95
GO
13
    
```



Now add the final section that draws the graph.

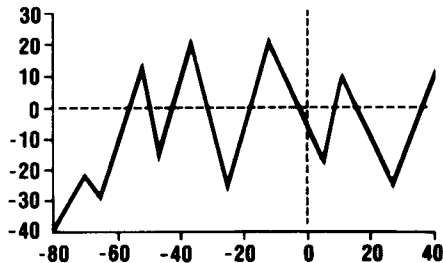
```
80X = -80
90CLEAR
100LINE X,@(1) - 44,0
110FOR N = 1TO A
120LINE X,@(N) - 44,1
130BOX X, - 43,1,2,1
140X = X + 150 + A - 1
150NEXT N
```

```
GO
LIST
GO
```



To start drawing the graph (line 80), the computer sets $X = -80$ (the left edge of your screen), clears the screen, and places the starting point for the series of lines that make the graph.

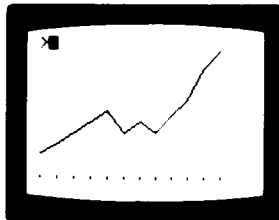
The number $@(1) - 44$ is the vertical distance (number of pixels) above or below the center of the screen. For example, if the first number in the $@$ array is 0, then the computer subtracts 44 to place this point on the bottom of the graph.



There are three instructions (120, 130 and 140) in the last FOR/NEXT loop. These instructions are run once for each item in the graph. In line 120, the computer draws a line from the last point to the next point. Line 130 places a small dot at the bottom of the graph. Line 140 changes the variable X to move each point on the graph a short distance to the right. The graph is 150 pixels wide and this distance is divided equally.

Run the program and draw a graph with these twelve figures. Don't forget to push GO after each number.

RUN
GO
12
15
21
28
35
42
28
35
28
39
49
68
79



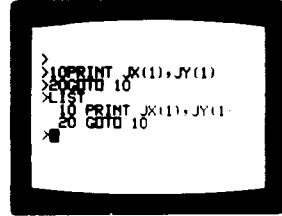
Remember — no single entry can be larger than 87 and no decimal points are accepted.

Use your graph drawing program to make a graph of your grocery expenses, your company sales, or your favorite stock.

LESSON 7 VIDEO GAMES

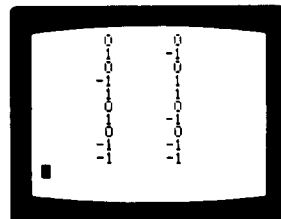
In this lesson you will learn how to build your own video game, using the hand controls. The program will be assembled one piece at a time, so you can see what each section does. First, plug a hand control into the number 1 port (next to the power cord), and then enter this program to see how it works:

```
RESET  
10PRINT JX(1),JY(1)  
20GOTO 10  
GO  
LIST  
GO
```



JX and JY are two-letter variables used by your computer to check the position of hand control joysticks 1 through 4. For example: JX (2)=0 means the joystick on hand control 2 is in the center and has not been pushed either left or right. With the number 1 joystick control centered, JX(1) and JY(1) are zero. Moving the joystick to the right makes JX(1) = + 1, and moving it to the left makes JX(1) = - 1. Similarly, moving the joystick forward or back makes JY(1) either + 1 or - 1. Run the program and change the numbers on your screen by moving the joystick left and right, back and forth. Turning (rotating) the knob has no effect right now.

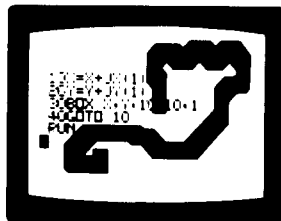
```
RUN  
GO
```



Now use the joystick to move a box on the screen with this program. Two variables, X (left and right) and Y (up and down), keep track of where the box is. When you move the box with the joystick you will be adding +1 or -1 to the variables. Run the program and move the box.

```

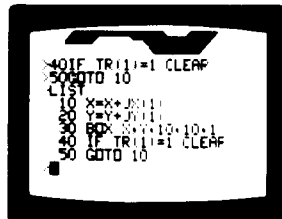
RESET
10X = X + JX(1)
20Y = Y + JY(1)
30BOX X,Y,10,10,1
40GOTO 10
GO
RUN
GO
    
```



The trigger for the first hand control is called TR(1). TR is another two-letter variable that checks the positions of the triggers. TR(1) = 1 when the trigger is pulled and TR(1) = 0 when the trigger is not pulled. Add this line to your program so you can clear the screen by pulling the trigger.

