Thomas E. Rowley

ATARI BASIC

LEARNING BY USING

Thomas E. Rowley

ATARI BASIC Learning By Using

Editor: Carl M. Evans *Slave Driver*
Assistant Editor: Cindy Hall *Right oar*
Production: Debbie Cooke *Left oar*
Graphics: Don Smith *Rudder*

Printed in the USA
IJG, Inc.
1953 W. 11th Street
Upland, CA. 91786
USA

Thomas E. Rowley

ATARI BASIC Learning By Using

Published simultaneously in the United States of America and Germany

IJG, Inc.
1953 W. 11th Street
Upland, CA. 91786
USA

ISBN 0-936200-35-9
(previously) **ISBN-3-92-1682-86-X**
10 9 8 7 6 5 4 3 2 1

Credits

This book is a result of a project initiated by Sam Calvin. He has been a constant source of information and motivation. Much of the credit goes to him for it's completion.

Thanks to Ron Leckron, Larry Cook, David Robinson and Vincent Senter for their contribution of programs. A special thanks to Melinda, my wife, for her support and enduring the many hours of work needed to complete this project.

Reference is made to *Atari* throughout this book. Atari is a trademark of Atari Inc., a Warner Communications Company. Ted Kahn of Atari is recognized for his support.

Preface

This book has been written to provide a supplementary resource for learning BASIC programming on the *"Atari Personal Computer."* Short programs and learning exercises are intended to motivate and stimulate learning of Atari BASIC programming. "Hands On" interaction with a computer is essential.

Some programs in this book use sophisticated programming techniques, while others are written using simple Atari BASIC statements. Many of these programs are appropriate for beginners as well as experienced computer users. It is recommended that the user of this book has an elementary knowledge of BASIC programming, and an *Atari Basic Reference Manual* available.

Though many routines provided will be useful, it is not the intention of the author to provide a book of finished software products. The routines can be adapted as subroutines to other programs, or expanded to meet the needs of the programmer. The programs in this book will work properly on either an Atari 800 or an Atari 400 computer. 16K of memory is required for some of the routines.

It is hoped that this book will provide the motivation to grow in BASIC programming of the *"Atari Personal Computer."* Much has been learned in the process of writing it.

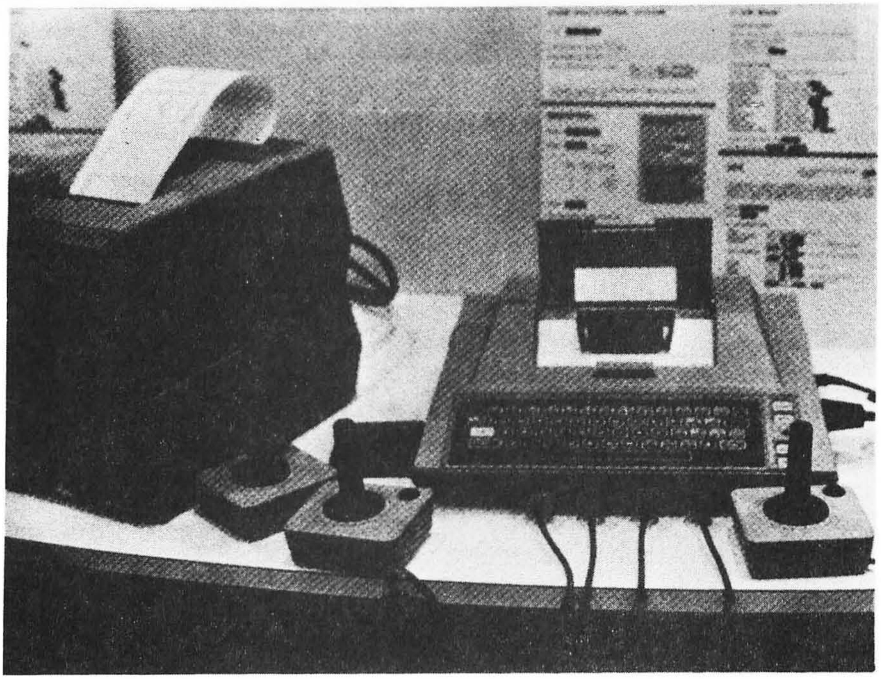
ATARI BASIC Learning By Using

Table of Contents

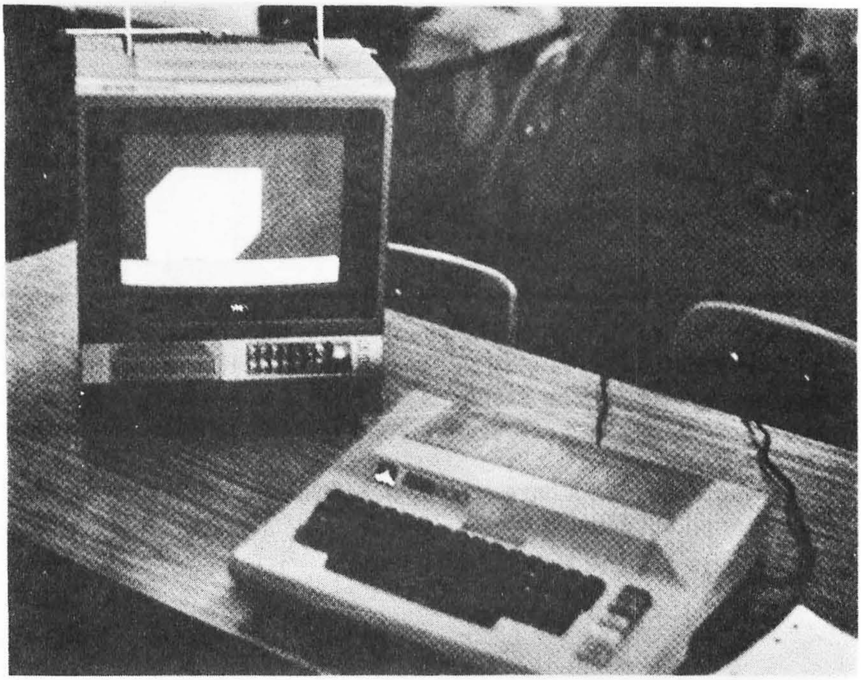
Introduction

This is an "ACTION" book. You do more than read it. You use it, discover with it and create with it. Learn Atari BASIC through the short programs provided. The examples challenge you to change and write programs on your own. They are a great source of instruction for any teacher or student.

Enough words, let's have some "ACTION." Sit down to your Atari computer and **"Learn by Using."**



ATARI 400



ATARI 800

1 Atari

Screen Drawings

This set of programs for "Screen Drawings" provides a good starting point to learn Atari graphics.

Try to become proficient in the use of the following Atari statements used in this section.

COLOR SETCOLOR GRAPHICS PLOT DRAWTO

Refer to your *Basic Reference Manual* for a description of these statements.

German Flag

This routine provides a short example of using COLOR, PLOT and DRAWTO in the GRAPHICS 3 MODE. It's simple because the German flag is so easy to draw.

```
100 REM DRAW THE GERMAN FLAG
110 GRAPHICS 3+16:SETCOLOR 4,7,2
120 SETCOLOR 0,0,0
130 SETCOLOR 1,3,4
140 SETCOLOR 2,13,10
150 REM DRAW FLAG
160 FOR C=1 TO 3
170 COLOR C
180 FOR Y=5*(C-1)+3 TO 5*(C-1)+8
190 PLOT 7,Y:DRAWTO 35,Y
200 NEXT Y
210 NEXT C
220 GOTO 220
```

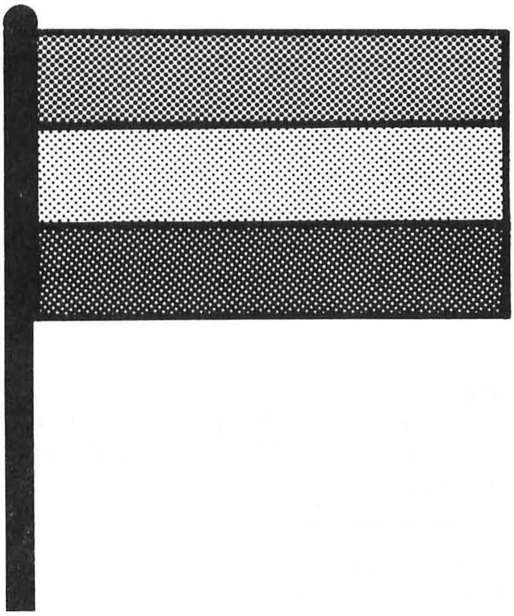
SETCOLOR 0, 1 and 2 sets COLOR 1, 2 and 3 respectively.

Suggestion:

1. Try a different background color. Change line 110 to
   `110 GRAPHICS 2+16:SETCOLOR 4,13,4`

2. Change the colors of the flag in lines 120,130 and 140.
3. Add a nice flag pole with a ball on the top.
4. Draw the flag in GRAPHICS MODE 7.

Figure 1.1 *German Flag*

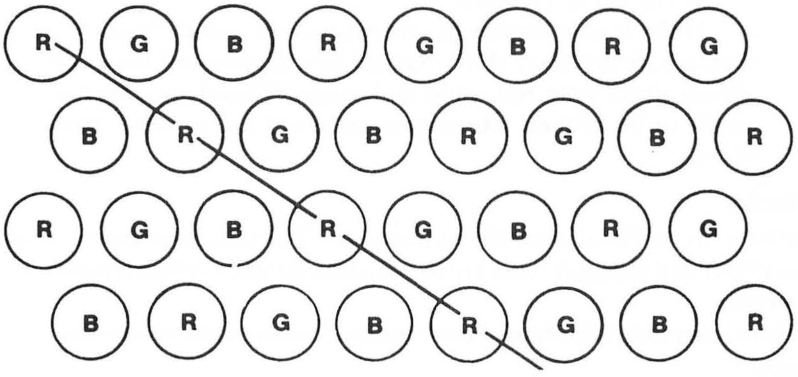


Design

Take a close look at this one. Colors are produced in GRAPHICS 8 by selectively drawing lines on the screen. By taking advantage of the physical separation between color dots on the television screen, the three primary colors can be individually turned ON and OFF.

The line shown in the diagram would be drawn (red). A series of lines drawn adjacent to one another would turn on all of the color dots to produce (white).

Figure 1.2 *Color Dots on the Television Screen*



R=red, B=blue, G=Green

```
10 REM A DESIGN IN GRAPHICS 8
15 GRAPHICS 8+16
20 SETCOLOR 2,13,2:SETCOLOR 1,13,14
25 REM DRAW ENTIRE SCREEN
30 COLOR 1
35 FOR A=0 TO 319 STEP 3
40 PLOT A,0:DRAWTO 319-A,191
45 NEXT A
50 FOR A=191 TO 0 STEP -2
55 PLOT 0,A:DRAWTO 319,191-A
60 NEXT A
65 REM DRAW CENTER OF SCREEN
70 COLOR 0
75 FOR A=80 TO 239 STEP 3
80 PLOT A,122 TO 70 STEP -2
85 NEXT A
90 FOR A=122 TO 70 STEP -2
95 PLOT 80,A:DRAWTO 239,191-A
100 NEXT A
105 GOTO 105
```

Suggestion:

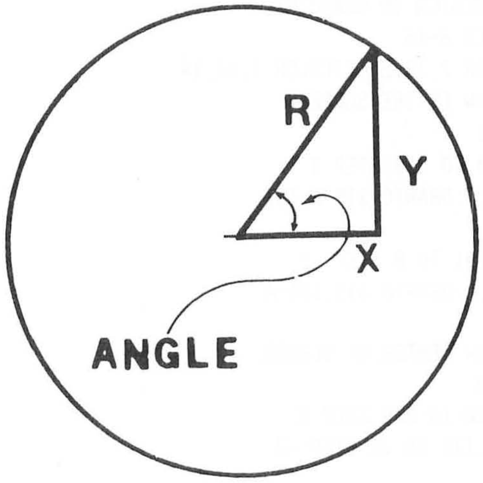
1. Change the color in line 20.
   ```
   SETCOLOR 2,4,2:  SETCOLOR 1,4,14
   ```
2. Change the step in line 35 and 50.
   ```
   35 FOR A=Ø TO 319 STEP 3
   5Ø FOR A=191 TO Ø STEP -2
   ```
3. Make a new design by drawing a new set of lines. For example, draw every third vertical line. Horizontal lines will always draw over red, blue and green. Therefore, the line needs to be broken into segments to get colors other than white.

Circle

Plot a circle. You choose the radius (size). The trigonometric functions sine (SIN) and cosine (COS) are used in line 60 to calculate a position on the circle.

Here is a quick definition of how sine and cosine relate to a position on a circle.

Figure 1.3 *Plot a Circle*



```
    SIN(ANGLE) = Y/R OR Y = R*SIN(ANGLE)
    COS(ANGLE) = X/R OR X = R*COS(ANGLE)
```

```
10 REM  DRAW A CIRCLE
20 R=40:DEG
23 GRAPHICS 7:SETCOLOR 2,4,5:COLOR 1
25 REM  DRAW AXIS
30 PLOT 20,40:DRAWTO 140,40
40 PLOT 80,0:DRAWTO 80,80
45 REM  PLOT CIRCLE
50 FOR ANGLE=0 TO 360 STEP 10
60 X=R*COS(ANGLE):Y=R*SIN(ANGLE)
70 PLOT X+80,Y+40
80 NEXT ANGLE
85 REM  ANOTHER CIRCLE
90 PRINT "ENTER RADIUS(10-40) ":INPUT R
100 GOTO 50
```

Suggestions:

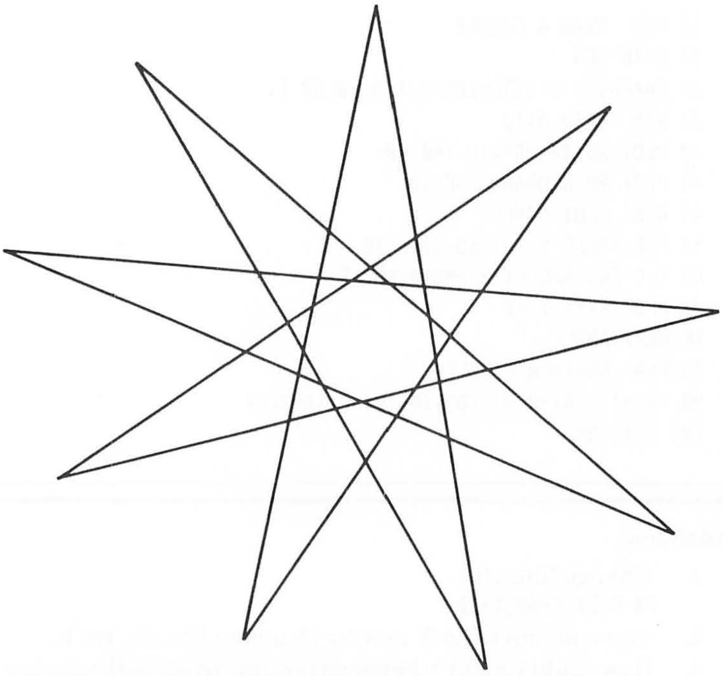
1. Change line 70.
   `70 PLOT X+60,Y+40`
2. You can move the Y axis by changing line 40, try it.
3. How could you fill in between the dots to make the circle a continuous curve?

Star

This program draws a Star of Solomon. Simple changes have some interesting effects on the size and shape. See the program "CIRCLE" for a definition of the SIN and COS functions.

```
10 REM  DRAW A PRETTY STAR
15 DEG :R=50:COLOR 1
20 GRAPHICS 8:SETCOLOR 4,14,0:SETCOLOR 2,12,2
22 REM  DRAW STAR
25 PLOT R+180,80
30 FOR T=0 TO 1800 STEP 160
40 DRAWTO 180+R*COS(T),80+R*SIN(T)
50 NEXT T
55 REM  ANOTHER STAR
60 ? "ENTER RADIUS(10-80)":INPUT R
70 GOTO 25
```

Figure 1.4 *Star of Solomon*



Suggestions:

1. Change line 30.

```
30 FOR T-0 TO 1800 STEP 100
```

2. Can you put several stars at different places on the screen?

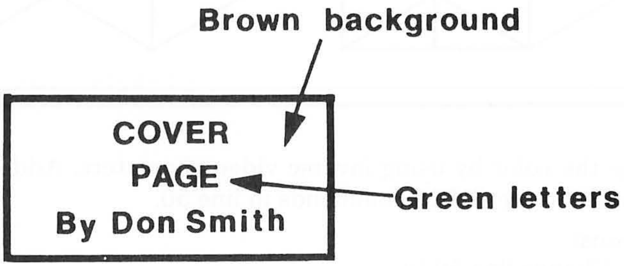
3. Draw several stars, each with a different number of points on the same screen.

Title Page

Put a title page on the front of your programs. It can make a good program look even better. Try this one for practice with large characters in different colors.

```
10 REM  MAKE A NICE TITLE PAGE
20 GRAPHICS 2:SETCOLOR 4,13,2
30 POSITION 5,1:? #6;"###########"
40 POSITION 5,2:? #6;"# program #"
50 POSITION 5,4:? #6;"#  title  #"
60 POSITION 5,4:? #6;"###########"
70 ? "              By the Author"
```

Figure 1.5 *Title Page*



Suggestions:

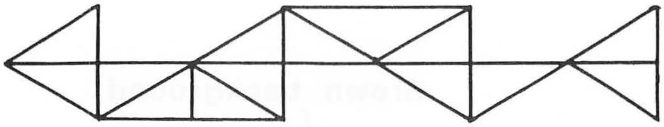
1. Add these lines.
```
25 SETCOLOR 2,13,2
65 POKE 752,1
90 GOTO 90
```
2. Make the title a different color changing the program title in lines 40 and 50 to upper-case or inverse video.
3. Try making a title page in GRAPHICS 1.
4. Make a screen like the one to the right. You can change the color of a character with the SETCOLOR statement.

Symbols in GRAPHICS 2

Make a graphics design on the screen. You can use special symbols in GRAPHIC MODES 1 and 2. Just change the value in the character base to 226 (POKE 756,226).

```
1Ø REM  SYMBOLS IN GRAPHICS 2
2Ø GRAPHICS 2+16
3Ø POKE 756,226
4Ø FOR J=1 TO 8Ø
5Ø ? #6;CHR$(8);CHR$(1Ø);CHR$(1Ø);
6Ø NEXT J
1ØØ GOTO 1ØØ
```

Figure 1.6 *Make a Graphic Design*



Change the color by using inverse video characters. Add 128 to the numbers inside the CHR$ commands in line 50.

Suggestions:

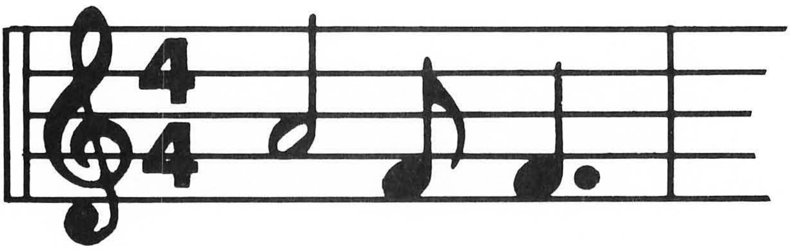
1. Change line 50 to
   `5Ø ?#6;CHR$(8);CHR$(138);CHR$(1Ø);`
2. Try it in GRAPHICS 1.
3. If you liked that, try a new design with some other graphic symbols.

2 Special Sounds

Perhaps you've heard the saying, "A picture is worth a thousand words." Sound also does more than words. Beyond time, it adds a fifth dimension. Sound reaches the subconsciousness of the human mind. Sounds make your programs come alive. Use this section to work with the Atari "SOUND" statement. Vary tone, distortion and volume to produce unique and exciting sounds.

Try some sound effects or a musical tune. Try to integrate some sounds with the "Screen Drawings."

Figure 2.1 *Screen Drawings*



Sound Effects

A variety of sound effects can be created using the SOUND statement. The changing patterns of pitch and volume determine the special quality of the sound. Here are some simple sound effects. First, try the "Dropped Coin."

```
100 REM A DROPPED COIN
105 P=60
110 FOR J=10 TO 0 STEP -0.2
115 REM  INCREASE VOLUME
120 FOR V=1 TO J:SOUND 0,P,10,V:NEXT V
122 REM  DECREASE VOLUME
125 FOR V=2*J TO 1 STEP -1:SOUND 0,P,10,V:NEXT V
130 NEXT J
150 END
```

A decreasing pitch, and a random volume produce a little bit of thunder. Add the lightning and you'll have a real storm.

```
10 REM  THUNDER
15 FOR P=5 TO 100 STEP RND(0)*5+0.2
20 SOUND 0,P,8,(RND(0)*10+5)/(0.1*P)
25 SOUND 1,P+20,8,(RND(0)*10+5)/(0.1*P)
30 NEXT P
50 GOTO 10
```

The volume goes UP and DOWN while the length of each cycle decreases.

Here is a good imitation of a European police siren.

```
10 REM  EUROPEAN SIREN
15 LOW=57:HIGH=45:P=45
20 FOR AGAIN=1 TO 20
30 SOUND 0,P,10,14
40 FOR WAIT=1 TO 180:NEXT WAIT
50 P=LOW:LOW=HIGH:HIGH=P
60 NEXT AGAIN
70 SOUND 0,0,0,0
80 END
```

The pitch alternates between a HIGH and LOW value. The length of each tone is set in line 40.

In this routine a random pitch and volume are generated. Close your eyes and listen. Does it sound like a swarm of flies?

```
10 REM  A SWARM OF FLIES
20 P=INT(RND(0)*6)+250
30 V=INT(RND(0)*4)+6
40 SOUND 0,P,14,V
50 FOR X=1 TO 10:NEXT X
60 GOTO 10
```

Produce a series of intermittent sounds for a noise that sounds like an engine.

```
10 REM  ENGINE
15 SOUND 0,250,10,10
20 FOR J=1 TO 5
30 SOUND 0,0,0,0
40 NEXT J
50 GOTO 15
```

You can call this one random noise, or perhaps space age music.

```
10 REM  RANDOM NOISE
20 P=INT(RND(0)*256)
30 V=INT(RND(0)*16)
40 SOUND 0,P,10,V
50 GOTO 10
```

Suggestions:

1. Modify the "Dropped Coin" to sound like three coins are dropped at once.
2. Make the engine sound like it speeds up.
3. Make a little more thunder.
4. Write a routine for:
   - Footsteps
   - Car Race
   - Machine Gun
   - Telephone
   - A rocket at blast-off
   - Knock on the door

Musical Tune

Listen to the musical tune generated by this routine. A musical note such as G is assigned a number (G=162) which sets the pitch in the

SOUND statement. The data in lines 300 through 320 gives the musical note and the duration. Run it, and see if you recognize the tune.

```
10 REM  MUSICAL TUNE
20 DIM PITCH(7),SCALE$(7),N$(1)
25 REM  SET NOTES ON MUSICAL SCALE
30 FOR P=1 TO 7
32 READ PITCH:PITCH(P)=PITCH
36 NEXT P
40 DATA 144,128,121,108,96,91,162
55 REM  READ NOTES AND DURATION
57 SCALE$="ABCDEFG"
60 READ N$,T
62 REM  "Z" IS END OF TUNE
65 IF N$="Z" THEN 400
70 FOR X=1 TO 7
80 IF N$=SCALE$(X,X) THEN 130
90 NEXT X
120 REM  SET DURATION AND PLAY NOTE
130 T=T*20
135 FOR N=1 TO T
140 SOUND 0,PITCH(X),10,8
150 NEXT N
160 SOUND 0,0,0,0
200 GOTO 60
300 DATA G,1,B,1,C,1,D,4,G,1,B,1,C,1,D,4
305 DATA G,1,B,1,C,1,D,2,B,2,G,2,B,2,A,4
310 DATA B,1,B,1,A,1,G,2,G,1,B,1,D,1,S,1
315 DATA D,1,C,2,C,1,B,1,C,1,D,2,B,2,A,2
320 DATA A,2,G,4,Z,1
400 SOUND 0,0,0,0
410 END
```

Suggestions:

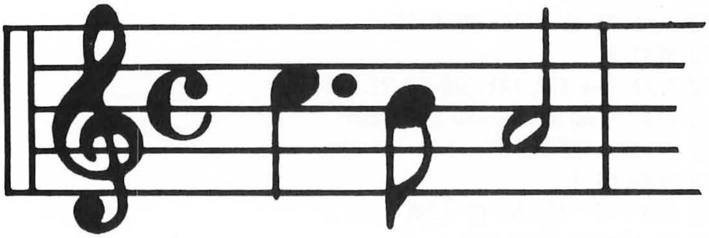
1. Add another sound channel for a little harmony. Enter these lines into your program.

```
    130 T=T*10
    145 SOUND 1,PITCH(X) +2,10,8
```

2. Try another song such as Row, Row, Row Your Boat. Change the notes and the duration in the DATA statements.

3. Make a musical round out of suggestion 2 such that the melody begins repeating before the first round has finished.

Figure 2.2 *Musical Tune*

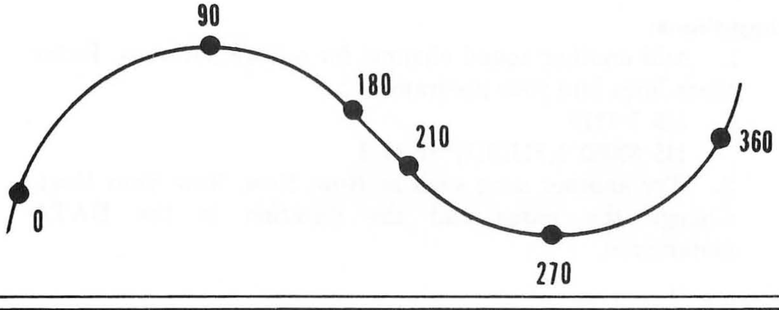


Up and Down Sound

In this program a pattern of sound is created as a sine wave is displayed on the screen. The value of SIN(ANGLE) ranges from−1 to +1. Here is a partial table of some angles and their corresponding sines.