```

H
40IF TR(1)=1CLEAR
50GOTO 10
GO
LIST
GO
    
```



Line 40 checks to see if TR(1) is pulled every time the program cycles. Remember the chapter on IF statements? If line 40 expresses a true statement its value is 1 and the following command, CLEAR, is carried out. If the statement is false its value is 0 and the rest of that line is ignored by the computer, which goes on to the next line. A byte-saving hint--Line 40 could be shortened to read as follows:

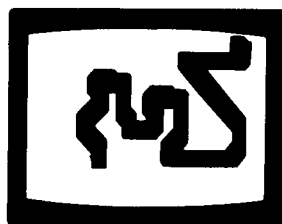
```

40IF TR(1)CLEAR
    
```

The "equals one" expression isn't needed because the statement has a value of 1 if it's true, and the computer knows this. Now run your program, draw some lines, and clear the screen with the trigger.

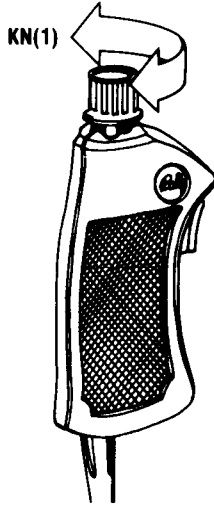
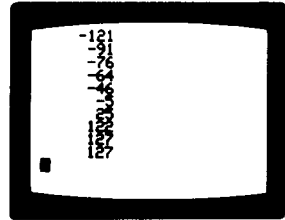
```

RUN
GO
    
```



Now change your program and see what happens when you turn the knob. The knob on hand control number 1 is called KN(1). Run the program and turn the knob.

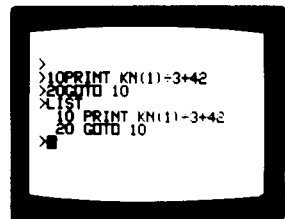
```
RESET
10 PRINT KN(1)
20 GOTO 10
GO
RUN
GO
```



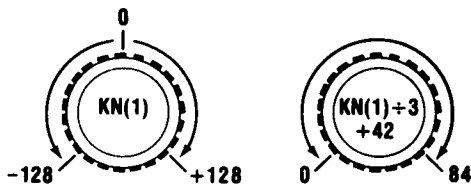
With the knob turned all the way to the left, $KN(1) = -128$. With the knob turned to the right $KN(1) = 127$. Try to dial your age. This is hard to do because the numbers are very close together on the knob.

This next program spreads the numbers out and makes it easier to dial your age.

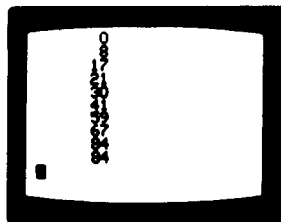
```
RESET
10 PRINT KN(1) + 3 + 42
20 GOTO 10
GO
LIST
GO
```



Here's what you have done to make it easier. KN(1) still has a range from -128 to 127. When you divide KN(1) by 3 this range is reduced to -42 on the left and 42 on the right. When the computer adds 42 to KN(1) + 3, the final range is 0 on the left and 84 on the right. In a similar way you can write an instruction and change the numbers on the dial to match any range you would like. Run this program and see that the knob rotates from 0 to 84.



RUN
GO



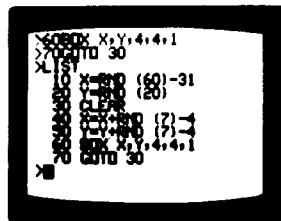
Now you can use the hand control to build your own video game. Begin with this portion of the program that makes a blinking target move around on the screen.

```

RESET
10X = RND (60) - 31
20Y = RND (20)
30CLEAR
40X = X + RND (7) - 4
50Y = Y + RND (7) - 4
60BOX X,Y,4,4,1
70GOTO 30

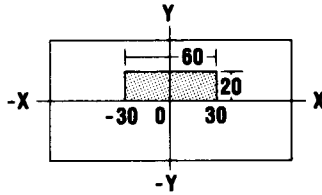
```

GO
LIST
GO



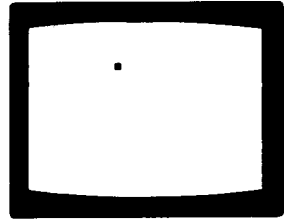
First, the computer picks a value for X between -30 and 30, and then picks a value for Y between 1 and 20. These values for X and Y are in the shaded area of the diagram below. Line 40 and 50 cause the target to wander around the screen. In Line 40, the computer adds a random number to X, moving the target to the right or left. The number added to X is RND (7) - 4. RND (7) is a random number between 1 and 7. Subtracting 4 makes this equal to a random number between -3 and 3.

In Line 50, RND (7) - 4 is added to Y, and this moves the target up or down. The BOX is drawn at X and Y, and the program loops.



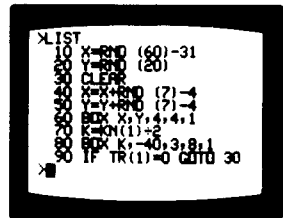
Now run the program and see that it puts a 4 x 4 black box somewhere in the shaded area.

RUN
GO



Add a second box at the bottom of the screen. You will move this box left and right with the knob. Notice that you will replace the old line 70 with a new instruction.

H
70K = KN(1) + 2
80BOX K, -40,3,8,1
90IF TR(1) = 0GOTO 30
GO
LIST
GO



In Line 70, the variable K is set to the value of the knob KN(1) + 2. Line 80 draws a black box that is three squares wide and eight squares tall. The box can be moved left or right as the value of K changes. The center of the box can be moved left or right as the value of K changes. The center of the box will be at -40, near the bottom of your screen.