ANGLE	SIN(ANGLE)
Ø	Ø.ØØØ
9Ø	1.ØØØ
18Ø	Ø.ØØØ
21Ø	−Ø.5ØØ
27Ø	−1.ØØØ
36Ø	Ø.ØØØ

Figure 2.3 *Up and Down Sound*



```
100 REM  UP AND DOWN SOUND
200 GRAPHICS 7+16:SETCOLOR 4,2,2
205 DEG :COLOR 2
207 REM  VARY SOUND AND PLOT
210 FOR ANGLE=0 TO 720 STEP 6
215 SOUND 0, (SIN(ANGLE)+1)*80,10,10
220 PLOT ANGLE/5,SIN(ANGLE)*35+45
230 NEXT ANGLE
240 END
```

Suggestions:

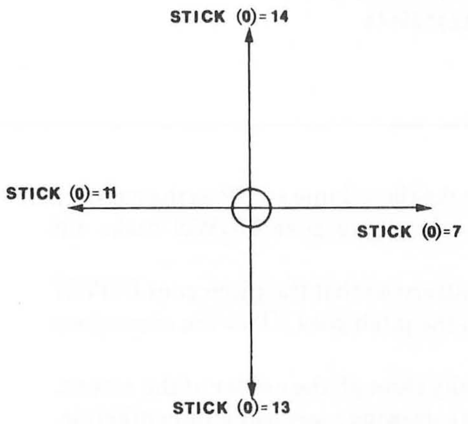
1. Change line 215 to make the volume go UP as the curve on the screen goes UP. As the curve goes DOWN make the volume go DOWN.
2. Reverse the sound pattern so that the pitch goes DOWN as the curve goes UP, and the pitch goes UP as the curve goes DOWN.
3. Draw a line horizontally through the center of the screen. Make the pitch go UP when drawing away from the center line, and DOWN when drawing toward the center line.

Audible Joystick

Make your programs communicate with both "sight and sound." Listen as you make the colored square move horizontally. Before using this program, you need to plug a joystick into the left hand jack of the computer (STICK (0)).

```
10 REM AUDIBLE JOYSTICK
15 PAUSE=200:MOVE=100:PX=20
20 GRAPHICS 3+;16:SETCOLOR 4,8,2:COLOR 1
30 PLOT PX,10
35 REM  CHECK JOYSTICK POSITION
40 IF STICK(0)=11 OR STICK(0)=7 THEN GOSUB MOVE
60 GOTO 40
100 REM  MOVE SQUARE ON SCREEN
105 COLOR 0:PLOT PX,10
106 TEMP=(PX-SGN(STICK(0)-9)*2)
107 IF TEMP<1 OR TEMP>37 THEN 112
110 PX=PX-SGN(STICK(0)-9)*2
112 COLOR 1:PLOT PX,10
115 REM A LITTLE SOUND
120 SOUND 0,15,10,8:GOSUB PAUSE:SOUND 0,0,0,0
130 RETURN
200 REM  PAUSE SUBROUTINE
210 FOR T=1 TO 100:NEXT T:RETURN
```

Figure 2.4 *Joystick Positions*



The values of several joystick positions are shown at the left.

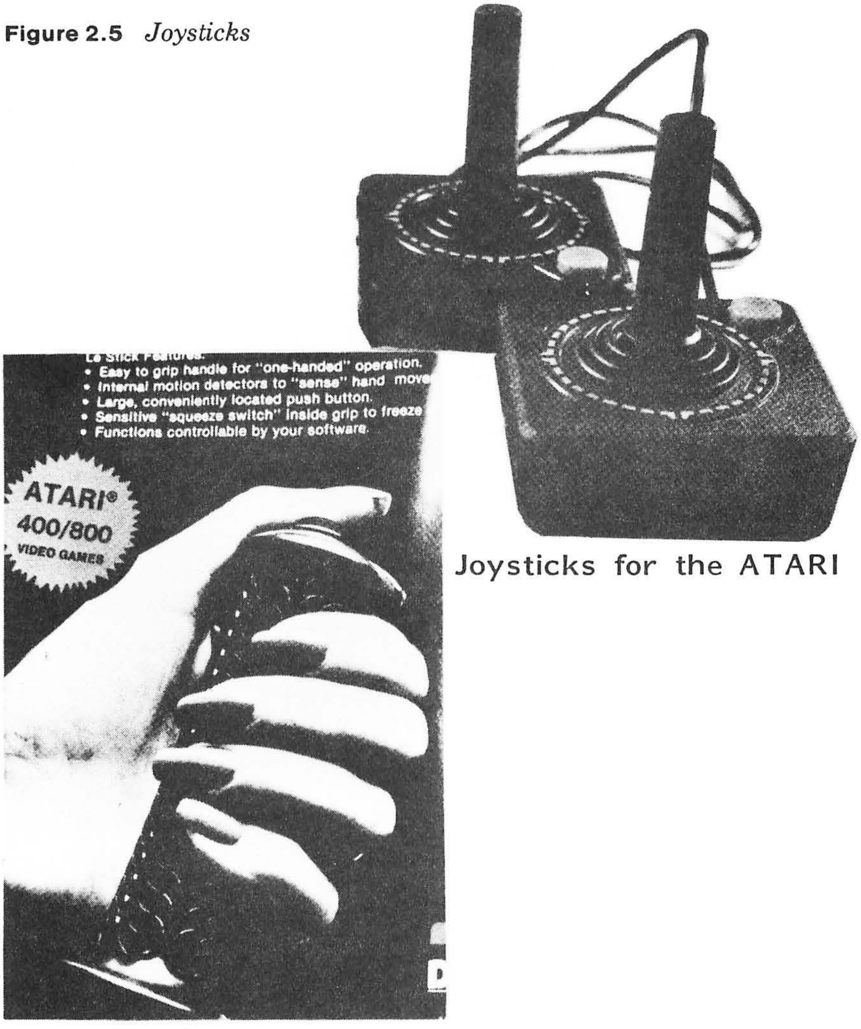
Suggestions:

1. For a different pitch change line 120

```
        12Ø SOUND Ø,2ØØ,1Ø,8:GOSUB PAUSE:SOUND Ø,Ø,Ø,Ø,
```

2. Make the sound a LOW pitch when going left and a HIGH pitch when going right.
3. Make the square move vertically as well as horizontally. Add a different sound for each direction.

Figure 2.5 *Joysticks*

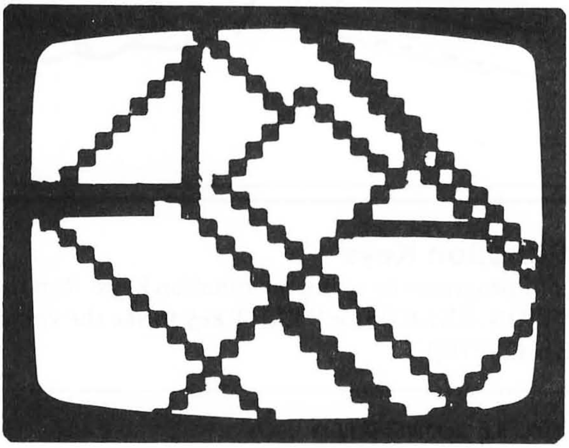

Le Stick Features:
• Easy to grip handle for "one-handed" operation.
• Internal motion detectors to "sense" hand move
• Large, conveniently located push button.
• Sensitive "squeeze switch" inside grip to freeze
• Functions controllable by your software.
ATARI®
400/800
VIDEO GAMES

Joysticks for the ATARI

3 Keys Paddles and Joysticks

The excitement of using a computer can be enhanced by using various methods of input. Program interaction can be made easier through the use of joysticks, paddles and special keys.

Figure 3.1 *Screen Drawing*



A program like "Joystick Drawing" requires an efficient method of handling input from a joystick.

In this section, we will work with the following functions:

PADDLE STICK PEEK

Refer to your *Basic Reference Manual* for details.

After you feel comfortable with this section, work on combining the ideas of the first three sections of this book into a single program.

Figure 3.2 *Cartoon*

Console Function Keys

Spice up your programs by using the function keys. Run this program. Press the OPTION, SELECT or START key to see the value of address 53279. (PEEK (53279)).

```
10 REM CONSOLE DEMONSTRATION
20 GRAPHICS 0
30 PRINT "PRESS THE START KEY TO BEGIN"
40 IF PEEK(53279)<>6 THEN 40
50 PRINT "ADDRESS 53279 = ";PEEK(53279)
70 GOTO 50
```

You need to know a little about binary numbers to understand this routine. Each key controls a single bit (binary digit) of address 53279. The bit is 0 if the key is pressed, and 1 if it is not. A binary to decimal conversion is shown.

Figure 3.3 *Binary Numbers*

	Address 53279			Decimal Number
Binary Number	bit 2	bit 1	bit Ø	PEEK(53279)
Ø11	$\boxed{Ø}$ x 2 +	$\boxed{1}$ x 2 +	$\boxed{1}$ x 2	= 3 $\boxed{\text{OPTION}}$
1Ø1	$\boxed{1}$ x 2 +	$\boxed{Ø}$ x 2 +	$\boxed{1}$ x 2	= 5 $\boxed{\text{SELECT}}$
11Ø	$\boxed{1}$ x 2 +	$\boxed{1}$ x 2 +	$\boxed{Ø}$ x 2	= 6 $\boxed{\text{START}}$
111	$\boxed{1}$ x 2 +	$\boxed{1}$ x 2 +	$\boxed{1}$ x 2	= 7 No Switch pressed

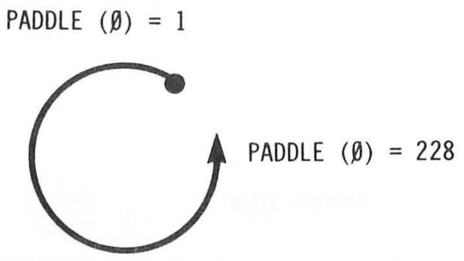
Suggestions:

1. Press a combination of keys. Does the value of address 53729 change as expected?
2. Change the program to PRINT the correct name keys you pressed.
3. Write a short program to select a number from 1 to 5. For example, the first time you press the SELECT key, a 1 is displayed on the screen. The second time you press SELECT, a 2 is displayed and so on. This could be used to select game options.

Paddle Motion

Use the paddle to move a colored square horizontally across the screen. By turning the paddle, a value from 1 to 228 is produced. For this program the paddle must be plugged into the left hand jack of the computer.

Figure 3.4 *Paddle Movement*



PADDLE (Ø) increases as the
paddle is turned counterclockwise.

```
10 REM PADDLE DEMONSTRATION
20 GRAPHICS 3+16
30 X=0
40 REM CONVERT PADDLE VALUE TO POSITION
50 X1=PADDLE(0)/6
55 REM  PLOT BACKGROUND COLOR
60 COLOR 0:PLOT X,10
65 REM  PLOT COLOR 1
70 COLOR 1:PLOT X1,10
80 X=X1
90 GOTO 50
```

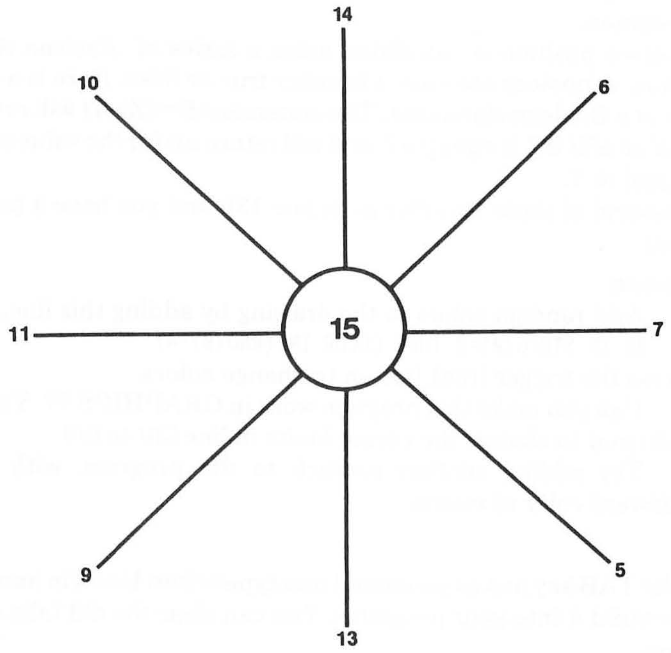
Suggestion:

1. Experiment with the color. Change line 70 to
   `70 COLOR 2:PLOT X1,10`
2. Try another color change. Change line 60 to
   `60 COLOR 1:PLOT X,10`
3. Add some colors with these line changes. Press the trigger (red) button to get another color.
   ```
   30 X=0:C=0
   55 IF PTRIG(0) =0 THEN C=C+1
   60 COLOR C+1 :PLOT X,10
   70 COLOR C:PLOT X1,10
   ```
4. Try this program in 2 dimensions by adding another paddle for vertical movement.

Joystick Drawing

Use the joystick to draw on the screen. Here is a diagram of the values the computer returns to STICK(0).

Figure 3.5 *Values of STICK(0)*



```
10 REM  JOYSTICK DRAWING
50 REM  SET UP SCREEN
60 GRAPHICS 3+16
70 X=19:Y=9
75 COLOR 1
80 PLOT X,Y
100 REM  STICK POSITION
110 Z=STICK(0)
120 REM  SCREEN POSITION
130 X=X+((Z=5)+(Z=7)+(Z=6))
```

```
140 X=X-((Z=10)+(Z=11)+(Z=9))
150 IF X<0 OR X>39 THEN X=39*(X>39)
160 Y=Y-((Z=5)+(Z=9)+Z=13))
170 Y=Y-((Z=6)+(Z=14)+(Z=10))
180 IF Y<0 OR 7>23 THEN Y=23*(Y>23)
190 GOTO 80
```

The stick position is read in line 110. Lines 130 to 180 change the cursor position.

The cursor position is calculated using a series of *Boolean Algebra* statements. A boolean statement is either true or false. Here is a simple example of a Boolean statement. The statement B=(Z=7) will return a 1 for the value of B if Z is equal to 7, and will return a 0 for the value of B if Z is not equal to 7.

Add several of these together as in line 130 and you have a powerful statement.

Suggestions:

1. Add random colors to the drawing by adding this line.
   `85 IF STRIG(0)=0 THEN COLOR INT(RND(0)*4)`
   Press the trigger (red) button to change colors.
2. Can you make this program work in GRAPHICS 7? You will need to change the cursor limits in line 130 to 180
3. Try adding another joystick to the program, with a different color of course.

TAB

Use the TAB key just as you would on a typewriter. Use it in immediate mode or build it into your programs. You can clear the old tabs and set new ones.

```
10 REM  TAB DEMONSTRATION
15 REM  CLEAR TABS
20 FOR X=1 TO 6:? CHR$(127);CHR$(8158);:NEXT X
40 REM  SET NEW TABS
45 PRINT "       ";CHR$(159);"                ";
50 PRINT "                   ";CHR$(159);"                ";
55 PRINT CHR$(159)
60 REM DISPLAY RANDOM NUMBERS IN COLUMNS
70 PRINT CHR$(125)
80 FOR X=1 TO 60
90 R=INT(RND(0)*100):? R;CHR$(127);
100 NEXT X
120 END
```

To clear a tab press TAB, then control TAB. Repeat this for as many tabs as you wish to clear. In program mode you can PRINT CHR$(127);CHR$(158).

To set a tab press SHIFT TAB after you have moved the cursor to desired position on the screen. In program mode the CHR$ function can be used (ie. PRINT" ";CHR$(159).)

Suggestions:

1. Change line 45

```
45 PRINT"      ";CHR$(159) ;"       ";CHR$(159)
```

2. The tabs are normally set 10 spaces apart. This can be changed with a POKE to address 201. Try this.
   a) Press SYSTEM RESET
   b) Press the TAB key several times and note the movement.
   c) Type POKE 201,18
   d) Press the TAB key several times again and note the movement of the cursor.
3. Write a program that prints random numbers in 3 columns tabbed at positions 15, 30 and 35.

Key Control

This subroutine is used to convert all key entries to upper-case normal video. This demonstration is designed to accept one character at a time by a GET statement. The variable A holds the Atari ASCII value of the key pressed. This subroutine is especially useful when only normal upper-case input is desired. As an example, enter a letter between A and J. Even if the reverse VIDEO key or CAPS lower key is pressed, the computer will check and reset itself for upper-case normal video.

```
100 REM  KEY CONTROL
110 KEYCHECK=24500
200 CLOSE #1:OPEN #1,4,0,"K:".GET #1,A
205 GOSUB KEYCHECK
210 PRINT CHR$(A);
220 GOTO 200
230 END
24500 REM  CHECK AND RESET KEYS
24515 REM  CHECK RETURN KEY
24520 IF A=155 THEN 24560
24525 REM  CHECK INVERSE VIDEO
24530 IF A>=128 THEN A=A-128
24535 REM  CHECK LOWER CASE
24550 IF PEEK(702)=0 AND A>96 THEN A=A-31
```

```
24555 REM  SET INV-VIDEO AND CAP-UP
24560 POKE 702,64:POKE 694,0
24590 RETURN
```

Suggestions:

1. Change lines 24530 and 24560, so that inverse video keys are accepted.
2. Modify this program to get a string of up to 8 alphanumeric entries in "inverse" video only.
3. Make this routine accept the letters A or B only. This routine could be used to restrict key entry to a specific set of characters.

Pick a Key

If you are looking for a method to get input from the keyboard while not stopping the execution of your program, a PEEK to the keyboard might be your answer. Address 764 holds the keycode of the last key pressed. The keycode is not the same as the Atari ASCII code. Here is a program to help you find the keycode values.

```
10 REM PICK A KEY
15 GRAPHICS 0
20 PRINT "PRESS A KEY"
30 IF PEEK(764)=255 THEN 30
35 KEY=PEEK(764)
40 CLOSE #1:OPEN #1,4,0,"K:":GET #1,A
50 PRINT "THE KEYCODE VALUE OF ";
60 PRINT CHR$(A);" IS ";KEY
70 GOTO 30
```

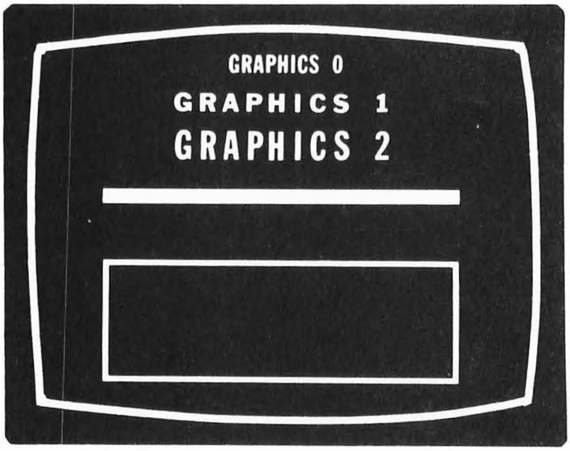
Suggestions:

1. Make a table of key names, keycodes and Atari ASCII.
2. Modify the "Musical Tune" program so that it plays normal, but starts over any time a key is pressed.
3. Modify the "Circle" program so that you can change the radius of the circle as it is being drawn.

4 Specialized Screen Routines

Give your screen displays something extra. Add more colors, change the Atari character set or perhaps mix several graphic modes.

Figure 4.1 *Graphics*



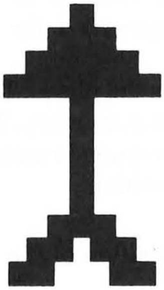
(In MIXED MODE GRAPHICS five graphic modes are displayed on a single screen.)

These programs all require direct access to memory by using a BASIC PEEK or POKE. Keep Appendix 1 Description of "Memory Addresses" handy as you go through these programs.

Player-Missile Graphics

An entirely new dimension in graphics is possible by using players and missiles. A player, or a missile is a special graphics design which can be easily moved from one part of the screen to another. Try it.

Figure 4.2 *Graphic Missile*



```
1000 REM PLAYER/MISSILE DEMONSTRATION
1010 GRAPHICS 8:SETCOLOR 2,3,4
1015 REM  SET RESOLUTION
1020 POKE 559,62
1025 REM  SET HOR. POS. AND COLOR
1030 POKE 53248,120
1040 POKE 704,200
1042 POKE 53256,2
1045 REM  ACTIVATE P/M GRAPHICS
1050 I=PEEK(106)-8
1060 POKE 54279,I:POKE 53277,3
1070 REM  BUILD PLAYER SHAPE
1080 J=I*256+1024
1090 FOR N=100 TO 128
1095 READ D
1100 POKE J+N,D
1110 NEXT N
1200 DATA 24,24,24,60,60,60,126,126,126
1210 DATA 255,255,255,24,24,24,24,24,24
1220 DATA 24,24,60,60,60,102,102,102
1230 DATA 195,195,195
1400 REM  CHANGE HOR. POSITION
1490 PRINT "PRESS SYSTEM RESET TO EXIT"
1500 PRINT "ENTER POSITION (50 TO 200)"
1510 TRAP 1510:INPUT P
1520 IF P<50 OR P>200 THEN 1510
1600 POKE 53248,P:GOTO 1500
```

Line 120 determines the resolution (in this case, single line). The top part of memory is used to store the shape of a player. Line 1050 calculates the beginning of this memory section. A memory map is shown in Appendix 2. Also, refer to Appendix 1 for a description of "Memory Addresses."

Line 1060 is needed to access the player. Lines 1090 to 1110 read the data and store the player shape in memory. The vertical position on the screen depends on this data's position in memory. A single POKE in line 1500 changes the horizontal position.

Suggestions:

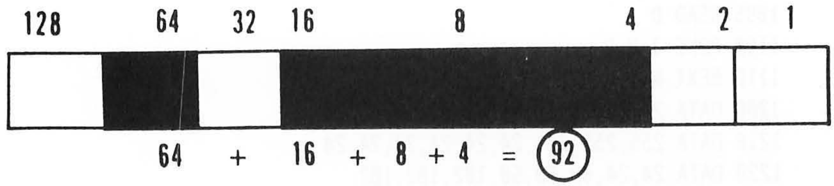
1. Change the vertical position of the player
```
1Ø9Ø FOR N=5Ø TO 78
```
2. Change the color in line 1040
```
1Ø4Ø POKE 7Ø4,134
```

Note, to calculate a color, multiply a BASIC color by 16 and add the luminance value. For example, SETCOLOR 0,12,8 translates to a color of 12*16+8=200.

3. Change the size of the player.
```
1Ø42 POKE 53256,3
2 = normal size, 1 = double width, 3 = quadruple width
```
4. Change the Data in line 1200 for a different shape.

Note, each player is 8-bits wide, and each line of the player can be represented by a decimal number from 0 to 255.

Figure 4.3 *Player*



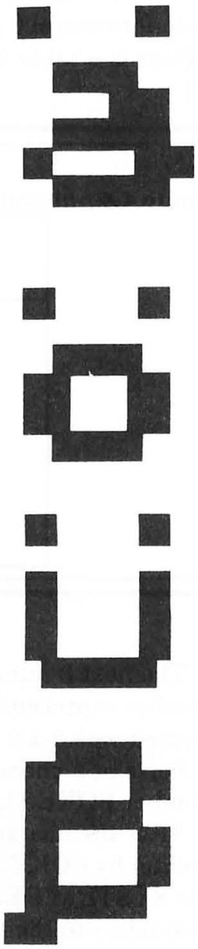
Data of 92 will turn on these segments of a line in the player.

5. Write the word "PLAYER" vertically on the screen by changing the numbers in the DATA statements.
6. Refer to the Appendices and add another player. Make it a different color, of course.

German Font

This routine is used to modify the Atari character set. Four graphic characters are changed to represent ä, ö, ü, and ß.
By pressing CONTROL A, O, U and S respectively these special German characters can be generated.

```
1Ø REM GERMAN FONT DEMONSTRATION
2ØØ GERMFONT=28ØØØ
5ØØ GOSUB GERMFONT
51Ø FOR X=97 TO 122:? CHR$(X);:NEXT X
52Ø ? CHR$(1);CHR$(15);CHR$(21);CHR$(19)
6ØØ END
28ØØØ REM GERMAN FONT . . . . . . . . . . . . .
28ØØ5 REM CONTROL -a,o,u,s PRODUCES
28ØØ6 REM GERMAN UMLAUT a,o,u & ss
28ØØ7 DIM B$(8Ø)
28Ø1Ø POKE 756,224
28Ø4Ø MEMEND=PEEK(1Ø6)-4:CHSET-MEMEND:256
28Ø6Ø POKE 1Ø6,MEMEND-1:GRAPHICS Ø
28Ø65 REM  READ MACHINE LANGUAGE COPY
28Ø7Ø FOR X=1 TO 32
28Ø75 READ A:B$(X,X)=CHR$(A)
28Ø8Ø NEXT X
28Ø82 DATA 1Ø4,1Ø4,133,213,1Ø4,133,212
28Ø83 DATA 1Ø4,133,215,1Ø4,133,214,162
28Ø84 DATA 4,16Ø,Ø,177,212,145,214
28Ø85 DATA 2ØØ,2Ø8,249,23Ø,213,23Ø,215
28Ø86 DATA 2Ø2,2Ø8,24Ø,96
28Ø88 REM  COPY CHARACTER SET
28Ø9Ø I=USR(ADR(B$),224*256,CHSET)
281Ø5 REM MODIFY CHARACTERS
2811Ø FOR X=1 TO 4
2812Ø READ CHAR:N=CHSET+CHR*8
2813Ø FOR I=Ø TO 7:READ T:POKE N+I,T:NEXT I
2814Ø NEXT X
286ØØ DATA 79,Ø,1Ø2,Ø,6Ø,1Ø2,1Ø2,6Ø,Ø
2861Ø DATA 85,Ø,1Ø2,Ø,1Ø2,1Ø2,1Ø2,62,Ø
2862Ø DATA 65,1Ø8,Ø,6Ø,6,62,1Ø8,62,Ø
2863Ø DATA 83,Ø,6Ø,1Ø2,124,1Ø2,1Ø2,124,224
287ØØ POKE MEMEND-1,Ø:POKE 756,MEMEND
29ØØØ RETURN
```



In line 28060 four pages of memory are reserved for a new character set. This section of memory is determined by PEEKing the top of memory, PEEK(106) and subtracting 4 from it. A description of "Memory Addresses" can be found in Appendix 1.

The machine language routine that executes in line 28090 copies the Atari character set from ROM to the new memory location in RAM. Now the characters can be modified. Line 28120 is used to read new data, and change a particular character in the new character set. The data can be interpreted like this.

```
    28600 DATA 79,0,102,0,60,102,102,60,0
```

The 79 is the internal code for control 0. 79 is the Atari ASCII character code for control 0 plus 64.

Here is a table for finding internal codes from ASCII, see Appendix 4.

Figure 4.4 *Internal ASCII Codes*

Atari ASCII Value	Operation
0 - 31	Value + 64
32 - 95	Value - 32
96 - 127	None
128 - 159	Value + 64
160 - 223	Value - 32
224 - 255	None

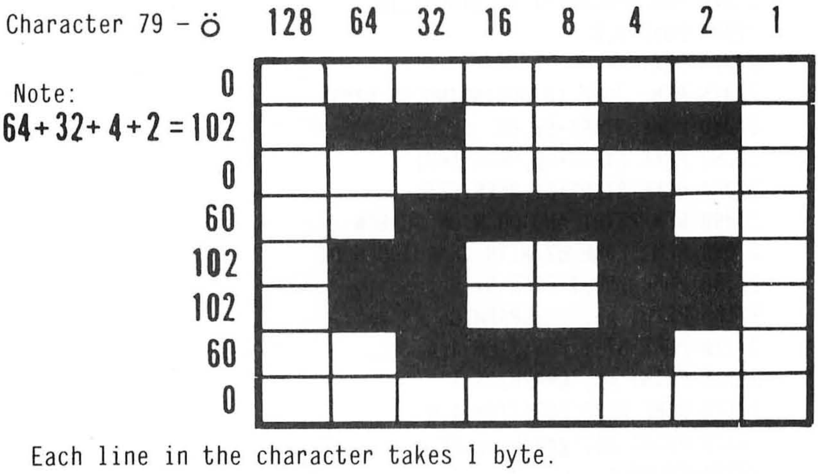
The next 8 numbers in the data statement form the character. Each number represents the decimal equivalent of a byte. Each character is formed on a 8 x 8 matrix.

Parts of a character are turned either ON or OFF depending on the number POKEd into the memory address representing that character.

After the character set has been modified, change the character set pointer by POKE 756,MEMEND in line 28700. A graphics command or the SYSTEM RESET key will change the pointer back, and return the characters to their original state. Before executing the routine a second time you must press SYSTEM RESET or POKE 106, MEMEND+4.

This new character set (font), can be used in a running program or in the immediate mode.

Figure 4.5 *Special Screen Display*



Suggestions:

1. Try a new character. Change line 28600
   ```
   286ØØ DATA 79,255,195,195,195,195,195,195,255
   ```
   Run the program and press CONTROL 0.
2. Try making some other characters or designs.
3. Try it in another graphics mode.
4. Make a Greek character set.

Mixed Mode Screen

Would you like to see a combination of 5 graphic modes on a single screen display? This and more can be accomplished by building a custom display list in memory. Run this program to see an example of a mixed mode screen.

```
27ØØØ REM MIXED MODE SCREEN
27Ø1Ø GRAPHICS 7+16
27Ø2Ø SETCOLOR 4,8,2:SETCOLOR 2,8,2
27Ø25 REM BUILD CUSTOM DISPLAY LIST
27Ø27 REM  FIND START OF DISPLAY LIST
27Ø3Ø START=PEEK(56Ø)+PEEK(561)*256+4
27Ø35 REM  GRAPHICS Ø,  2 LINES
27Ø4Ø POKE START-1,66:POKE START+2,2
27Ø45 REM  GRAPHICS 1,  2 LINES
27Ø5Ø POKE START+3,6:POKE START+4,6
```

```
27Ø55 REM  GRAPHICS 2, 2 LINES
27Ø6Ø POKE START+5,7:POKE START+6,7
27Ø65 REM  GRAPHICS 3, 4 LINES
27Ø7Ø FOR X=START+7 TO START+;1Ø
27Ø72 POKE X,8
27Ø73 NEXT X
27Ø75 REM  JUMP TO BEGINNING OF LIST
27Ø8Ø POKE START+59,65
27Ø82 POKE START+6Ø,PEEK(56Ø)
27Ø84 POKE START+61,PEEK(561)
27Ø9Ø REM PRINT AND DRAW ON SCREEN
27Ø95 REM  POKE 87 WITH GRAPHICS MODE
271ØØ POKE 752,1:POKE 87,Ø
271Ø5 PRINT #6,"    GRAPHICS Ø"
2711Ø POKE 87,1:POSITION 4,4
27115 PRINT #6;"GRAPHICS 1"
2712Ø POKE 87,2:POSITION 4,6
27122 PRINT #6;"graphics 2"
27125 COLOR 1
2713Ø POKE 87,3:PLOT 5,16:DRAWTO 3Ø,16
27135 COLOR 2
2714Ø POKE 87,7:PLOT 2Ø,12:DRAWTO 12Ø,12
2715Ø DRAWTO 12Ø,4Ø:DRAWTO 2Ø,4Ø
2716Ø DRAWTO 2Ø,12
272ØØ GOTO 272ØØ
```

The computer generates 192 scan lines on the TV screen. A single line of a particular graphics mode (mode line), is made up of a number of scan lines. For example, in GRAPHICS MODE 3 eight scan lines are required to produce 1 mode line. In GRAPHICS MODE 8, only one scan line is required to produce a mode line.

The computer uses a short sequence of code called the " display list" to generate these mode lines. To build a custom display list, you need to modify the existing display list. First, decide on the GRAPHICS MODES and their positions on the screen. Then change the mode lines in the display list to correspond to your screen. This procedure requires several well planned steps. Refer to Appendix 3 "Building a Display List" for more details.

Suggestions:

1. Change the color in line 27020
   `27Ø2Ø SETCOLOR 4,8,2:SETCOLOR 2,4,2`
2. Draw a line in the GRAPHICS 7 area at the bottom of the screen.

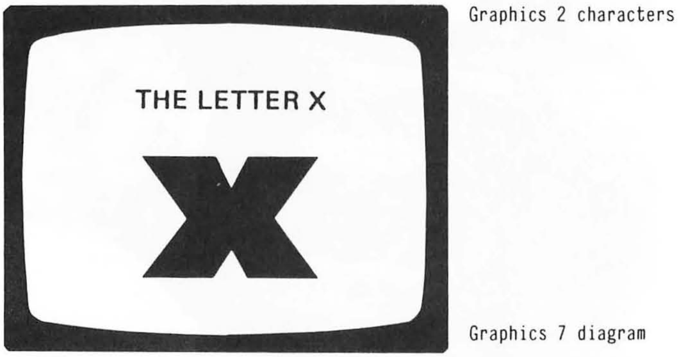
```
27150 PLOT 40,26:DRAWTO 100,26
```

3. Changing any part of the display list usually requires the whole list to be rebuilt. A couple of graphic sections, however, can be switched without too much trouble.
Change these lines.

```
27050 POKE START+3,7:POKE START+4,7
27060 POKE START+5,6:POKE START+6,6
```

4. Refer to Appendix 3 and build a display list to produce the following screen display.

Figure 4.6 *Graphic Character Set*



Hints:

The LMS byte in line 27040 should be 71.
Make an even number of GRAPHICS MODE 2 lines.
The display list will contain 82 mode lines if you have 2 mode-2 lines and 80 mode-7 lines.

Characters in GRAPHICS 4, 6 and 8

If you are looking for characters in GRAPHIC MODES 4, 6 or 8, keep reading. In the two color modes (4,6,8) characters can be read from the Atari character set and POKEd to the screen. The character sizes range from twice the GRAPHICS 2 size to the normal GRAPHICS 0 size.

ATARI 800

```
100 REM  GRAPHICS 8 CHARACTERS DEMO
110 GRAPHICS 8:SETCOLOR 2,4,2
200 DIM CHAR$(40)
220 GR8CHAR=20200
250 PRINT "ENTER CHARACTER STRING";
255 INPUT CHAR$
260 PRINT "ENTER CHARACTER STRING";
270 INPUT PX,PY:GOSUB GR8CHAR
280 GOTO 250
20200 REM  POKE CHARACTERS ON SCREEN
20205 IF LEN(CHAR$)=0 THEN RETURN
20207 REM  CALCULATE SCREEN POSITION
20210 I0=PEEK(560)+PEEK(561)*256
20215 I1=PEEK(I0+4)+PEEK(I0+5)*256
20220 REM  FIND POSITION IN CHARACTER SET
20230 FOR U=1 TO LEN(CHAR$)
20240 I2=57344+((ASC(CHAR$(U,U))-32)*8)
20245 I3*I3=PY*40+PX+U-1
20250 REM  POKE CHARACTER ON SCREEN
20260 FOR Z=0 TO 7
20265 POKE I3+Z*40,PEEK(I2+Z)
20270 NEXT Z
20275 NEXT U
20280 RETURN
```

The subroutine to POKE characters to the screen starts at line 20200. This routine is actually as complicated as it appears. In line 20210 the variable I1 is the position in memory at the top left-hand corner of the screen. In line 20240, I2 is a position of a character in the Atari character generator. I3 is the screen position. In line 20260 characters from the Atari character set, PEEK (I2+Z), are POKEd to a screen position I2+Z*40. An explanation of the procedure to calculate a screen position is found in Appendix 4.

Suggestions:

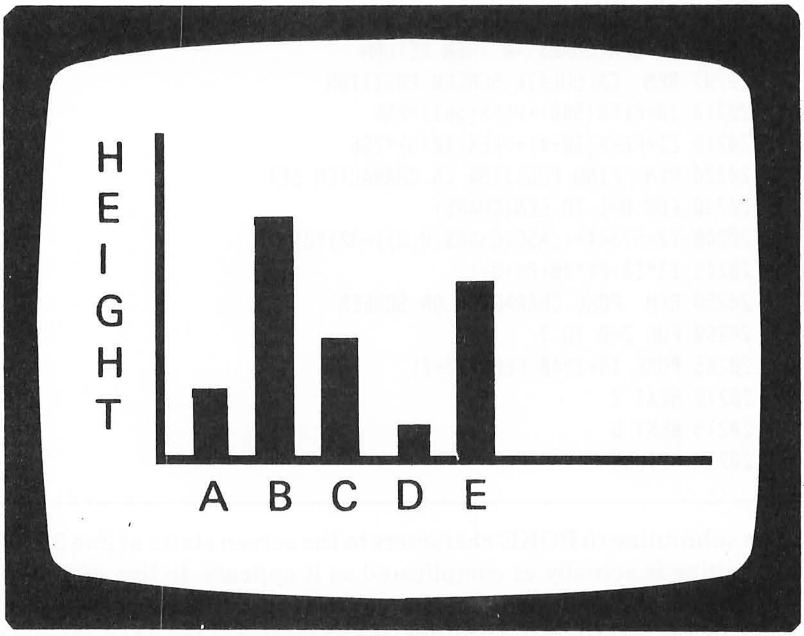
1. Change the 32 in line 20240 to 0.
   ```
   20240 12=57344+((ASC(CHAR$(U,U))-0)*8)
   ```

Rewrite the program to display putting lower-case letters in from the keyboard. This change will get characters from a different part of the memory holding the character set.

2. Try it in another graphics mode.
   ```
   110 GRAPHICS 6 :SETCOLOR 2,4,2
   ```

3. Try this program in GRAPHICS 4. Only ten bytes of RAM per line are required for this mode.
4. Set up a screen using GRAPHICS 6 that looks like the following.

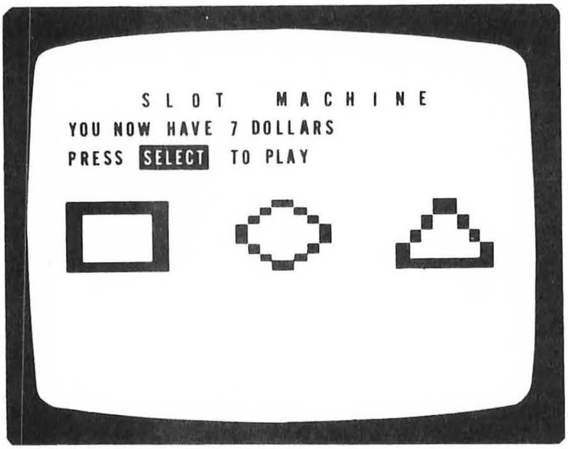
Figure 4.7 *Graphics Screen*



5 Graphics and Sound Applications

This is the section that let's you tie all of your programming techniques together. Play a game like "DUEL" or "SLOT MACHINE." Watch "VIDEO ART" or plot an equation with "LINEAR X-Y PLOT."

Figure 5.1 *Slot Machine*



```
     S L O T    M A C H I N E
YOU NOW HAVE 7 DOLLARS
PRESS SELECT  TO PLAY
```

"SLOT MACHINE," a game of luck, puts 3 colored players on a GRAPHICS 0 screen.

Make a note of how Graphics, Sound and Special Features can be used together. As you complete this section, you are ready to let your creativity do it's thing. If you can imagine it, you can do it.

Duel

Use the joystick to duel with an opponent. Eat up your opponents squares faster than yours are eaten and you win. This game takes two players. Put the joysticks into the first two jacks.

```
10 REM  A GAME OF DUEL
12 REM  GAME LENGTH
13 GRAPHICS 0:POSITION 4,5
14 ? "HOW LONG OF A GAME?":?
15 ? "  1,2 OR 4 MINUTE";:INPUT LENGTH
16 IF LENGTH<1 OR LENGTH>4 THEN 13
17 REM  SET UP SCREEN
18 GRAPHICS 3:SETCOLOR 0,0,8:SETCOLOR 1,8,2
19 SETCOLOR 4,1,6:SETCOLOR 2,1,6
20 FOR N=0 TO 19
21 COLOR 1
22 PLOT 0,N:DRAWTO 19,N:COLOR 2:DRAWTO 39,N
23 NEXT N
25 REM  INITIALIZE VARIABLES
27 DIM PX(2),PY(2)
30 P=0:PX(0)=39:PX(1)=0:PY(0)=19:PY(1)=19
40 REM  POSITION AND COLOR
42 P=(P=0)
45 X=PX(ABS(P-1)):Y=PY(ABS(P-1))
50 COLOR P+1:PLOT X,Y
61 REM  STICK POSITION
62 Z=STICK(P)
63 REM  SCREEN POSITION
65 X=X+((Z=5)+(Z=7)+(Z=6))
70 X=X-((Z=10)+(Z=11)+(Z=9))
75 IF X<0 OR X>39 THE X=39*(X>39)
80 Y=Y+((Z=5)+(Z=9)+(Z=13))
100 Y=Y-((Z=6)+(Z=14)+(Z=10))
110 IF Y<0 OR Y>19 THEN Y=19*(Y>19)
160 COLOR ABS(P-2):PLOT X,Y
170 PX(ABS(P-1))=X:PY(ABS(P-1))=Y
175 REM  POSITION DEPENDENT SOUND
180 SOUND 1,6*(ABS(41-X)+1),10,8
190 SOUND 2,12*(ABS(21-Y)+1),10,8
200 SOUND 1,0,0,0:SOUND 2,0,0,0
300 REM  LENGTH OF GAME
310 L=L+1:IF L=LENGTH*550 THEN 320
315 GOTO 40
```

```
317 REM  LOCATE SQUARES FOR SCORE
32Ø POKE 752,1:?
322 PRINT "          ONE MOMENT PLEASE. . . . ."
325 FOR I=0 TO 39
33Ø FOR J=Ø TO 19
34Ø LOCATE I,J,Q
35Ø IF Q=2 THEN T=T+1
36Ø NEXT J:NEXT I
37Ø ? :? "SCORE:  GRAY D;8ØØ-T,"BLUE ";T;
39Ø END
```

Suggestions:

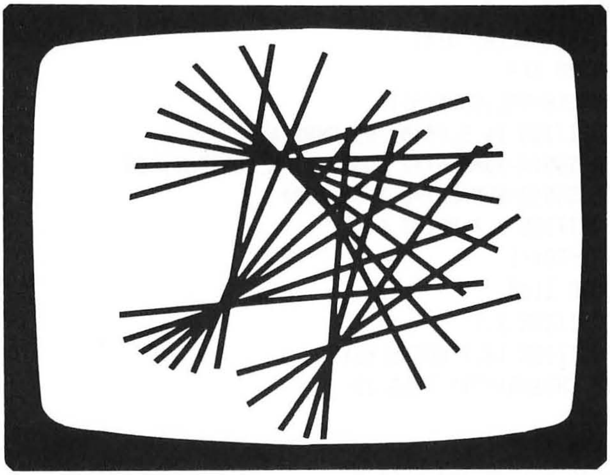
1. For a more difficult game, make this program work in GRAPHICS 5.
2. Add an option to "Play Again?" to this program.
3. Rewrite the program so one person can play the computer.

Video Art

Sit back and enjoy this one. Let it run for several minutes and watch the screen pattern develop.

```
1Ø RAM  VIDEO ART
15 GRAPHICS 7+16
17 REM  GENERATE 2 RANDOM POINTS
2Ø X1=RND(Ø)*159:X2=RND(Ø)*159
3Ø Y1=RND(Ø)*91:Y2=RND(Ø)*91
35 REM  GENERATE RANDOM INCREMENTS
4Ø A1=RND(Ø)*2:A2=RND(Ø)*4
5Ø B1=Ø.5-ABS(A1):B2=1-ABS(A2)
6Ø REM  A LITTLE COLOR
65 SETCOLOR RND(Ø)*4,RND(Ø)*16,RND(Ø)*15
75 COLOR RND(Ø)*4
77 REM  NEW POSITION
8Ø X1=X1+A1:X2=X2+A2
9Ø Y1=Y1+B1:YX=Y2+B2
95 REM  START AGAIN IF OUT OF RANGE
1ØØ TRAP 2Ø
1Ø2 REM  DRAW LINES
11Ø PLOT X1,Y1:DRAWTO X2,Y2
12Ø GOTO 8Ø
```

Figure 5.2 *Video Art*



Suggestions:

1. Change this line for a different pattern.
   ```
   12Ø GOTO 65
   ```
2. Add some sound.
   ```
   1Ø5 SOUND Ø,X1,1Ø,X2/1Ø
   ```
3. Make this program interactive so that the sound and pattern change when a key is pressed.

Guessing Game

This is a classic computer game. The computer generates a random number for you to guess. For a flashy screen, the large characters of GRAPHICS MODE 2 are used.

```
1Ø REM NUMBER GUESSING GAME
2ØØ REM SUBROUTINE VALUES
21Ø TIME=2ØØØØ:LOW=21ØØØ:HIGH=22ØØØ
22Ø CORRECT=23ØØØ:KEY=24ØØØ
1ØØØ REM SET UP SCREEN
1ØØ5 TRAP 1ØØØ
1Ø1Ø GRAPHICS 2+16:SETCOLOR 4,8,2
11ØØ POSITION 1,2
```

```
1104 PRINT #6;"I'M THINKING OF A"
1106 PRINT #6;" NUMBER BETWEEN 1&9"
1110 POSITION 1,5:PRINT #6;"YOUR GUESS? "
1112 REM GENERATE NUMBER AND CHECK
1113 TRY=1
1115 R=INT(RND(0)*9)+1
1120 GOSUB KEY
1125 NUMBER=VAL(CHR$(A))
1130 POSITION 14,5:PRINT #6;CHR$(A)
1210 ON SGN(R-NUMBER)+2 GOSUB HIGH,CORRECT,LOW
1220 IF SGN(R-NUMBER)=0 THEN 1300
1230 POSITION 2,7:PRINT #6;"
1240 TRY=TRY+1
1250 GOTO 1120
1300 POSITION 2,7
1307 POSITION 13,7:GOSUB KEY
1310 IF CHR$(A)="Y" THEN 10
1400 END
20000 REM TIME
20010 FOR T=1 TO 400:NEXT T
20020 RETURN
21000 REM TOO LOW
21010 POSITION 2,7:PRINT #6;"too low"
21015 GUSUB TIME
21020 RETURN
22000 REM TOO HIGH
22010 POSITION 2,7:PRINT #6;"too high"
22015 GOSUB TIME
22020 RETURN
23000 REM CORRECT
23010 POSITION 2,7
23015 PRINT #6;"correct in "";TRY;" trys"
23017 GOSUB TIME
23020 RETURN
24000 REM GET KEY
24010 CLOSE #1:OPEN #1,4,0,"K:";GET #1,A
24020 RETURN
```

Suggestions:

1. Change these lines to make your game work for numbers from 10 to 99.

```
1106 PRINT#6;"NUMBER FROM 10 TO 99"
1115 R=INT (RND(0)*90)+10
```

2. Add these lines
```
1126 GOSUB KEY:NUMBER2=VAL(CHR$(A))
1127 NUMBER=NUMBER*1Ø+NUMBER2
```
3. Make the game work for numbers from 1 to 1000.
4. Add a "buzzer sound" if your guess is wrong, and a "musical jingle" if your guess is right.

Slot Machine

This game will not cost you a penny to play. You start with 10 dollars. Each time you press the SELECT button a new set of figures appear. If all three match, you're a winner. You will, however find it hard to put your winnings in the bank.

```
1ØØ REM  SLOT MACHINE
2ØØ REM  SET VARIABLES
21Ø DIM FRUIT(3)
22Ø X=1Ø:PLAYER=1ØØØ:ERASE=12ØØ
4ØØ REM  SET PM GRAPHICS
4Ø5 GRAPHICS Ø
41Ø POKE 559,46:PMBASE=PEEK(1Ø6)-8
412 POKE 54279,PMBASE:POKE 53277,3
413 POKE 623,1
415 POSITION 8,1
417 PRINT "" S L O T   M A C H I N E"2
42Ø GOTO 8ØØ
5ØØ REM  PLAY GAME
54Ø REM  RANDOM SELECTION
6ØØ FOR PM=1 TO 3
61Ø FRUIT(PM)=INT(RND(Ø)*3)+1:GOSUB PLAYER
62Ø NEXT PM
71Ø IF FRUIT(1)<>FRUIT(2) THEN 72Ø
715 IF FRUIT(2)=FRUIT(3) THEN 75Ø
72Ø X=X-1
73Ø GOTO 8ØØ
75Ø X=X+INT(RND(Ø)*1Ø)+5
8ØØ POKE 752,1:POSITION 2,2:?
8Ø5 PRINT ""YOU NOW HAVE "";X;" DOLLARS."
81Ø IF X=Ø THEN 815
812 GOTO 82Ø
815 POSITION 2,5:? ""YOU'RE OUT OF MONEY."
816 GOSUB ERASE:PRINT CHR$(125):END
82Ø ? :? "PRESS OPTION TO QUIT"
84Ø IF PEEK(53279)=5 THEN 85Ø
842 IF PEEK(53279)=3 THEN 816
```

```
844 GOTO 84Ø
85Ø GOSUB ERASE
9ØØ GOTO 5ØØ
1ØØØ REM PLAYER
1ØØ5 POKE 53247+PM,PM*55
1ØØ6 POKE 53255+PM,3
1Ø1Ø J=PMBASE*256+512+((PM-1)*128)
1Ø15 LINE=FRUIT(PM)*1Ø+11ØØ
1Ø2Ø RESTORE LINE
1Ø3Ø READ PIGMENT
1Ø35 POKE 7Ø3+PM,PIGMENT
1Ø37 FOR Q=5Ø TO 8Ø
1Ø4Ø READ SHAPE
1Ø5Ø IF SHAPE=999 THEN RETURN
1Ø6Ø POKE J+Q,SHAPE
1Ø7Ø NEXT Q
11ØØ REM  PLAYER DATA
11Ø5 REM PLAYER ONE
111Ø DATA 234,255,255,255,129,129,129
1111 DATA 129,129,129,129,129,129,129
1112 DATA 129,129,129,129,255,255,999
1115 REM PLAYER TWO
112Ø DATA 199,24,24,24,24,36,36,36,36
1121 DATA 66,66,66,66,129,129,129,129
1122 DATA 255,255,255,999
1125 REM PLAYER THREE
113Ø DATA 68,24,24,36,36,66,66,66,129
1131 DATA 129,129,129,129,66,66,66,36
1132 DATA 36,24,24,999
12ØØ REM  ERASE PM PLAYERS
12Ø5 FOR M=Ø TO 256 STEP 128
121Ø FOR Q=5Ø TO 8Ø
1214 POKE (PEEK(1Ø6)-8)*256+512+M+Q,Ø
1216 NEXT Q
122Ø NEXT M
125Ø RETURN
```

Suggestions:

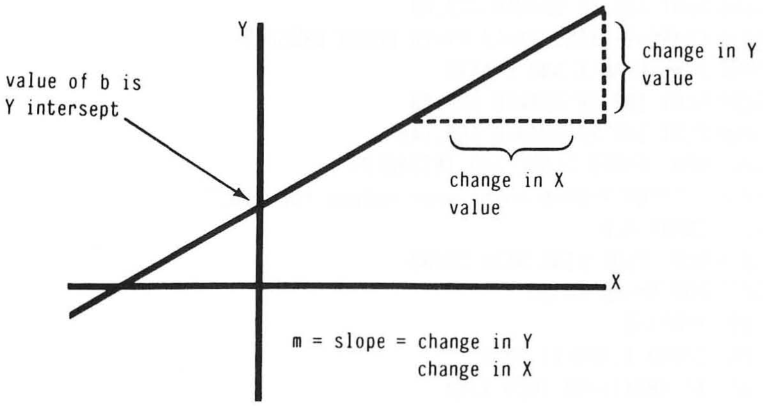
1. Change the data in lines 1100 through 1132 to get some different figures.
2. Add some sound as the figures appear on the screen.

3. Modify the program so that it keeps displaying new figures, and it stops only when you press SELECT. You then press START to continue.

Linear X- Y Plot

This math program draws lines on an X-Y axis. A full understanding requires you to be familiar with simple linear equation. Here is a brief introduction to the equation of a straight line. Y=mX+b is a linear equation in slope intercept form.

Figure 5.3 *Slope intercept form*



The line is a set of X and Y values that fit the equation Y=mX+b. In line 1072, the values of m and b are entered. In line 1080 Y values are calculated for the set of X values ranging from–80 to +80. In line 1090 the point is plotted on the screen.

```
1000 REM  LINEAR X-Y PLOT
1014 GR8CHAR=20200
1017 DIM CHAR$(40)
1018 REM  SET UP SCREEN
1020 GRAPHICS 8
1030 COLOR 1:SETCOLOR 2,12,2:SETCOLOR 4,12,0
1035 REM  DRAW BORDER
1038 FOR I=0 TO 3
```

```
1040 PLOT I,I:DRAWTO 319-I,I
1041 DRAWTO 319-I,159,I
1042 DRAWTO I,159-I:DRAWTO I,I
1043 NEXT I
1044 REM  CHARACTERS ON GR.8 SCREEN
1045 CHAR$="WHERE:":PX=1:PY=5
1046 GOSUB GR8CHAR
1047 CHAR$="WHERE:":PX=1:PY=12:GOSUB GR8CHAR
1048 CHAR$="M= SLOPE":PX=1:PY=19
1049 GOSUB GR8CHAR
1051 GOSUB GR8CHAR
1052 CHAR$="AND  X=-80 TO 80":PX=1:PY=35
1053 GOSUB GR8CHAR
1054 CHAR$="X-AXIS":PX=30:PY=74:GOSUB GR8CHAR
1055 PLOT 100,80:DRAWTO 220,80
1056 CHAR$=Y-AXIS":PX=17:PY=10:GOSUB GR8CHAR
1058 REM  DRAW X AND Y AXIS
1059 PLOY 100,80:DRAWTO 220,80
1060 PLOT 160,20:DRAWTO 160,140
1065 REM  ENTER SLOPE AND INTERCEPT
1070 ? ""FOR Y=mX+b enter your values for m,b."
1072 INPUT M,B
1075 REM  PLOT LINE WITH SOUND
1077 FOR X=-80 TO 80
1080 Y=M*X+B
1082 SOUND 1,ABS(Y),10,8
1086 IF ABS(Y)>80 THEN 1100
1090 PLOT X+10,80-Y
1100 NEXT X
1105 SOUND 1,0,0,0
1110 GOTO 1070
20200 REM POKE CHARACTERS ON MODE 8 SCREEN
20205 IF LEN(CHAR$)=0 THEN RETURN
20210 I0=PEEK(560)+PEEK(561)*256
20220 I1=PEEK(I0+4)+PEEK(I0+5)*256
20230 FOR U=1 TO LEN(CHAR$)
20240 I2=57344+((ASC(CHAR$(U,U))-32)*8)
20250 I3=I1+PY*40+PX+U-1
20260 FOR Z=0 TO 7
20264 POKE I3+Z*40,PEEK(I2+Z)
20266 NEXT Z
20270 NEXT U
20280 RETURN
```

Suggestions:

1. Use this program with different values of slope and Y intercept.
2. Substitute a different equation into line 1080, perhaps an equation of the form Y=AX2. You enter the value for A. Try .01 for a value of A. Try some negative values as well.
3. Make this program work in another graphics mode.
4. Plot different algebra equations such as those for a hyperbola, circle or parabola.

6 PEEKS, POKES and Special Stuff

PEEK and POKE let you unleash a power not available through other BASIC commands. With these two key words, it is possible to look directly into a memory address and change the content of many of the addresses. PEEK is used to "read" the contents of a memory address. POKE is used to "write" to an address. Because each address contains a single byte, the decimal value of the contents will always be between 0 and 255.

Refer to Appendix 1 "Description of Memory Addresses" and the *Atari Basic Reference Manual* as you go through this section.

Margins 82,83

Change the left margin.

```
    POKE 82,2Ø
```

Change the right margin.

```
    POKE 83,22
```

POKE things back to normal or press SYSTEM RESET.

Inverse Video 694

To change the keystrokes to inverse video try this.

```
  1Ø DIM Q$(4Ø)
  2Ø POKE 694,128
  3Ø INPUT Q$
  4Ø POKE 694,Ø
  5Ø INPUT Q$
  6Ø END
```

Run it. Can you figure it out?

CURSOR 752

Get rid of that cursor.

```
  POKE 752,1
```

Get it back.

```
  POKE 752,Ø
```

Flip a Character 755

Can you read upside down?

```
POKE 755,4
```

I can't either.

```
POKE 755,Ø
```

Loudspeaker 53279

Make the keyboard speaker produce sound.

```
1Ø POKE 53279,Ø:GOTO 1Ø
```

Run this one.

Where Are You? 84,85

What row are you on?

```
PRINT PEEK(84)
```

What column are you in?

```
PRINT PEEK(85)
```

Recorder 54019

Put a music cassette into your program recorder.

```
POKE 54Ø18,52
```

Enjoy the music while you work. Turn it off.

```
POKE 54Ø18,6Ø
```

Colors (0-4) 708-712

Change the background color in GRAPHICS 0

```
POKE 71Ø,12
```

Change the character luminance.

```
POKE 7Ø9,Ø
```

Characters 702

Try this demonstration.

```
5 DIM Q$(4Ø)
1Ø POKE 7Ø2,Ø
2Ø INPUT Q$
3Ø POKE 7Ø2,128
4Ø INPUT Q$
5Ø POKE 7Ø2,64
6Ø INPUT Q$
7Ø END
```

Press some keys and hit RETURN

The Atari Clock 18, 19, 20

Memory address 18,19 and 20 hold the Atari clock. Address 18 is the "most" significant byte, and address 20 is the "least" significant byte. Every 1/60 of a second (known as a jiffy) address 20 incremented by one. Each time address 20 exceeds 255, it returns to 0 and a 1 is added to address 19. Likewise, when address 19 exceeds 255, it returns to 0 and a 1 is added to address 18.

Try this routine.

```
    1Ø PRINT PEEK(18), PEEK(19), PEEK(2Ø)
    2Ø GOTO 1Ø
```

Now translate these to a single decimal number. The time will be in 60ths of a second.

```
    1Ø PRINT PEEK(18)*256*256+PEEK(1Ø)*256+PEEK(2Ø)
    2Ø GOTO 1Ø
```

Can you make the time print in seconds?

Try this to RESET the CLOCK to 0.

```
    5 POKE18,Ø:POKE19,Ø:POKE2Ø,Ø
```

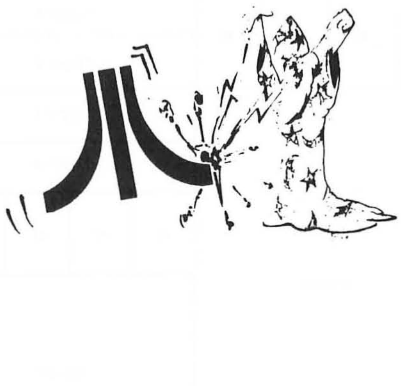
Appendix 1

Description of Memory Addresses

Address	Description
18,19,20	Clock. Address 20 increses by 1 every 60th second.
65	Input/Output sound flag:0=quiet
77	Screen color shift flag: 128=on
82,83	Left, Right margin (normally 2,39)
84	Current cursor row (GRAPHICS 0)
85,86	Current cursor column
87	GRAPHICS MODE number for screen output
88,89	Upper-left hand screen corner address
106	Size of memory in 256 byte pages
201	Print-tab width (normally 10)
559	Enable Player Missile DMA control: 62=single line resolution, 46=double line resolution.
560,561	Low and High bytes of starting address of display list
649	Reverse video flag: 0=normal, 128=reverse
702	Caps-lock flag: 0=1 lower-case, 64=upper-case, 128=control characters
704	Color of Player 0
705	Color of Player 1
706	Color of Player 2
707	Color of Player 3
708	Color of Playfield 0
709	Color of Playfield 1
710	Color of Playfield 2
711	Color of Playfield 3
712	Color of Playfield 4

752	Cursor inhibit: 0=visible, 1 =invisible
755	Character mode: 1 =Blank, 2=Normal, 3=Flip
756	Character base register: 226=lower-case 224=upper-case
764	Last key pressed (internal code)
53248	Horizontal position of Player 0
53249	Horizontal position of Player 1
53250	Horizontal position of Player 2
53251	Horizontal position of Player 3
53252	Horizontal position of Missile 0
53253	Horizontal position of Missile 1
53254	Horizontal position of Missile 2
53255	Horizontal position of Missile 3
53256-53259	Players 0-3 Size: 0 or 2 = Normal size 1 = twice normal size 3 = 4x normal size
53260	Missiles 0-3 sizes
53277	Player Missile enable, 3=on
54018	Recorder motor control line: 52=On 60=Off
54279	Holds page number of Player Missile data

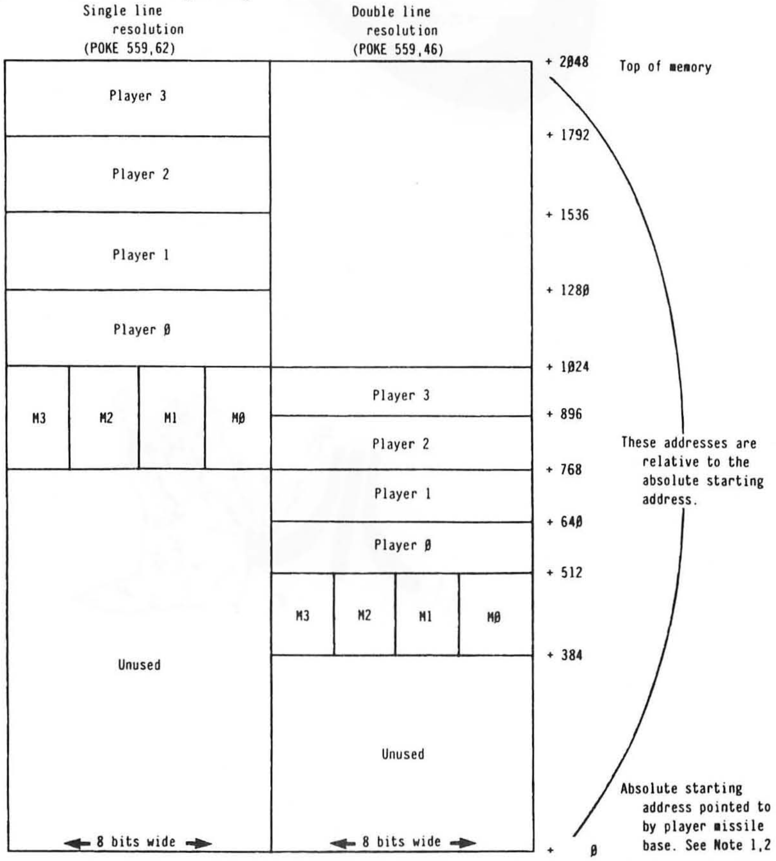
ATARI 400
ATARI 800



Appendix 2

Player Missile Memory Map

Figure 6.1 *Memory Map*



Single line resolution (POKE 559,62)
Double line resolution (POKE 559,46)
Player 3
Player 2
Player 1
Player Ø
M3
M2
M1
MØ
Unused
8 bits wide
Player 3
Player 2
Player 1
Player Ø
M3
M2
M1
MØ
Unused
8 bits wide
+ 2Ø48
Top of memory
+ 1792
+ 1536
+ 128Ø
+ 1Ø24
+ 896
+ 768
+ 64Ø
+ 512
+ 384
+ Ø
These addresses are relative to the absolute starting address.
Absolute starting address pointed to by player missile base. See Note 1,2

Notes:

1. The player missile base address is referenced by page number. One page equals 256 bytes. The starting address is the player missile page number times 256 bytes.
2. The player missile starting address must be on a 2K memory boundary for single line resolution players, and on 1K boundary for double line resolution players.
3. Each player missile section maps directly onto the total height of the TV screen.
4. Refer to Appendix 1 for a description of "Player Missile Memory Addresses."

Appendix 3

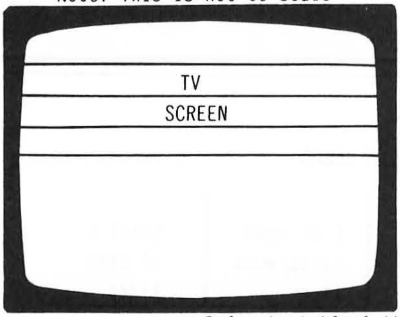
Building a Display List

As a demonstration of the procedure to build a display list, the following screen display will be produced.

Note: This is not to scale

Figure 6.2 *Display List*

Note: This is not to scale



```
(Mode Ø is graphics Ø)

Mode Ø - 2 mode lines
Mode 1 - 2 mode lines
Mode 2 - 2 mode lines
Mode 3 - 4 mode lines

Mode 7 - 48 mode lines
```

Refer to table 1 throughout this procedure.

Table 1. *Graphics Mode Byte Codes*

mode byte	scan lines per mode line	mode byte	RAM per line
Ø	8	2	4Ø
1	8	6	2Ø
2	16	7	2Ø
3	8	8	1Ø
4	4	9	1Ø
5	4	1Ø	2Ø
6	2	11	2Ø
7	2	13	4Ø
8	1	15	4Ø

Table 1. Graphics mode byte codes

Procedure:

1. Determine the number of mode lines such that the total number of scan lines 192. Refer to the scan line requirements in Table 1.

Figure 6.3

Table 1. Graphics mode byte codes

Mode	# of mode lines		# of scan lines/mode line		Total # of scan lines
Ø	2	x	8	=	16
1	2	x	8	=	16
2	2	x	16	=	32
3	4	x	8	=	32
7	48	x	2	=	96
					Total = 192

2. Determine the mode byte codes for each graphics mode. Refer to Table 1.

Figure 6.4

Mode	Mode byte number
Ø	2
1	6
2	7
3	8
7	13

3. Calculate the LMS (Load Memory Scan) byte. Add 64 to the mode byte number for the graphics mode at the top of the screen (MODE 0 in this case).

```
    LMS byte = 64 + 2 = 66
```

4. The display list is ready to be built. First, execute the BASIC graphics command that requires the most memory that you will need. In this example it would be GRAPHICS 7+16. Here is a table of memory requirements

Figure 6.5 *Graphics Mode*

Mode	REM required
8+16	8138 bytes
8	8112
7+16	42ØØ
7	419Ø
6+16	2184
6	2174
5+16	1176
5	1174
4+16	696
4	694
3+16	432
3	434
2+16	42Ø
2	424
1+16	672
1	674
Ø	992

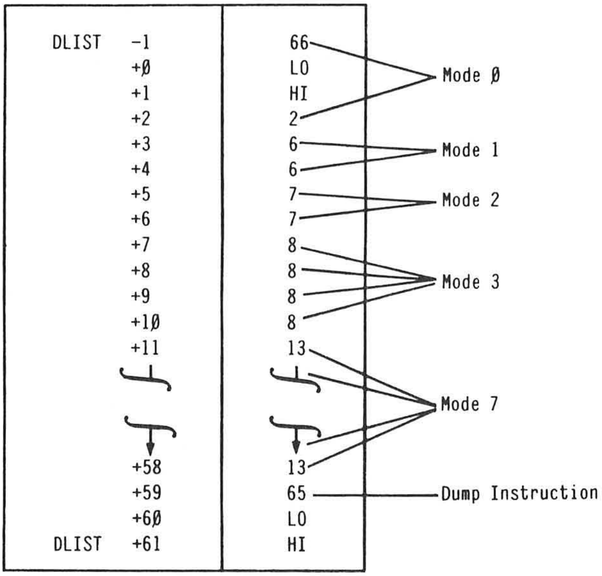
5. Calculate the display list pointer.
```
DLST = PEEK(56Ø)+PEEK(561)*256+4
```
6. Poke the LMS byte.
```
POKE DLIST-1,66
```
7. Poke a mode byte for all corresponding mode lines.

For mode.
```
Ø        POKE DLIST+2,2
1        POKE DLIST+3,6:POKE DLIST+4,6
2        POKE DLIST+5,7:POKE DLIST+6,7
3        FOR X=DLIST+7 TO DLIST+1Ø:POKE X,8:NEXT X
```
Note: The first line is part of the LMS byte.

The MODE 7 display data is already correct because of the execution of the GRAPHICS 7+16 statement.

8. Since the display list is actually a short machine code program, the list is finished with a JUMP instruction to the beginning of the list.
```
POKE DLIST+59,65
POKE DLIST+6Ø,PEEK(56)
POKE DLIST+61,PEEK(561)
```
9. Here is a memory map of the completed display list.

Figure 6.6 *Display List*



To write to the screen, first POKE the graphics mode being used into address 87. For example, if you want to print in GRAPHICS MODE 2

```
        Poke 87,2
```

Now you can PRINT or PLOT to the GRAPHICS 2 part of the screen.

10. You may PRINT or PLOT to a mixed mode screen as long as you do not exceed the normal cursor range. The vertical position is found, by counting the number of mode lines from the top of the screen. The horizontal position is found in the normal way for the particular graphics mode being used.

If you want to PRINT or PLOT to a portion of the screen, but the cursor is out of range for that graphics mode, you will have to calculate a screen position and POKE the informaton to the screen. See Appendix 4 "Calculating Screen Position" for details.

Figure 6.6 *Display List*

To write to the screen, first POKE the graphics mode being used into address 87. For example, if you want to print to GRAPHICS MODE 2

POKE 87,2

Now you can PRINT or PLOT to the GRAPHICS 2 part of the screen.

TIP: You may PRINT or PLOT on a mixed mode screen as long as you do not exceed the normal cursor range. The vertical position is found, by counting the number of mode lines from the top of the screen. The horizontal position is found in the normal way for the particular graphics mode being used.

If you want to PRINT or PLOT to a portion of the screen, but the cursor is out of range for that graphics mode, you will have to calculate a screen position and POKE the information to the screen. See Appendix 4 "Calculating Screen Position" for details.

Appendix 4

Calculating Screen Position

1. Calculate the memory address of the upper-left hand corner of the screen (T1). Here are two methods.

```
a) TØ=PEEK(56Ø)+PEEK(561)*256
   T1=PEEK (TØ+4)+PEEK(TØ+5)*256
b) T1=PEEK(88)+PEEK(89)*256
```

2. Calculate the amount of screen memory between the upper-left hand corner and the target position on the screen. Call this SCRMEM.

To calculate SCRMEM, add up the total memory requirements for each mode line above the target position and add the horizontal position of the preceeding value. You will need to know the amount of memory required for each mode line above the target line. Refer to Table 1 in Appendix 3.

Example, suppose the entire screen were made of GRAPHICS 2 MODE lines. Suppose you want to print a character 4 lines down, and 2 spaces to the right of the upper-left hand corner. The value for SCRMEM would be:

```
SCRMEM = 3 lines x 2Ø bytes/line + 2
SCRMEM = 62
```

3. To find the memory value representing the target position, add the value from part 1 (upper-left hand corner address) to the value from part 2 (memory between upper-left hand corner and target position).

```
In our example:  T1+62
```

4. Here is an example for you to try in GRAPHICS 0. To POKE a character to the screen, you must POKE the internal code, not the Atari ASCII code.

Example:

```
5   DIM Q$(1)
10  GRAPHICS 0:REM 40 BYTES/LINE
20  T1=PEEK(88)+PEEK(89)*256
30  PRINT"VERTICAL POSITION":INPUT V
40  PRINT"HORIZONTAL POSITION":INPUT H
50  PRINT"LETTER TO BE POKED":INPUT Q$
60  POKE T1+40*V+H,ASC(Q$)-32
70  GOTO 30
```

Here is a Table for finding internal code from ASCII

Figure 6.7 *Internal Code form ASCII*

Atari ASCII Value	Operation
0 - 31	Value + 64
32 - 95	Value - 32
96 - 127	None
128 - 159	Value + 64
160 - 223	Value - 32
224 - 255	None

HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER