TEK INSTRUCTION MANUAL

Part No. 070-4639-00
Product Group 14

4050 SERIES

R12 GRAPHICS ENHANCEMENT ROM PACK

Tektronix®
COMMITTED TO EXCELLENCE

TEK INSTRUCTION MANUAL

Part No. 070-4639-00
Product Group 14

4050 SERIES

R12 GRAPHICS ENHANCEMENT ROM PACK

Please Check for CHANGE INFORMATION at the Rear of this Manual

First Printing MAY 1983

Tektronix®
COMMITTED TO EXCELLENCE

WARNING

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for Class A computing devices pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment Operation of this equipment in a residential area is likely to cause interference in which case the users at their own expense will be required to take whatever measures may be required to correct the interference.

MANUAL REVISION STATUS

PRODUCT: 4051R12/4052R12 Graphics Enhancement ROM Pack

This manual supports the following versions of this product: Serial Numbers B010100 and up.

REV DATE	DESCRIPTION
MAY 1983	Original Issue

CONTENTS

Section 3 (cont)

ILLUSTRATIONS

TABLES

OPERATORS SAFETY SUMMARY

This general safety information is for both operating and servicing personnel. Specific warnings and cautions will be found throughout the manual where they apply, but may not appear in this summary.

TERMS

IN THIS MANUAL

CAUTION statements identify conditions or practices that can result in damage to the equipment or other property.

WARNING statements identify conditions or practices that can result in personal injury or loss of life.

AS MARKED ON EQUIPMENT

CAUTION indicates a personal injury hazard not immediately accessible as one reads the marking, or a hazard to property including the equipment itself.

DANGER indicates a personal injury hazard immediately accessible as one reads the marking.

SYMBOLS

IN THIS MANUAL.


This symbol indicates where applicable cautionary or other information is to be found.

AS MARKED ON EQUIPMENT


DANGER high voltage.


Protective ground (earth) terminal.

ATTENTION—refer to manual.


Refer to manual

POWER SOURCE

This product is designed to operate from a power source that does not apply more than 250 volts rms between the supply conductors or between either supply conductor and ground. A protective ground connection by way of the grounding conductor in the power cord is essential for safe operation.

GROUNDING THE PRODUCT

This product is grounded through the grounding conductor of the power cord. To avoid electrical shock, plug the power cord into a properly wired receptacle before connecting to the power input or output terminals. A protective ground connection by way of the grounding conductor in the power cord is essential for safe operation.

DANGER ARISING FROM LOSS OF GROUND

Upon loss of the protective-ground connection, all accessible conductive parts (including knobs and controls that may appear to be insulating) can render an electric shock.

USE THE PROPER POWER CORD

Use only the power cord and connector specified for your product.

Use only a power cord that is in good condition.

Refer cord and connector changes to qualified service personnel.

USE THE PROPER FUSE

To avoid fire hazard, use only the fuse specified in the parts list for your product, and which is identical in type, voltage rating, and current rating.

Refer fuse replacement to qualified service personnel.

DO NOT OPERATE IN EXPLOSIVE ATMOSPHERES

To avoid explosion, do not operate this product in an atmosphere of explosive gases unless it has been specifically certified for such operation.

DO NOT REMOVE COVERS OR PANELS

To avoid personal injury, do not remove the product covers or panels. Do not operate the product without the covers and panels properly installed.

DO NOT OPERATE PLUG-IN UNIT WITHOUT COVERS

To avoid personal injury, do not operate this product without covers or panels installed. Do not apply power to the plug-in via a plug-in extender.

SERVICE SAFETY SUMMARY

FOR QUALIFIED SERVICE PERSONNEL ONLY

Refer also to the preceding Operators Safety Summary.

DO NOT SERVICE ALONE

Do not perform internal service or adjustment of this product unless another person capable of rendering first aid and resuscitation is present.

USE CARE WHEN SERVICING WITH POWER ON

Dangerous voltages may exist at several points in this product. To avoid personal injury, do not touch exposed connections and components while power is on.

Disconnect power before removing the power supply shield, soldering, or replacing components.

DO NOT WEAR JEWELRY

Remove jewelry prior to servicing. Rings, necklaces, and other metallic objects could come into contact with dangerous voltages and currents.

X-RADIATION

X-ray emission generated within this instrument has been sufficiently shielded. Do not modify or otherwise alter the high voltage circuitry or the CRT enclosure.

POWER SOURCE

This product is intended to operate from a power source that will not apply more than 250 volts rms between the supply conductors or between either supply conductor and ground. A protective ground connection by way of the grounding conductor in the power cord is essential for safe operation.

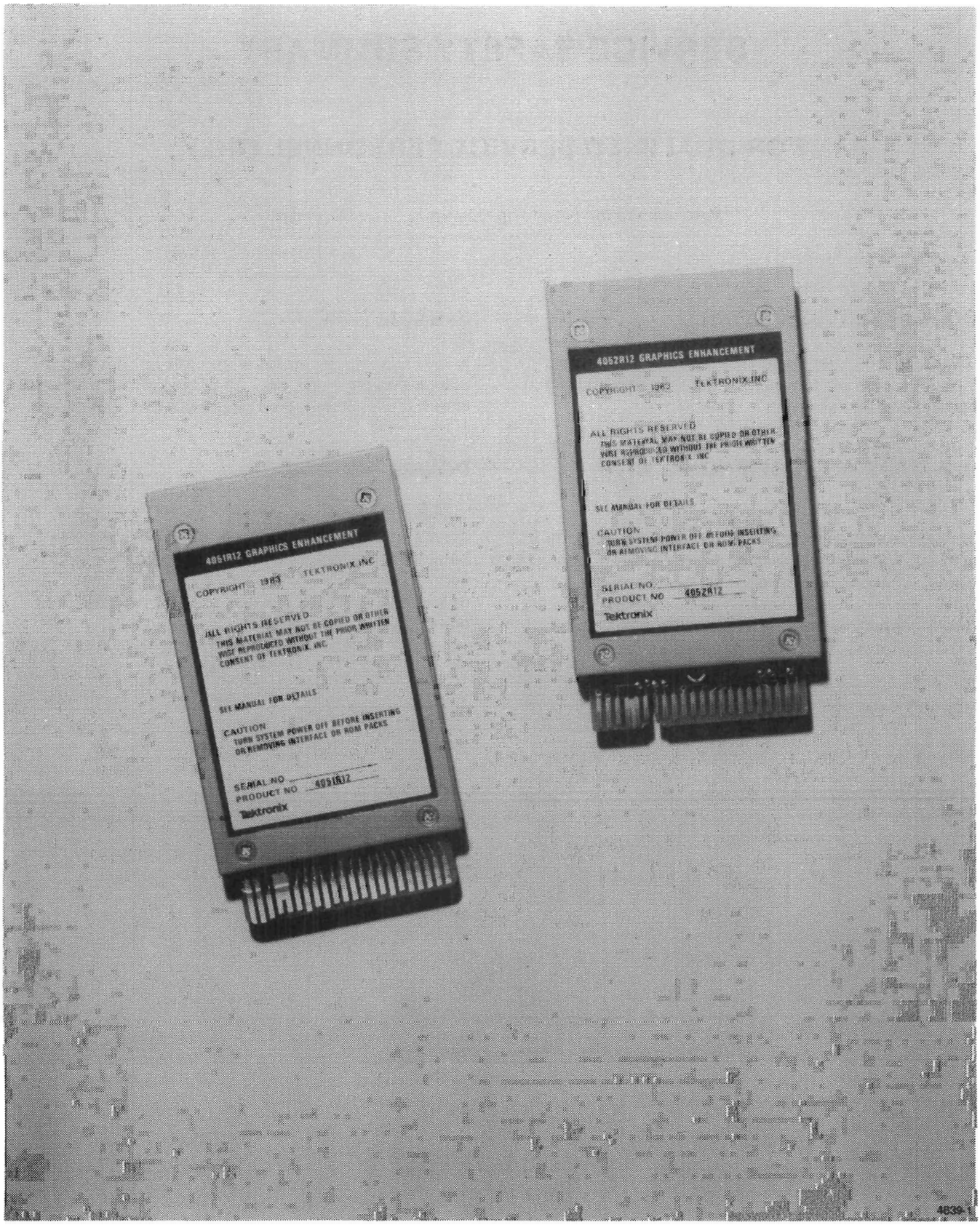


Figure 1-1. 4050 Series Graphics Enhancement ROM Pack.

Section 1

GENERAL INFORMATION

INTRODUCTION

The TEKTRONIX 4050 Series R12 Graphics Enhancement ROM Packs (Figure 1-1) are read-only memory (ROM) devices that enhance the interactive graphics capability of the TEKTRONIX 4050 Series Graphic Computing Systems. The major features in these ROM packs are:

- Sixty-four high level graphics commands for refresh drawing, image manipulation, and programmable sound generation. The ROM pack commands can locate cursors and image points; display cursors, images, and text; create or modify images; and create, print, and sound special character strings.
- More flexible keyboard input.
- Greatly increased speed in graphic writing due to improved storage density (capacity) of complex graphic images. They are encoded into binary form and packed into ASCII character strings.
- Capability to easily convert the standard coordinates of images from floating point to a packed string format, as required by this ROM Pack for graphic display. Another command provides the opposite conversion to standard coordinates.
- Capability to manipulate, save, and read images in string form from magnetic tape files.
- Full-screen crosshair cursor (from screen edge to screen edge) on the Graphic System display, when called by any prefixed CROSS command (e.g., CALL "JCROSS"). You can manipulate the cursor without interfering with the displayed graphics.
- All image display commands restore the graphic cursor to the position it had before you executed the image display call. Thus, you can independently maintain the graphic cursor and the joystick cursor.
- Use of all of the 4050 Series BASIC string functions with string-form images, in order to locate a vector, replace or extract part of an image, or concatenate images. You can concatenate standard two-dimensional operations (scale, translate, rotate, shear, and taper) for multiple transformations. Scaling is applicable to the x- and y-directions independently.
- Generation of music through a pitch range of eight octaves. Ten tempos are available. Duration may vary from a sixty-fourth to a whole note.

Use the 405X R12 demonstration magnetic tape (located inside the manual binder) to preview the graphics enhancement of the ROM Pack.

MATERIAL COVERED

This manual provides the following information:

- Section 1 includes the ROM Pack specifications and installation instructions.
- Section 2 contains a tutorial program to acquaint you with various command routines and capabilities.
- Section 3 contains an overview of command functions and a detailed description of each command. The command descriptions include syntax forms and descriptive forms.
- Section 4 contains the replaceable parts lists (electrical and mechanical) for each ROM pack, and an exploded view diagram.
- Section 5 provides the circuit board layout of component locations and the circuitry schematic for each ROM pack.
- Appendix A is a summary of the ROM pack commands, listed alphabetically. Appendix B lists a few of the more common error messages. Appendix C gives information on the format of the three byte string data items, and Appendix D contains the ASCII Code Chart. Appendix E contains a wide range of graduated transformations which the Graphics Enhancement commands draw.

DOCUMENTATION

This manual, the *4050 Series Graphics Enhancement ROM Pack Instruction Manual*, and the related demonstration tape, are the only standard accessories.

There are helpful programming instructions and basic operating information in the following manuals, but this manual supercedes any conflicting data on command routines.

- *PLOT 50 Introduction to Graphic Programming in BASIC Instruction Manual*
- *4050 Series Graphic System Operators Manual*
- *4050 Series Graphic System Reference Manual*

The ROM packs require no adjustments or preventative maintenance. This instruction manual contains schematics for the ROM packs, but you should refer to the following manuals for a general description of ROM pack circuitry:

- *4051 Service Manual, Volume 1*
- *4052/4054 Technical Data Service Manual*

SPECIFICATIONS

The statements under the heading "Performance Requirements" in Tables 1-1 and 1-2 define characteristics in quantitative terms of performance.

Environmental Characteristics

The 4050 Series R12 Graphics Enhancement ROM Packs meet all environmental specifications (Table 1-1) of the 4050 Series Graphic Computing Systems.

Table 1-1

ENVIRONMENTAL CHARACTERISTICS

Characteristics	Performance Requirements
Temperature	
Operating	+50°F to +104°F (+10°C to +40°C)
Nonoperating	–40°F to +149°F (–40°C to +65°C)
Altitude	
Operating	15,000 ft (4,572 m) maximum
Nonoperating	50,000 ft (15,240 m) maximum
Humidity	
Operating	0% to 80% noncondensing
Nonoperating	0% to 95% noncondensing

Physical Characteristics

Table 1-2 gives the ROM pack's dimensions and weight.

Table 1-2

PHYSICAL CHARACTERISTICS

Characteristics	Supplemental Information
Dimensions[a]	
Length	4.7 in (119 mm)
Width	2.62 in (66.5 mm)
Depth	0.875 in (22.2 mm)
Weight	about 8 oz (230 gm)

[a] **Includes edge-board connector which extends beyond housing.**

INSTALLING THE ROMPACK

Procedure:

1. Be sure the Graphic System power switch is OFF.

CAUTION

Inserting the ROM pack while the Graphic System power is ON can damage the ROM pack. The Graphic System memory contents may also be lost. Turn the power OFF before inserting the ROM pack.

2. Insert the Graphics Enhancement ROM Pack into one of the slots in the rear of the Graphic System, as shown in Figure 1-2. Press down and gently rock the plastic case until the ROM pack edge connector is firmly seated in the backpack slot.

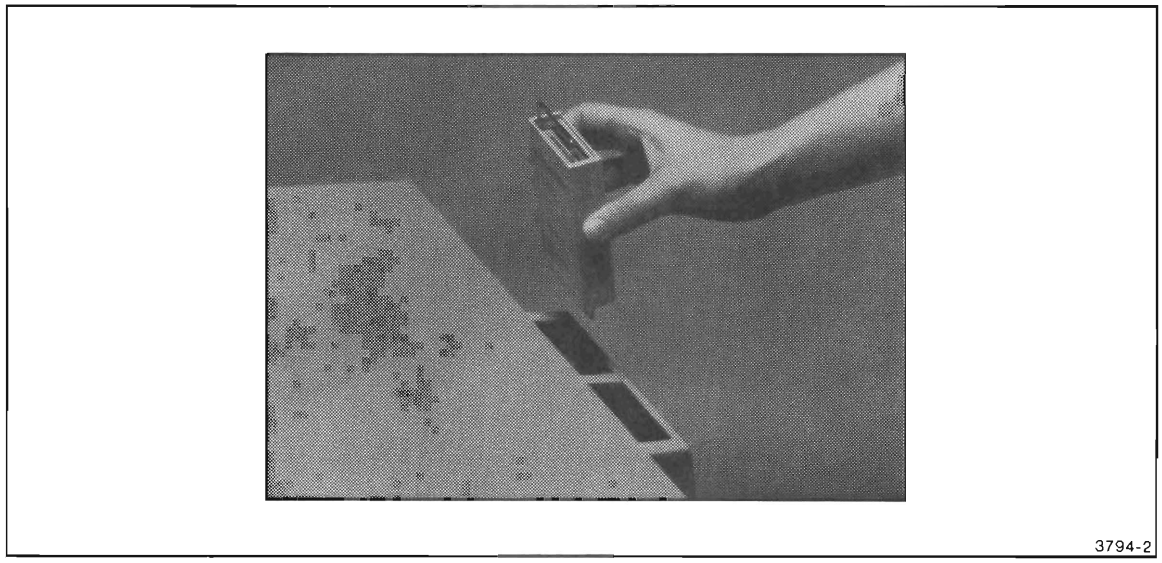


Figure 1-2. Installing the ROM Pack in the Graphic System.

3. Turn the power switch on and wait a few seconds for the system to warm up.

Now you are ready to try some of the ROM pack commands. The Tutorial follows this section. But first, a word about how to adjust the refresh brightness.

REFRESH BRIGHTNESS ADJUSTMENT

You may vary the brightness of the refresh (blinking) cursor by adjusting the cursor intensity trimmer potentiometer (see Figure 1-3). Safely tilt the 4051 or 4052 Graphic System on its side, and use a plastic slotted adjustment tool to adjust R683 until the cursor just begins to store on the screen. To check for storage and to find the optimum point, move the cursor across the screen by pressing the space bar while making the adjustment. The 4054 pot adjustment (labeled Write Thru) is to the left of the main power switch.

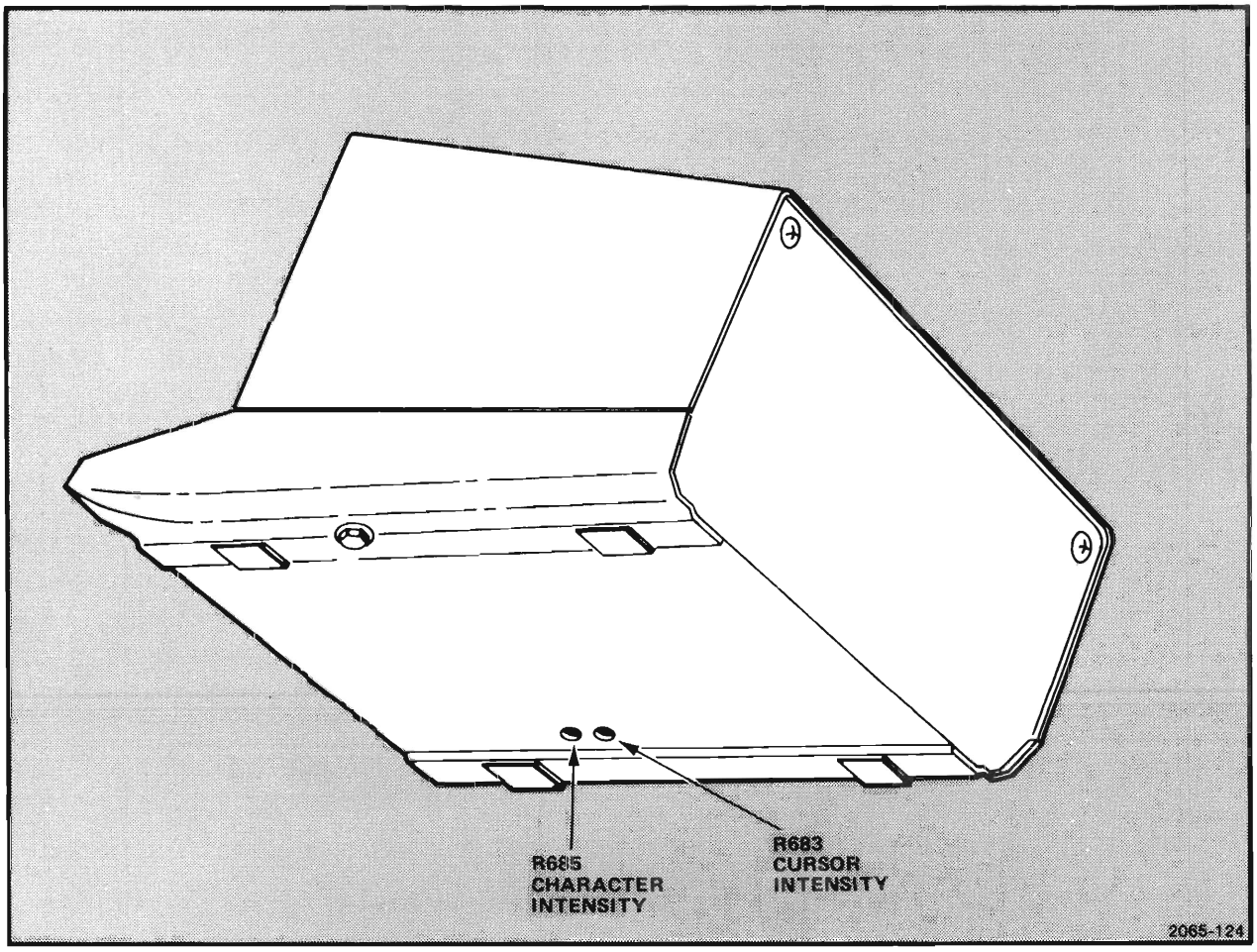


Figure 1-3. Intensity Adjustment.

Section 2

TUTORIAL

PURPOSE

This tutorial acquaints you with the Graphics Enhancement ROM Pack commands and capabilities. This section does not include all the commands, but by doing the following exercises you will sample features of different manipulative commands. Use the demonstration tape to exercise tutorial functions.

TUTORIAL

The following tutorial takes about 30 minutes to do. If you do not complete it at one sitting, and you turn the graphic system off, you lose your previous entries. Therefore, either plan a straight-through tutorial session or save the entered portion on a magnetic tape. Do not skip around in the tutorial; take each section in the order given.

The tutorial is treated in the following order of program segments:

Lines 1 to 160	Drawing a BASIC Rectangle.
Lines 170 to 250	Changing Floating Point to Image String.
Lines 260 to 300	Using Relative Draw (CALL "RDRAW").
Lines 310 to 380	Using Relative Rotate (CALL "RROTATE").
Lines 390 to 560	Using Relative Scale (CALL "RSCALE").
Lines 570 to 640	Using Relative Shear (CALL "RSHEAR").
Lines 650 to 770	Using Relative Taper (CALL "RTAPER"):

Drawing a Basic Rectangle

After installing the Graphics Enhancement ROM Pack and turning on the Graphic Computing System, press the HOME PAGE key. Enter the following BASIC program, lines 1 through 160, on the keyboard to draw a rectangle on the screen (see Figure 2-1). Press Autoload if using the demonstration tape's tutorial.

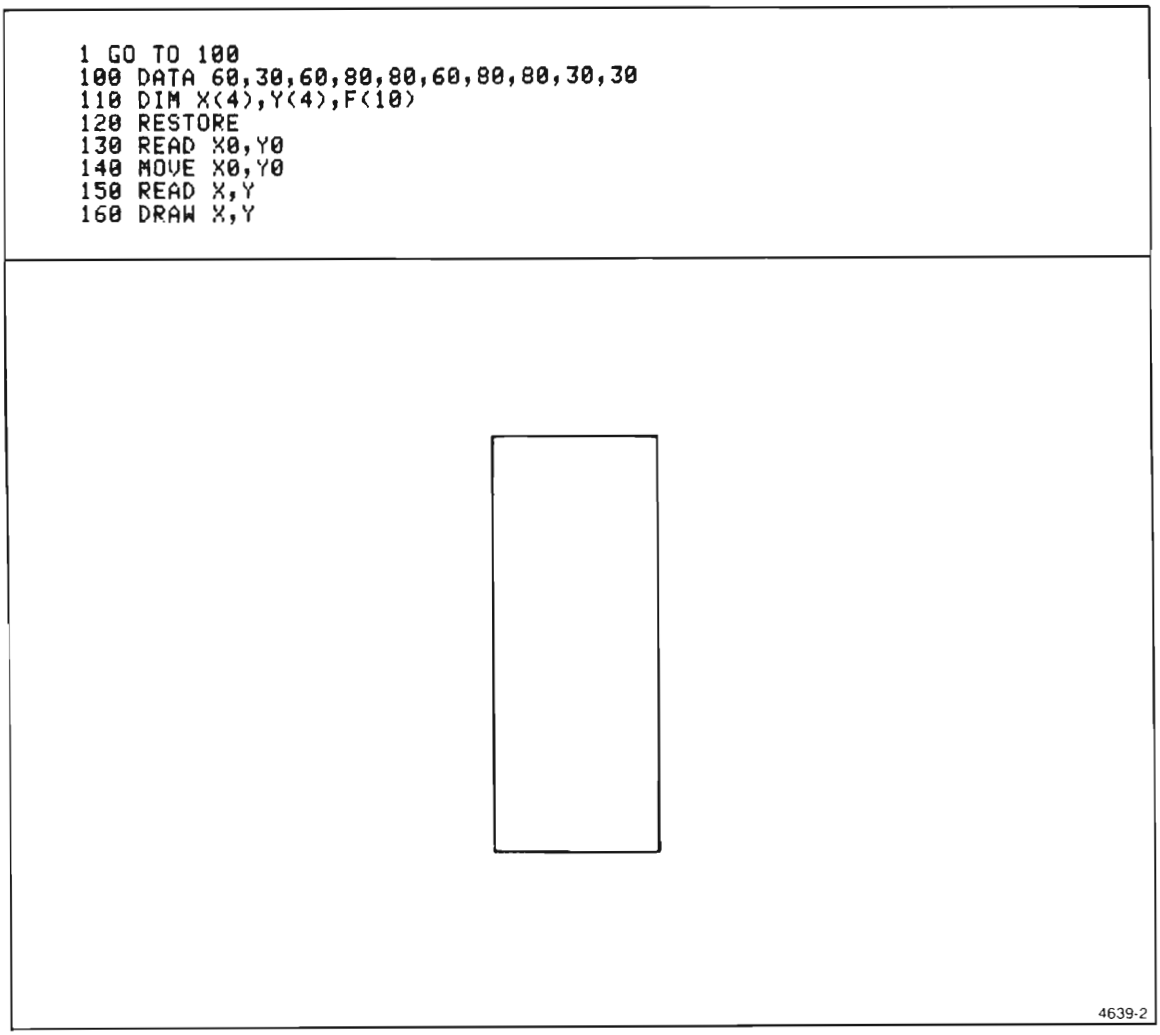


Figure 2-1. BASIC Rectangle.

Explanation

This is one of many ways to DRAW an image. As you will see in the *Command Description* section, an image drawn in regular BASIC is easily convertible to the form used by the Graphics Enhancement ROM pack, once you put the X,Y data points in an array.

Line 100 enters the data for MOVES and DRAWS. The first number in the data statement is the value of X0; the second is Y0. The next four numbers are X values; the last four are Y.

Line 110 sets up the variables in a dimension statement, and line 120 reads from the start of data statement.

Line 130 reads the start position, and line 140 moves to the start position. Line 150 reads X and Y, then line 160 draws X and Y.

Changing Floating Point to Image String

Now enter the following program, lines 170 through 250, to convert the image from floating point to image string format:

```
170 F(1)=X0
180 F(2)=Y0
190 FOR I=3 TO 10 STEP 2
200 F(I)=X(INT(I/2))
210 F(I+1)=Y(INT(I/2))
220 NEXT I
230 CALL "CHANGE",F,I$
240 N=1
250 CALL "TOGGLE",I$,N
```

Explanation

Lines 170 and 180 set the first two elements of array F to the destination point. Lines 190 through 220 fill the remainder of array F as X,Y,X,Y,X,Y,X,Y.

Line 230 converts the array to an image string. Lines 240 and 250 change the first point to MOVE.

The ROM pack stores MOVEs and DRAWs as three character strings. A Graphics Enhancement MOVE or DRAW takes only three characters (bytes) of a string, allowing you to put together many MOVEs and DRAWs in one long string.

Using just three bytes of memory for MOVEs and DRAWs means you can store more complex images. (A numeric array element takes eight bytes of memory for each X and each Y value — a total of 16 bytes.) However, to gain this advantage limits the display resolution of MOVEs and DRAWs to 1024 x 768. This is also true of the 4054 screen.

Using Relative Draw (CALL ''RDRAW'',I$,C,X,Y)

Now enter lines 260 through 300 to redraw the rectangle in four successive locations relative to and overlapping the original location — to the northeast, northwest, southwest, and southeast (see Figure 2-2).

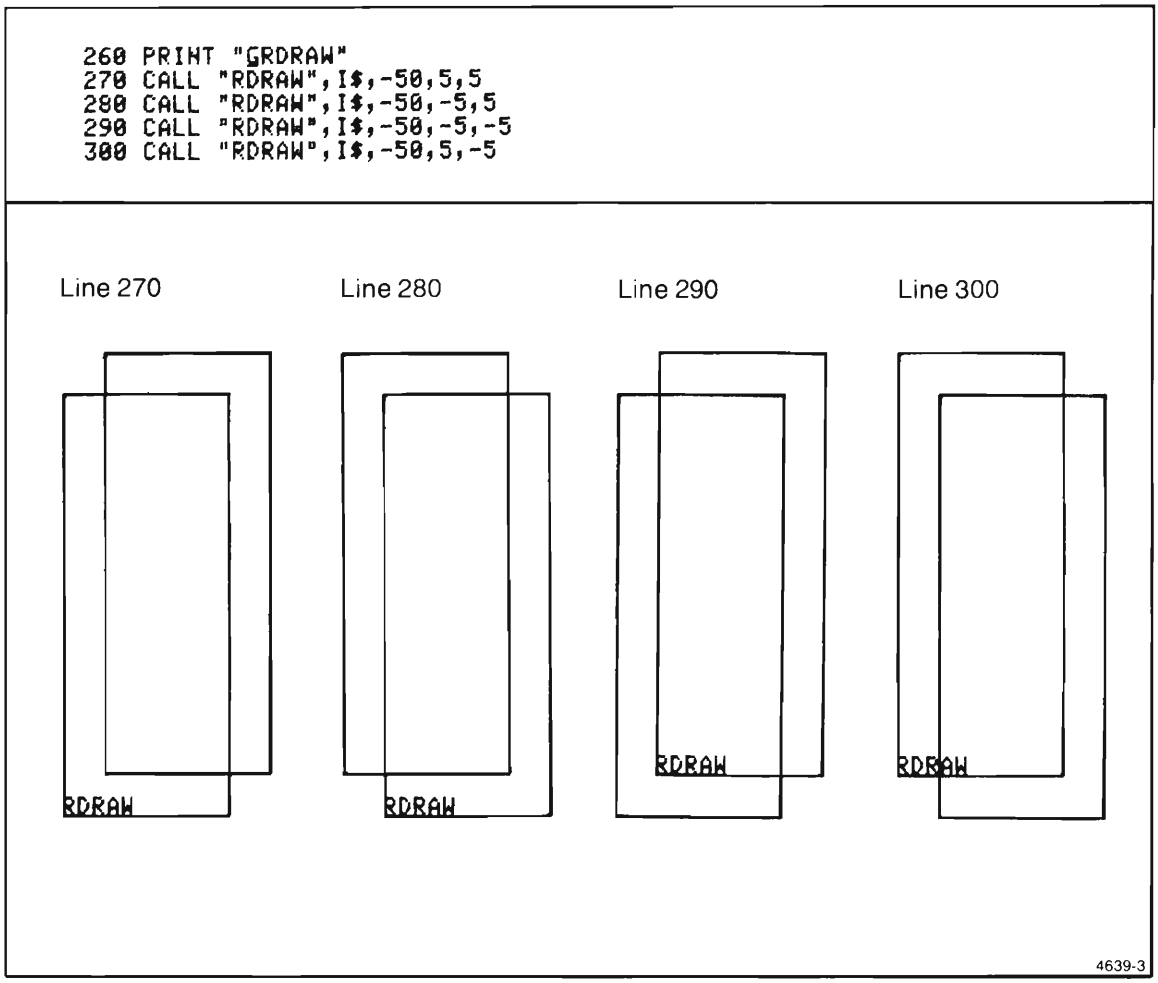


Figure 2-2. RDRAW Images

Explanation

The rectangle image I$ is defined to start at 60,30 (the first MOVE and DRAW, lines 100-160). RDRAW draws the image relative to where it is defined. The last two parameters (given as ±5 in lines 270 through 300) are the X,Y offset, from where the rectangle was defined to be, to where you want it to be.

The statement CALL ''RDRAW'',I$,–50,0,0 draws the image 50 times, starting at its original location. The negative sign on the 50 means the image does not store on the screen; the positive sign would draw the image in storage 50 times.

Using Relative Rotate (CALL ''RROTATE'',I$,A,X,Y)

Now enter lines 310 through 380 to rotate the rectangle 19 times counterclockwise at 20-degree increments. See Figure 2-3.

```
310 PRINT "GRROTATE"
320 C$=I$
330 SET DEGREES
340 FOR I=0 TO 360 STEP 20
350 CALL "RROTATE",I$,20,0,0
360 CALL "RDRAW",I$,-5,0,0
370 NEXT I
380 I$=C$
```

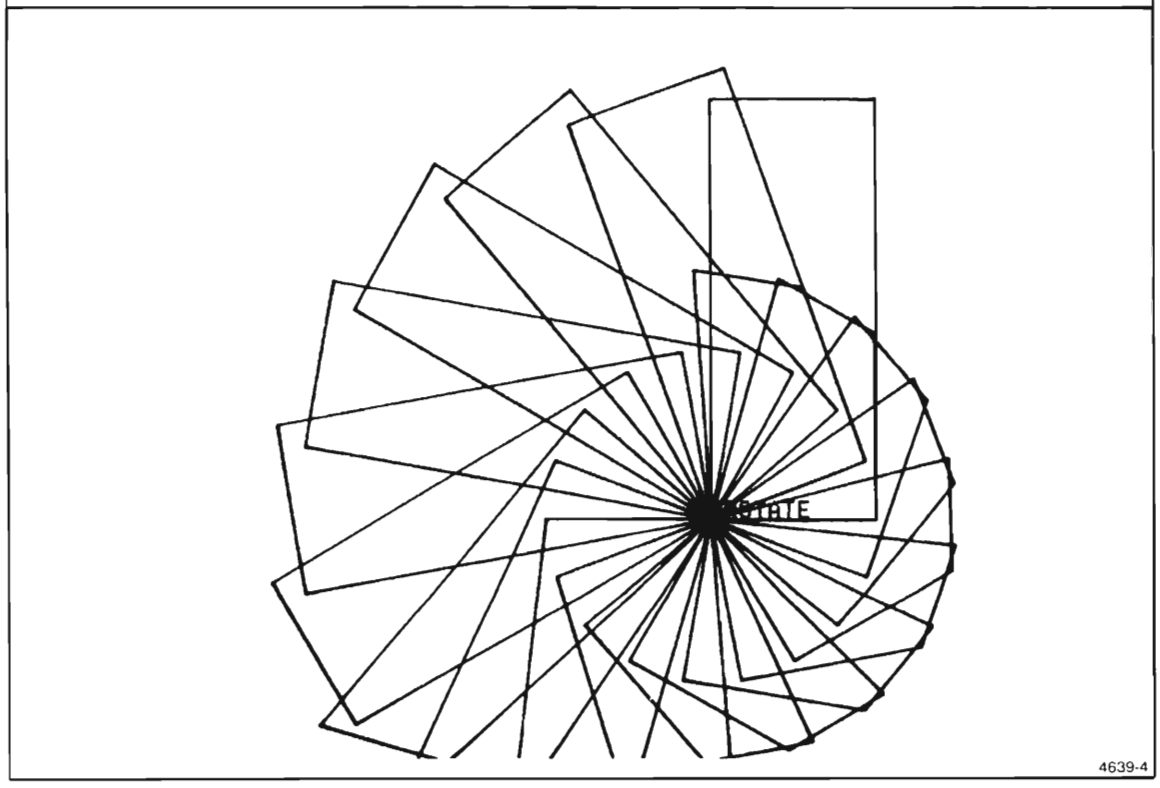


Figure 2-3. RROTATE Images.

Explanation

RROTATE changes the image string I$ by rotating it about the X,Y point you specify, according to the angle given. As with other Graphics Enhancement commands, if the image is changed to exceed the boundaries of the screen, the image is limited (not clipped) to the screen. The X and Y coordinate values are limited independently, as appropriate.

Because of the 1024 x 768 resolution, each rotation causes an approximation of the image. Repeated transformation like that done above with RROTATE can cause degradation of the image.

Here is an example program that keeps a backup image for use as input to each rotation in order to minimize image degradation.

```
C$ = I$                          (This copies the image)
FOR A = 0 TO 360 STEP 20
CALL "RROTATE",I$,A,0,0
CALL "RDRAW",I$,-5,0,0
I$ = C$                          (This restores the image.)
NEXT A
```

Using Relative Scale (CALL "RSCALE",I$,H,V,X,Y)

Now enter lines 390 through 560 to scale the rectangle through different transformations. See Figure 2-4.

```
390 PRINT "GRSCALE"
400 X1=1.1
410 Y1=1
420 GOSUB 510
430 X1=1/1.1
440 GOSUB 510
450 X1=1
460 Y1=1.1
470 GOSUB 510
480 Y1=1/1.1
490 GOSUB 510
500 GO TO 570
510 REM CUMULATIVELY RESCALE AND REDRAW THE IMAGE
520 FOR I=1 TO 10
530 CALL "RSCALE",I$,X1,Y1,0,0
540 CALL "RDRAW",I$,-5,0,0
550 NEXT I
560 RETURN
```

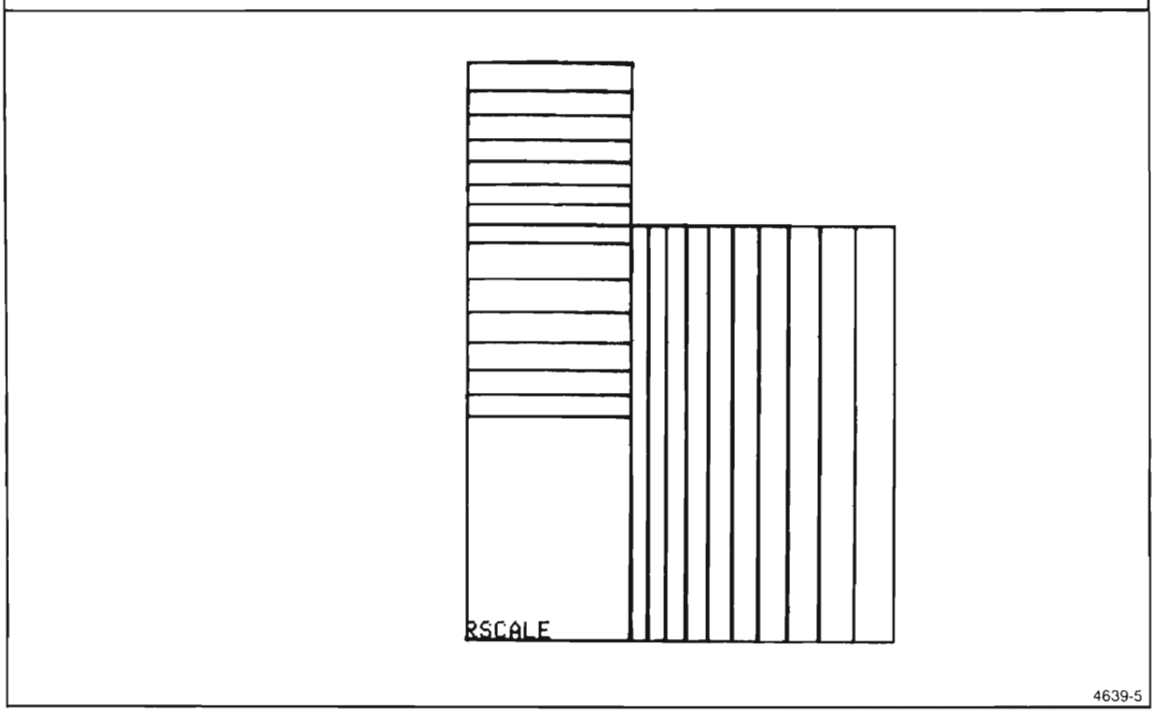


Figure 2-4. RSCALE Images.

Explanation

You can scale images independently in X and Y around any X,Y point. Observe that, when the scaled rectangle moves partially off the screen, its emerging image is different from its preceding image. If you scale an image down so small that it loses detail, and then expand it, the new image looks different.

Using Relative Shear (CALL ''RSHEAR'',I$,H,V,X,Y)

Now enter lines 570 through 640 to shear the image through different transformations. See Figure 2-5.

```
570 PRINT "GRSHEAR"
580 I$=C$
590 FOR X1=1 TO 180 STEP 12
600 I$=C$
610 CALL "RSHEAR",I$,X1,0,0,0
620 CALL "RDRAW",I$,-5,0,0
630 NEXT X1
640 I$=C$
```

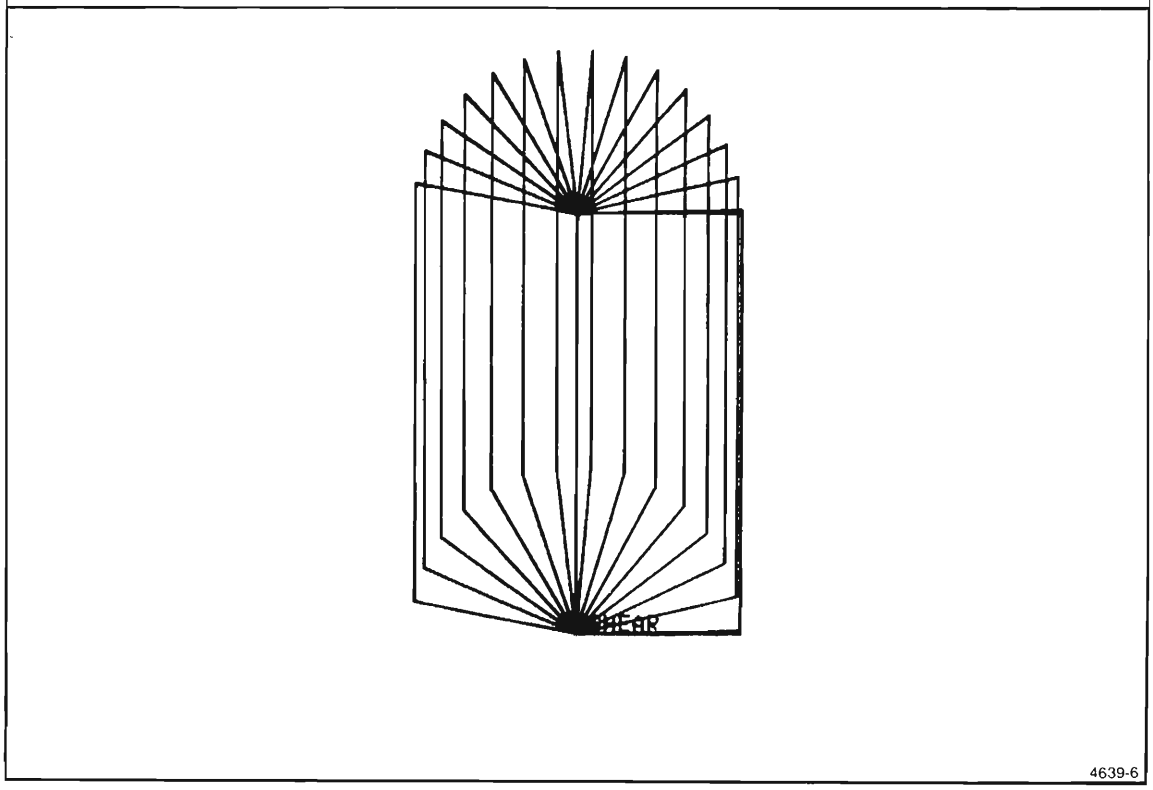


Figure 2-5. RSHEAR Images.

Explanation

RSHEAR rotates the X and Y axes of the image independently. In this case the X axis rotates 90° about the relative image's starting point (0,0), causing the image to flip over like the page of a book.

If you were to rotate both the X and Y axes the same degrees (CALL ''RSHEAR'',I$,X1,Y1,0,0,0), the result would resemble the first half of the earlier RROTATE example.

Using Relative Taper (CALL ''RTAPER'',I$,H,V,X,Y)

Changing Taper in Y

Now enter lines 650 through 700 to change RTAPER in Y (see Figure 2-6a).

Taper with Moving Y Center

Now enter lines 710 through 770 to move the taper center from the bottom to the top of the screen. Refer to Figure 2-6b.

```
710 PRINT "GRTAPER WITH MOVING Y CENTER"
720 I$=C$
730 FOR P=-30 TO 60 STEP 15
740 I$=C$
750 CALL "RTAPER",I$,0,1,10,P
760 CALL "RDRAW",I$,-5,0,0
770 NEXT P
```

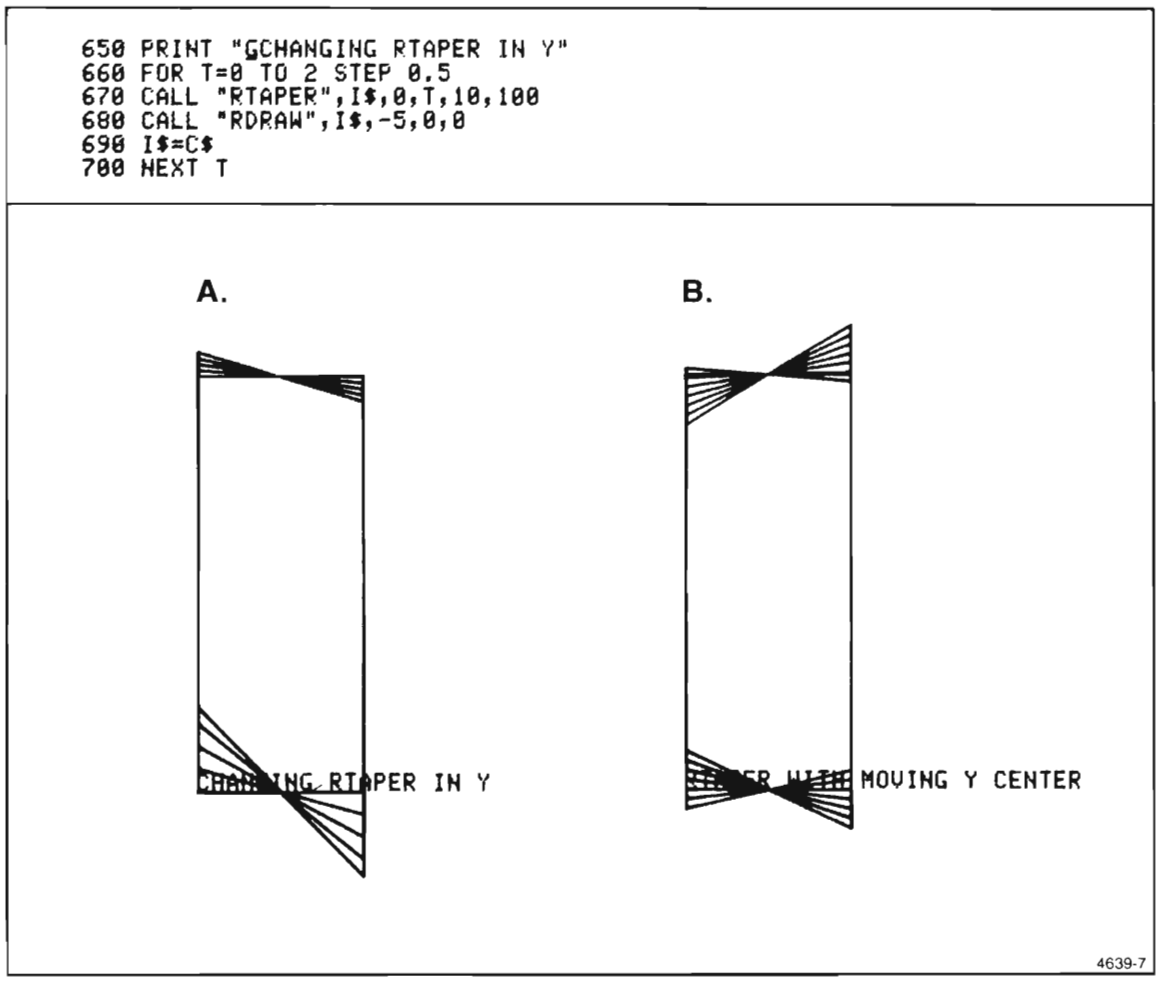


Figure 2-6. RTAPER Images.

Explanation

RTAPER scales the X and Y values of a coordinate according to the taper factor and the absolute X,Y distance of the coordinate from the taper center. Positive taper factors move a point that is above and right of the taper center further above and further right, and a point below and left of the taper center further below and further left. Negative taper factors reverse this effect.

The purpose of RTAPER is to provide pseudo-perspective. You may have no application for this command; others may have many applications.

USING THE JOYSTICK

If you have a TEKTRONIX 4952 Joystick, Option 2, to operate with the 4050 Series Graphic System, refer to the demonstration tape for examples of using the joystick. If a joystick is not available, a simulated use appears under ACROSS in the *Command Descriptions* section.

Section 3

COMMAND DESCRIPTIONS

INTRODUCTION

The Graphics Enhancement ROM Pack adds 64 new commands to your TEKTRONIX 4050 Series Graphic Computing System. This section contains an overview of the command functions, as well as a detailed description of each command. The explanations are intended for users who are familiar with the operation of the TEKTRONIX 4050 Series of Graphic Computing Systems. The explanations contain basic information about how you use the commands. Examples help clarify the use of the ROM pack.

COMMAND FUNCTIONS

The ROM pack commands perform four basic types of functions: location, display, image formation/transformation, and special character strings. Table 3-1 groups all command functions under these four categories.

Table 3-1

COMMAND FUNCTIONS BY CATEGORIES

Category	Function	Comments
LOCATION	Locates Graphics cursor Locates Joystick cursor Locates any point or points in any image	Never modifies image strings
DISPLAY	Displays cursors Displays images Displays text	Never modifies image strings
IMAGE FORMATION & TRANSFORMATION[a]	Creates images Modifies images Manipulates images	Can modify image strings
SPECIAL CHARACTER STRINGS	Creates special character strings Prints special character strings Sounds special character strings	Can modify strings

[a] **You can use all of the standard 4050 string functions to locate a vector, replace or extract part of an image, or concatenate images.**

COMMAND SYNTAX

The following symbols define the command syntax in this manual:

Symbol	Meaning
< >	Treat the enclosed definition as one element.
[ ]	You have the OPTION of choosing one element or of omitting all.
{ }	You MUST choose one element.
. . .	You can repeat the preceding item (that is, all characters back to the preceding space character or left bracket).
UPPERCASE	Use boldface, uppercase characters exactly as written; treat as literal elements.
lowercase	Do not use lowercase characters as written; they represent a parameter.

CALL STATEMENTS

You access any ROM pack command by specifying the command name in the CALL statement in 4050 Series BASIC. The BASIC programming line number (when used) precedes the command name.

A CALL statement always begins with CALL, followed by the command within quotation marks and a comma (for example, **CALL "JGIN",**). The rest of the command is represented by a combination of string variables or constants, numeric variables or constants, and arrays. A comma separates each of these elements. Refer to the *4050 Series Graphic System Reference Manual* for more information on CALL statements.

You must enter the quotation marks around the command name and insert the commas within the command syntax. For example, the JGIN syntax is:

```
                {numeric constant}{. . .}{. . .}
CALL "JGIN",{numeric variable},{. . .},{. . .},<string variable>
                {array element}{. . .}{. . .}
```

Here is the sample format of the JGIN program statement:

[Line number] **CALL "JGIN"**,T,X,Y,K$

COMMAND FORMS

Graphics Enhancement commands consist of prefixed forms and non-prefixed forms. Table 3-2 defines the four prefixes that the prefixed commands have.

Table 3-2

COMMAND PREFIXES

Prefix	Definition
R	Relative Display Screen
A	Absolute Display Screen
G	Relative Graphics Cursor
J	Absolute Joystick Cursor

IMAGE STRING POSITIONING

Image strings contain absolute MOVEs and DRAWs. Therefore, a string is defined to be at a fixed place on the screen. When it comes time to draw the image, however, you have a number of options in how to specify the location.

Relative Display Screen (Image String Relative)

When drawing an image with a 0,0 offset, the Relative form of a command (with R prefix) draws the image where it was defined to be. THIS IS AN IMPORTANT CONCEPT.

Drawing Relative with an X,Y offset adds X,Y to every point on the image as it is drawn. Thus, while this image has the same configuration as though with a 0,0 offset, it appears at location X,Y relative to where it was defined.

NOTE

The first point in an image becomes the point of reference for all Relative image transformations or operations. See Table 3-3.

Table 3-3

IMAGE POINT LOCATIONS

Command Form	Image Drawn[a]
RELATIVE DISPLAY SCREEN	If first point = Xo,Yo, then image drawn at Xr + Xo, Yr + Yo
ABSOLUTE DISPLAY SCREEN	If first point = Xo,Yo, then image drawn at Xa,Ya
RELATIVE GRAPHICS CURSOR	If first point = Xo,Yo, then image drawn at Xg + Xr, Yg + Yr
ABSOLUTE JOYSTICK CURSOR	If first point = Xo,Yo, then image drawn at Xj,Yj

[a] **r = relative, a = absolute, g = graphic, j = joystick**

Absolute Display Screen

The Absolute form of a command (with the A prefix) calculates the X,Y difference between the given Absolute position and the start (first vector) of the image. This difference is added to every point in the image as it is drawn. Thus, the image looks the same but starts at the Absolute X,Y position that you specify. Refer again to Table 3-3.

Relative Graphics Cursor

The Relative Graphics Cursor form of a command (with the G prefix) is similar to Absolute Display Screen except that the graphics cursor location and the specified offset determine the X,Y difference. (The graphics cursor location is the last point to which you MOVE or DRAW in BASIC.) As a result, the graphics cursor and the X,Y location determine where the image starts. Refer again to Table 3-3.

Absolute Joystick Cursor

The Absolute Joystick Cursor form of a command (with the J prefix) is similar to Relative Graphics Cursor except that the X,Y difference is between the joystick cursor location and the image start. Thus, the image starts where the joystick is. Refer again to Table 3-3.

COMMAND LIST

All of the ROM pack commands (except R12LVL), whether prefixed or nonprefixed, appear in Table 3-4 according to function type. The prefixed commands are further divided by command form.

CALL "R12LVL" simply prints the ROM Pack identification, version level, and copyright statement.

Table 3-4

COMMAND CHART

Function Type	Prefixed Commands				Non-Prefixed Commands
	Relative	Absolute	Graphics	Joystick	
LOCATION	RGIN RPOINT	AGIN APOINT	GGIN GPOINT	JGIN JPOINT	BOUNDS LOCATE
DISPLAY	RCROSS RDOTS RDRAW RINPUT RPRINT	ACROSS ADOTS ADRAW AINPUT APRINT	GCROSS GDOTS GDRAW GINPUT GPRINT	JCROSS JDOTS JDRAW JINPUT JPRINT	DASHED DOTTED RUBBER VERTEX
IMAGE FORMATION & TRANSFORMATION	RMOVE RROTATE RSCALE RSHEAR RTAPER	AMOVE AROTATE ASCALE ASHEAR ATAPER	GMOVE GROTATE GSCALE GSHEAR GTAPER	JMOVE JROTATE JSCALE JSHEAR JTAPER	CHANGE DEFINE IMAGES POINTS TOGGLE
SPECIAL CHARACTER STRINGS					INPUTS MUSIC PRINTS SOUNDS STRING

COMMAND CONDITIONS

You should give careful attention to the following command conditions:

- All X,Y positions are in graphic display units (GDU's).
- Coordinates are limited to default screen (0-130,0-100).
- Commands assume the power-up default window and viewport conditions.
- Any vectors that exceed the default screen boundary are limited to the default screen boundary.
- All graphic display commands restore the graphic cursor. A PAGE in the middle of graphics is followed by a move to the previous graphic position when the graphics routine ends; the cursor does not remain at home.
- Relative image commands are relative to the first image vector.
- Many of the commands allow undefined numeric and undefined string parameters.
- Any undefined string variable is dimensioned to the 4050 BASIC default of 72 characters.
- All defined image strings must have a length of 0 mod 3 (that is, be an exact multiple or integer of 3). Thus, a partial vector (less than three bytes) is not allowed. Also, all defined image strings must have a length greater than 0. Null images are not allowed.

DISPLAY COUNT

Commands that draw images have a repeat parameter named the display count, interpreted as follows:

- If the display count C is greater than or equal to +0.5, you draw the image ABS(C) times in Storage mode.
- If the display count C is $-0.5 < C < +0.5$, you will not draw the image.
- If the display count C is less than or equal to –0.5, you draw the image ABS(C) times in Refresh mode.

NOTE

The repeat count can be a constant or a variable. If any Joystick command uses a variable for the repeat count, then, when the command terminates, the variable is updated to represent the actual repeat count.

COMMAND SYNTAX DESCRIPTIONS

The remainder of this section gives you detailed command descriptions, including functional purpose, syntax form, descriptive form, format example, sample program, and an explanation of the action and arguments of each command. The commands are in alphabetical order, with the command function statement directly under each command.

The syntax form shows the type of parameter; the descriptive form shows the nature, source, or result of the parameter. For example, in describing the ACROSS command, the syntax form of one parameter is listed as "numeric simple/constant argument" whereas the descriptive form of that parameter is listed as "X position." This tells you that the horizontal coordinate position (X) is a simple numeric value (like 65).

ACROSS (Absolute Cross)

This command draws the cross-hair cursor C times at coordinate position X,Y.

Syntax Form

[line number] **CALL "ACROSS"**, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "ACROSS"**, C times of cross-hair cursor, X position, Y position

Format Example

```
500 CALL "ACROSS", C,X,Y
```

Explanation

A crosshair is drawn C times at location X,Y.

- If C $\leqslant$ –0.5, then write-thru occurs.
- If C $\geqslant$ + 0.5, then storage occurs.
- Otherwise, nothing happens.

Where C = 1 (that is, the crosshair cursor is drawn one time, making C greater than + 0.5), the crosshair is stored.

Using the ACROSS format example, CALL "ACROSS",C,X,Y, the sample program (illustrated in Figure 3-1) defines X and Y as 65,50 (lines 240-250), then defines C as –3 in the statement CALL "ACROSS",–3,X,Y. (The one-step parameter definition is: CALL "ACROSS",–3,65,50.) Parameters C, X, and Y for this command must be defined (although some parameters for other commands may be undefined). To enter the cited format example without defining the parameter values produces an error message on the display. See the appendix on error messages.

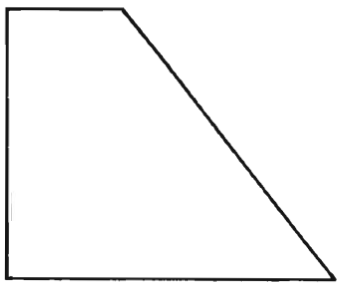
Sample Program

See Figure 3-1.

```
1 REM ACROSS example
2 GO TO 100
4 REM go left
5 X=X-1
6 RETURN
12 REM go up
13 Y=Y+1
14 RETURN
20 REM go right
21 X=X+1
22 RETURN
24 REM generate 'MOVE', with a point to mark it.
25 MOVE X,Y
26 DRAW X,Y
27 RETURN
32 REM go down
33 Y=Y-1
34 RETURN
40 REM generate 'DRAW'
41 DRAW X,Y
42 RETURN
80 END
100 INIT
110 PAGE
120 PRINT "Fcn key  Action"
130 PRINT "    1      Move Cross-hairs to left 1 unit."
140 PRINT "    5      Move Cross-hairs to right 1 unit."
150 PRINT "    3      Move Cross-hairs up 1 unit."
160 PRINT "    8      Move Cross-hairs down 1 unit."
170 PRINT "    6      Generate MOVE."
180 PRINT "   10      Generate DRAW."
190 PRINT "   20      Quit."
200 PRINT
210 PRINT "To move the cross-hairs quickly, hold the function key dow
220 PRINT "    for a while."
230 SET KEY
240 X=65
250 Y=50
260 REM I keep the cross-hairs glowing while you push the keys.
270 REM    There are three refresh passes per ACROSS call.
280 CALL "ACROSS",-3,X,Y
290 GO TO 280
300 END
```

```
Fcn key  Action
    1      Move Cross-hairs to left 1 unit.
    5      Move Cross-hairs to right 1 unit.
    3      Move Cross-hairs up 1 unit.
    8      Move Cross-hairs down 1 unit.
    6      Generate MOVE.
   10      Generate DRAW.
   20      Quit.

To move the cross-hairs quickly, hold the function key down
   for a while.
```



4639-8

Figure 3-1. Example of ACROSS.

ADOTS (Absolute Dots)

This command draws the image dots C times at coordinate position X,Y.

Syntax Form

[line number] **CALL "ADOTS"**, image string argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "ADOTS"**, image dots string, C times, X position, Y position

Format Example

```
500 CALL "ADOTS",I$,C,X,Y
```

Explanation

ADOTS draws the image dots (I$) with the first point at position X,Y.

Sample Program

See Figure 3-2.

```
100 REM ADOTS example
110 INIT
120 REM  Create the image of a box.
130 DIM B(24),B$(36)
140 DATA -50,-50,50,60,60,60,60,50,50,50,-50,-60,58.66,65,68.66,65
150 DATA 68.66,55,60,50,-60,-60,68.66,65
160 READ B
170 CALL "CHANGE",B,B$
180 REM  Now place dots, drawing them once, and repositioning
190 REM    so that the first dot is at 65,50.
200 CALL "ADOTS",B$,1,65,50
210 END
```

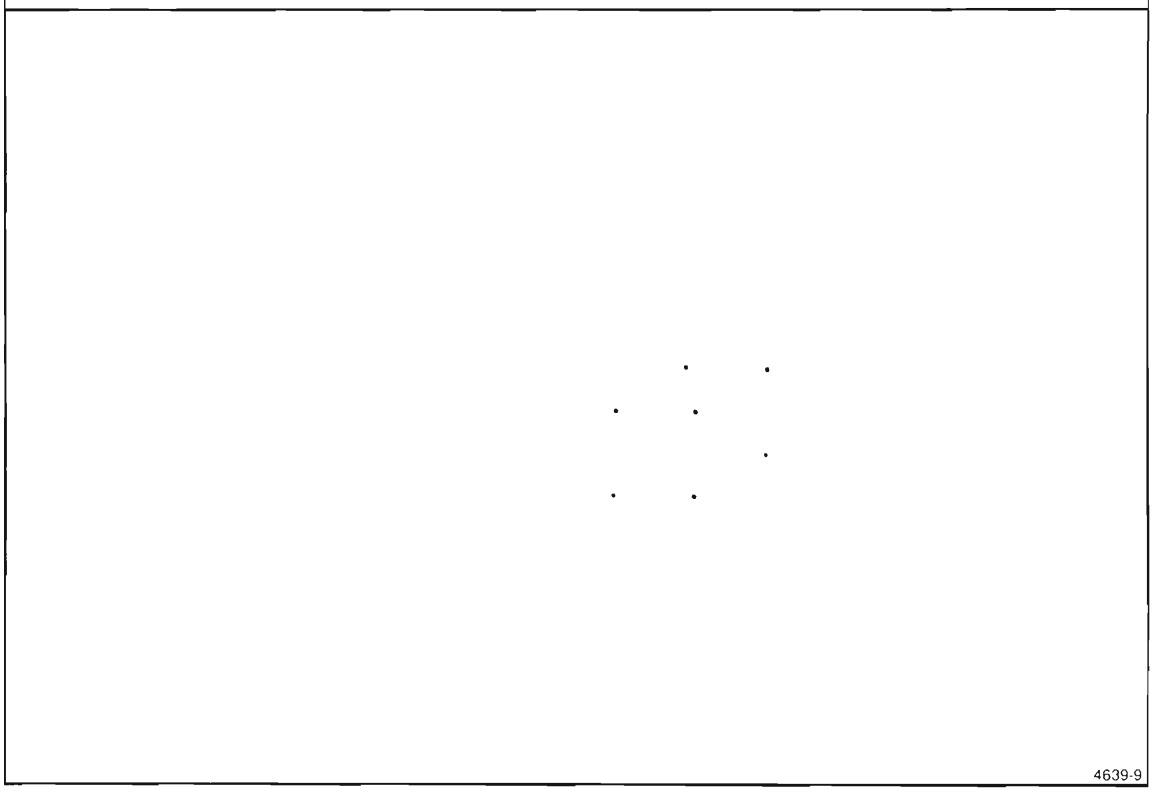


Figure 3-2. Example of ADOTS.

ADRAW (Absolute Draw)

This command draws the image vectors C times at coordinate position X,Y.

Syntax Form

[line number] **CALL "ADRAW"**, image string argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "ADRAW"**, image vectors string, C times, position X, position Y

Format Example

```
500 CALL "ADRAW",I$,C,X,Y
```

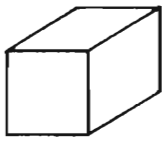
Explanation

ADRAW draws the image vectors with the first image point at the X,Y position. The image string is not altered by this command.

Sample Program

See Figure 3-3.

```
100 REM ADRAW example
110 INIT
120 REM Create the image of a box.
130 DIM B(24),B$(36)
140 DATA -50,-50,50,60,60,60,60,50,50,50,-50,-60,58.66,65,68.66,65
150 DATA 68.66,55,60,50,-60,-60,68.66,65
160 READ B
170 CALL "CHANGE",B,B$
180 REM Now draw the box at 65,50.
190 CALL "ADRAW",B$,1,65,50
200 END
```



4639-10

Figure 3-3. Example of ADRAW.

AGIN (Absolute Gin)

This command finds the graphic cursor's X,Y coordinate position.

Syntax Form

[line number] **CALL "AGIN"**, numeric simple result, numeric simple result

Descriptive Form

[line number] **CALL "AGIN"**, graphic cursor X position, graphic cursor Y position

Format Example

```
500 CALL "AGIN",X,Y
```

Explanation

BASIC GIN returns User Display Units (UDU's), whereas Graphics Enhancement GIN returns Graphic Display Units (GDU's). The following program shows the use of AGIN with both MOVE and RMOVE commands.

Sample Program

See Figure 3-4.

```
100 REM AGIN example
110 INIT
120 REM Mark the starting cursor location.
130 MOVE 65,50
140 RDRAW 0,0
150 REM Now wander about a bit.
160 FOR I=1 TO 10
170 RMOVE 5*RND(1),5*RND(1)
180 NEXT I
190 REM Mark the destination
200 RDRAW 0,0
210 REM Now find out where we ended up.
220 CALL "AGIN",X,Y
230 PRINT X;Y
240 END
```

```
                                        94.08 71.68
```

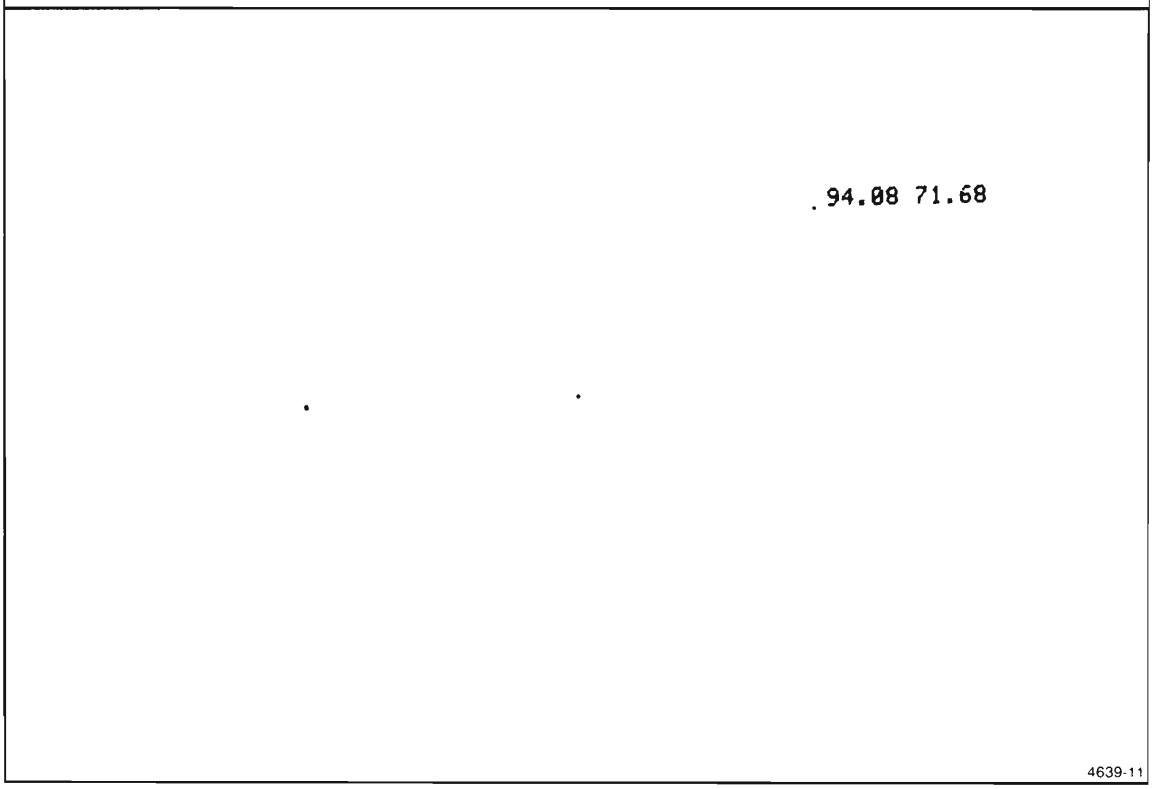


Figure 3-4. Example of AGIN.

AINPUT (Absolute Input)

This command prints the string S$ and any keyboard keys (K$) at the X,Y coordinate position C times (or until $^{C}_{R}$ is typed).

Syntax Form

[line number] **CALL "AINPUT"**, text string/constant argument, numeric simple argument & result (can be a constant if desired), numeric simple/constant argument, numeric simple/constant argument, text string result.

Descriptive Form

[line number] **CALL "AINPUT"**, text string S$, C times, X position, Y position, keyboard characters typed during call (K$)

Format Example

```
500 CALL "AINPUT",S$,C,X,Y,K$
```

Explanation

Input commands print the string S$ and any keyboard characters C times or until you press the carriage return, whichever occurs first.

AINPUT allows up to 28 characters, including the prompt, and terminates on the 28th character. On the Graphic System, characters typed after the 28th one go into the type-ahead buffer. The 28th character is not displayed and does not ring the bell. Refresh print commands require at least a one character prompt.

AINPUT throws away the type-ahead buffer characters before it starts accepting keyboard keys. If you enter character strings after reaching a PAGE FULL condition, you lose them when you erase them with the PAGE key. Keep this in mind.

After input command exit:

- C = actual display count.
- Any pending keyboard keys are saved in K$.

Sample Program #1

```
100 CALL "AINPUT","TEST : ",-1000,65,50,K$
110 PRINT K$
```

Figure 3-5a shows the result. If you type the word BOZO while the call command is active, Figure 3-5b shows the result.

At CALL exit (after 1000 printings or $^{C}_{R}$ typed), K$ = "BOZO".

AINPUT does not allow a null prompt string. If S$ is "" in the program statement, 100 CALL "AINPUT",S$,-1000,65,50,K$, then you get the following message on your screen:

"UNDEFINED VARIABLE IN LINE 100 – ERROR MESSAGE #36"

HOWEVER, S$ can be <Control G> or any other non-printing character string.

You can use BREAK (once) to terminate the input and to return a null string. This clears the keyboard buffer.

Sample Program #2

See Figure 3-6.

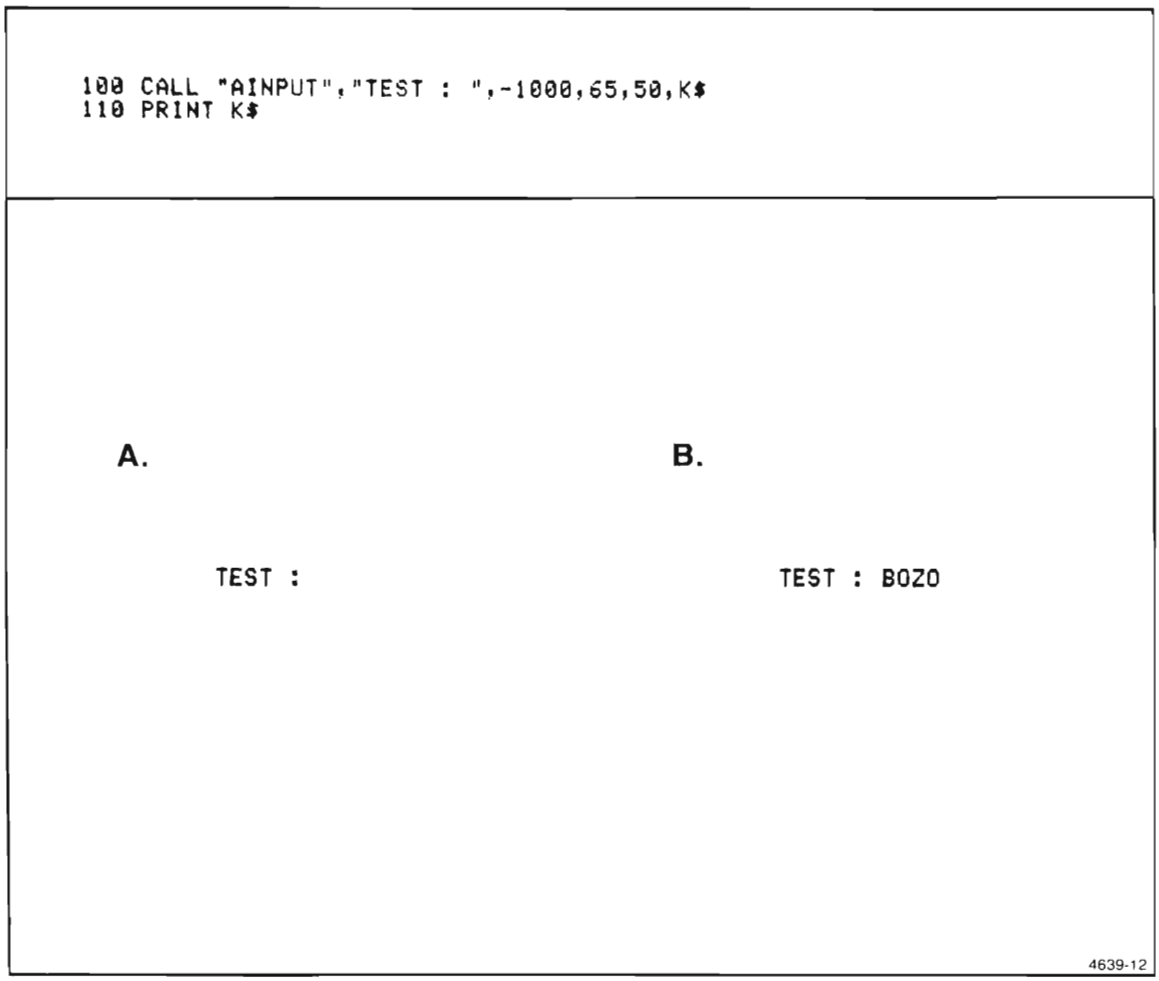


Figure 3-5. First Example of AINPUT.

```
100 REM AINPUT example
110 REM Type in city names along this simulated road.
120 INIT
130 X=0
140 Y=0
150 REM The names will appear P,Q off of the graphic position X,Y.
160 P=2
170 Q=-2
180 MOVE 0,0
190 FOR I=1 TO 8
200 REM Draw a segment of road.
210 X=X+10
220 Y=Y+10*RND(I)
230 DRAW X,Y
240 REM Now get the city name (k$); the question appears at the place
250 REM   the name will be printed.
260 CALL "AINPUT","CITY: ",-3000,X+P,Y+Q,K$
270 CALL "APRINT",K$,1,X+P,Y+Q
280 NEXT I
290 END
```

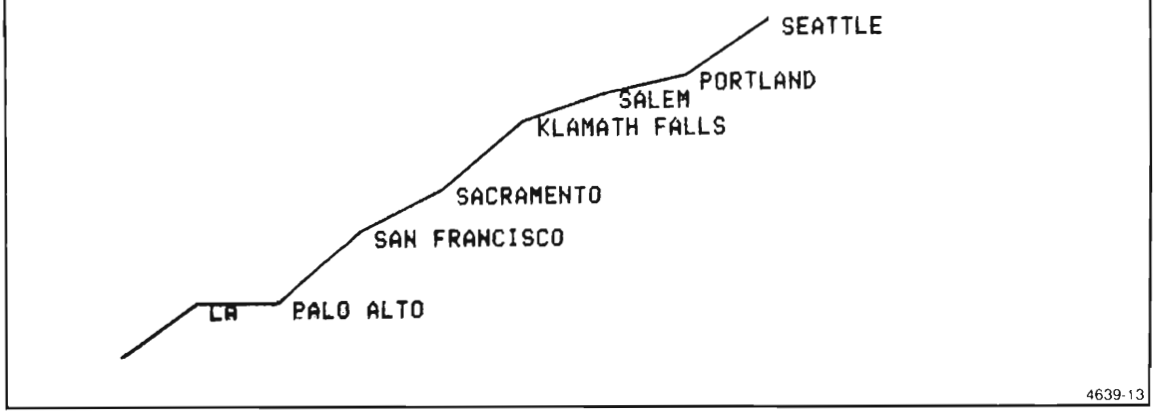


Figure 3-6. Second Example of AINPUT.

AMOVE (Absolute Move)

This command moves the image I$ to the X,Y position.

Syntax Form

[line number] **CALL "AMOVE"**, image string argument & result, numeric simple/ constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "AMOVE"**, image string, position X, position Y

Format Example

```
500 CALL "AMOVE",I$,X,Y
```

Explanation

The entire image is translated so that the first image point is at the X,Y position. This command alters the image string.

Sample Program

See Figure 3-7.

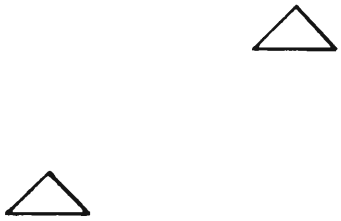
```
100 REM AMOVE example
110 INIT
120 PAGE
130 REM Create the image of a triangle.
140 DIM F(8),F$(12),G(8)
150 DATA -50,-40,60,40,55,45,50,40
160 READ F
170 PRINT "F",F
180 REM Print the starting values, for later comparison.
190 CALL "CHANGE",F,F$
200 REM Draw the triangle at the location of its data.
210 CALL "RDRAW",F$,1,0,0
220 REM Move the image to start at 20,20.
230 CALL "AMOVE",F$,20,20
240 REM Draw it at its new data location.
250 CALL "RDRAW",F$,1,0,0
260 CALL "CHANGE",F$,G
270 PRINT "G",G
280 END
```

```
F

 -50            -40            60             40
 55             45             50             40

G

 -19.968        -19.968        29.952         19.968
 25.088         24.96          19.968         19.968
```



4639-14

Figure 3-7. Example of AMOVE.

APOINT (Absolute Point)

This command finds the image point N and the X,Y (output) coordinate position nearest absolute position X,Y (input).

Syntax Form

[line number] **CALL "APOINT"**, image string argument, numeric simple argument & result, numeric simple argument & result, numeric simple argument & result

Descriptive Form

[line number] **CALL "APOINT"**, image, vector number where search starts, X position nearest X absolute, Y position nearest Y absolute

Format Example

```
500 CALL "APOINT",I$,N,X,Y
```

Explanation

At CALL entry:

- X,Y = search center.
- N = image point where search starts (point number).
- I$ = image.

At CALL exit:

- The sign (N) indicates if the vector is a MOVE or a DRAW.
- X,Y = location of point nearest search center.
- N = image point (number) nearest search center.
- I$ = image (not altered).

The POINT command finds the image I$ point N and the X,Y position that is nearest to absolute X,Y. The variables are received, redefined, and returned.

Sample Program

See Figure 3-8.

```
100 REM APOINT example
110 INIT
120 REM Get the image of an 'A'.
130 GOSUB 270
140 REM Always let all points be searched for, via N=1.
150 N=1
160 PRINT "Type in X an Y for search center: ";
170 INPUT X,Y
180 REM Find the nearest point.
190 CALL "APOINT",F$,N,X,Y
200 REM Mark the point with the cross-hairs.
210 CALL "ACROSS",-50,X,Y
220 CALL "AINPUT","This point? ",-1000,X,Y,K$
230 REM Repeat if desired by typing 'no'.
240 IF K$<>"yes" AND K$<>"y" THEN 150
250 PRINT X,Y
260 END
270 REM Define image.
280 INIT
290 DIM F(10),F$(15)
300 DATA -50,-50,53,60,56,50,-51,-53.3,55,53.3
310 READ F
320 CALL "CHANGE",F,F$
330 CALL "ADRAW",F$,1,50,50
340 RETURN
```

```
Type in X an Y for search center: 2.5,3.3
 49.92              49.92
```

4639-15

Figure 3-8. Example of APOINT.

APRINT (Absolute Print)

This command prints the characters (S$) C times at the X,Y coordinate position.

Syntax Form

[line number] **CALL "APRINT"**, text string/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "APRINT"**, characters to be printed, print count, X position, Y position

Format Example

```
500 CALL "APRINT",S$,C,X,Y
```

Explanation

APRINT prints the characters in S$ a total of C times at the X,Y position (write-thru or storage). This command does not alter the S$ string.

Sample Program

See Figure 3-9.

```
100 REM APRINT example
110 REM Type in city names along the simulated road.
120 INIT
130 PAGE
140 X=0
150 Y=0
160 REM Names appear P,Q off of the graphic position X,Y.
170 P=2
180 Q=-2
190 MOVE 0,0
200 FOR I=1 TO 8
210 REM Draw a segment of the road.
220 X=X+10
230 Y=Y+10*RND(I)
240 DRAW X,Y
250 REM Now get the city name (k$).  The question appears
260 REM   at the place the name will be printed.
270 CALL "AINPUT","CITY: ",-3000,X+P,Y+Q,K$
280 CALL "APRINT",K$,1,X+P,Y+Q
290 NEXT I
300 END
```

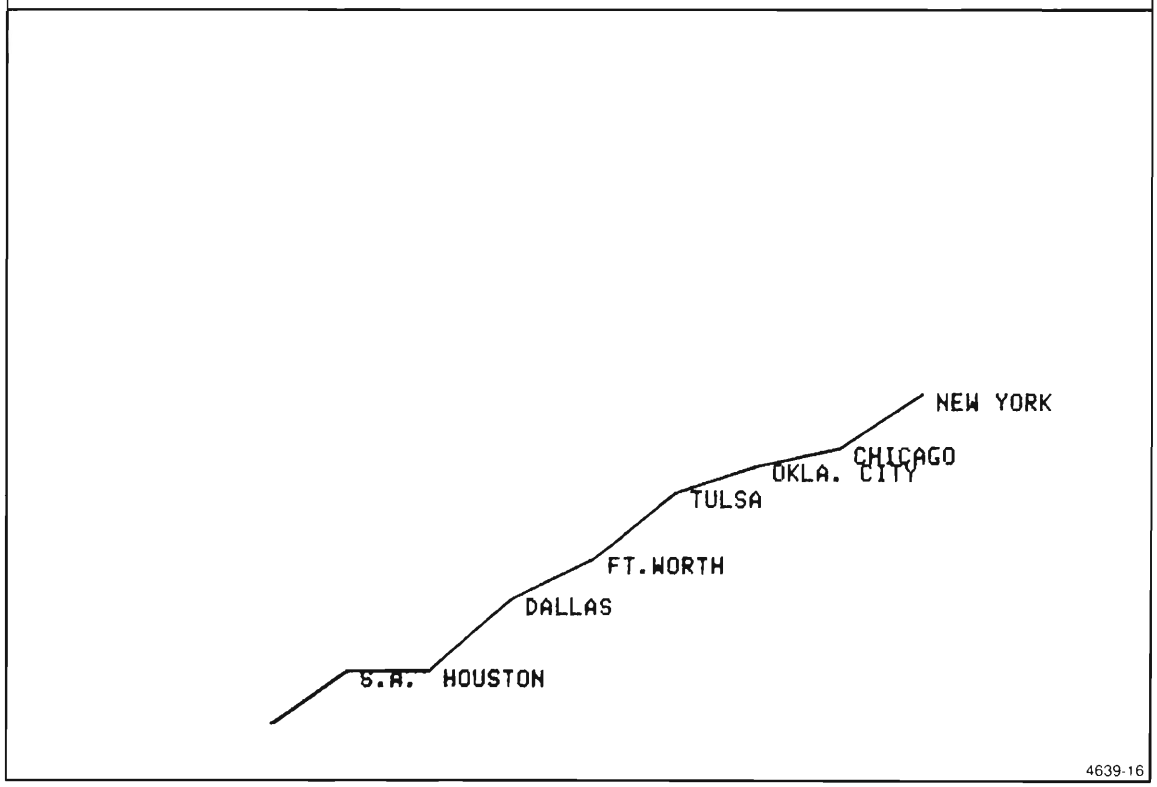


Figure 3-9. Example of APRINT.

AROTATE (Absolute Rotate)

This command rotates the image I$ by angle A around the X,Y coordinate position.

Syntax Form

[line number] **CALL "AROTATE"**, image string argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "AROTATE"**, image to be rotated, angle of rotation, X position, Y position

Format Example

```
500 CALL "AROTATE",I$,A,X,Y
```

Explanation

All angles are measured counterclockwise and in current trigonometric units. This command alters the image string.

NOTE

The radians specified in ROTATE and SHEAR commands are limited to:

$-2PI*65.536 \leqslant Angle \leqslant 2PI*65.536$ (radians); that is, $= \pm 131.072*PI = 411.774$ (radians).

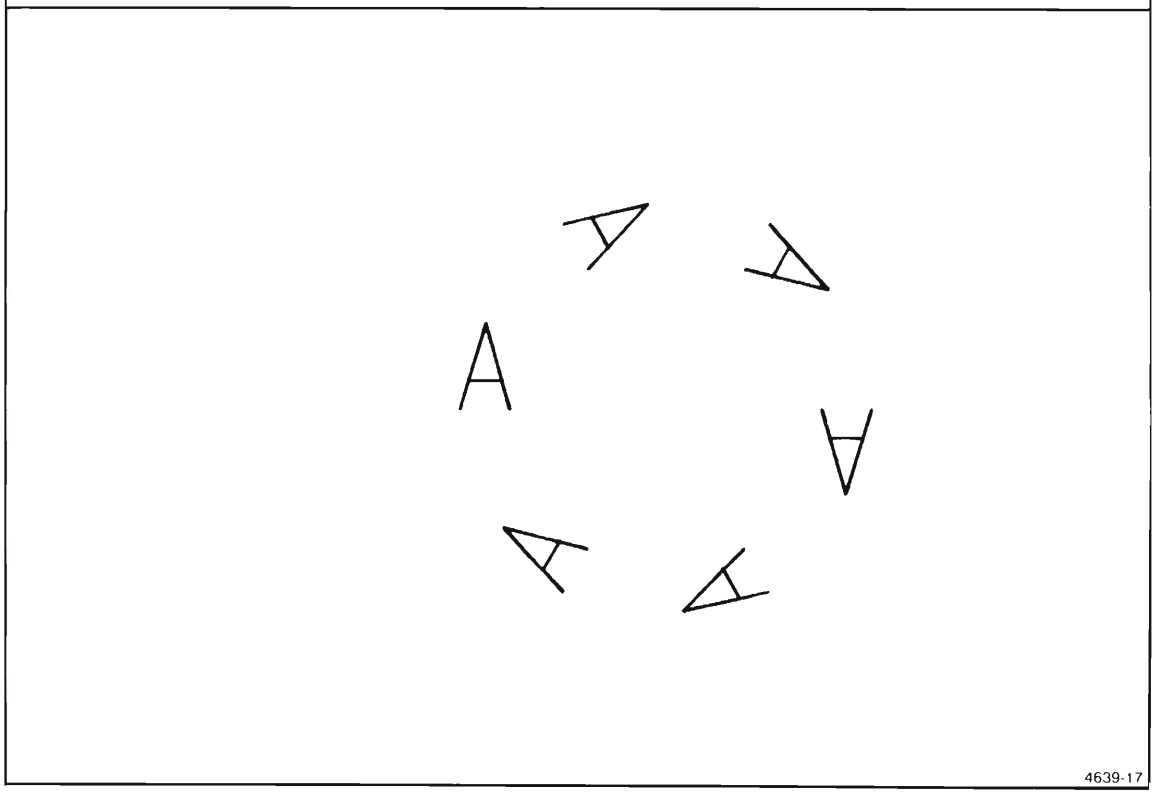
Or, by degrees: $-2PI*65.536\,(180/PI) = -360*65.536 \leqslant Angle \leqslant 360*65.536 = 23592.96$ °.

The image I$ is rotated by angle A around the rotation center X,Y. If angle A is less than 0° (that is, negative), the image rotates clockwise. If angle A is greater than zero, the image rotates counterclockwise.

Sample Program

See Figure 3-10.

```
100 REM AROTATE example
110 INIT
120 SET DEGREES
130 REM Get the image of an 'A'.
140 DIM F(10),F$(15),G$(15)
150 DATA -50,-50,53,60,56,50,-51,-53.3,55,53.3
160 READ F
170 CALL "CHANGE",F,F$
180 REM Save the original image for each rotation.
190 G$=F$
200 FOR A=0 TO 360 STEP 60
210 REM Find X and Y increments for a point at radius 30 and angle A.
220 X=30*COS(A)
230 Y=30*SIN(A)
240 F$=G$
250 REM Rotate around a point 25 to the right of the image.
260 CALL "AROTATE",F$,A,75,50
270 REM Draw the image at its data location.
280 CALL "RDRAW",F$,1,0,0
290 NEXT A
300 END
```



4639-17

Figure 3-10. Example of AROTATE.

ASCALE (Absolute Scale)

This command scales the image I$ by factors H,V around coordinate position X,Y.

Syntax Form

[line number] **CALL "ASCALE"**, image string argument & result, numeric simple/ constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "ASCALE"**, image to be scaled, horizontal factor, vertical factor, X position, Y position

Format Example

```
500 CALL "ASCALE",I$,H,V,X,Y,
```

Explanation

This command alters the image string.

If either the horizontal (H) or vertical (V) factor is less than zero, that is, a negative value, then a "mirror image" occurs. $-65.535 \leqslant$ scale factor range $\leqslant +65.535$ (17-bit resolution).

NOTE

SCALE and TAPER commands operate over a range of -65.535 to + 65.535. This gives a resolution of 1 in 2∧10 (about 0.001). The input of a number less than -65.535 or greater than 65.535 has the same effect as an input of -65.535 or 65.535 respectively. The full range is rarely used. The screen usually limits the image before reaching the range extremities.

Sample Program

See Figure 3-11.

```
100 REM ASCALE example
110 INIT
120 SET DEGREES
130 REM Get the image of an 'A' and draw it.
140 DIM F(10),F$(15)
150 DATA -50,-50,53,60,56,50,-51,-53.3,55,53.3
160 READ F
170 CALL "CHANGE",F,F$
180 CALL "RDRAW",F$,1,0,0
190 REM Mark the scaling center.
200 MOVE 40,50
210 DRAW 40,50
220 REM Scale by 2 in X and 3 in Y.
230 CALL "ASCALE",F$,2,3,40,50
240 CALL "RDRAW",F$,1,0,0
250 END
```

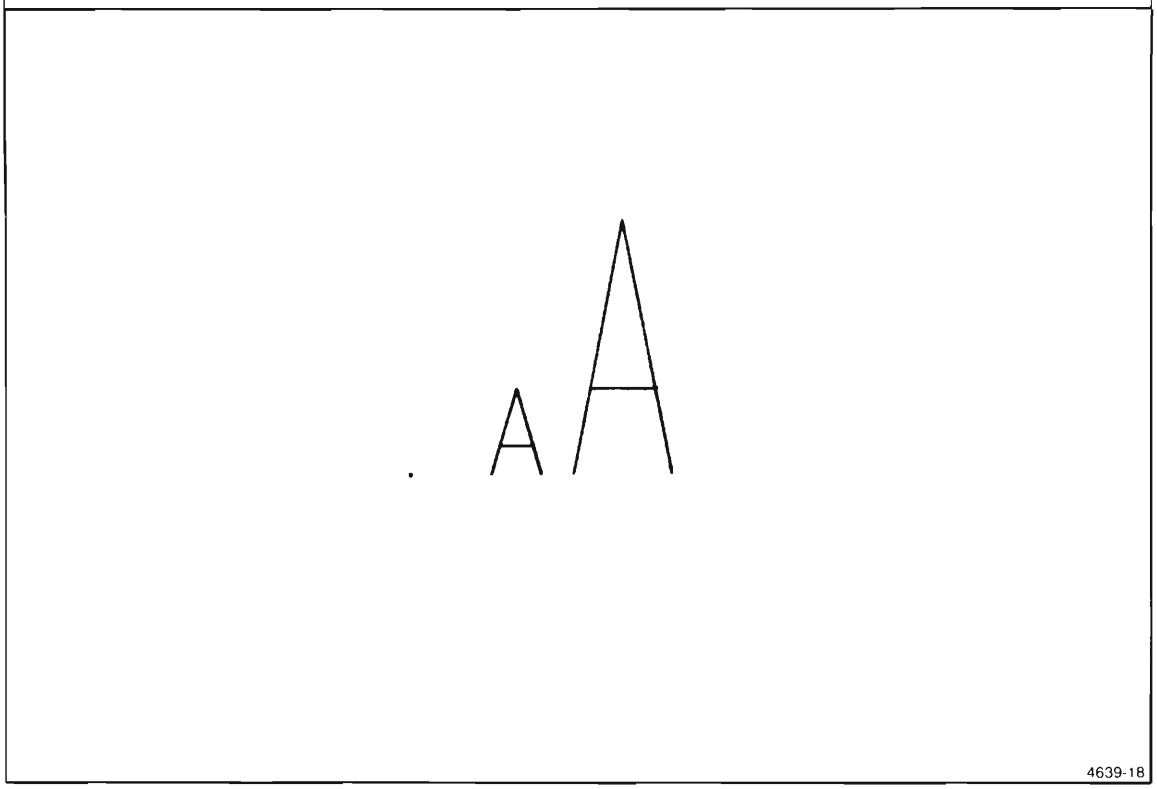


Figure 3-11. Example of ASCALE.

ASHEAR (Absolute Shear)

The command shears the image I$ by angles H,V around coordinate position X,Y.

Syntax Form

[line number] **CALL "ASHEAR"**, image string argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "ASHEAR"**, image string, H angle, V angle, X position, Y position

Format Example

```
500 CALL "ASHEAR",I$,H,V,X,Y
```

Explanation

All angles are measured counterclockwise (CCW) and in current trigonometric units.

NOTE

The radians specified in SHEAR and ROTATE commands are limited to:

$-2PI*65.536 \leqslant Angle \leqslant 2PI*65.536$ (radians); that is, $= \pm 131.072*PI =$ 411.774 (radians).

Or, by degrees: $-2PI*65.536\,(180/PI) = -360*65.536 \leqslant Angle \leqslant 360*65.536$ $= 23592.96$ °.

If either the H or V angle is greater than 0°, the image rotates counterclockwise. If the H or V angle is less than 0°, the image rotates clockwise. This command alters the image string.

ASHEAR rotates the X and Y axes independently.

Sample Program

See Figure 3-12.

```
100 REM ASHEAR example
110 INIT
120 SET DEGREES
130 REM Get the image and its back-up copy.
140 DIM F(12),F$(18),G$(18)
150 DATA -53,-55,-50,-50,53,60,56,50,-51,-53.3,55,53.3
160 READ F
170 CALL "CHANGE",F,F$
180 G$=F$
190 X=50
200 Y=5
210 U=1
220 X0=1
230 Y0=0
240 GOSUB 310
250 X=80
260 X0=0
270 Y0=1
280 U=-1
290 GOSUB 310
300 END
310 FOR V=-90 TO 90 STEP 9
320 REM Restore the original image.
330 F$=G$
340 REM Shear at an angle V about the vertical, then horizontal
350 CALL "ASHEAR",F$,X0*V,Y0*V,53,53.3
360 REM Draw the image moving up or down
370 Y=Y+U*4
380 CALL "ADRAW",F$,1,X,Y
390 NEXT V
400 RETURN
```

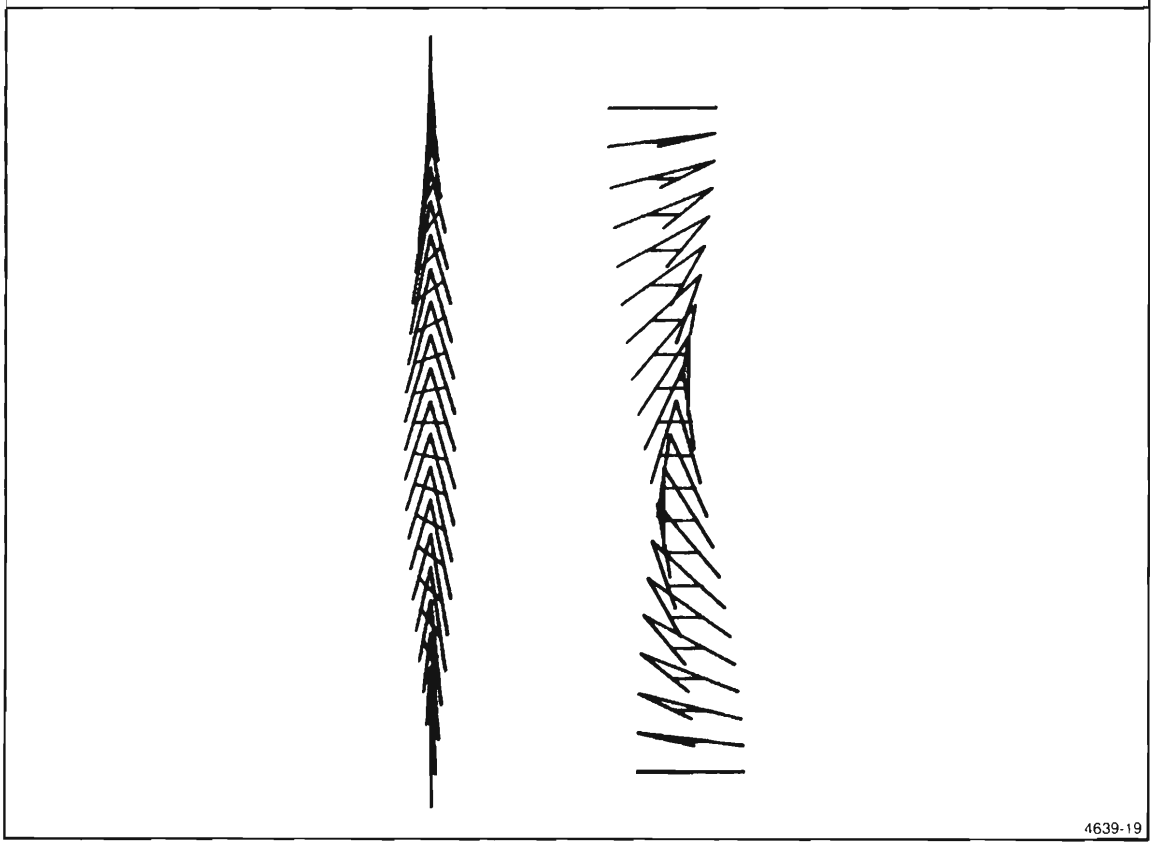


Figure 3-12. Example of ASHEAR with X-shear Rotation.

ATAPER (Absolute Taper)

This command tapers the image I$ by factors H,V around coordinate position X,Y.

Syntax Form

[line number] **CALL "ATAPER"**, image string argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "ATAPER"**, image, factor H, factor V, position X, position Y

Format Example

```
500 CALL "ATAPER",I$,H,V,X,Y
```

Explanation

This command alters the image string.

Negative values for H and/or V result in a mirror image. $-65.535 \leqslant$ scale factor range $\leqslant +65.535$ (17-bit resolution).

NOTE

TAPER and SHEAR commands operate over a range of -65.535 to +65.535. This gives a resolution of 1 in 2∧10 (about 0.001). The input of a number less than -65.535 or greater than 65.535 has the same effect as an input of -65.535 or 65.535 respectively. The full range is rarely used. The screen usually limits the image before reaching the range extremities.

Sample Program

See Figure 3-13.

```
100 REM ATAPER example
110 REM Allows you to experiment with tapering as a pseudo-perspective.
120 INIT
130 PAGE
140 REM Get the image of a box and its back-up.
150 DIM B(24),B$(36),C$(36)
160 DATA -50,-50,50,60,60,60,60,50,50,50,-50,-60,58.66,65,68.66,65
170 DATA 68.66,55,60,50,-60,-60,68.66,65
180 READ B
190 CALL "CHANGE",B,B$
200 C$=B$
210 CALL "RDRAW",B$,1,0,0
220 REM Mark the taper center.
230 MOVE 30,45
240 DRAW 30,45
250 X=-20
260 B$=C$
270 REM Starting at zero, the images move right 20 each repetition.
280 X=X+20
290 IF X>110 THEN 390
300 HOME
310 PRINT "Horizontal and Vertical Taper factors (e.g. .5,.1): ";
320 INPUT H,V
330 REM Taper and draw.
340 CALL "ATAPER",B$,H,V,30,45
350 CALL "ADRAW",B$,1,X,20
360 MOVE X,15
370 PRINT H;V;
380 GO TO 260
390 END
```

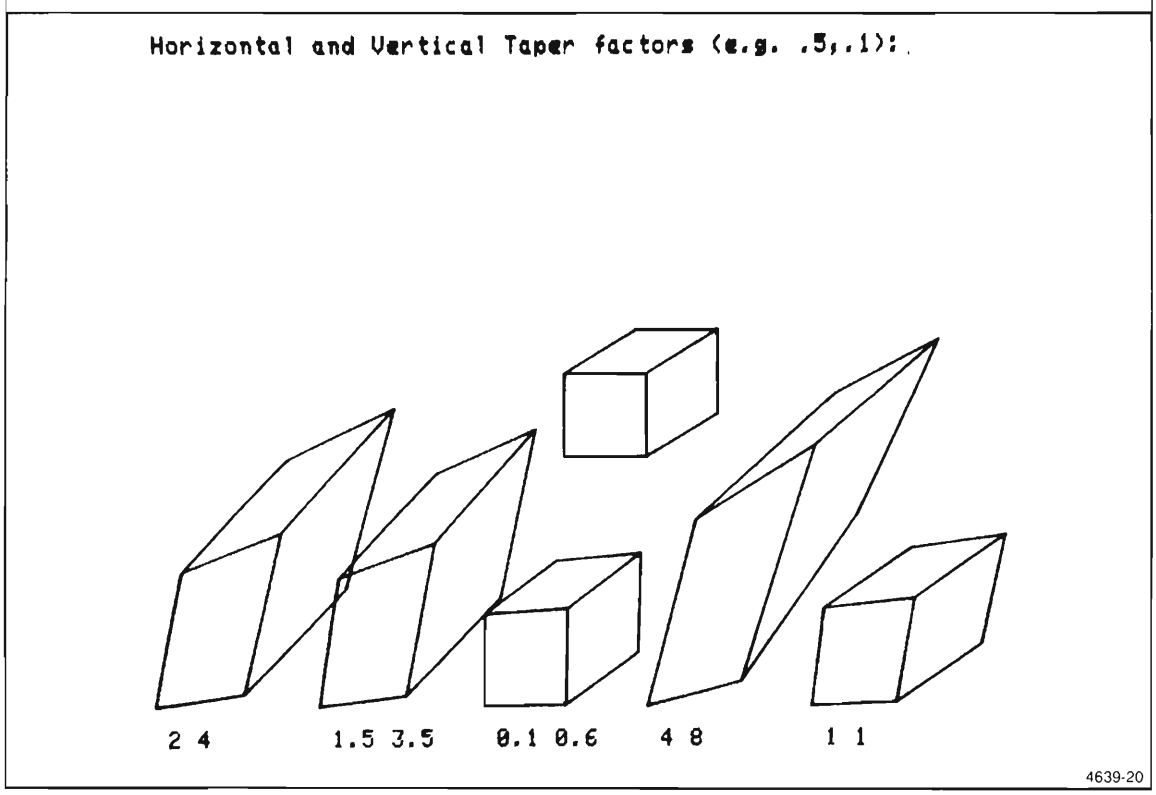


Figure 3-13. Example of ATAPER.

BOUNDS (Image Bounds)

This command finds the minimum X, minimum Y, maximum X, and maximum Y coordinate positions of the image I$.

Syntax Form

[line number] **CALL "BOUNDS"**, image string argument, numeric simple result, numeric simple result, numeric simple result, numeric simple result

Descriptive Form

[line number] **CALL "BOUNDS"**, image string, minimum X position, minimum Y position, maximum X position, maximum Y position

Format Example

```
500 CALL "BOUNDS",I$,X0,Y0,X1,Y1
```

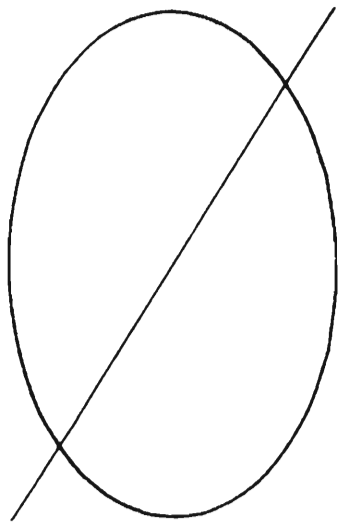
Explanation

This command does not alter the image string.

Sample Program

See Figure 3-14.

```
100 REM BOUNDS example
110 INIT
120 SET DEGREES
130 J=1
140 REM Generate an ellipse and draw it in place.
150 DIM F(146),F$(219)
160 FOR I=0 TO 360 STEP 5
170 F(J)=65+20*COS(I)
180 F(J+1)=50+30*SIN(I)
190 J=J+2
200 NEXT I
210 F(1)=-F(1)
220 F(2)=-F(2)
230 CALL "CHANGE",F,F$
240 CALL "RDRAW",F$,1,0,0
250 REM Find lower left and upper right of enclosing rectangle.
260 CALL "BOUNDS",F$,X0,Y0,X1,Y1
270 REM Slash the '0'.
280 MOVE X0,Y0
290 DRAW X1,Y1
300 END
```



4639-21

Figure 3-14. Example of BOUNDS.

CHANGE

Function #1

This command changes the floating point image array F into image string I$. Negative values in array F indicate moves. (The description of the reverse of this procedure appears below under Function #2.)

Syntax Form

[line number] **CALL "CHANGE"**, numeric array argument, image string result

Descriptive Form

[line number] **CALL "CHANGE"**, floating point image array, desired image string

Format Example

```
500 CALL "CHANGE",F,I$
```

Explanation

The floating point image array must have an even number of elements (that is, consist of X,Y pairs). This F array can have any form-factor, but the elements are assumed to be X,Y pairs. For example, DIM F(100) = DIM F(10,10) = DIM F(20,5).

For different form factors, the input order of X,Y values into I$ is shown below (in bold type), progressing from 1 to 6:

1 2 3	1 2 **3**	1 2 3
4 5 6	**4** 5 6	4 **5 6**

Push the HOME/PAGE key after executing the CHANGE command if you wish to home the cursor.

The array is not altered by this command, but the image string is. The string must be at least big enough to hold the converted characters.

Function #2

This alternative use of the CHANGE command changes the image string S$ into floating point image array F. Negative values in array F indicate moves. (See the earlier Function #1 for the reverse function.)

Syntax Form

[line number] **CALL "CHANGE"**, image string argument, numeric array result

Descriptive Form

[line number] **CALL "CHANGE"**, image string, floating point image array

Format Example

```
500 CALL "CHANGE",I$,F
```

Explanation

The floating point image array F values will be integer multiples of 0.128 GDU's (display resolution). The array must be dimensioned so that the number of elements in the array are equal to the string length divided by three: DIM F(2*LEN(S$)/3).

The array can have any form-factor. The array is altered, but the image string is not.

DASHED (Dashed Grid)

This command draws a dashed grid with horizontal pitch H and vertical pitch V. The dashed grid starts at X0,Y0 and ends at X1,Y1.

Syntax Form

[line number] **CALL "DASHED"**, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "DASHED"**, horizontal pitch, vertical pitch, starting position X0, starting position Y0, ending position X1, ending position Y1

Format Example

```
500 CALL "DASHED",H,V,X0,Y0,X1,Y1
```

Explanation

The dashed grid starts at the lower left point (X0,Y0) of the grid, and ends at the upper right point (X1,Y1). The result looks like solid lines.

The grid is approximate, and not suitable as a reference grid. H,V pitch parameters need to be multiples of .128 if placement is to be without placement error.

Sample Program

See Figure 3-15.

```
100 REM DASHED example
110 CALL "DASHED",1.28,1.28,0,0,64,64
120 REM Increments not multiples of .128 are more likely
130 REM   to result in an approximate grid.
140 CALL "DASHED",2,2,70,70,90,90
150 END
```

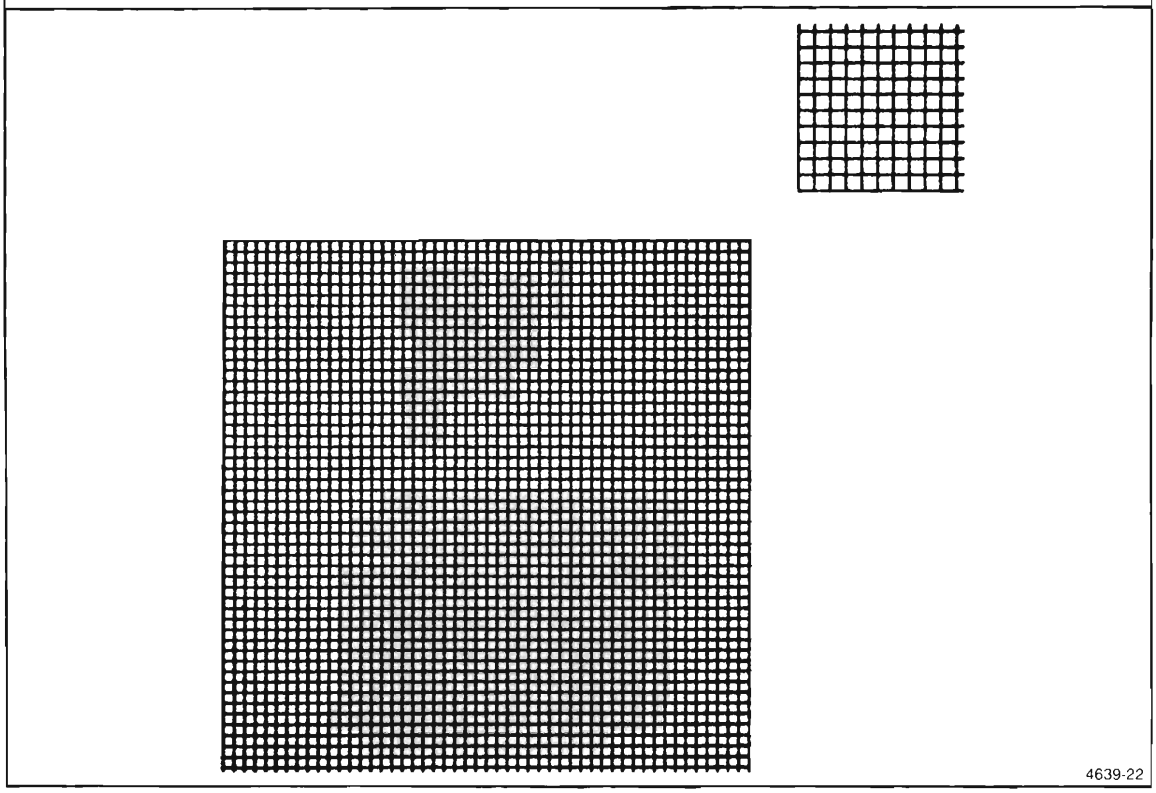


Figure 3-15. Example of DASHED.

DEFINE (Define Point)

This command defines the image point number N to be a MOVE X,Y or a DRAW X,Y. The sign (N) indicates if the vector is a MOVE or a DRAW. The image point N must exist prior to the call.

Syntax Form

[line number] **CALL "DEFINE"**, image string argument & result, numeric simple argument & result, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "DEFINE"**, image, point N, position X, position Y

Format Example

```
500 CALL "DEFINE",I$,N,X,Y
```

Explanation

This command allows any point in any image to be modified (thus altering the image string) without having to go through the CHANGE command. Image point numbers are 1-based; that is, the first point is number 1.

```
                  #1              N-1    N    N+1
Image string XXX . . . XXX  XXX  XXX  XXX  XXX . . . (where X = 1 byte)

                            Modify point N
                            to DRAW (+) or
                            MOVE (–) X,Y.
```

Sample Program

See Figure 3-16.

```
100 REM DEFINE example
110 INIT
120 DIM F(14),F$(21)
130 SET DEGREES
140 REM  Arbitrary 21 character string.
150 F$="123456789012345678901"
160 REM Now set the points to their desired arc locations.
170 CALL "DEFINE",F$,-1,95,50
180 FOR N=2 TO 7
190 X=65+30*COS((N-1)*10)
200 Y=50+30*SIN((N-1)*10)
210 CALL "DEFINE",F$,N,X,Y
220 NEXT N
230 CALL "RDRAW",F$,10,0,0
240 END
```

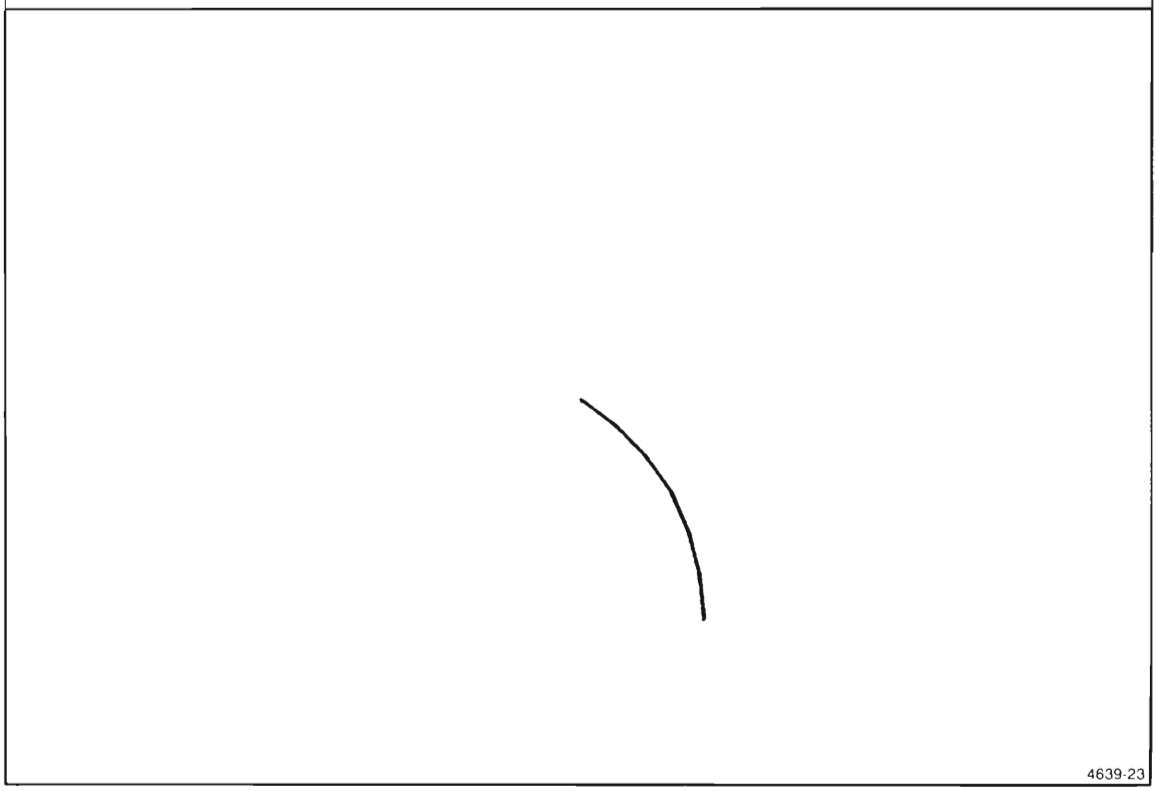


Figure 3-16. Example of DEFINE.

DOTTED (Dotted Grid)

This command draws a dotted grid with horizontal pitch H and vertical pitch V. The dotted grid starts at X0,Y0 and ends at X1,Y1.

Syntax Form

[line number] **CALL "DOTTED"**, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "DOTTED"**, horizontal pitch, vertical pitch, starting position X0, starting position Y0, ending position X1, ending position Y1

Format Example

```
500 CALL "DOTTED",H,V,X0,Y0,X1,Y1
```

Explanation

The dotted grid starts at the lower left point (X0,Y0) of the grid, and ends at the upper right point (X1,Y1).

The grid is approximate, and not suitable as a reference grid. H,V pitch parameters need to be multiples of .128 if placement is to be without placement error.

Sample Program

See Figure 3-17.

```
100 REM DOTTED example
110 CALL "DOTTED",1.28,1.28,0,0,64,64
120 CALL "DOTTED",2,2,70,70,90,90
130 REM Mark the ideal upper right.
140 MOVE 90,90
150 DRAW 90,90
160 END
```

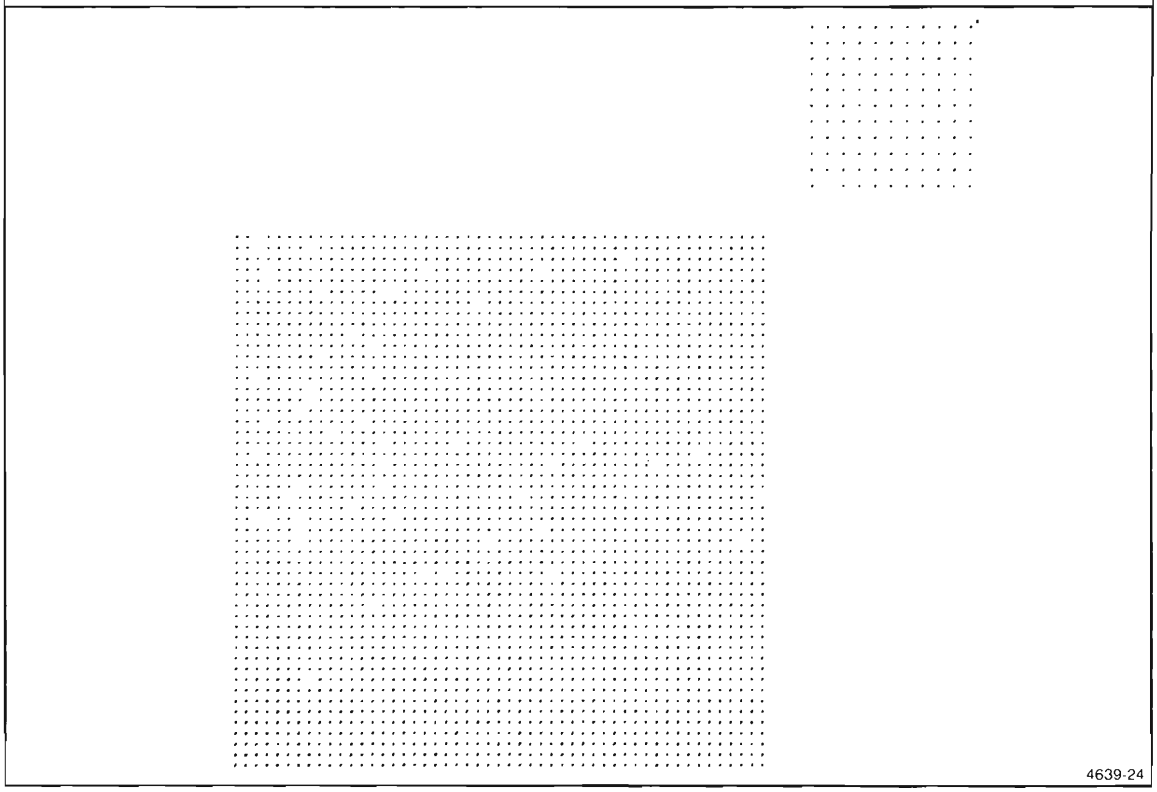


Figure 3-17. Example of DOTTED.

GCROSS (Graphics Cross)

This command draws the cross-hair cursor C times at coordinate position X,Y relative to the graphics cursor.

Syntax Form

[line number] **CALL "GCROSS"**, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "GCROSS"**, count, X position relative to graphics cursor, Y position relative to graphics cursor

Format Example

```
500 CALL "GCROSS",C,X,Y
```

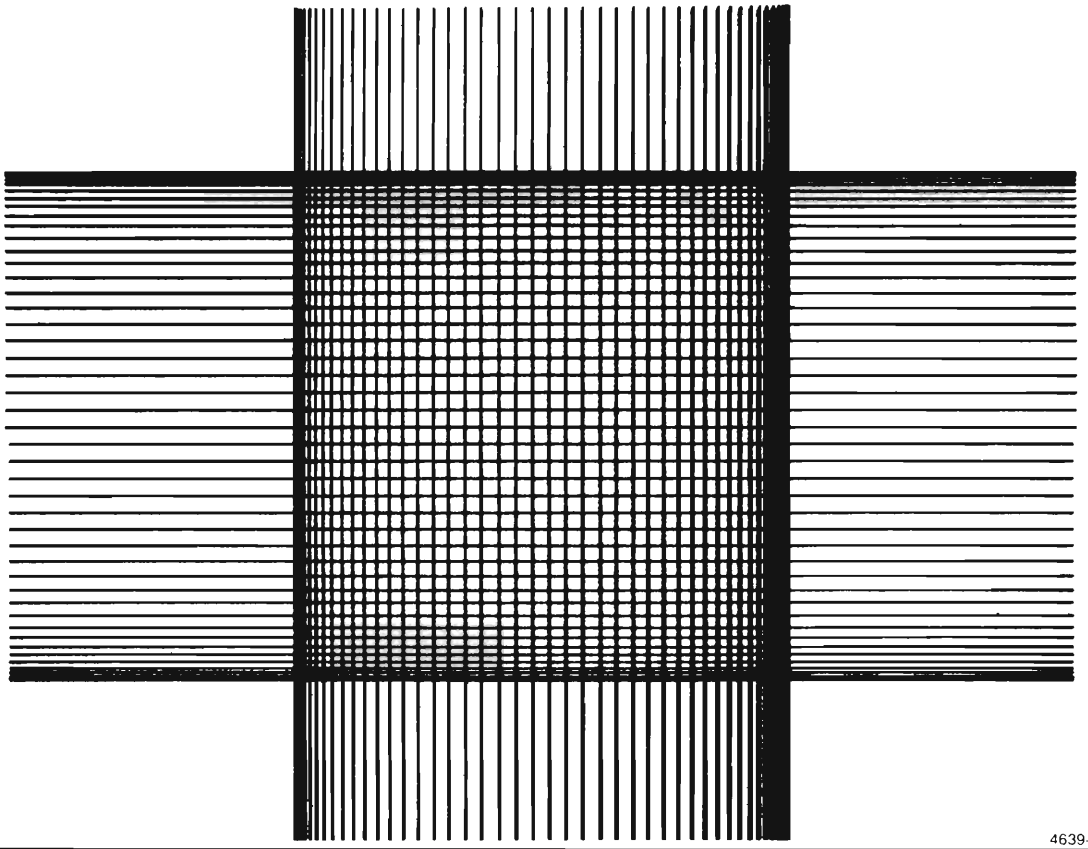
Explanation

If the graphics cursor position equals Xg,Yg, GCROSS draws a crosshair at X + Xg,Y + Yg. Although GCROSS is not very useful, it is included for the consistency of command structure.

Sample Program

See Figure 3-18.

```
100 REM Make the cross-hairs go in a circle.
110 INIT
120 SET DEGREES
130 MOVE 65,50
140 FOR A=0 TO 360 STEP 4
150 REM Compute X,Y increments for radius 30, angle A.
160 X=30*COS(A)
170 Y=30*SIN(A)
180 REM Present cross-hairs at X,Y from the graphic cursor 65,50.
190 CALL "GCROSS",2,X,Y
200 NEXT A
210 END
```



4639-25

Figure 3-18. Example of GCROSS.

GDOTS (Graphics Dots)

This command draws the image dots C times at coordinate position X,Y relative to the graphic cursor.

Syntax Form

[line number] **CALL "GDOTS"**, image string argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "GDOTS"**, image dots, count, X position relative to graphic cursor, Y position relative to graphic cursor

Format Example

```
500 CALL "GDOTS",I$,C,X,Y
```

Explanation

The first image point is drawn at Xg + X,Yg + Y. This command does not alter the image string.

Sample Program

See Figure 3-19.

```
100 REM GDOTS example
110 INIT
120 SET DEGREES
130 Z=-10
140 REM Get the image of an ellipse.
150 DIM C(96),C$(144)
160 C(1)=-65
170 C(2)=-50
180 C(3)=-69
190 C(4)=-50
200 FOR I=5 TO 95 STEP 2
210 C(I)=65+4*COS(4*(I-3))
220 C(I+1)=50+3*SIN(4*(I-3))
230 NEXT I
240 CALL "CHANGE",C,C$
250 REM Start a broken line at 10,10.
260 MOVE 10,10
270 REM All the following graphics are 'relative'.
280 FOR I=1 TO 50
290 REM Generate the offset of ellipse from graphic cursor location.
300 R=2*RND(1)
310 S=2*RND(1)
320 IF I<>50 THEN 340
330 Z=10
340 CALL "GDOTS",C$,Z,R,S
350 REM Draw to the new ellipse center.
360 RDRAW R,S
370 NEXT I
380 END
```

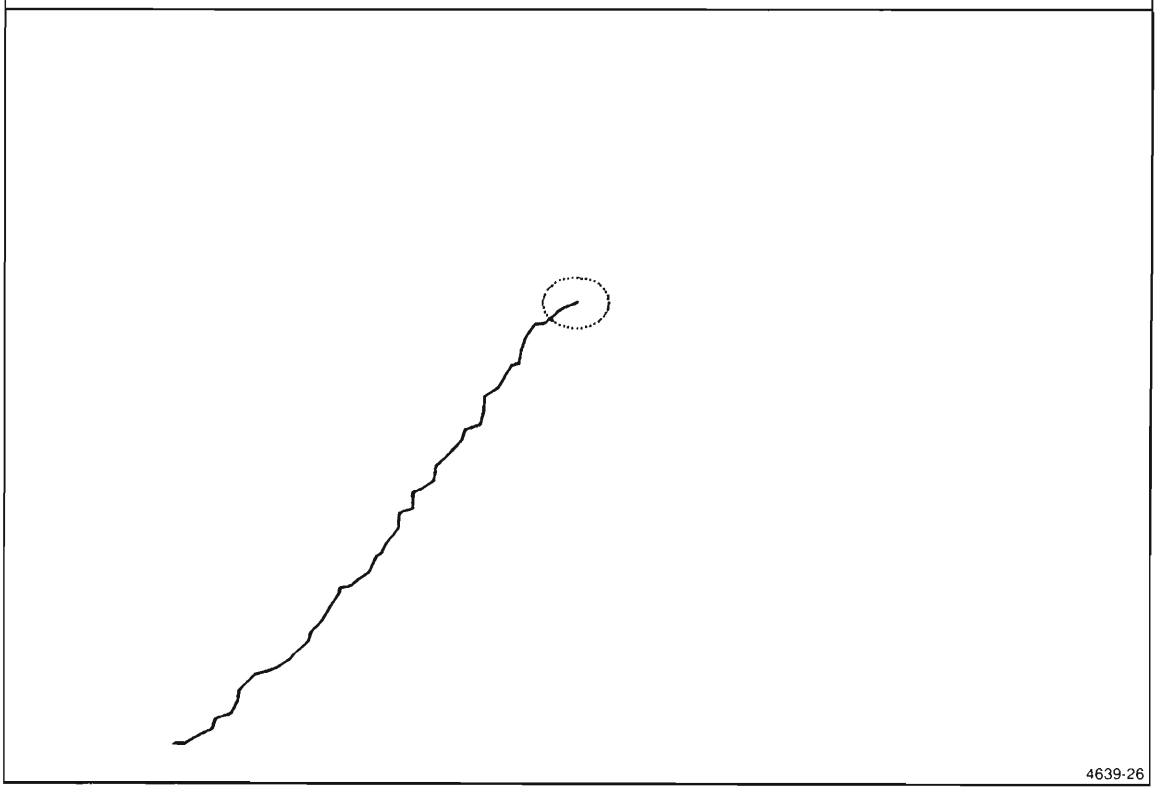


Figure 3-19. Example of GDOTS.

GDRAW (Graphics Draw)

This command draws the image I$ vectors C times at coordinate position X,Y relative to the graphic cursor.

Syntax Form

[line number] **CALL "GDRAW"**, image string argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "GDRAW"**, image vectors, count, X position relative to graphics cursor, Y position relative to graphics cursor

Format Example

```
500 CALL "GDRAW",I$,C,X,Y
```

Explanation

The first image point is drawn at Xg + X,Yg + Y. This command does not alter the image string.

Sample Program

See Figure 3-20.

```
100 REM GDRAW example
110 INIT
120 SET DEGREES
130 REM Get the image of a P.
140 DIM C(20),C$(30)
150 DATA -50,-50,50,56,51.5,56,52.25,55.8,52.8,55.25,53,54.5,52.8,53.75
160 DATA 52.25,53.2,51.5,53,50,53
170 READ C
180 CALL "CHANGE",C,C$
190 REM Start a broken line at 10,10.
200 MOVE 10,10
210 REM All the following graphics are 'relative'.
220 FOR I=1 TO 50
230 REM Generate offset of P from graphic cursor location.
240 R=2*RND(1)
250 S=2*RND(1)
260 CALL "GDRAW",C$,-10,R,S
270 REM Draw to the new P start.
280 RDRAW R,S
290 NEXT I
300 END
```

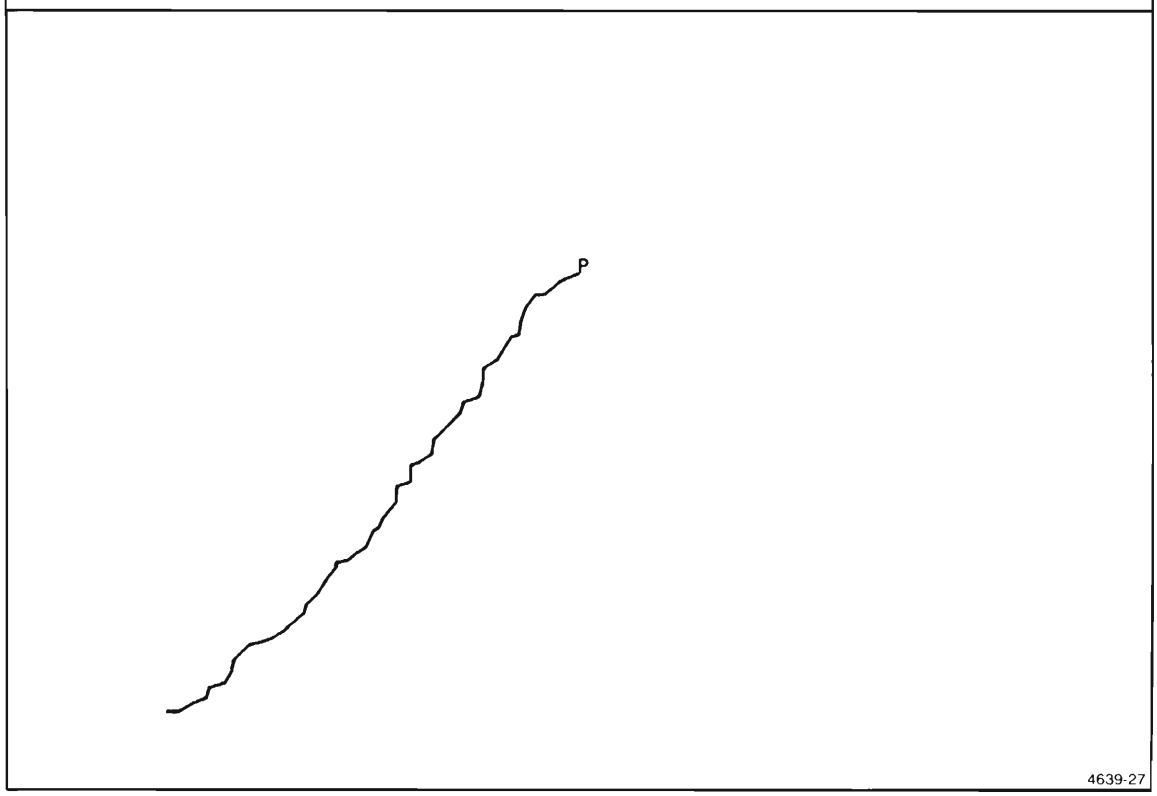


Figure 3-20. Example of GDRAW.

GGIN (Graphics Gin)

This command finds the X,Y coordinate position at X,Y relative to the graphic cursor.

Syntax Form

[line number] **CALL ''GGIN''**, numeric simple argument & result, numeric simple argument & result

Descriptive Form

[line number] **CALL ''GGIN''**, X position relative to graphic cursor, Y position relative to graphic cursor

Format Example

```
500 CALL ''GGIN'',X,Y
```

Explanation

Sample Program

```
100 MOVE 10,10
110 X = 5
120 Y = -5
130 CALL ''GGIN'',X,Y
140 PRINT X,Y
150 END
```

This program results in X = 15,Y = 5.

GINPUT (Graphics Input)

This input command prints the string and any keyboard keys C times (or until $^{C}_{R}$ or the 28th character, including the prompt, is typed) at coordinate position X,Y relative to the graphic cursor.

Syntax Form

[line number] **CALL "GINPUT"**, text string/constant argument, numeric simple argument & result or numeric constant, numeric simple argument or numeric constant, numeric simple argument or numeric constant , numeric simple/constant argument, numeric simple/constant argument, text string result

Descriptive Form

[line number] **CALL "GINPUT"**, string, count (if variable, will be actual count at exit), X position relative to graphic cursor, Y position relative to graphic cursor, keyboard characters K$

Format Example

```
500 CALL "GINPUT",C$,C,X,Y,K$
```

Explanation

Input commands print the string S$ and any keyboard characters C times or until you press the carriage return, whichever occurs first.

You cannot type more than 28 characters into the K$ string, including the prompt. For example, if C$ = 1234567890 (that is, ten characters), K$ will accept only 18 more characters. Suppose those characters are OTTFFSSENT12345678. When you type the number 8 (the 28th character), it does not appear in refresh with the prompt and the previous 17 input characters. However, the string K$ (OTTFFSSENT12345678) does contain the 8. Processing continues with the next BASIC statement.

If the graphic cursor position equals Xg,Yg, then GINPUT prints S$ at Xg + X,Yg + Y a total of C times or until $^{C}_{R}$ is typed (whichever occurs first).

At CALL exit (assuming a variable for refresh count), C equals the actual number of times S$ (and K$) were printed. S$ is not altered by this command, but K$ might be altered (if any keyboard keys are typed). Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-21.

```
100 REM GINPUT example
110 INIT
120 PAGE
130 X=0
140 Y=0
150 REM The prompt will be offset from the graphic cursor by P,Q.
160 P=2
170 Q=-2
180 MOVE 0,0
190 FOR I=1 TO 8
200 REM Generate drawn segment.
210 X=X+10
220 Y=Y+10*RND(I)
230 DRAW X,Y
240 REM Input city name and place it at graphic cursor plus offset.
250 CALL "GINPUT","CITY: ",-3000,P,Q,K$
260 CALL "APRINT",K$,1,X+P,Y+Q
270 NEXT I
280 END
```

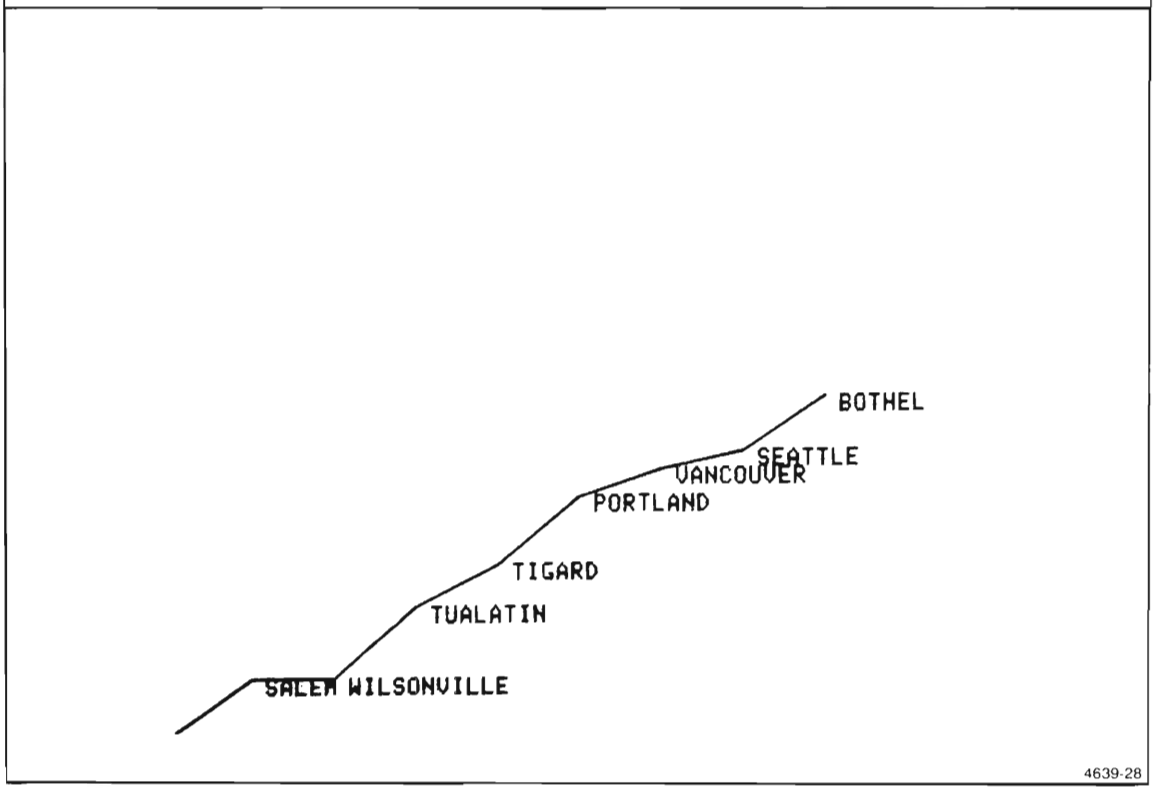


Figure 3-21. Example of GINPUT.

GMOVE (Graphics Move)

This command moves the image I$ to coordinate position X,Y relative to the graphic cursor.

Syntax Form

[line number] **CALL "GMOVE"**, image string argument & result, numeric simple/ constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "GMOVE"**, image, position X relative to graphic cursor, position Y relative to graphic cursor

Format Example

```
500 CALL "GMOVE",I$,X,Y
```

Explanation

This command alters every entry in the image. If the graphic cursor position equals Xg,Yg, then the first image point will be X + Xg,Y + Yg. If Xo,Yo represents the first point in the image, then X + Xg + Xi-Xo is added to each Xi, and Y + Yg + Yi-Yo is added to each Yi.

Sample Program

See Figure 3-22.

```
100 REM GMOVE example
110 INIT
120 REM Generate and draw the image of an A.
130 DIM F(10),F$(15)
140 DATA -50,-40,53,50,56,40,-51,-43.3,55,43.3
150 READ F
160 CALL "CHANGE",F,F$
170 CALL "RDRAW",F$,1,0,0
180 REM Now move the image by 65-50+10 = 25 to the right.
190 MOVE 65,40
200 CALL "GMOVE",F$,10,0
210 CALL "RDRAW",F$,1,0,0
220 END
```

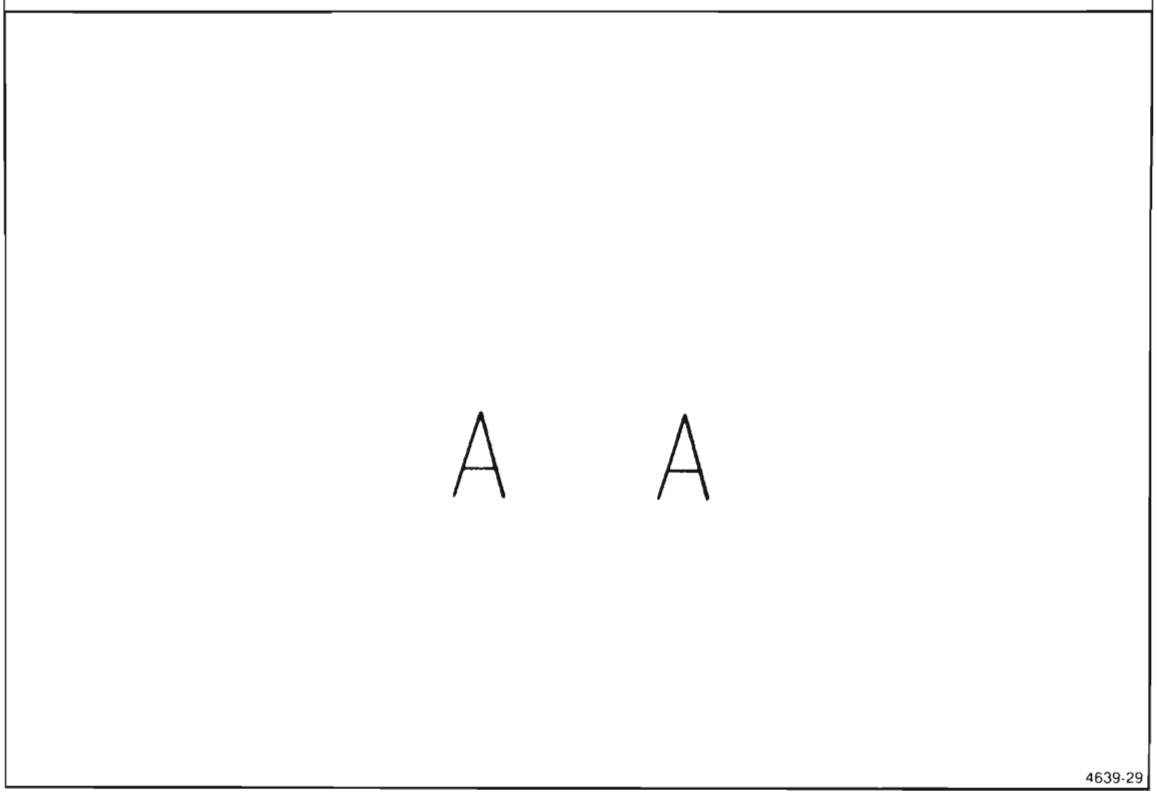


Figure 3-22. Example of GMOVE.

GPOINT (Graphics Point)

This command finds the image point N and coordinate position (output in X,Y) nearest the input X,Y relative to the graphic cursor.

Syntax Form

[line number] **CALL "GPOINT"**, image string argument, numeric simple argument & result, numeric simple argument & result, numeric simple argument & result

Descriptive Form

[line number] **CALL "GPOINT"**, image, beginning vector number where search begins, X position nearest X relative to graphic cursor (at exit), Y position nearest Y relative to graphic cursor (at exit)

Format Example

```
500 CALL "GPOINT",I$,N,X,Y
```

Explanation

This commmand does not alter the image string.

GPOINT finds an image I$ point with a number greater than or equal to the input argument N. The number of this discovered point is returned in N. GPOINT returns in X,Y the absolute position that is nearest to Xi + Xg,Yi + Yg (where Xi and Yi are the input values). The variables N, X, and Y are received, redefined, and returned.

At CALL entry:

- N = the vector number where the search starts.
- X,Y = the search center relative to the graphic cursor.

At CALL exit:

- The sign (N) indicates if the Vector is a MOVE or a DRAW
- N = the nearest point.
- X,Y = the absolute location of point N in string I$.

Sample Program

See Figure 3-23.

```
1 REM GPRINT example
100 INIT
110 X=0
120 Y=0
130 P=2
140 Q=-2
150 MOVE 0,0
160 REM Generate line segments with city names at junctions.
170 FOR I=1 TO 8
180 X=X+10
190 Y=Y+10*RND(I)
200 DRAW X,Y
210 REM Input and print city name at offset P,Q from graphic location.
220 CALL "GINPUT","CITY: ",-3000,P,Q,K$
230 CALL "GPRINT",K$,1,P,Q
240 NEXT I
250 END
```

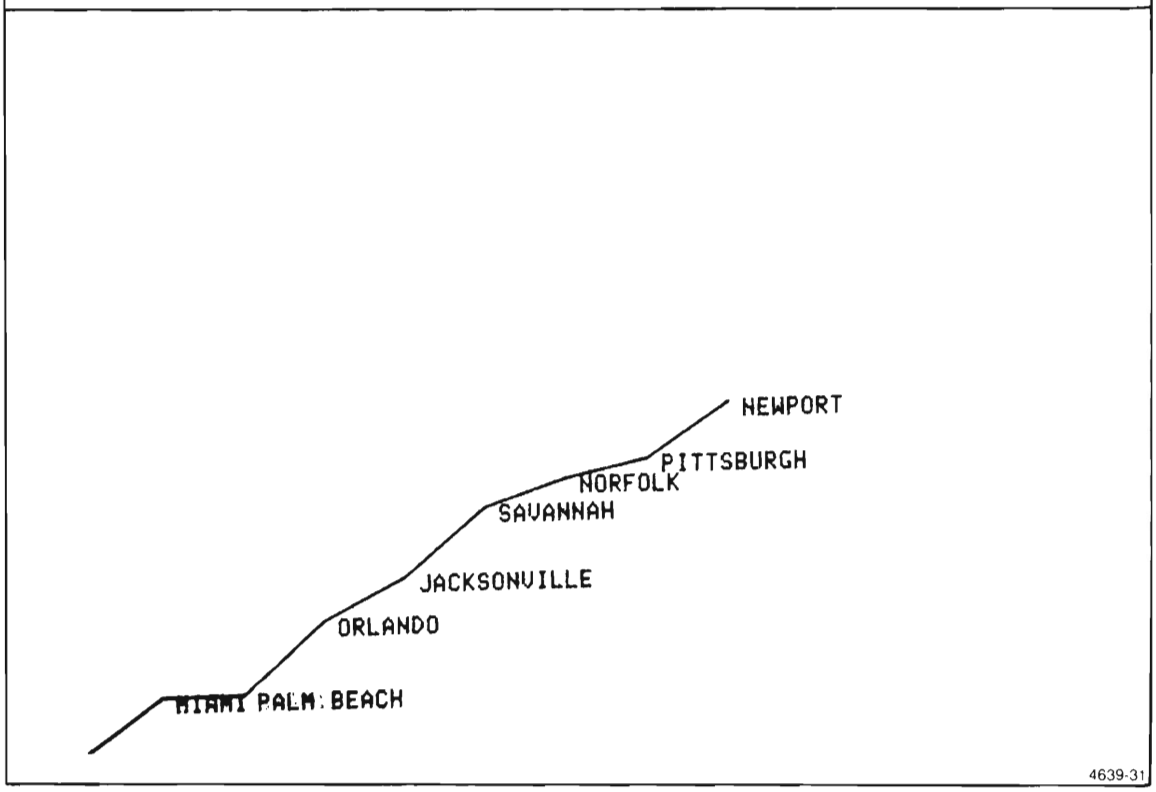


Figure 3-24. Example of GPRINT

GROTATE (Graphics Rotate)

This command rotates the image by angle A around coordinate position X,Y relative to the graphic cursor.

Syntax Form

[line number] **CALL "GROTATE"**, image string argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "GROTATE"**, image to be rotated, angle of rotation, position X relative to graphic cursor, position Y relative to graphic cursor

Format Example

```
500 CALL "GROTATE",I$,A,X,Y
```

Explanation

This command alters the image string.

All angles are measured counterclockwise (CCW) and in current trigonometric units.

NOTE

The radians (or degrees) specified in ROTATE and SHEAR commands are limited to:

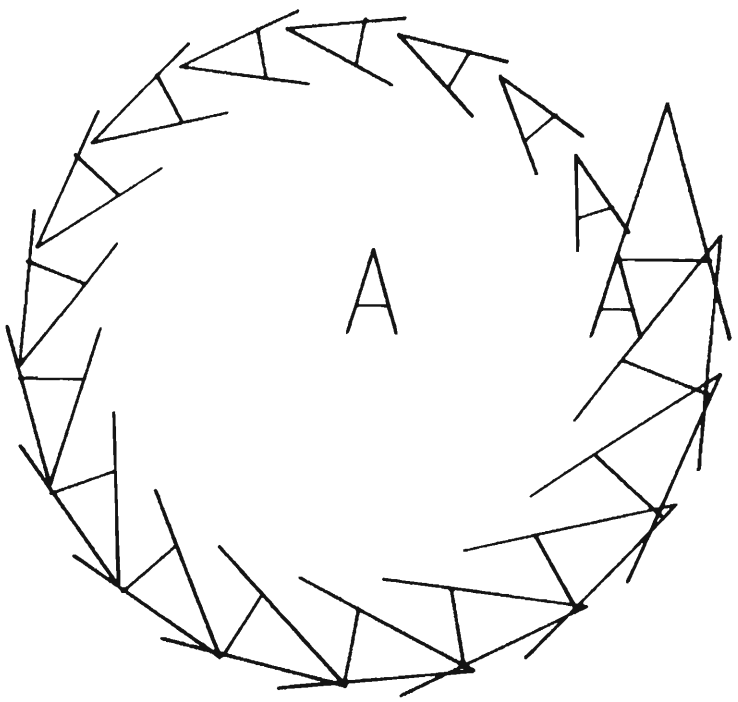
$-2PI*65.536 \leqslant Angle \leqslant 2PI*65.536$ (radians); that is, $= \pm 131.072*PI = 411.774$ (radians).

Or, by degrees: $-2PI*65.536\ (180/PI) = -360*65.536 \leqslant Angle \leqslant 360*65.536 = 23592.96$ °.

Sample Program

See Figure 3-25.

```
100 REM GROTATE and GSCALE example
110 INIT
120 SET DEGREES
130 REM Generate the image of an A.
140 DIM F(10),F$(15),G$(15)
150 DATA -50,-50,53,60,56,50,-51,-53.3,55,53.3
160 READ F
170 CALL "CHANGE",F,F$
180 REM Draw the image with lower left at 65,50.
190 CALL "ADRAW",F$,1,65,50
200 REM Keep the original image in G$.
210 G$=F$
220 REM The transforms have lower left at 65,50.
230 MOVE 65,50
240 S=1
250 FOR A=0 TO 360 STEP 20
260 REM Compute X,Y increments for radius 30 and angle A.
270 X=30*COS(A)
280 Y=30*SIN(A)
290 F$=G$
300 REM Rotate and scale centers are at 65+0,50+0.
310 CALL "GROTATE",F$,A,0,0
320 CALL "GSCALE",F$,S,S,0,0
330 REM The image data now has the larger images farther out.
340 REM The GDRAW will adjust the drawn position to 65+X,50+Y.
350 CALL "GDRAW",F$,1,X,Y
360 S=S+0.1
370 NEXT A
380 END
```



4639-32

Figure 3-25. Example of ROTATE and GSCALE

GSCALE (Graphics Scale)

This command scales the image by factors H,V around coordinate position X,Y relative to the graphic cursor.

Syntax Form

[line number] **CALL "GSCALE"**, image string argument & result, numeric simple/ constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "GSCALE"**, image to be scaled, horizontal factor, vertical factor, X position relative to the graphic cursor, Y position relative to the graphic cursor

Format Example

```
500 CALL "GSCALE",I$,H,V,X,Y
```

Explanation

This command alters the image string.

Negative values for H and/or V result in a mirror image. $-65.535 \leqslant$ scale factor range $\leqslant +65.535$ (17-bit resolution).

NOTE

SCALE and TAPER commands operate over a range of -65.535 to + 65.535. This gives a resolution of 1 in 2∧10 (about 0.001). The input of a number less than -65.535 or greater than 65.535 has the same effect as an input of -65.535 or 65.535 respectively. The full range is rarely used. The screen usually limits the image before reaching the range extremities.

Sample Program

See previous example at Figure 3-25).

GSHEAR (Graphics Shear)

This command shears the image by angles H,V around the X,Y coordinate position relative to the graphic cursor.

Syntax Form

[line number] **CALL "GSHEAR"**, string argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "GSHEAR"**, image to be sheared, horizontal angle, vertical angle, horizontal position relative to graphic cursor, vertical position relative to graphic cursor

Format Example

```
500 CALL "GSHEAR",I$,H,V,X,Y
```

Explanation

All angles are measured counterclockwise (CCW) and in current trigonometric units. This command alters the image string.

NOTE

The radians specified in SHEAR and ROTATE commands are limited to:

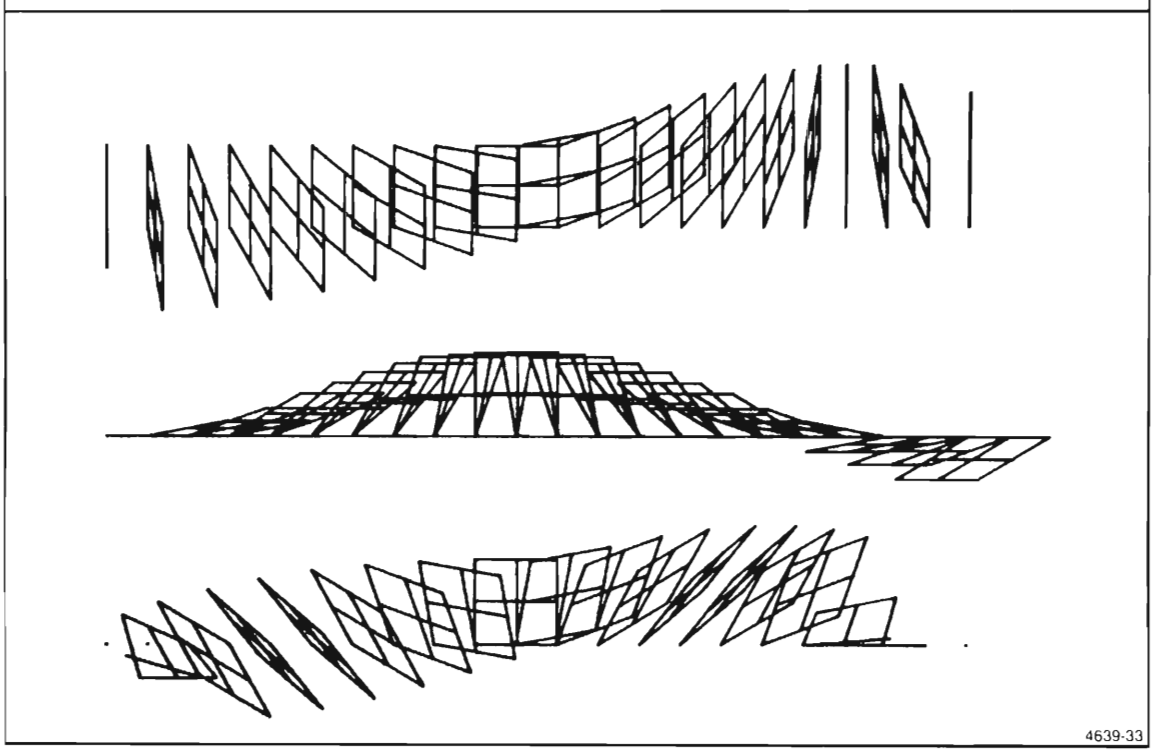
$-2PI*65.536 \leqslant Angle \leqslant 2PI*65.536$ (radians); that is, $= \pm 131.072*PI = 411.774$ (radians).

Or, by degrees: $-2PI*65.536\,(180/PI) = -360*65.536 \leqslant Angle \leqslant 360*65.536 = 23592.96$ °.

Sample Program

See Figure 3-26).

```
100 REM GSHEAR example
110 INIT
120 SET DEGREES
130 REM Generate and draw the grid image.
140 DIM F(18),F$(27),G$(27)
150 DATA -50,-50,50,60,60,60,60,50,50,50,-55,-50,55,60,-50,-55,60,55
160 READ F
170 CALL "CHANGE",F,F$
180 G$=F$
190 MOVE 5,80
200 E=0
210 FOR D=-90 TO 120 STEP 10
220 GOSUB 340
230 NEXT D
240 D=0
250 MOVE 5,55
260 FOR E=-90 TO 120 STEP 10
270 GOSUB 340
280 NEXT E
290 MOVE 5,30
300 FOR D=-90 TO 120 STEP 10
310 E=-D
320 GOSUB 340
330 NEXT D
335 END
340 REM Subroutine to draw a sheared grid image.
350 F$=G$
360 REM When the image goes off screen in the shear it collapses
370 REM into a point due to limiting at the screen boundary.
380 CALL "GSHEAR",F$,D,E,0,0
390 CALL "GDRAW",F$,1,0,0
400 RMOVE 5,0
410 RETURN
```



4639-33

Figure 3-26. Example of GSHEAR.

GTAPER (Graphics Taper)

This command tapers the image by factors H,V around coordinate position X,Y relative to the graphic cursor.

Syntax Form

[line number] **CALL "GTAPER"**, image string argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "GTAPER"**, image to be tapered, horizontal taper factor, vertical taper factor, position X relative to graphic cursor, position Y relative to graphic cursor

Format Example

```
500 CALL "GTAPER",I$,H,V,X,Y
```

Explanation

Negative values for H and/or V result in a mirror image. $-65.535 \leqslant$ scale factor range $\leqslant +65.535$ (17-bit resolution). This command alters the image string.

NOTE

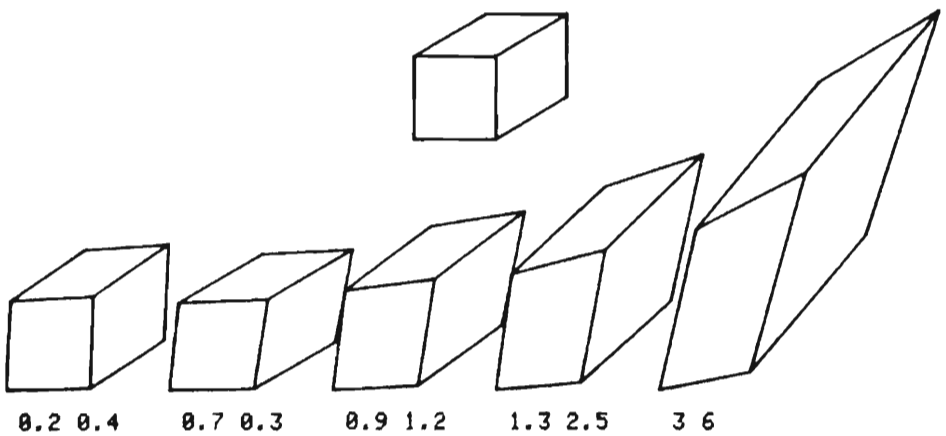
TAPER and SCALE commands operate over a range of -65.535 to +65.535. This gives a resolution of 1 in 2∧10 (about 0.001). The input of a number less than -65.535 or greater than 65.535 has the same effect as an input of -65.535 or 65.535 respectively. The full range is rarely used. The screen usually limits the image before reaching the range extremities.

Sample Program

See Figure 3-27.

```
100 REM GTAPER example
110 INIT
120 PAGE
130 REM Get image of box and back-up.
140 DIM B(24),B$(36),C$(36)
150 DATA -50,-50,50,60,60,60,60,50,50,50,-50,-60,58.66,65,68.66,65
160 DATA 68.66,55,60,50,-60,-60,68.66,65
170 READ B
180 CALL "CHANGE",B,B$
190 CALL "RDRAW",B$,1,0,0
200 REM Mark taper center.
210 MOVE 30,45
220 DRAW 30,45
230 C$=B$
240 X=-20
250 REM Taper image several times; draw and label them.
260 B$=C$
270 REM Calculate X position of drawn image.
280 X=X+20
290 IF X>110 THEN 400
300 HOME
310 PRINT "Horizontal and Vertical Taper factors (e.g. .5,.1): ";
320 INPUT H,V
330 REM Taper center is at the graphic cursor location.
340 MOVE 30,45
350 CALL "GTAPER",B$,H,V,0,0
360 CALL "ADRAW",B$,1,X,20
370 MOVE X,15
380 PRINT H;V;
390 GO TO 260
400 END
```

```
Horizontal and Vertical Taper factors (e.g. .5,.1):
```



4639-34

Figure 3-27. Example of GTAPER.

IMAGES

This command enters the magnetic tape ASCII file into an image string.

Syntax Form

[line number] **CALL "IMAGES"**, string result

Descriptive Form

[line number] **CALL "IMAGES"**, magnetic tape ASCII file

Format Example

```
500 CALL "IMAGES",I$
```

Explanation

Exit occurs when End of File (EOF) is encountered or the image string is full. No characters are lost if the string is full and the call is repeated.

While the primary purpose of the IMAGES command is to read image strings from the magnetic tape, this command is useful for entering any ASCII data into the image I$ string (provided $^{C}_{R}$ is ignored as end of string and each $^{C}_{R}$ is entered as a string character.) The resulting string length after call does not have to be an integer multiple of three. This command is very useful for duplicating ASCII files.

Using the Graphics Enhancement IMAGES command rather than INPUT@33:A$ is much easier and faster. Since image coordinates are encoded in valid ASCII character triplets, you might have a vector that consists of at least one carriage return $^{C}_{R}$, ASCII 13. BASIC uses a $^{C}_{R}$ to delimit string input (from magnetic tape or keyboard). Therefore, the image string would terminate prematurely.

Sample PROGRAM

See Figure 3-8.

```
100 REM IMAGES example
110 INIT
120 REM Generate image of an A.
130 DIM F(10),F$(15)
140 DATA -50,-50,53,60,56,50,-51,-53.3,55,53.3
150 READ F
160 CALL "CHANGE",F,F$
170 CALL "ADRAW",F$,1,50,50
180 PRINT "Number of pre-marked file to contain data image: ";
190 INPUT N
200 FIND N
210 REM Output image to magnetic tape.
220 PRINT @33:F$;
230 CLOSE
240 REM Now prepare variable F$ and input the image from tape.
250 DELETE F$
260 DIM F$(1000)
270 FIND N
280 CALL "IMAGES",F$
290 REM Draw it 10 units to the right of before.
300 CALL "ADRAW",F$,1,60,50
310 END
```

```
Number of pre-marked file to contain data image:
```

4639-35

Figure 3-28. Example of IMAGES.

INPUTS

This command enters the string characters into the keyboard buffer (28-character maximum).

Syntax Form

[line number] **CALL "INPUTS"**, text string/constant argument

Descriptive Form

[line number] **CALL "INPUTS"**, character string for keyboard buffer

Format Example

```
500 CALL "INPUTS",C$
```

Explanation

This command does not alter the text string.

INPUTS enters C$ into the keyboard buffer as though you had manually typed the string on the keyboard. This is useful for program defaults in BASIC.

Sample Programs

See Figure 3-29.

To use default, simply press the $^{C}_{R}$ (RETURN) key, or use the line edit keys to change the line (C$).

NOTE

If you want execution to continue without stopping, make the last character in C$ a $^{C}_{R}$.

NOTE

The INPUTS call must precede the 405X BASIC input, but not as a result of the pressing a function key or some other interrupt after reaching the BASIC input statement.

```
100 REM INPUTS example
110 PAGE
120 PRINT "Just type <CR> two times to execute program."
130 PRINT
140 C$="test"
150 REM String 'test' will be type-ahead for the POINTER command.
160 CALL "INPUTS",C$
170 PRINT "POINTER in progress"
180 POINTER X,Y,C$
190 PRINT
200 PRINT "POINTER string result was ";C$
210 PRINT
220 C$="100 miles"
230 REM String '100 miles' is type-ahead for INPUT A$ command.
240 CALL "INPUTS",C$
250 PRINT "Enter distance: ";
260 INPUT A$
270 PRINT "The distance was: ";A$
280 PRINT
290 C$="1,2,3,4"
300 REM String '1,2,3,4" is type-ahead for INPUT A,B command.
310 CALL "INPUTS",C$
320 DIM A(3)
330 PRINT "Enter 4 numbers: ";
340 INPUT A,B
350 PRINT "The 4 numbers were: ";A,B
360 END
```

```
Just type <CR> two times to execute program.

POINTER in progress

POINTER string result was t

Enter distance: 100 miles
The distance was: 100 miles

Enter 4 numbers: 1,2,3,4
The 4 numbers were:

 1 2 3

 4
```

4639-36

Figure 3-29. Example of INPUTS.

JCROSS (Joystick Cross)

This command draws the crosshair joystick cursor C times or until a keyboard key is pressed, whichever comes first.

Syntax Form

[line number] **CALL "JCROSS"**, numeric simple argument & result, numeric simple result, numeric simple result, text string result

Descriptive Form

[line number] **CALL "JCROSS"**, display count, joystick X position, joystick Y position, any pending keyboard keys K$

Format Example

```
500 CALL "JCROSS",C,X,Y,K$
```

Explanation

The crosshair joystick cursor will be drawn C times or until a keyboard key is pressed, whichever occurs first.

After joystick command exit:

- C = Actual display count (if C was a variable).
- X and Y are updated to the current joystick position.
- Any pending keyboard keys are saved in K$.

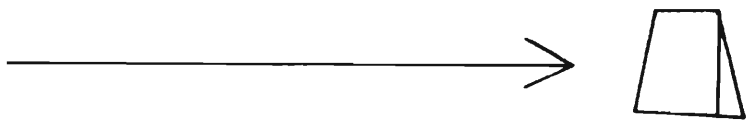
Unlike the pointer command (BASIC), all of the Graphics Enhancement joystick commands enter more than one keyboard character if more than one character is pending.

Sample Program

See Figure 3-30).

```
100 REM JCROSS example
110 PRINT "4952 Joystick required--"
120 PRINT "END:  Type E or e to quit."
130 PRINT "MOVE: Type M or m to move."
140 PRINT "DRAW: Type anything else for a drawn vector."
150 INIT
160 REM Generate image X,Y values from cross-hair input.
170 DIM F(800),F$(1200)
180 I=1
190 CALL "JCROSS",-1000,F(I),F(I+1),K$
200 IF K$="e" THEN 300
210 IF K$="m" THEN 240
220 DRAW F(I),F(I+1)
230 GO TO 280
240 REM M was typed.
250 MOVE F(I),F(I+1)
260 F(I)=-F(I)
270 F(I+1)=-F(I+1)
280 I=I+2
290 GO TO 190
300 REM E was typed.
310 REM Re-dimension to the computed point count times 2.
320 DIM F(I-1)
330 REM Now create and draw the image string.
340 CALL "CHANGE",F,F$
350 CALL "RDRAW",F$,5,0,0
360 END
```

```
4952 Joystick required--
END:  Type E or e to quit.
MOVE: Type M or m to move.
DRAW: Type anything else for a drawn vector.
```



4639-37

Figure 3-30. Example of JCROSS.

JDOTS (Joystick Dots)

This command draws the image dots C times at the joystick cursor.

Syntax Form

[line number] **CALL "JDOTS"**, image string argument, numeric simple argument & result (or constant number), numeric simple result, numeric simple result, text string result

Descriptive Form

[line number] **CALL "JDOTS"**, image dots, display count, joystick cursor X position, joystick cursor Y position, any pending keyboard keys K$

Format Example

```
500 CALL "JDOTS",I$,C,X,Y,K$
```

Explanation

This command does not alter the image string.

The crosshair joystick cursor will be drawn C times or until a keyboard key is pressed, whichever occurs first.

After joystick command exit:

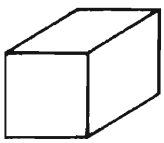
- C = Actual display count (if C was a variable).
- X and Y are updated to the current joystick position.
- Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-31.

```
100 REM JDOTS example
110 INIT
120 PAGE
130 REM Generate and draw the image of a box.
140 DIM B(24),B$(36)
150 DATA -50,-50,50,60,60,60,60,50,50,50,-50,-60,58.66,65,68.66,65
160 DATA 68.66,55,60,50,-60,-60,68.66,65
170 READ B
180 CALL "CHANGE",B,B$
190 CALL "RDRAW",B$,1,0,0
200 HOME
210 PRINT "Move the joystick, and type a character."
220 CALL "JDOTS",B$,-1000,X,Y,K$
230 REM Draw just dots of the vertices at the cross-hair location.
240 CALL "ADOTS",B$,1,X,Y
250 END
```

```
Move the joystick, and type a character.
```



4639-38

Figure 3-31. Example of JDOTS.

JDRAW (Joystick Draw)

This command draws the image vectors C times at the joystick cursor.

Syntax Form

[line number] **CALL "JDRAW"**, image string argument, numeric simple argument & result, numeric simple result, numeric simple result, text string result

Descriptive Form

[line number] **CALL "JDRAW"**, image vectors, display count, joystick cursor X position, joystick cursor Y position, any pending keyboard keys (K$)

Format Example

```
500 CALL "JDRAW",I$,C,X,Y,K$
```

Explanation

This command does not alter the image string.

The crosshair joystick cursor will be drawn C times or until a keyboard key is pressed, whichever occurs first.

After joystick command exit:

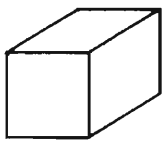
- C = Actual display count (if C was a variable).
- X and Y are updated to the current joystick position.
- Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-32.

```
100 REM JDRAW example
110 INIT
120 PAGE
125 REM Generate the image of a box.
130 DIM B(24),B$(36)
140 DATA -50,-50,50,60,60,60,60,50,50,50,-50,-60,58.66,65,68.66,65
150 DATA 68.66,55,60,50,-60,-60,68.66,65
160 READ B
170 CALL "CHANGE",B,B$
180 HOME
190 PRINT "Move the joystick, and type a character."
200 CALL "JDRAW",B$,-1000,X,Y,K$
210 CALL "ADRAW",B$,1,X,Y
220 END
```

```
Move the joystick, and type a character.
```



4639-39

Figure 3-32. Example of JDRAW.

JGIN (Joystick Gin)

This command waits a maximum of T milliseconds for a keyboard key, or until any keyboard key is typed. Finds the joystick cursor X,Y position and returns the typed keyboard keys (K$).

Syntax Form

[line number] **CALL "JGIN"**, numeric simple argument & result (or constant), numeric simple result, numeric simple result, text string result

Descriptive Form

[line number] **CALL "JGIN"**, milliseconds to wait (input)/milliseconds waited (output), joystick cursor X position, joystick cursor Y position, keyboard character typed during call or any pending keyboard keys typed before call

Format Example

```
500 CALL "JGIN",T,X,Y,K$
```

Explanation

This call can be used to clear the keyboard buffer (CALL "JGIN",0,X,Y,K$). It also can be used like CALL "WAIT". For example, to wait 5.0 seconds, use CALL "JGIN",5000,X,Y,K$.

The crosshair joystick cursor will be drawn C times or until a keyboard key is pressed, whichever occurs first.

After joystick command exit:

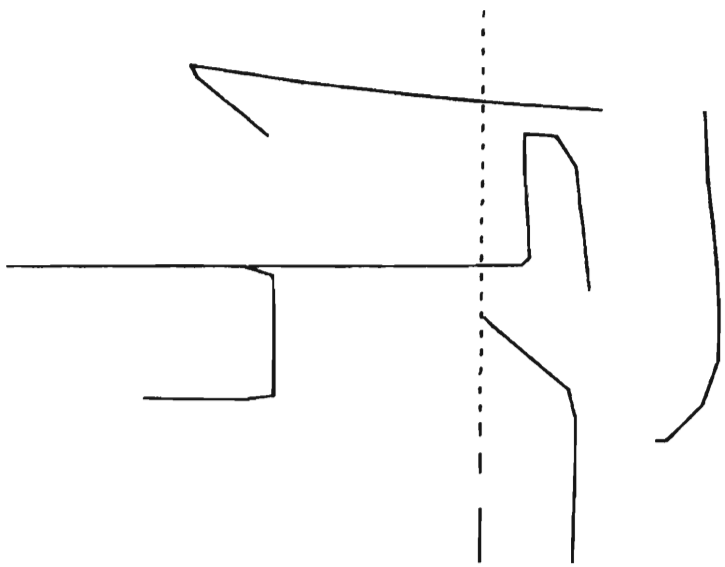
- C = Actual display count (if C was a variable).
- X and Y are updated to the current joystick position.
- Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-33.

```
100 REM JGIN example
110 INIT
120 DIM F(1000),F$(1500)
130 PAGE
140 PRINT "Begin stream digitizing by typing any character."
150 PRINT "Stream digitize by moving the joystick."
160 PRINT "Type any character to end a particular stream of data,"
170 PRINT "  and another to start again."
180 PRINT "Type E to end digitizing."
190 I=1
200 REM Wait for a typed character, then move to cross-hair position.
210 CALL "JCROSS",-1000,X,Y,K$
220 MOVE X,Y
230 F(I)=-X
240 F(I+1)=-Y
250 IF K$="e" THEN 350
260 I=I+2
270 REM Automatically digitize a drawn point after 100 milliseconds.
280 CALL "JGIN",100,F(I),F(I+1),K$
290 DRAW F(I),F(I+1)
300 IF K$="e" OR I=999 THEN 350
310 I=I+2
320 IF K$="" THEN 270
330 REM A character was typed.  Generate a move.
340 GO TO 200
350 REM End of digitizing.  Re-dimension F and draw its image.
360 DIM F(I+1)
370 CALL "CHANGE",F,F$
380 CALL "RDRAW",F$,5,0,0
390 END
```

```
Begin stream digitizing by typing any character.
Stream digitize by moving the joystick.
Type any character to end a particular stream of data,
  and another to start again.
Type E to end digitizing.
```



4639-40

Figure 3-33. Example of JGIN.

JINPUT (Joystick Input)

This command prints the string C$ and any keyboard keys (K$) C times (or until ^{C}R is typed or until the keyboard buffer is full, whichever occurs first) at the joystick cursor.

Syntax Form

[line number] **CALL "JINPUT"**, text string/constant argument, numeric simple argument & result (or constant numeric), numeric simple/constant argument, numeric simple/constant argument, text string result

Descriptive Form

[line number] **CALL "JINPUT"**, character string C$, display count, starting location X, starting location Y, pending keyboard keys saved

Format Example

```
500 CALL "JINPUT",C$,C,X,Y,K$
```

Explanation

JINPUT prints the string S$, any keyboard characters, and the crosshair joystick cursor for C times or until you press the carriage return, whichever occurs first. (See the 28 character limitation discussed under AINPUT.)

After the command exit:

- C = actual display count (if C is a variable).
- X and Y are updated to the current joystick position.
- Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-34.

```
100 REM JINPUT example
110 INIT
120 PAGE
130 REM Place city names wherever you wish on the screen
140 REM   through JINPUT selection.
150 PRINT "Move the joystick to a desired location."
160 PRINT "Then type a city name."
170 FOR I=1 TO 8
180 CALL "JINPUT","CITY: ",-3000,X,Y,K$
190 REM Print it where it was digitized.
200 CALL "APRINT",K$,1,X,Y
210 NEXT I
220 END
```

```
Move the joystick to a desired location.
Then type a city name.

                    Portland

                                    New York
                    Seattle

                                Oregon City
```

4639-41

Figure 3-34. Example of JINPUT.

JMOVE (Joystick Move)

This command moves the image to the joystick cursor.

Syntax Form

[line number] **CALL "JMOVE"**, image string argument & result, numeric simple argument & result, numeric simple result, numeric simple result, text string result

Descriptive Form

[line number] **CALL "JMOVE"**, image, display count, joystick cursor X position, joystick cursor Y position, pending keyboard keys saved

Format Example

```
500 CALL "JMOVE",I$,C,X,Y,K$
```

Explanation

This command alters the image string.

The crosshair joystick cursor will be drawn C times or until a keyboard key is pressed, whichever occurs first.

After joystick command exit:

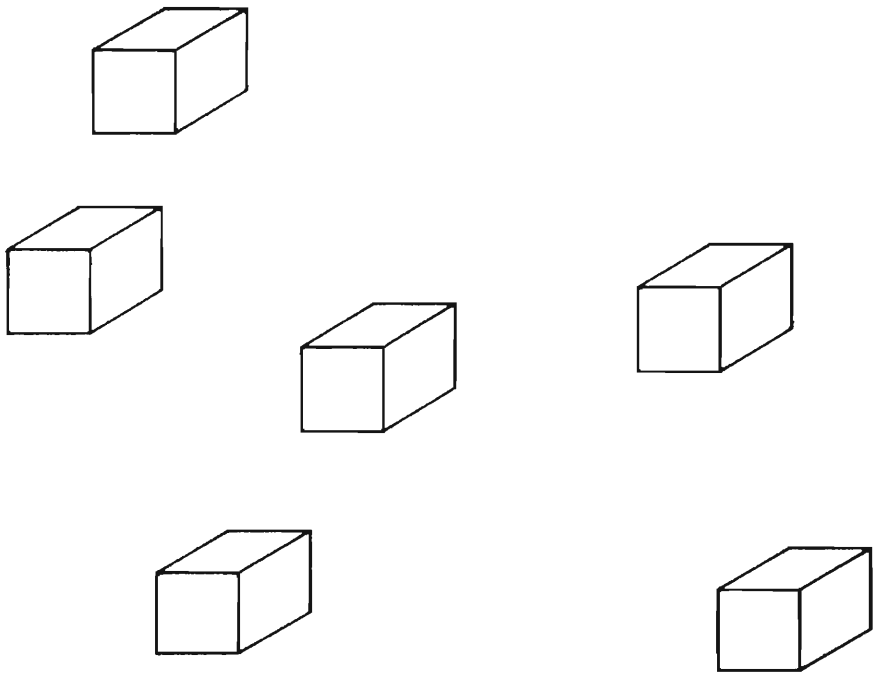
- C = Actual display count (if C was a variable).
- X and Y are updated to the current joystick position.
- Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-35.

```
100 REM JMOVE example
110 INIT
120 PAGE
130 REM Generate the image of a box.
140 DIM B(24),B$(36),C$(36)
150 DATA -50,-50,50,60,60,60,60,50,50,50,-50,-60,58.66,65,68.66,65
160 DATA 68.66,55,60,50,-60,-60,68.66,65
170 READ B
180 CALL "CHANGE",B,B$
190 C$=B$
200 PRINT "Move the joystick to a desired symbol location;"
210 PRINT "Type a character to place the symbol."
220 PRINT "Type E to end placement."
230 REM Use the joystick to place the box
240 B$=C$
250 CALL "JMOVE",B$,-1000,X,Y,K$
260 IF K$="e" THEN 290
270 CALL "RDRAW",B$,1,0,0
280 GO TO 230
290 END
```

```
Move the joystick to a desired symbol location;
Type a character to place the symbol.
Type E to end placement.
```



4639-42

Figure 3-35. Example of JMOVE.

JPOINT (Joystick Point)

This command finds the image point N and the X,Y position nearest the joystick cursor. The call is used most often to pick a point in an image with the joystick.

Syntax Form

[line number] **CALL "JPOINT"**, image string argument, numeric simple argument & result, numeric simple argument & result, numeric simple result, numeric simple result, text string result

Descriptive Form

[line number] **CALL "JPOINT"**, image, display count, vector number where search starts, joystick cursor position X, joystick cursor position Y, pending keyboard keys saved

Format Example

```
500 CALL "JPOINT",I$,C,N,X,Y,K$
```

Explanation

A Point command finds the image point N and the X,Y position nearest to absolute X,Y. The variables are received, redefined, and returned. The crosshair joystick cursor is drawn C times or until you press a keyboard key, whichever occurs first.

At JPOINT entry:

- I$ = Image.
- N = Point number of image vector where search starts.
- X,Y = Search center.
- C = Refresh count.

At JPOINT exit:

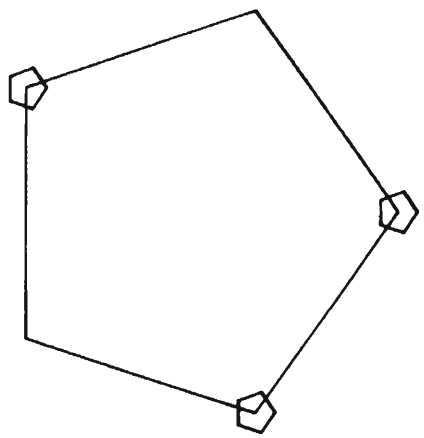
- I$ = Image (not altered).
- N = Nearest point number — the sign (N) indicates if vector is a MOVE or a DRAW.
- X,Y = Nearest point coordinates to current joystick position.
- C = Actual refresh count (if C is a numeric variable).
- K$ saves any pending keyboard keys.

Sample Program

See Figure 3-36.

```
100 REM JPOINT example
110 PAGE
120 PRINT "Select position with joystick.  Closest polygon point"
130 PRINT "  is then highlighted.  Type E to end selection."
140 INIT
150 SET DEGREES
160 REM Generate and draw a pentagon.
170 DIM F(14),F$(21),G$(21)
180 DATA -65,-50,-90,-50,72.7,73.8,44.8,64.7,44.8,35,72.7,26,90,50
190 READ F
200 CALL "CHANGE",F,F$
210 CALL "RDRAW",F$,1,0,0
220 REM Generate a scaled down version of the pentagon.
230 G$=F$
240 CALL "ASCALE",G$,0.1,0.1,65,50
250 REM Now choose a position with the joystick.
260 REM   Return a pentagon point near it.
270 CALL "JPOINT",F$,-1000,1,X,Y,K$
280 IF K$="e" THEN 320
290 REM Mark the chosen pentagon point with a small pentagon.
300 CALL "ADRAW",G$,20,X,Y
310 GO TO 250
320 END
```

```
Select position with joystick.  Closest polygon point
  is then highlighted.  Type E to end selection.
```



4639-43

Figure 3-36. Example of JPOINT.

JPRINT (Joystick Print)

This command prints the string C times at the joystick cursor.

Syntax Form

[line number] **CALL "JPRINT"**, text string/constant argument, numeric simple argument & result, numeric simple result, numeric simple result, text string result

Descriptive Form

[line number] **CALL "JPRINT"**, text string, count, joystick cursor X position, joystick cursor Y position, pending keyboard keys

Format Example

```
500 CALL "JPRINT",C$,C,X,Y,K$
```

Explanation

The crosshair joystick cursor will be drawn C times or until a keyboard key is pressed, whichever occurs first.

After joystick command exit:

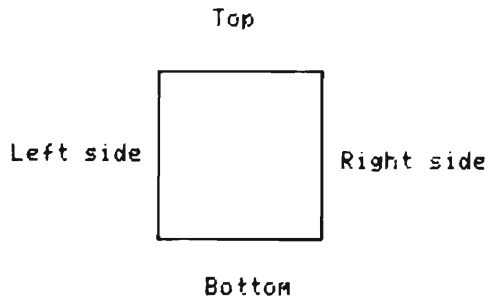
- C = Actual display count (if C was a variable).
- X and Y are updated to the current joystick position.
- Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-37.

```
100 REM JPRINT example
110 INIT
120 PAGE
130 PRINT "Locate each text string by moving joystick, then typing"
140 PRINT "  a character."
150 REM This is a square.
160 DIM F(10),F$(15)
170 DATA -55,-40,75,40,75,60,55,60,55,40
180 READ F
190 CALL "CHANGE",F,F$
200 CALL "RDRAW",F$,1,0,0
210 DATA "Right side","Left side","Top","Bottom"
220 FOR I=1 TO 4
230 READ C$
240 REM Allow placement of the string with the joystick.
250 CALL "JPRINT",C$,-1000,X,Y,K$
260 MOVE X,Y
270 PRINT C$;
280 NEXT I
290 END
```

```
Locate each text string by moving joystick, then typing
  a character.
```



4639-44

Figure 3-37. Example of JPRINT.

JROTATE (Joystick Rotate)

This command rotates the image by angle A around the joystick cursor.

Syntax Form

[line number] **CALL "JROTATE"**, image string argument & result, numeric simple argument & result, numeric simple/constant argument, numeric simple result, numeric simple result, text string result

Description Form

[line number] **CALL "JROTATE"**, image to be rotated, times that joystick cursor is drawn, angle of rotation, joystick cursor X position, joystick cursor Y position, pending keyboard keys

Format Example

```
500 CALL "JROTATE",I$,C,A,X,Y,K$
```

Explanation

All angles are measured counterclockwise and in current trigonometric units.

NOTE

The radians specified in ROTATE and SHEAR commands are limited to:

$-2PI*65.536 \leqslant Angle \leqslant 2PI*65.536$ *(radians); that is,* $= \pm 131.072*PI =$ *411.774 (radians).*

Or, by degrees: $-2PI*65.536\,(180/PI) = -360*65.536 \leqslant Angle \leqslant 360*65.536 = 23592.96°$.

The crosshair joystick cursor will be drawn C times or until a keyboard key is pressed, whichever occurs first.

After joystick command exit:

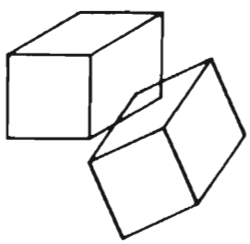
- C = Actual display count (if C was a variable).
- X and Y are updated to the current joystick position.
- Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-38.

```
100 REM JROTATE example
110 INIT
120 SET DEGREES
130 PAGE
140 REM Generate and draw the image of a box.
150 DIM B(24),B$(36),C$(36)
160 DATA -50,-50,50,60,60,60,60,50,50,50,-50,-60,58.66,65,68.66,65
170 DATA 68.66,55,60,50,-60,-60,68.66,65
180 READ B
190 CALL "CHANGE",B,B$
200 CALL "RDRAW",B$,1,0,0
210 C$=B$
220 REM Repeatedly enter rotation and joystick chosen center
230 REM   of rotation.
240 B$=C$
250 HOME
260 PRINT "Type rotation angle (0 to stop): ";
270 INPUT A
280 IF A=0 THEN 370
290 PRINT "Now select rotation center with joystick."
300 CALL "JROTATE",B$,-1000,A,X,Y,K$
310 REM Mark rotation center.
320 MOVE X,Y
330 DRAW X,Y
340 REM Draw transformed image.
350 CALL "RDRAW",B$,1,0,0
360 GO TO 220
370 END
```

```
Type rotation angle (0 to stop): 25
Now select rotation center with joystick.
```



4639-45

Figure 3-38. Example of JROTATE.

JSCALE (Joystick Scale)

This command scales the image by factors H,V around the joystick cursor.

Syntax Form

[line number] **CALL "JSCALE"**, image string argument & result, numeric simple argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple result, numeric simple result, image string result

Descriptive Form

[line number] **CALL "JSCALE"**, image to be scaled, actual display count, horizontal factor, vertical factor, joystick cursor position X, joystick cursor position Y, pending keyboard keys

Format Example

```
500 CALL "JSCALE",I$,C,H,V,X,Y,K$
```

Explanation

Negative values for H and/or V result in a mirror image. $-65.535 \leqslant$ scale factor range $\leqslant +65.535$ (17-bit resolution).

NOTE

SCALE and TAPER commands operate over a range of -65.535 to + 65.535. This gives a resolution of 1 in 2∧10 (about 0.001). The input of a number less than -65.535 or greater than 65.535 has the same effect as an input of -65.535 or 65.535 respectively. The full range is rarely used. The screen usually limits the image before reaching the range extremities.

The crosshair joystick cursor will be drawn C times or until a keyboard key is pressed, whichever occurs first.

After joystick command exit:

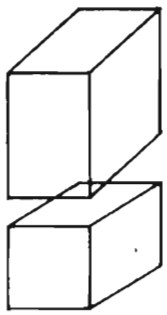
- C = Actual display count (if C was a variable).
- X and Y are updated to the current joystick position.
- Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-39.

```
100 REM JSCALE example
110 INIT
120 SET DEGREES
130 PAGE
140 REM Generate and draw box.
150 DIM B(24),B$(36),C$(36)
160 DATA -50,-50,50,60,60,60,60,50,50,50,-50,-60,58.66,65,68.66,65
170 DATA 68.66,55,60,50,-60,-60,68.66,65
180 READ B
190 CALL "CHANGE",B,B$
200 CALL "RDRAW",B$,1,0,0
210 C$=B$
220 REM Repeatedly enter scaling factors and joystick chosen
230 REM   scaling center. Transform and draw the image.
240 B$=C$
250 HOME
260 PRINT "Type X and Y scale factors (0,0 to end): ";
270 INPUT H,V
280 IF H=0 AND V=0 THEN 370
290 PRINT "Now select scaling center with joystick."
300 CALL "JSCALE",B$,-1000,H,V,X,Y,K$
310 REM Mark the scaling center.
320 MOVE X,Y
330 DRAW X,Y
340 REM Draw the transformed image.
350 CALL "RDRAW",B$,10,0,0
360 GO TO 220
370 END
```

```
Type X and Y scale factors (0,0 to end): 1,1.5
Now select scaling center with joystick.
```



4639-46

Figure 3-39. Example of JSCALE.

JSHEAR (Joystick Shear)

This command shears the image by angles H,V around the joystick cursor.

Syntax Form

[line number] **CALL "JSHEAR"**, image string argument & result, numeric simple argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple result, numeric simple result, text string result

Descriptive Form

[line number] **CALL "JSHEAR"**, image to be sheared, actual display count, horizontal angle, vertical angle, joystick cursor position X, joystick cursor position Y, pending keyboard keys

Format Example

```
500 CALL "JSHEAR",I$,C,H,V,X,Y,K$
```

Explanation

All angles are measured counterclockwise and in current trigonometric units.

NOTE

The radians specified in SHEAR and ROTATE commands are limited to:

$-2PI*65.536 \leqslant Angle \leqslant 2PI*65.536$ (radians); that is, $= \pm 131.072*PI = 411.774$ (radians).

Or, by degrees: $-2PI*65.536\ (180/PI) = -360*65.536 \leqslant Angle \leqslant 360*65.536 = 23592.96°$.

The crosshair joystick cursor will be drawn C times or until a keyboard key is pressed, whichever occurs first.

After joystick command exit:

- C = Actual display count (if C was a variable).
- X and Y are updated to the current joystick position.
- Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-40.

```
100 REM JSHEAR example
110 INIT
120 SET DEGREES
130 PAGE
140 REM Generate and draw the image of an A.
150 DIM F(10),F$(15),G$(15)
160 DATA -50,-50,53,60,56,50,-51,-53.3,55,53.3
170 READ F
180 CALL "CHANGE",F,F$
190 CALL "RDRAW",F$,1,0,0
200 G$=F$
210 REM Repeatedly select shear factors and joystick chosen
220 REM   shearing center.
230 F$=G$
240 HOME
250 PRINT "Type X and Y shear factors (0,0 to end): ";
260 INPUT H,V
270 IF H=0 AND V=0 THEN 360
280 PRINT "Now select shearing center with joystick."
290 CALL "JSHEAR",F$,-1000,H,V,X,Y,K$
300 REM Mark shearing center.
310 MOVE X,Y
320 DRAW X,Y
330 REM Draw the result.
340 CALL "RDRAW",F$,5,0,0
350 GO TO 210
360 END
```

```
Type X and Y shear factors (0,0 to end): 30,-30
Now select shearing center with joystick.
```

4639-47

Figure 3-40. Example of JSHEAR.

JTAPER (Joystick Taper)

This command tapers the image I$ by factors H,V around the joystick cursor.

Syntax Form

[line number] **CALL "JTAPER"**, image string argument & result, numeric simple argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple result, numeric simple result, text string result

Descriptive Form

[line number] **CALL "JTAPER"**, image to be tapered, actual display count, horizontal factor, vertical factor, joystick cursor position X, joystick cursor position Y, pending keyboard keys

Format Example

```
500 CALL "JTAPER",I$,C,H,V,X,Y,K$
```

Explanation

All Joystick commands allow a variable or a constant for the refresh count.

Negative values for H and/or V result in a mirror image. $-65.535 \leqslant$ scale factor range $\leqslant +65.535$ (17-bit resolution).

SCALE and TAPER commands operate over a range of –65.535 to +65.535. This gives a resolution of 1 in 2∧10 (about 0.001). The input of a number less than –65.535 or greater than 65.535 has the same effect as an input of –65.535 or 65.535 respectively. The full range is rarely used. The screen usually limits the image before reaching the range extremities.

The crosshair joystick cursor will be drawn C times or until a keyboard key is pressed, whichever occurs first.

After joystick command exit:

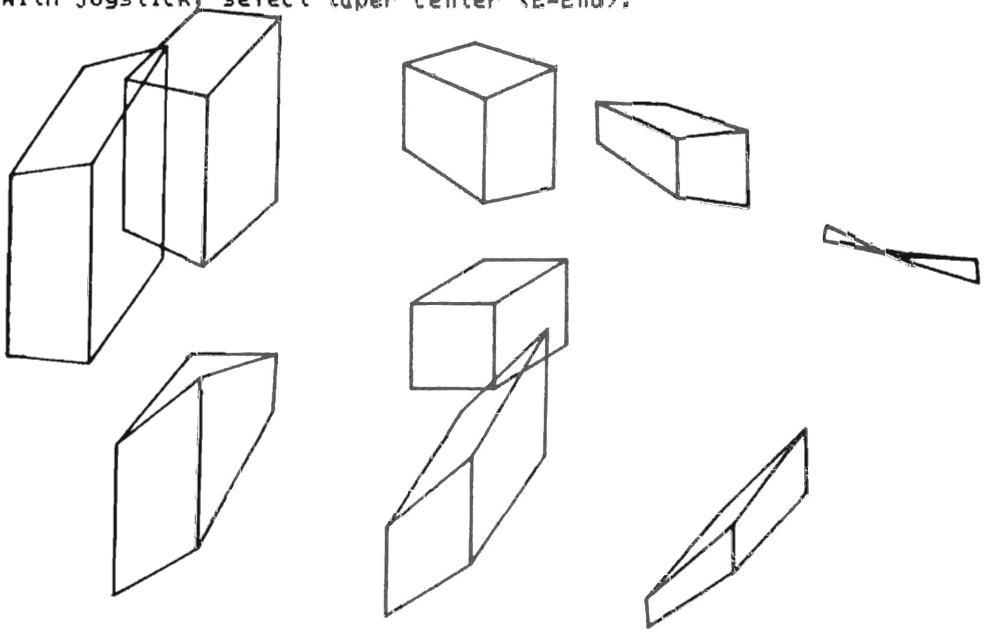
- C = Actual display count (if C was a variable).
- X and Y are updated to the current joystick position.
- Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-41.

```
100 REM JTAPER example
110 INIT
120 PAGE
130 REM Here is a box.
140 DIM B(24),B$(36),C$(36)
150 DATA -50,-50,50,60,60,60,60,50,50,50,-50,-60,50.66,65,60.66,65
160 DATA 60.66,55,60,50,-60,-60,60.66,65
170 READ B
180 CALL "CHANGE",B,B$
190 CALL "RDRAW",B$,1,0,0
200 C$=B$
210 HOME
220 PRINT "Horizontal and Vertical Taper factors (e.g. .5,.1): ";
230 INPUT H,V
240 PRINT "With joystick, select taper center (E=End)."
250 REM Repeatedly locate a taper center, seeing how its location
260 REM   affects the tapered image.
270 B$=C$
280 CALL "JTAPER",B$,-1000,H,V,X,Y,K$
290 IF K$="e" THEN 320
300 CALL "ADRAW",B$,1,X,Y
310 GO TO 250
320 END
```

```
Horizontal and Vertical Taper factors (e.g. .5,.1): .1,3
With joystick, select taper center (E=End).
```



4639-48

Figure 3-41. Example of JTAPER.

LOCATE (Joystick Location)

This command prints the joystick cursor location at the graphic cursor position, returns any keys typed, and refresh count if desired.

Syntax Form

[line number] **CALL "LOCATE"**, numeric simple argument & result (or constant), numeric simple result, numeric simple result, text string result

Descriptive Form

[line number] **CALL "LOCATE"**, refresh count, joystick cursor X location, joystick cursor Y location, return of any typed or pending keys

Format Example

```
500 CALL "LOCATE",C,X,Y,K$
```

Explanation

An example of this CALL is:

```
100 MOVE 65,50
110 CALL"LOCATE",-1000,X,Y,K$
```

A crosshair is drawn 1000 times in write-thru. If the joystick is moved, the crosshair tracks it. The joystick location is printed at 65,50.

LOCATE draws a crosshair cursor at the current joystick position, and prints the joystick X,Y position at the current graphic cursor location. See Figure 3-42.

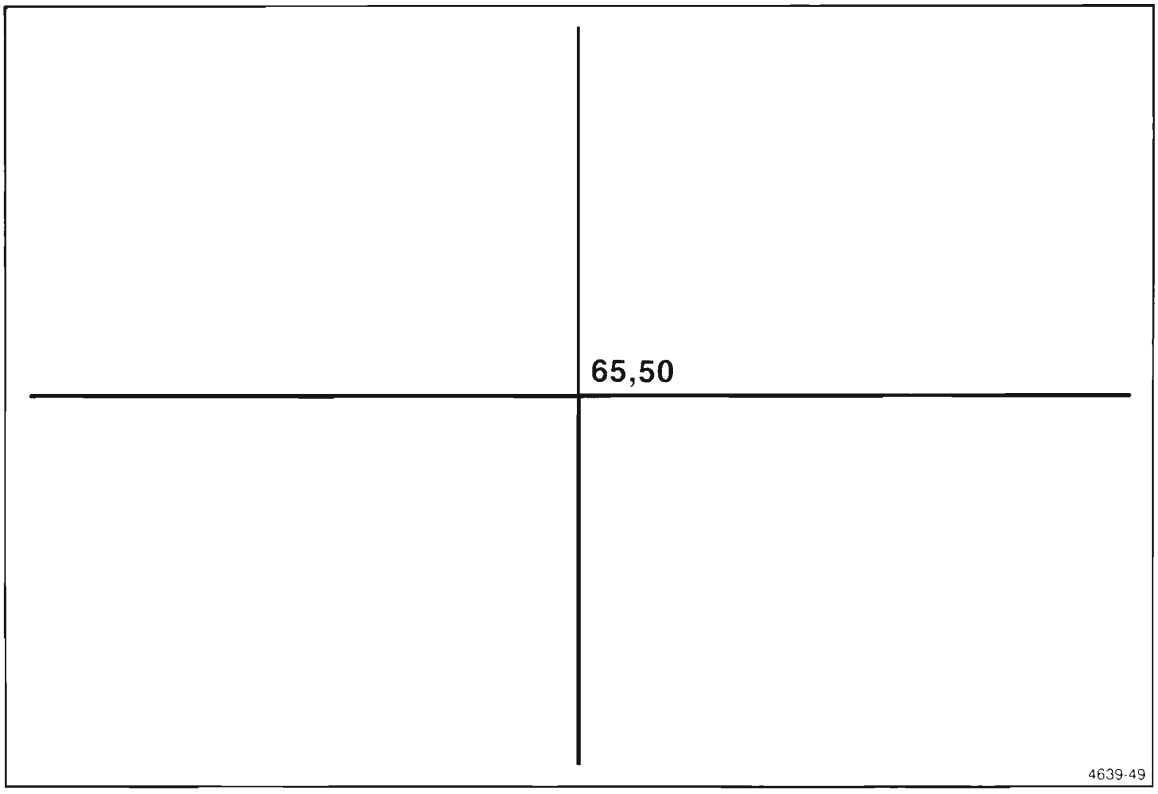


Figure 3-42. Example of Joystick Location.

MUSIC

This command plays musical notes that are encoded in an ASCII string.

Syntax Form

[line number] **CALL "MUSIC"**, text string/constant argument

Descriptive Form

[line number] CALL "MUSIC" <tempo> <note> [flat] [sharp] <octave> [dotted]

<length of note> [rest or next note] [. . .]

Format Sample

```
500 CALL "MUSIC",S$
```

Explanation

String Syntax Components. A <word> in the S$ string is defined as any one of the following:

- <tempo>
- <note>
- <note> <dot>
- <note> <octave>
- <note> <octave> <dot>
- <note> <octave> <length>
- <note> <octave> <length> <dot>
- <rest>

Tempo. The music string (S$) tempo is defined as <T> <0 . . 9>. Table 3-5 shows the beats per minute for each tempo code, T0 through T9. Select any of these tempos for the S$ string.

Table 3-5

MUSIC TEMPO

Tempo Code	Beats per Minute
T0	80
T1	90
T2	100
T3	110
T4	120
T5	130
T6	140
T7	150
T8	160
T9	170

Musical Notes. A note may be a <sharp>, <flat>, or natural:

- An upper case letter represents the note.
- The sharp symbol is <#>, the pound sign.
- The flat symbol is <b>, the lower case letter b.
- The natural note (neither flat nor sharp) needs no symbol.

Thus, a note is defined as any one of the following:

Natural	Sharp	Flat
<A>	<A#>	<Ab>
<B>	<B#>	<Bb>
<C>	<C#>	<Cb>
<D>	<D#>	<Db>
<E>	<E#>	<Eb>
<F>	<F#>	<Fb>
<G>	<G#>	<Gb>

Octaves. An octave is defined to be <0..7>. Select any of the following octaves for the S$ string:

<0> = first octave on standard piano keyboard (starts with A)
<1> = second octave
<2> = third octave
<3> = fourth octave (contains middle C as third note at 440 Hz)
<4> = fifth octave
<5> = sixth octave
<6> = seventh octave
<7> = eighth octave

NOTE

Do not confuse the first octave with octave <1>. The first octave is octave <0>; the second octave is octave <1>.

Octave <0> runs from the lowest <A> up to but not through the next higher <A>, as does each ascending octave. An "octave C" that runs from <C> to the next higher <C> overlaps a portion of two octaves as defined for MUSIC string parameters.) This manual uses the term "octave" to mean the sequence of notes from an <A> note to but not through the next higher <A> note, so that the beginning <A> of the sequence determines the octave number. See Figure 3-43.

In musical notation, an "octave A" runs from one A note to and including the next higher A note. However, in this MUSIC routine, octave <0> runs from <A> through <G> (<G#>, to be exact); the following <A> (eighth natural note up) is the first note of octave <1>. Thus, each <A> natural is the first note and each <G#> is the last note, of another octave.

The notation of the first 13 notes of the piano is as follows:

A0 A#0 B0 C0 C#0 D0 D#0 E0 F0 F#0 G0 G#0 A1 (A0 through G#0 composes octave <0>; A1 begins octave <1>).

Note Length. The length of a note is defined to be from a subset of <1 . . 99>. The number is usually a reciprocal of the length, like 8 is of 1/8. It does not need an alphabetical designator (like T in the Tempo parameter or R in the Rest parameter). Select any of the following lengths for the S$ string:

<1> = whole note
<2> = half note
<4> = quarter note
<8> = eighth note
<16> = sixteenth note
<32> = thirty-second note
<64> = sixty-fourth note

Dotted Notes. A dotted note is defined as any <word> whose last element is a period <.>. A dotted note increases its time value by half. For example, a dotted half note has the equivalent count of three quarter notes; and a dotted eighth note equals three sixteenth notes.

Thirds and Fifths. To enter a third or a fifth note-length, multiply the length by 3 or 5, respectively. For example, an F-sharp eighth note (F#8) times three (for a third) equals F#24, and times five (for a fifth) equals <F#40>.

Eighth-note triplets have a combined value of 1/4 (3/12). For example, on <F#> in octave <0>, an eighth-note triplet notation would be F#012F#F#. The latter two <F#F#> do not need to repeat the octave <0> and length <12> of the former <F#012>, since the three notes are identical.

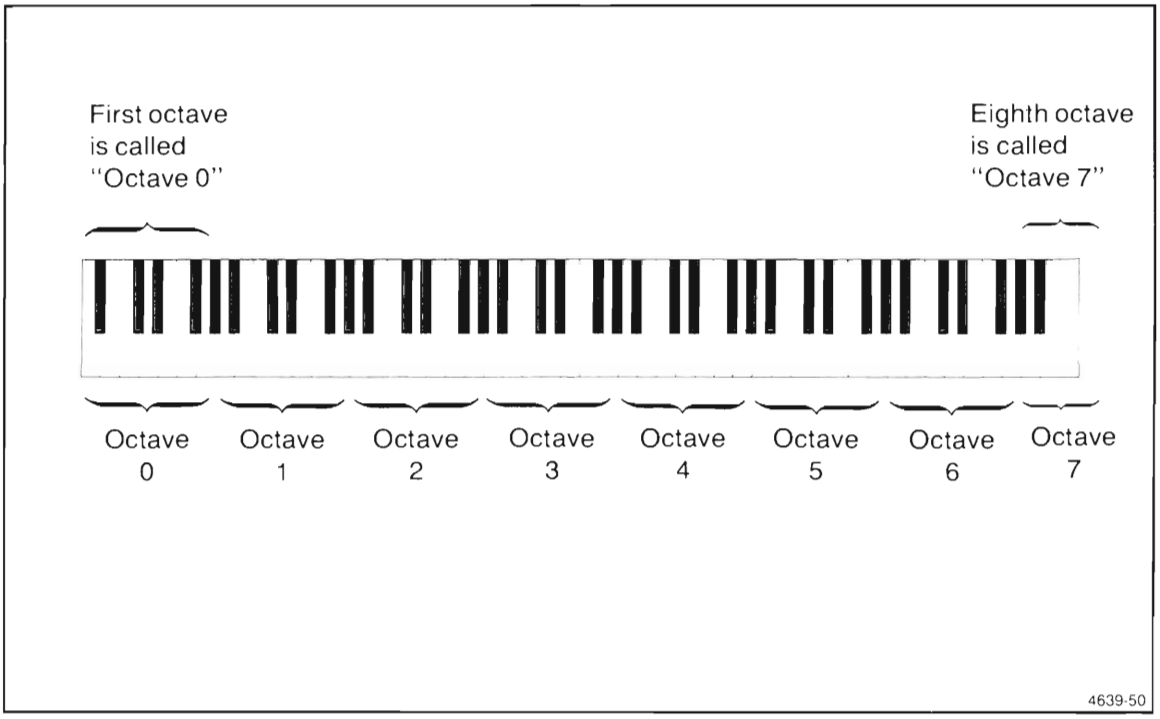


Figure 3-43. Keyboard Octaves.

Rests. A rest is defined to be <R><1..99>. The number is a reciprocal of the length. For example, 4 is the reciprocal of ¼th; thus, <R4> indicates a quarter rest. <R1> is a whole rest.

Select any of the following independent rest length reciprocals for the S$ string: <R1>, <R2>, <R4>, <R8>, <R16>, <R32>, <R64>.

You also can use one of the following triplet and fifth multiples: <3>, <5>, <6>, <10>, <12>, <20>, <24>, <40>, <48>, <80>, <96>.

String Example (S$)

```
500 CALL "MUSIC","T4E316G332B4EGB5EG58.G516G54F#R"
```

For an analysis of the string example, see Table 3-6.

Table 3-6

ANALYSIS OF STRING EXAMPLE (T4E316G332B4EGB5EG58.G516G54F#R)

Expression in String	Note or Element	Octave KB: L to R	Value
T4	Tempo = 4	-	120 beats/min
E316	E	3	1/16 note
G332	G	3	1/32 note
B4(32)[a]	B	4	1/32 note
E(432)[a]	E	4	1/32 note
G(432)[a]	G	4	1/32 note
B5(32)[a]	B	5	1/32 note
E(532)[a]	E	5	1/32 note
G58.	G	5	Dotted 1/8 (= 3/16)
G516	G	5	1/16 note
G54	G	5	1/4 note
F#(54)[a]	F#	5	1/4 note
R(4)[a]	Rest	-	1/4 note

[a] **Characters within parentheses in the first column are implied by default but not typed as part of the S$ string; that is, implicit, not explicit. The following paragraphs explain this.**

When both the octave and length of a note is the same as that of the preceding note, you need only add the note to the S$ string without repeating the octave and length. The expression defaults to the previous octave and length when those parameters remain the same. In the string example, G54 (G note, 5th octave, quarter note length) is followed by F# only, which defaults to F#54.

When a note is in a different octave but has the same length as the preceding note, you add the new octave without repeating the note length. The expression defaults to the previous note length.

When a note is in the same octave but has a different length from the preceding note , you must repeat the octave before adding the new length to the string. In the string example, G316 is followed by E332 rather than E32 (because E32 would be read as the 3rd octave and a half note (2), rather than as a 32nd note).

Invalid words or characters within the music string are ignored; however, such characters are not allowed between a note and its octave, between an octave and a note length, or between a note and its qualifier (b,#, or .). An example of allowable invalid characters (such as spaces and control characters) within the string example is:

T4 E316 G332 M B4 E G B5ZEQIMG58.:G516/G54>F#MR . . .

Push the HOME/PAGE key after executing the MUSIC command if you wish to home the cursor.

Interrupts. A single BREAK causes an immediate exit. This allows you to break out of a long sound sequence without the effects of BREAK BREAK.

POINTS (Locate Points)

This command finds the image I$ point N and X,Y position. Sign(N) indicates if the vector is a MOVE or a DRAW.

Syntax Form

[line number] **CALL "POINTS"**, image string argument, numeric simple argument & result, numeric simple result, numeric simple result

Descriptive Form

[line number] **CALL "POINTS"**, image, vector (point) number of interest, X position, Y position

Format Example

```
500 CALL "POINTS",I$,N,X,Y
```

Explanation

At CALL entry:

- I$ = Image.
- N = Point number of interest.
- X,Y = "Don't care" position (can be undefined).

At CALL exit:

- I$ = Image (not altered).
- N = Point number of interest (sign (N) indicates MOVE/DRAW).
- X,Y = Point number N coordinates.

Sample Program

See Figure 3-44.

```
100 REM POINTS example
110 INIT
120 SET DEGREES
130 REM Generate and draw a pentagon
140 DIM F(14),F$(21)
150 DATA -65,-50,-90,-50,72.7,73.8,44.8,64.7,44.8,35.3,72.7,26.2
160 DATA 90,50
170 READ F
180 CALL "CHANGE",F,F$
190 CALL "RDRAW",F$,1,0,0
200 REM Now find X,Y for the first four points and label them
210 REM   on the pentagon.
220 FOR I=1 TO 5
230 N=I
240 CALL "POINTS",F$,N,X,Y
250 T$="Draw␣␣␣␣␣"
260 IF N=>0 THEN 280
270 T$="Move␣␣␣␣␣"
280 MOVE X,Y
290 PRINT T$;X;Y;
300 NEXT I
310 END
```

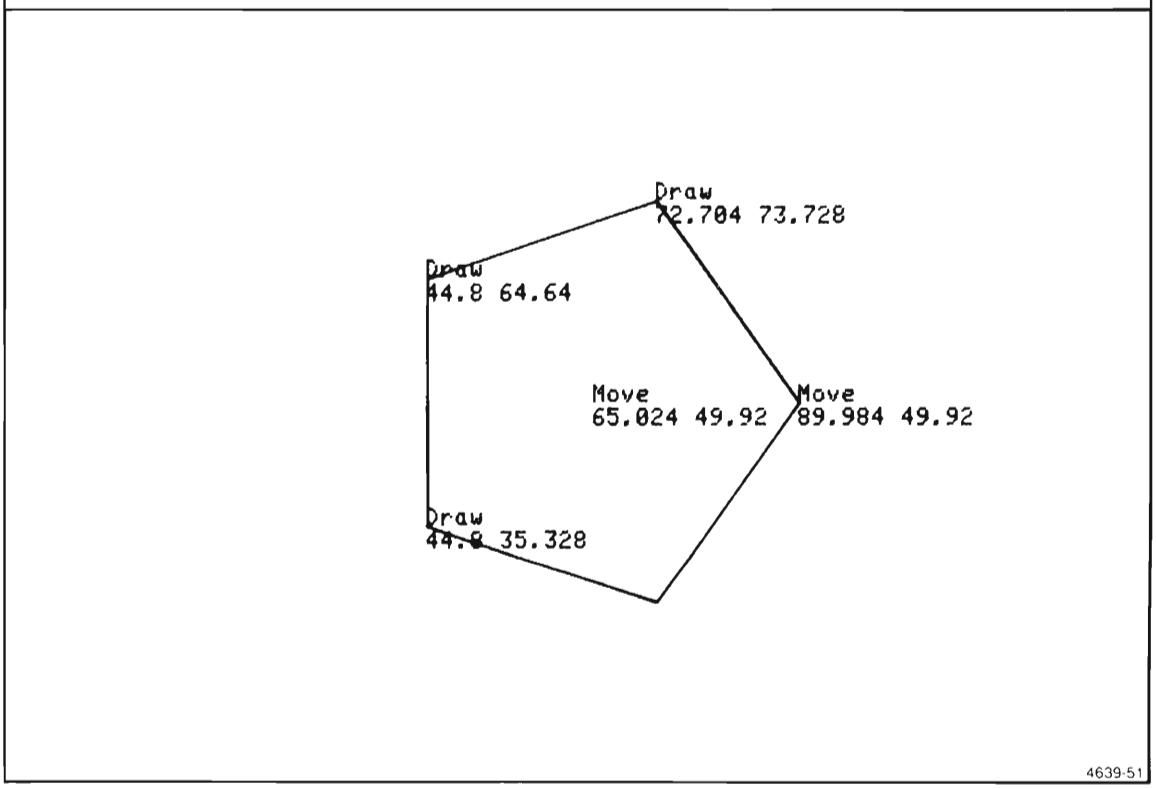


Figure 3-44. Example of POINTS.

PRINTS

This command prints the string C$ characters (printable control characters).

Syntax Form

[line number] **CALL "PRINTS"**, text string/constant argument

Descriptive Form

[line number] **CALL "PRINTS"**, printable control characters

Format Example

```
500 CALL "PRINTS",C$
```

Explanation

This CALL prints any string on the screen and makes all control characters printable.

Sample Program

```
100 A$ = ''''
110 FOR I = 0 TO 5
120 B$ = CHR(I)
130 A$ = A$ & B$
140 NEXT I
150 CALL"PRINTS",A$          Display: @ A B C D  M
```

The control characters are not executed. This command is useful for examining strings with embedded control characters.

RCROSS (Relative Cross)

This command draws the cross-hair cursor C times at X,Y relative to center screen.

Syntax Form

[line number] **CALL "RCROSS"**, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "RCROSS"**, actual display count, X position relative to center screen, Y position relative to center screen

Format Example

```
500 CALL "RCROSS",C,X,Y
```

Explanation

Since the image contains no absolute coordinates, it is relative to center screen. It is a relative command.

Sample Program

See Figure 3-45.

```
100 REM RCROSS example
110 INIT
115 REM Generate the image of a pentagon.
120 GOSUB 200
122 REM Now fill the entire screen with stored vectors.
124 REM Note that RCROSS uses offset from the middle of the screen.
130 D=-65
140 E=-50
150 CALL "RCROSS",1,D,E
160 D=D+0.256
170 E=E+0.256
180 IF D<=65 THEN 150
185 REM Now draw the Pentagon in a different kind of 'refresh'.
190 CALL "RDRAW",F$,50,0,0
195 END
200 REM Pentagon image.
210 INIT
220 SET DEGREES
230 DIM F(14),F$(21)
240 DATA -65,-50,-90,-50,72.7,73.8,44.8,64.7,44.8,35.3,72.7,26.2
250 DATA 90,50
260 READ F
270 CALL "CHANGE",F,F$
280 RETURN
```

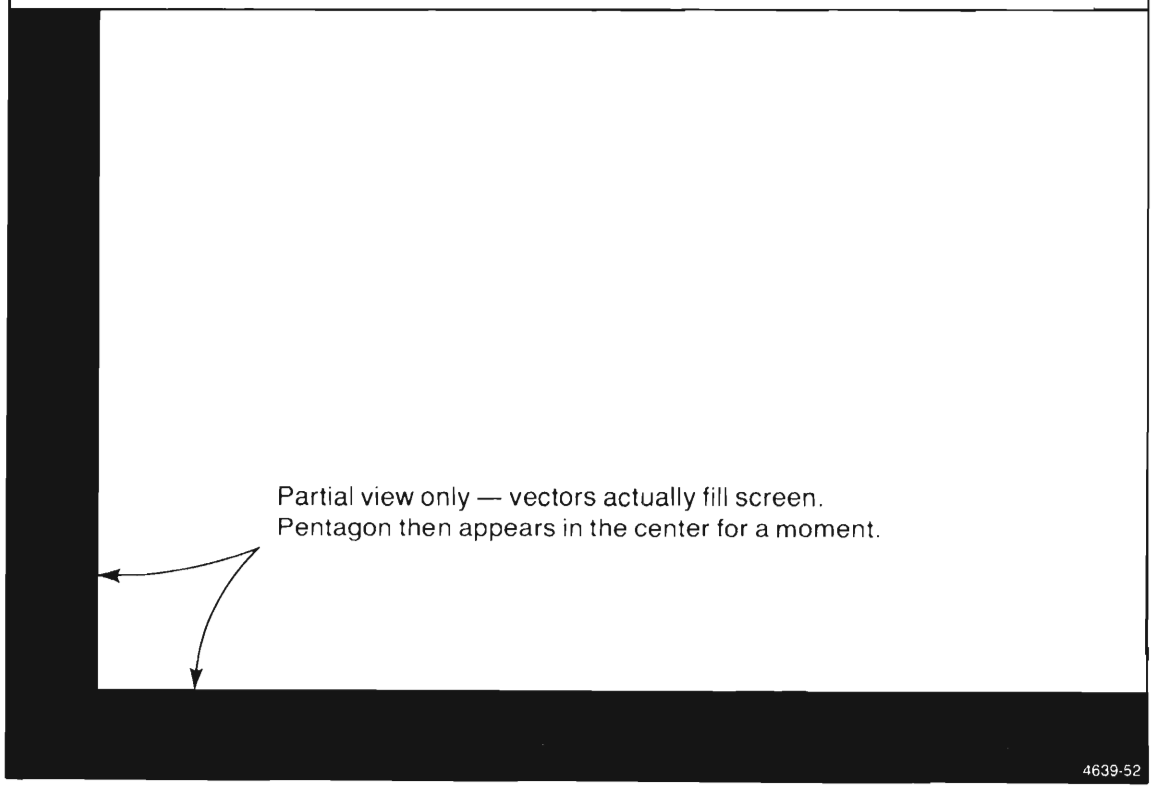


Figure 3-45. Example of RCROSS.

RDOTS (Relative Dots)

This command draws the image I$ dots C times at the X,Y relative position.

Syntax Form

[line number] **CALL "RDOTS"**, image string argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "RDOTS"**, image dots, actual display count, relative position X, relative position Y

Format Example

```
500 CALL "RDOTS",I$,C,X,Y
```

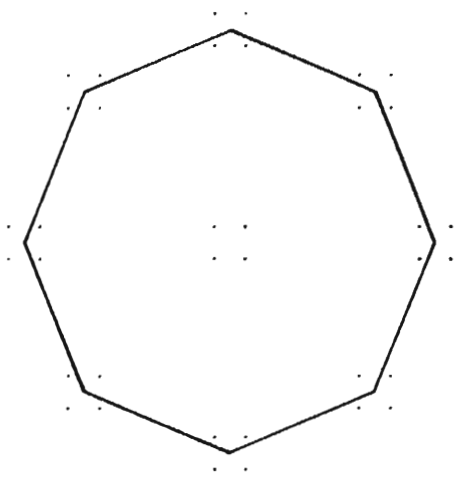
Explanation

If the first image point coordinates equal Xd,Yd, the image is drawn at Xd + X,Yd + Y. This command does not alter the image string.

Sample Program

See Figure 3-46.

```
100 REM RDOTS example
110 INIT
120 SET DEGREES
130 REM Generate and draw the image of an octagon.
140 DIM F(20),F$(30)
150 DATA -65,-50,-90,-50,82.7,67.7,65,75,47.3,67.7,40,50,47.3,32.3
160 DATA 65,25,82.7,32.3,90,50
170 READ F
180 CALL "CHANGE",F,F$
190 CALL "RDRAW",F$,1,0,0
200 REM Now draw the dots in its vertices with position
210 REM   incremented by I,J.
220 FOR I=-2 TO 2 STEP 4
230 FOR J=-2 TO 2 STEP 4
240 CALL "RDOTS",F$,30,I,J
250 NEXT J
260 NEXT I
270 END
```



4639-53

Figure 3-46. Example of RDOTS.

RDRAW (Relative Draw)

This command draws the image I$ vectors C times at the relative X,Y position.

Syntax Form

[line number] **CALL "RDRAW"**, image string argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "RDRAW"**, image vectors, actual display count, relative position X, relative position Y

Format Example

```
500 CALL "RDRAW",I$,C,X,Y
```

Explanation

If the first image point coordinates equal Xv,Yv, the image is drawn at Xv + X,Yv + Y. This command does not alter the image string.

Sample Program

See Figure 3-47.

```
100 REM RDRAW example
110 INIT
120 SET DEGREES
130 REM Generate and draw an octagon.
140 DIM F(20),F$(30)
150 DATA -65,-50,-90,-50,82.7,67.7,65,75,47.3,67.7,40,50,47.3,32.3
160 DATA 65,25,82.7,32.3,90,50
170 READ F
180 CALL "CHANGE",F,F$
190 CALL "RDRAW",F$,1,0,0
200 REM Draw it repeatedly with position incremented by I,J.
210 FOR I=-2 TO 2 STEP 4
220 FOR J=-2 TO 2 STEP 4
230 CALL "RDRAW",F$,1,I,J
240 NEXT J
250 NEXT I
260 END
```

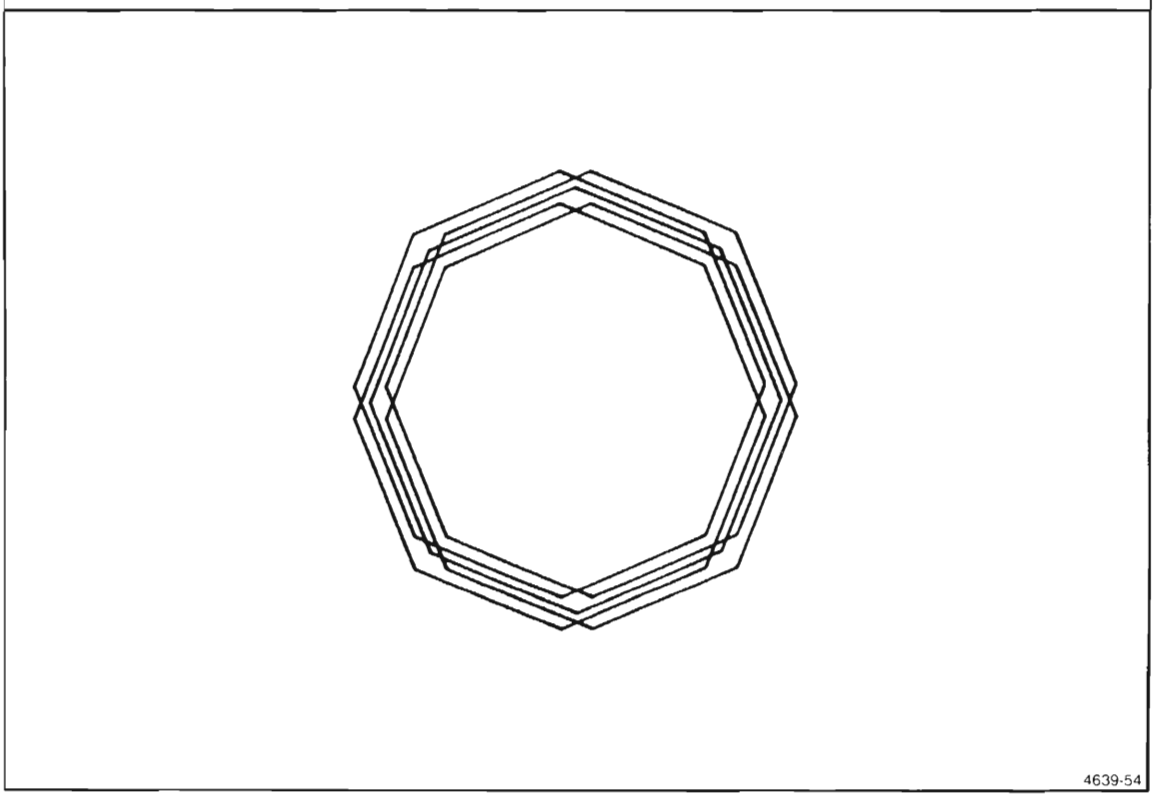


Figure 3-47. Example of RDRAW.

RGIN (Relative Gin)

Find graphic cursor X,Y position relative to X,Y.

Syntax Form

[line number] **CALL "RGIN"**, numeric simple argument & result, numeric simple argument & result

Descriptive Form

[line number] **CALL "RGIN"**, graphic cursor X position relative to absolute X position, graphic cursor Y position relative to absolute Y position

Format Example

```
500 CALL "RGIN",X,Y
```

Explanation

At CALL entry:

- X,Y = Absolute location on screen

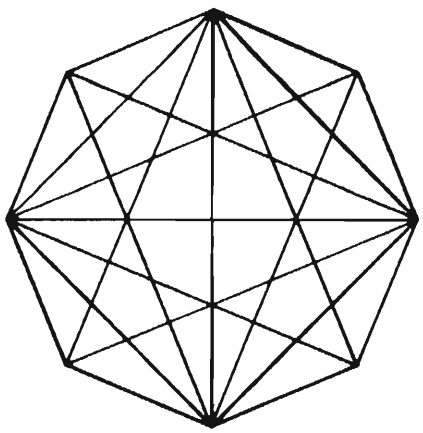
At CALL exit:

- X,Y output = Xg-X,Yg-Y input (graphic cursor proximity to X,Y)

Sample Program

See Figure 3-48.

```
100 REM RGIN example
110 INIT
120 SET DEGREES
130 REM Generate and draw an octagon.
140 DIM F(18),F$(27)
150 DATA -90,-50,82.7,67.7,65,75,47.3,67.7,40,50,47.3,32.3
160 DATA 65,25,82.7,32.3,90,50
170 REM Every other vertex, repeated.
180 DATA 90,50,65,75,40,50,65,25
190 READ F
200 CALL "CHANGE",F,F$
210 CALL "RDRAW",F$,1,0,0
220 REM Get every other vertex and draw to its
230 REM   companion vertices in the octagon.
240 FOR K=1 TO 4
250 READ U,V
260 FOR N=1 TO 8
270 N1=N
280 REM Move to the vertex.
290 MOVE U,V
300 REM Find a companion.
310 CALL "POINTS",F$,N1,X,Y
320 REM Find the difference in X,Y between them.
330 CALL "RGIN",X,Y
340 REM Draw the line.
350 RDRAW -X,-Y
360 NEXT N
370 NEXT K
380 END
```



4639-55

Figure 3-48. Example of RGIN.

RINPUT (Relative Input)

This command prints the string and any keyboard keys C times at X,Y relative to center screen.

Syntax Form

[line number] **CALL "RINPUT"**, text string/constant argument, numeric simple argument & result, numeric simple/constant argument, numeric simple/constant argument, text string result

Descriptive Form

[line number] **CALL "RINPUT"**, string C$, C times, X position relative to center screen, Y position relative to center screen, keyboard character typed during call

Format Example

```
500 CALL "RINPUT",S$,C,X,Y,K$
```

Explanation

Since the text string contains no absolute coordinates, it is made relative to center screen. It is a relative command.

Input commands print the S$ string and any K$ keyboard characters C times or until you press the carriage return, whichever occurs first. (See the 28-character limitation discussed under AINPUT.)

After the input command exit:

- C = actual display count
- Any pending keyboard keys are saved in K$.

Sample Program

See Figure 3-49.

```
100 REM RINPUT and RPRINT example
110 INIT
120 SET DEGREES
130 PAGE
140 PRINT "'Pie chart' percentage input"
150 REM Generate a circle.
160 MOVE 90,50
170 FOR A=5 TO 360 STEP 5
180 DRAW 65+25*COS(A),50+25*SIN(A)
190 NEXT A
200 REM Beginning angle A of first pie segment.
210 A=0
220 REM Find X,Y for radius 25 and angle A.
230 X=25*COS(A)
240 Y=25*SIN(A)
250 REM Draw the line between pieces.
260 MOVE 65,50
270 RDRAW X,Y
280 IF A=>360 THEN 500
290 REM Input the percentage of a piece of pie.
300 CALL "RINPUT","Percentage: ",-5000,X,Y,K$
310 IF K$="" THEN 500
320 K=VAL(K$)
330 REM Compute angle in degrees and X,Y at center of piece.
340 B=A+1.8*K
350 X=25*COS(B)
360 Y=25*SIN(B)
370 REM Input the Label for this piece.
380 CALL "RINPUT","Label: ",-5000,X,Y,M$
390 REM Add the percentage to the label.
400 M$=M$&" "
410 M$=M$&K$
420 REM If this is the left side of the pie, right justify.
430 IF B<90 OR B>270 THEN 460
440 X=X-LEN(M$)*1.8
450 REM Print the label.
460 CALL "RPRINT",M$,1,X,Y
470 REM Update the angle and do the next piece.
480 A=A+3.6*K
490 GO TO 230
500 END
```

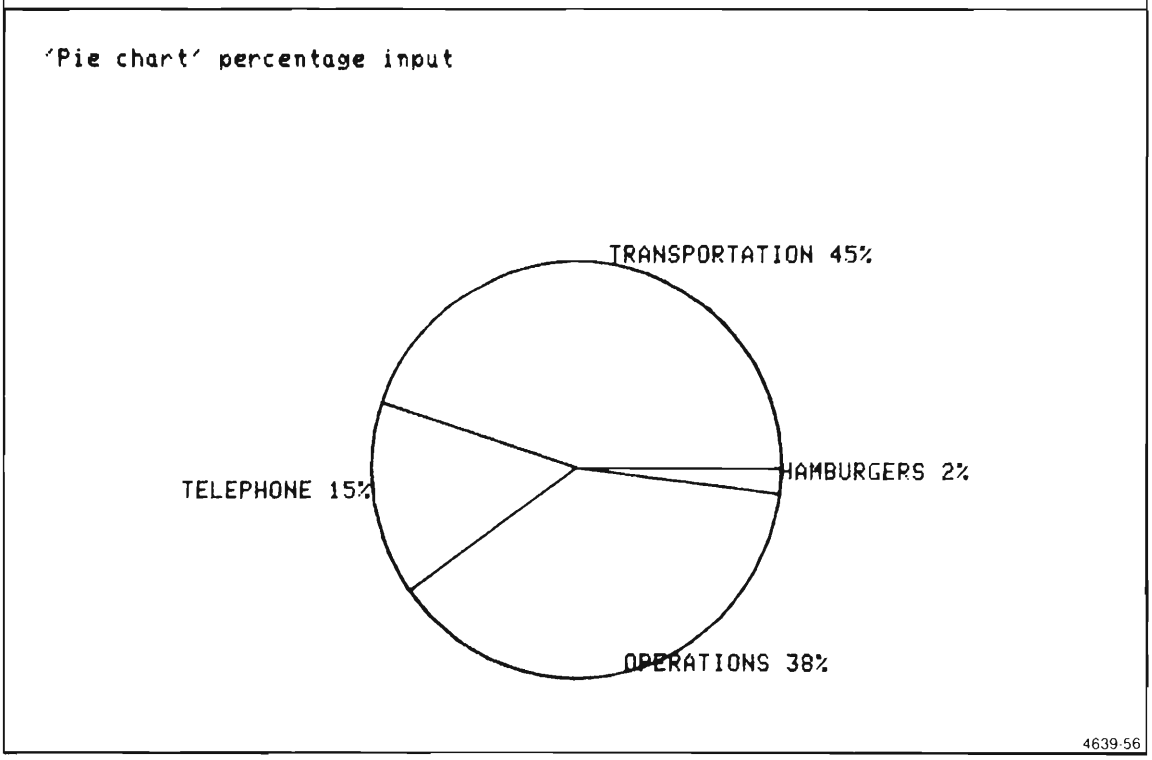


Figure 3-49. Example of RINPUT and RPRINT.

RMOVE (Relative Move)

This command moves the image I$ to the relative X,Y position.

Syntax Form

[line number] **CALL "RMOVE"**, image string argument & result, numeric simple/ constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "RMOVE"**, image, relative position X, relative position Y

Format Example

```
500 CALL "RMOVE",I$,X,Y
```

Explanation

This command adds X to all image horizontal coordinates, and Y to all image vertical coordinates. It alters the image string.

Sample Program

See Figure 3-50.

```
100 REM RMOVE example
110 INIT
120 REM Generate and draw the image for an A.
130 DIM F(10),F$(15)
140 DATA -50,-50,53,60,56,50,-51,-53.3,55,53.3
150 READ F
160 CALL "CHANGE",F,F$
170 CALL "RDRAW",F$,1,0,0
180 REM Repeatedly move the image to the right and draw it.
190 FOR I=1 TO 4
200 CALL "RMOVE",F$,10,0
210 CALL "RDRAW",F$,5,0,0
220 NEXT I
230 END
```

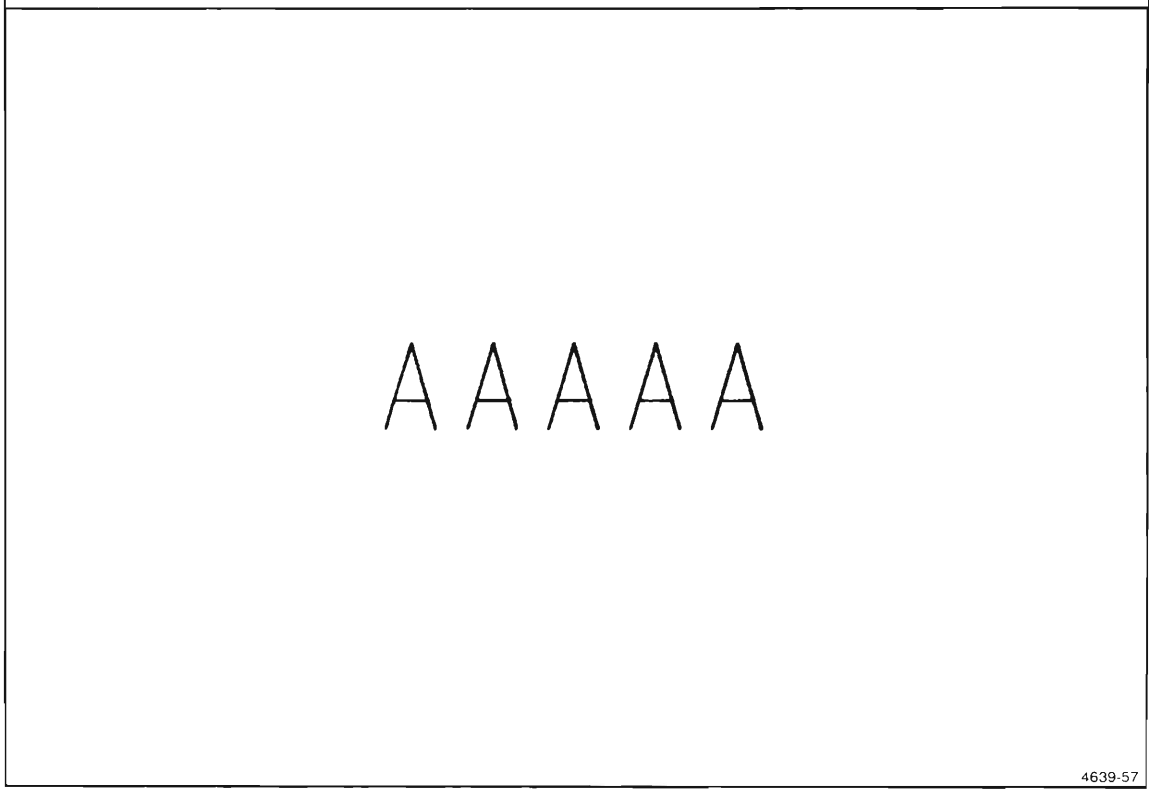


Figure 3-50. Example of RMOVE.

RPOINT (Relative point)

This command finds the image I$ point N and X,Y nearest the absolute X,Y position.

Syntax Form

[line number] **CALL "RPOINT"**, image string argument, numeric simple argument & result, numeric simple argument & result, numeric simple argument & result

Descriptive Form

[line number] **CALL "RPOINT"**, image, vector number where search starts, position X nearest relative position X, position Y nearest relative position Y

Format Example

```
500 CALL "RPOINT",I$,N,X,Y
```

Explanation

The POINT command finds the image I$ point N and the X,Y position that is nearest to absolute X,Y. The variables are received, redefined, and returned.

At CALL entry:

- I$: Image.
- N: Point number at which to start search.
- X,Y: When these are added to the first image point coordinates, they form the search center.

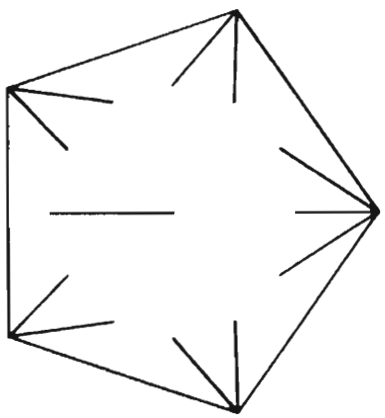
At CALL exit:

- I$: Image (not altered).
- N: Closest number (point or vector).
- X,Y: Coordinates of point closest to search center (point number N).
- The sign (N) indicates if the vector is a MOVE or a DRAW.

Sample Program

See Figure 3-51.

```
100 REM RPOINT example
110 INIT
120 SET DEGREES
130 REM Generate and draw a pentagon.
140 DIM F(14),F$(21)
150 DATA -50,-50,-75,-50,57.7,73.8,29.8,64.7,29.8,35.3,57.7,26.2,75,50
160 READ F
170 CALL "CHANGE",F,F$
180 CALL "RDRAW",F$,1,0,0
190 REM Generate points of a circle inside the pentagon.
200 FOR I=0 TO 360 STEP 30
210 X=15*COS(I)
220 Y=15*SIN(I)
230 MOVE 50+X,50+Y
240 REM Find the pentagon point that is closest and draw to it.
250 CALL "RPOINT",F$,1,X,Y
260 DRAW X,Y
270 NEXT I
280 END
```



4639-58

Figure 3-51. Example of RPOINT.

RPRINT (Relative Print)

This command prints the characters C$ C times at X,Y relative to center screen.

Syntax Form

[line number] **CALL ''RPRINT''**, text string/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL ''RPRINT''**, printable characters, display count, position X relative to center screen, position Y relative to center screen

Format Example

```
500 CALL ''RPRINT'',C$,C,X,Y
```

Explanation

Since the print string contains no absolute coordinates, it is made relative to center screen. It is a relative comand. For a sample program, refer to the combined example at Figure 3-49 under the RINPUT description.

RROTATE (Relative Rotate)

This command rotates the image I$ by angle A around the relative X,Y position.

Syntax Form

[line number] **CALL "RROTATE"**, image string argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "RROTATE"**, image to be rotated, angle of rotation, relative position X, relative position Y

Format Example

```
500 CALL "RROTATE",I$,A,X,Y
```

Explanation

All angles are measured counterclockwise and in current trigonometric units.

NOTE

The radians specified in ROTATE and SHEAR command are limited to:

$-2PI*65.536 \leqslant Angle \leqslant 2PI*65.536$ (radians); that is, $= \pm 131.072*PI = 411.774$ (radians).

Or, by degrees: $-2PI*65.536\ (180/PI) = -360*65.536 \leqslant Angle \leqslant 360*65.536 = 23592.96°$.

If the image's first point equals Xi,Yi, the image will be rotated by angle A around point Xi + X,Yi + Y.

Sample Program

See Figure 3-52.

```
100 REM RROTATE example
110 INIT
120 SET DEGREES
130 REM Here's an A.
140 DIM F(10),F$(15),G$(15)
150 DATA -50,-50,53,60,56,50,-51,-53.3,55,53.3
160 READ F
170 CALL "CHANGE",F,F$
180 G$=F$
190 FOR A=0 TO 300 STEP 60
200 F$=G$
210 REM Rotate the A with center 50+15,50+0.
220 CALL "RROTATE",F$,A,15,0
230 CALL "RDRAW",F$,1,0,0
240 NEXT A
250 END
```

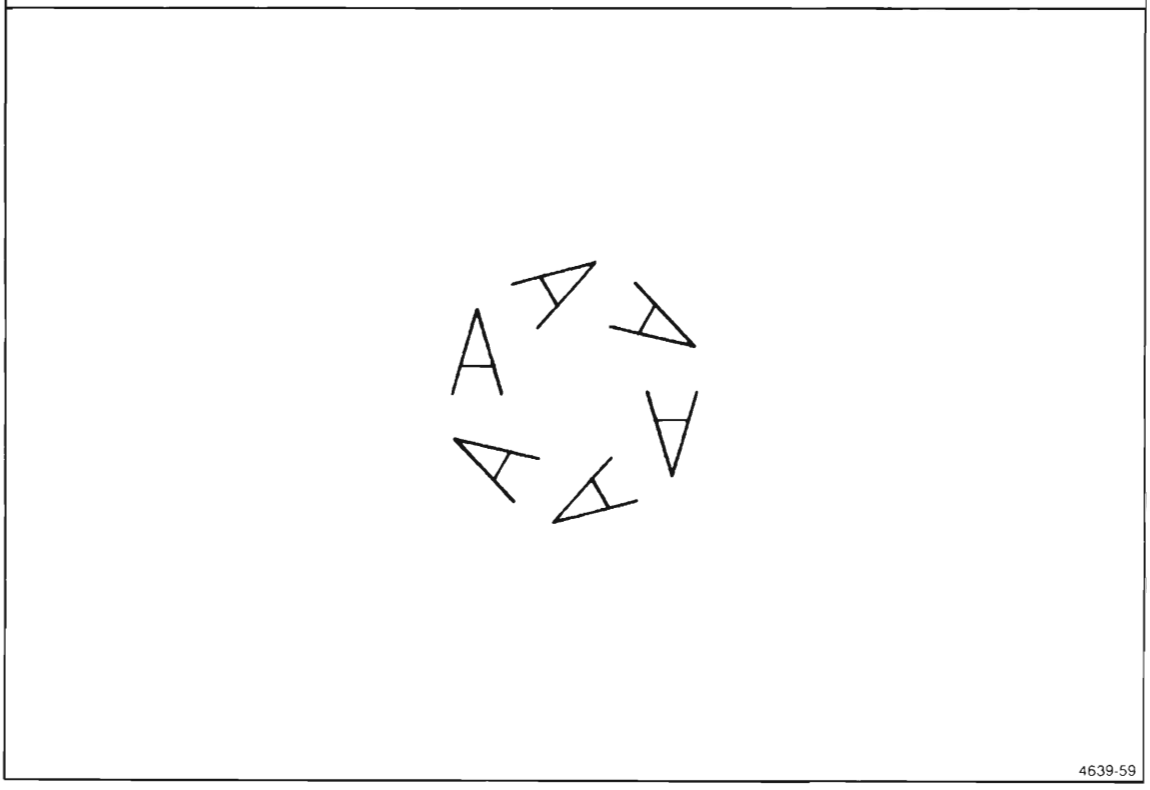


Figure 3-52. Example of RROTATE.

RSCALE (Relative Scale)

This command scales the image I$ by factors H,V around the relative X,Y position.

Syntax Form

[line number] **CALL "RSCALE"**, string argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "RSCALE"**, image to be scaled, horizontal factor, vertical factor, relative position X, relative position Y

Format Example

```
500 CALL "RSCALE",I$,H,V,X,Y
```

Explanation

The scale center is Xi + X,Yi + Y.

Negative values for H and/or V result in a mirror image. –65.535 ⩽ scale factor range ⩽ + 65.535 (17-bit resolution).

NOTE

SCALE and TAPER command operate over a range of -65.535 to + 65.535. This gives a resolution of 1 in 2∧10 (about 0.001). The input of a number less than -65.535 or greater than 65.535 has the same effect as an input of -65.535 or 65.535 respectively. The full range is rarely used. The screen usually limits the image before reaching the range extremities.

Sample Program

See Figure 3-53.

```
100 REM RSCALE example
110 INIT
120 REM Generate and draw an A.
130 DIM F(10),F$(15)
140 DATA -50,-50,53,60,56,50,-51,-53.3,55,53.3
150 READ F
160 CALL "CHANGE",F,F$
170 CALL "RDRAW",F$,1,0,0
180 REM Mark the center of the scaling transform.
190 MOVE 50-15,50+10
200 RDRAW 0,0
210 REM Now scale and draw the image.
220 CALL "RSCALE",F$,1.5,2,-15,10
230 CALL "RDRAW",F$,1,0,0
240 END
```

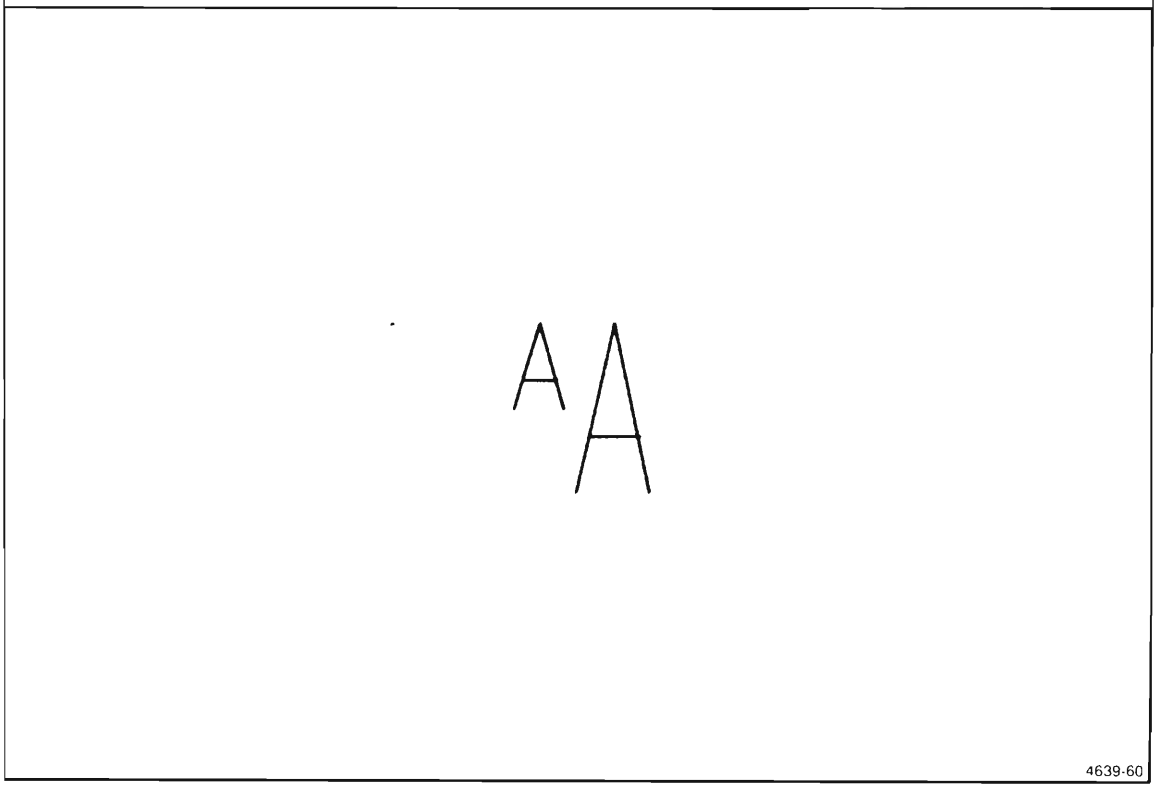


Figure 3-53. Example of RSCALE.

RSHEAR (Relative Shear)

This command shears the image I$ by angles H,V around the relative X,Y position.

Syntax Form

[line number] **CALL "RSHEAR"**, string argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "RSHEAR"**, image to be sheared, horizontal angle, vertical angle, relative position X, relative position Y

Format Example

```
500 CALL "RSHEAR",I$,H,V,X,Y
```

Explanation

The shear center is Xi + X,Yi + Y.

All angles are measured counterclockwise and in current trigonometric units.

NOTE

The radians specified in SHEAR and ROTATE command are limited to:

$-2PI*65.536 \leq Angle \leq 2PI*65.536$ (radians); that is, $= \pm 131.072*PI = 411.774$ (radians).

Or, by degrees: $-2PI*65.536\ (180/PI) = -360*65.536 \leq Angle \leq 360*65.536 = 23592.96°$.

Sample Program

See Figure 3-54.

```
100 REM RSHEAR example
110 INIT
120 SET DEGREES
130 REM Generate and draw the image of a grid.
140 DIM F(18),F$(27),G$(27)
150 DATA -50,-50,50,60,60,60,60,50,50,50,-55,-50,55,60,-50,-55,60,55
160 READ F
170 CALL "CHANGE",F,F$
180 CALL "RDRAW",F$,20,0,0
190 G$=F$
200 FOR D=-360 TO 360 STEP 15
210 F$=G$
220 REM Center of shear is 50+D/20,50-D/20.
230 CALL "RSHEAR",F$,-D,D,D/20,-D/20
240 CALL "RDRAW",F$,1,0,0
250 NEXT D
260 END
```

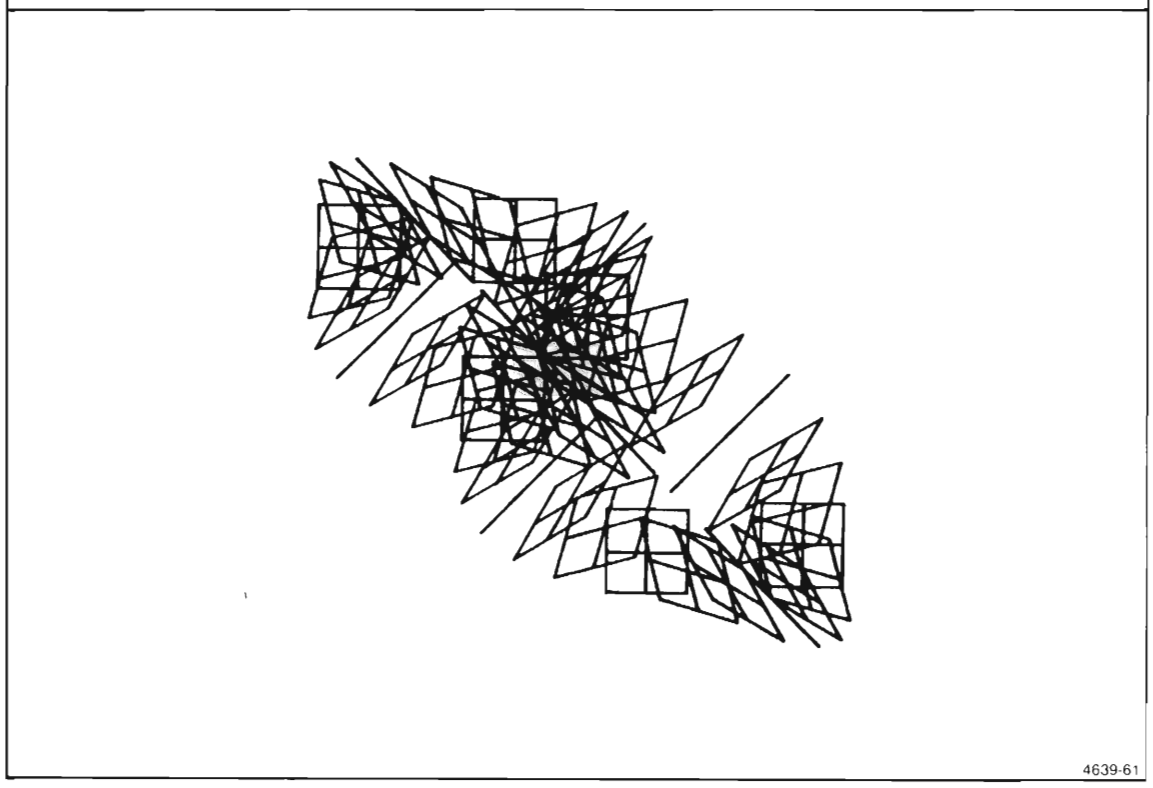


Figure 3-54. Example of RSHEAR.

RTAPER (Relative Taper)

This command tapers the image I$ by factors H,V around the relative X,Y position.

Syntax Form

[line number] **CALL "RTAPER"**, string argument & result, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument, numeric simple/constant argument

Descriptive Form

[line number] **CALL "RTAPER"**, image to be tapered, horizontal factor, vertical factor, relative X position, relative Y position

Format Example

```
500 CALL "RTAPER",I$,H,V,X,Y
```

Explanation

The taper center is Xi + X,Yi + Y.

Negative values for H and/or V result in a mirror image. $-65.535 \leqslant$ scale factor range $\leqslant +65.535$ (17-bit resolution).

NOTE

TAPER and SCALE commands operate over a range of -65.535 to + 65.535. This gives a resolution of 1 in 2∧10 (about 0.001). The input of a number less than -65.535 or greater than 65.535 has the same effect as an input of -65.535 or 65.535 respectively. The full range is rarely used. The screen usually limits the image before reaching the range extremities.

Sample Program

See Figure 3-55.

```
100 REM RTAPER example
110 INIT
120 PAGE
130 REM Generate and draw a box.
140 DIM B(24),B$(36),C$(36)
150 DATA -50,-50,50,60,60,60,60,50,50,50,-50,-60,58.66,65,68.66,65
160 DATA 68.66,55,60,50,-60,-60,68.66,65
170 READ B
180 CALL "CHANGE",B,B$
190 CALL "RDRAW",B$,1,0,0
200 REM Mark the center of taper.
210 MOVE 50-40,50-5
220 RDRAW 0,0
230 C$=B$
240 REM Repeatedly enter taper factors, drawing the tapered objects
250 REM   at the bottom of the screen.
260 X=0
270 B$=C$
280 HOME
290 PRINT "Horizontal and Vertical Taper factors (e.g. .5,.1): ";
300 INPUT H,V
310 CALL "RTAPER",B$,H,V,-40,-5
320 CALL "ADRAW",B$,1,X,20
330 MOVE X,15
340 PRINT H;V;
350 X=X+20
360 IF X>100 THEN 380
370 GO TO 270
380 END
```

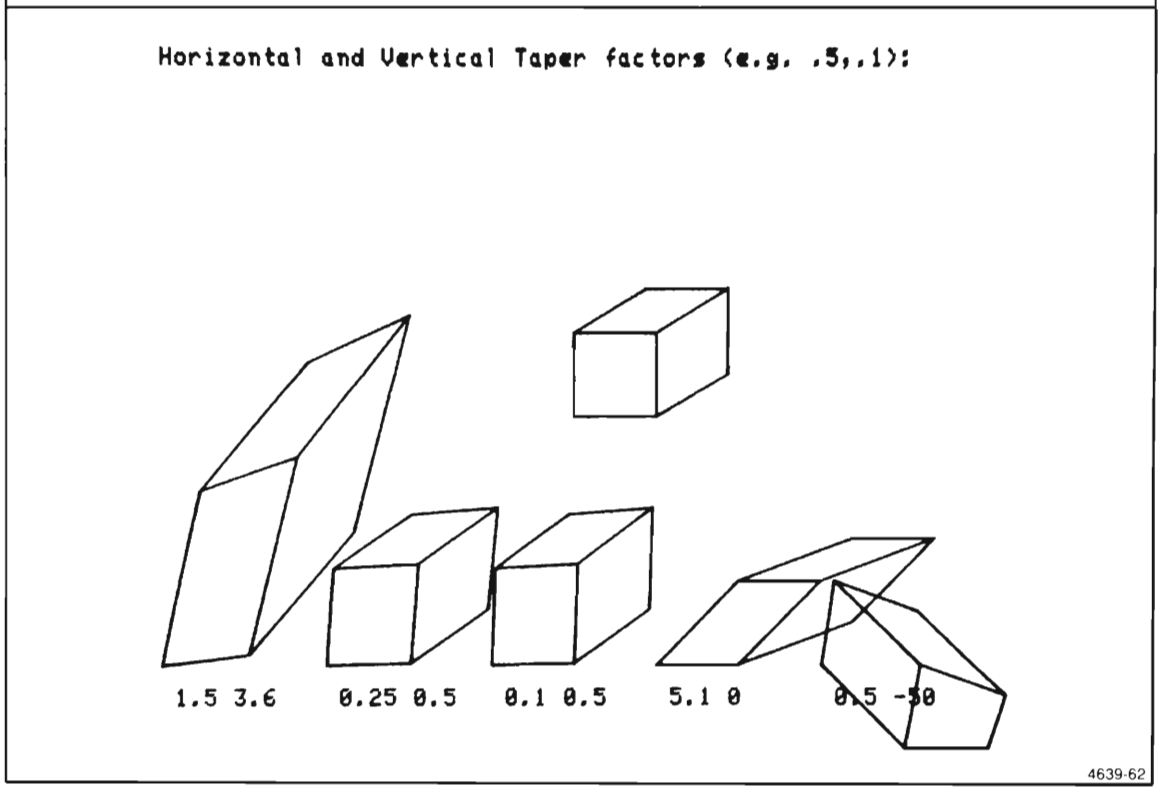


Figure 3-55. Example of RTAPER.

RUBBER (Rubber Band Line)

This command draws a "rubberband line" from the joystick cursor to the graphic cursor.

Syntax Form

[line number] **CALL "RUBBER"**, numeric simple argument & result (or constant), numeric simple result, numeric simple result, text string result

Descriptive Form

[line number] **CALL "RUBBER"**, display count, joystick horizontal position, joystick vertical position, any pending keyboard keys (K$)

Format Example

```
500 CALL "RUBBER",C,X,Y,K$
```

Explanation

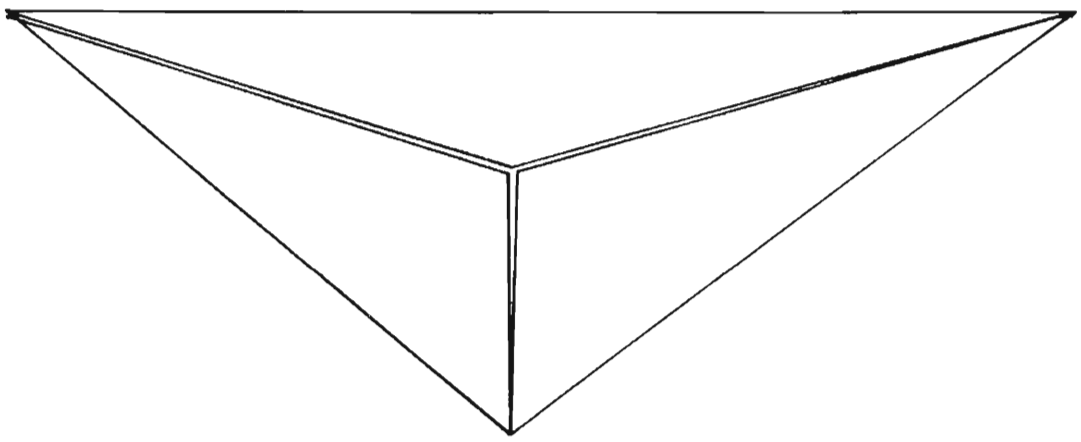
The RUBBER command is like JGIN, except RUBBER uses the display count instead of the time parameter. Len(K$) can be greater than 1, unlike POINTER and JCROSS.

Sample Program

See Figure 3-56.

```
100 REM RUBBER example
110 PRINT "4952 Joystick required--"
120 PRINT "END:  Type E or e to quit."
130 PRINT "MOVE: Type M or m to move."
140 PRINT "DRAW: Type anything else for a drawn vector."
150 INIT
160 REM Let there exist a potentially large image.
170 DIM F(800),F$(1200)
180 I=1
190 REM Get a vector.
200 CALL "RUBBER",-1000,F(I),F(I+1),K$
210 IF K$="e" THEN 310
220 IF K$="m" THEN 250
230 DRAW F(I),F(I+1)
240 GO TO 290
250 REM M was typed.
260 MOVE F(I),F(I+1)
270 F(I)=-F(I)
280 F(I+1)=-F(I+1)
290 I=I+2
300 GO TO 190
310 REM E was typed.
320 REM Re-dimension to the computed size of F, create and draw image.
330 DIM F(I-1)
340 CALL "CHANGE",F,F$
350 CALL "RDRAW",F$,5,0,0
360 END
```

```
4952 Joystick required--
END:  Type E or e to quit.
MOVE: Type M or m to move.
DRAW: Type anything else for a drawn vector.
```



4639-63

Figure 3-56. Example of RUBBER.

SOUNDS

This command turns the internal 4050 Series speaker on and off at rates dependent on the bit-weighted value of the characters in the provided string.

Syntax Form

[line number] **CALL "SOUNDS"**, text string/constant argument

Descriptive Form

[line number] **CALL "SOUNDS"**, character string

Format Example

```
500 CALL "SOUNDS",S$
```

Explanation

Lower bit-weighted characters result in higher frequencies, as shown in Table 3-7.

Table 3-7

FREQUENCY RANGE

Description	Frequence
CHR(0)[ASCII NUL]	~ 13 KHz tone.
CHR(127)[ASCII RUBOUT]	~ 100 Hz tone.
CHR(1) through CHR(126)	See ASCII Code Chart for intermediate values and characters. See formula.

$$\text{Frequency} = \frac{1}{(N+1)(7.68E\text{-}5)} \text{ Hz} \quad \text{where N is N in CHR(N)}$$

Each character in the provided string determines one half-cycle at its frequency; that is, the ASCII character value determines the sounds half period time.

This command is useful for making attention-getting sounds (non-musical) as audible cues of certain functions in progress or just completed. SOUNDS can simulate voice patterns to a marked degree.

Push the HOME/PAGE key after executing SOUND if you wish to home the cursor.

A single BREAK causes an immediate exit. This allows you to break out of a long sound sequence without the effects of BREAK BREAK.

Sample Program

```
LIS
1 REM  SOUNDS example
2 GO TO 100
4 REM Push function key 1 to repeat without recomputing sounds string.
5 GO TO 250
100 INIT
110 SET KEY
120 REM Generate a magic string.
130 DIM L$(2500)
140 L$=" "
150 FOR J=1 TO 5
160 FOR I=0 TO 2500 STEP 127
170 C$=CHR(SIN(I)*63+64)
180 FOR K=1 TO J
190 C$=C$&C$
200 NEXT K
210 L$=L$&C$
220 NEXT I
230 NEXT J
240 REM How do it.
250 CALL "SOUNDS",L$
260 GO TO 250
270 END
```

STRING

Function #1

This command changes the floating point array F into ASCII string S$. The ASCII string will be valid ASCII characters (ASCII 0 . . . 127 on the 4051; 0. . . 255 on the 4052 and 4054). The description of the reverse procedure follows this description, under the caption of Function #2.)

Syntax Form

[line number] **CALL "STRING"**, numeric array argument, text string result

Descriptive Form

[line number] **CALL "STRING"**, floating point array, ASCII character string

Format Example

```
500 CALL "STRING",F,S$
```

Explanation

This command can auto-dimension the string result, if necessary (72 characters). The result string S$ must be at least big enough. For example:

```
100 DIM F(100),S$(100)
                (or bigger)
```

Sample Program

See Figure 3-57.

```
100 REM STRING example
110 INIT
120 PAGE
130 SET DEGREES
140 DIM F(14),F$(21),A(21),G$(21),G(14)
150 DATA -50,-50,-75,-50,57.7,73.8,29.8,64.7,29.8,35.3,57.7,26.2,75,50
160 READ F
170 PRINT "Point X,Y values: ",F
180 CALL "CHANGE",F,F$
190 PRINT "CHANGE image string:  ";
200 CALL "PRINTS",F$
210 PRINT
220 PRINT
230 CALL "STRING",F$,A
240 PRINT "STRING ASCII values:";
250 PRINT A
260 CALL "STRING",A,G$
270 PRINT "STRING string:  ";
280 CALL "PRINTS",G$
290 PRINT
300 PRINT
310 CALL "CHANGE",G$,G
320 PRINT "CHANGE point X,Y values: ",G
330 END
```

```
Point X,Y values:

 -50              -50              -75              -50
 57.7             73.8             29.8             64.7
 29.8             35.3             57.7             26.2
 75               50

CHANGE image string:  ÉEEcJE≥C@KiyJiIYCL#JE

STRING ASCII values:

 91               6                6                99
 74               6                28               67
 64               11               105              121
 10               105              20               25
 67               76               35               74
 6

STRING string:  ÉEEcJE≥C@KiyJiIYCL#JE

CHANGE point X,Y values:

 -49.92           -49.92           -75.008          -49.92
 57.728           73.728           29.824           64.64
 29.824           35.328           57.728           26.112
 75.008           49.92
```

4639-64

Figure 3-57. Example of STRING (Function #1).

Function #2

This command changes the ASCII string S$ into a floating point array F. The floating point array F will be valid ASCII character codes. (See the earlier Function #1 of STRING for converting the other way.)

Syntax Form

[line number] **CALL "STRING"**, text string argument, numeric array result

Descriptive Form

[line number] **CALL "STRING"**, ASCII character string, floating point array

Format Example

```
500 CALL "STRING",S$,F
```

Explanation

The array must be dimensioned to a correct length. For example:

```
100 DIM S$(N),F(N*2/3)
```

The floating point "N" must be equal to LEN(S$), no more, no less. DIM F(LEN(S$)).

Sample Program

See Figure 3-58.

```
100 REM STRING example
110 INIT
120 REM Create a sample string.
130 DIM A(6),A$(6)
140 A$="aA1!GJ"
150 REM Convert to ASCII.
160 CALL "STRING",A$,A
170 PRINT "STRING ASCII values:";
180 PRINT A
190 REM Convert back to character form.
200 CALL "STRING",A,A$
210 PRINT "STRING string:  ";
220 CALL "PRINTS",A$
230 PRINT
240 END
```

```
STRING ASCII values:

 97                65                49                33
 7                 10

STRING string:  aA1!GJ
```

4639-65

Figure 3-58. Example of STRING (Function #2).

TOGGLE (Toggle Move Flag)

This command toggles the image string I$ point N Move Flag. Sign(N) at exit indicates if the vector is a MOVE or a DRAW. A DRAW changes to a MOVE, and a MOVE changes to a DRAW.

Syntax Form

[line number] **CALL "TOGGLE"**, image string argument & result, numeric simple argument & result

Descriptive Form

[line number] **CALL "TOGGLE"**, image string, sign (at CALL exit) indicating if vector is MOVE or DRAW

Format Example

```
500 CALL "TOGGLE",I$,N
```

Explanation

This command alters the image string.

Sample Program

See Figure 3-59.

```
100 REM TOGGLE example
110 INIT
120 SET DEGREES
130 REM Create a pentagon.
140 DIM F(14),F$(21)
150 DATA -50,-50,-75,-50,57.7,73.8,29.8,64.7,29.8,35.3,57.7,26.2,75,50
160 READ F
170 CALL "CHANGE",F,F$
180 REM Blink each segment in turn.
190 FOR J=1 TO 7
200 I=J
210 CALL "RDRAW",F$,-40,0,0
220 REM Change from draw to move, or vice versa.
230 CALL "TOGGLE",F$,-I
240 CALL "RDRAW",F$,-40,0,0
250 REM Change it back the way it was.
260 CALL "TOGGLE",F$,I
270 NEXT J
280 CALL "RDRAW",F$,-40,0,0
290 END
```

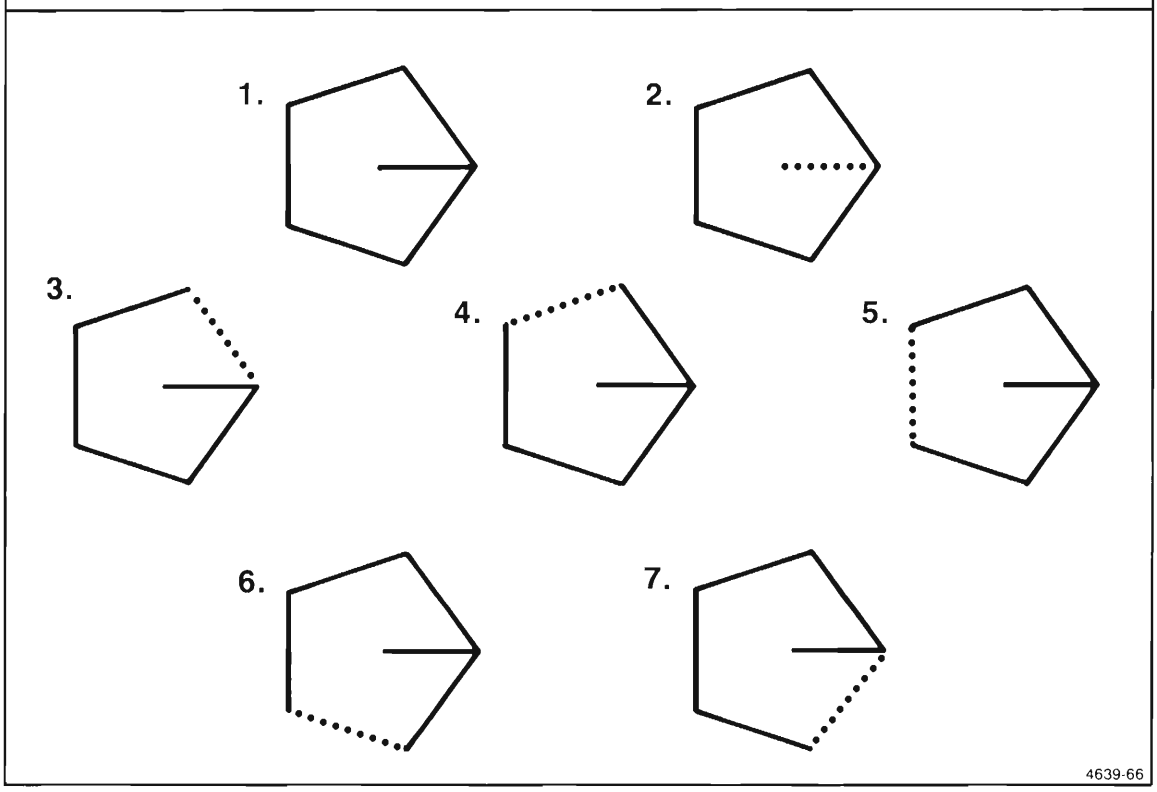


Figure 3-59. Example of TOGGLE.

VERTEX

This command draws "rubber band lines" from the joystick cursor to image I$ points.

Syntax Form

[line number] **CALL "VERTEX"**, image string argument, numeric simple argument & result, numeric simple result, numeric simple result, image string result

Descriptive Form

[line number] **CALL "VERTEX"**, image, display count, joystick horizontal position, joystick vertical position, any pending keyboard keys K$

Format Example

```
500 CALL "VERTEX",I$,C,X,Y,K$
```

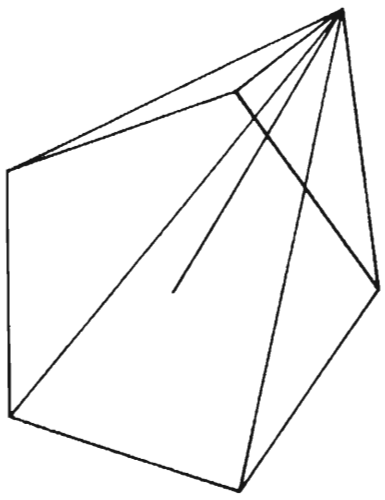
Explanation

The VERTEX command produces the following display when you enter the sample program.

Sample Program

See Figure 3-60.

```
100 REM VERTEX example
110 INIT
120 SET DEGREES
130 REM Generate a pentagon.
140 DIM F(14),F$(21)
150 DATA -50,-50,-75,-50,57.7,73.8,29.8,64.7,29.8,35.3,57.7,26.2,75,50
160 READ F
170 CALL "CHANGE",F,F$
180 CALL "RDRAW",F$,1,0,0
190 REM Refresh from vertices of the pentagon to the joystick location.
200 CALL "VERTEX",F$,-1000,X,Y,K$
210 REM When a key is depressed, store the lines to the vertices.
220 CALL "VERTEX",F$,1,X,Y,K$
230 END
```



4639-67

Figure 3-60. Example of VERTEX.

COMMAND CAPABILITIES

In Table 3-8, "yes" or "no" indicates which functions a command can do. "Maybe" generally means that the refresh count parameter determines whether Store or Refresh is activated, as follows:

$> +0.5$ = Store
< -0.5 = Refresh
$-0.5 <$ count $< +0.5$ = No drawing done

Table 3-8

COMMAND CAPABILITY MATRIX

Name	Refresh	Store	Image	Cursor	String	Key	#Input	#Output
BOUNDS	No	No	Yes	No	Yes	No	No	Yes
LOCATE	Maybe	Maybe	No	Yes	Yes	Yes	Yes	Yes
AGIN	No	No	No	No	No	No	No	Yes
RGIN	No	No	No	No	No	No	Yes	Yes
GGIN	No	No	No	No	No	No	Yes	Yes
JGIN	No	No	No	No	Yes	Yes	Yes	Yes
APOINT	No	No	Yes	No	Yes	No	Yes	Yes
RPOINT	No	No	Yes	No	Yes	No	Yes	Yes
GPOINT	No	No	Yes	No	Yes	No	Yes	Yes
JPOINT	Maybe	Maybe	Yes	Yes	Yes	Yes	Yes	Yes
DASHED	No	Yes	No	No	No	No	Yes	No
DOTTED	No	Yes	No	No	No	No	Yes	No
RUBBER	Maybe	Maybe	No	Yes	Yes	Yes	Yes	Yes
VERTEX	Maybe	Maybe	Yes	Yes	Yes	Yes	Yes	Yes
ACROSS	Maybe	Maybe	No	Yes	No	No	Yes	No
RCROSS	Maybe	Maybe	No	Yes	No	No	Yes	No
GCROSS	Maybe	Maybe	No	Yes	No	No	Yes	No
JCROSS	Maybe	Maybe	No	Yes	Yes	Yes	Yes	Yes
ADRAW	Maybe	Maybe	Yes	No	Yes	No	Yes	No
RDRAW	Maybe	Maybe	Yes	No	Yes	No	Yes	No
GDRAW	Maybe	Maybe	Yes	No	Yes	No	Yes	No
JDRAW	Maybe	Maybe	Yes	Yes	Yes	Yes	Yes	Yes
ADOTS	Maybe	Maybe	Yes	No	Yes	No	Yes	No
RDOTS	Maybe	Maybe	Yes	No	Yes	No	Yes	No
GDOTS	Maybe	Maybe	Yes	No	Yes	No	Yes	No
JDOTS	Maybe	Maybe	Yes	Yes	Yes	Yes	Yes	Yes
APRINT	Maybe	Maybe	No	No	Yes	No	Yes	No
RPRINT	Maybe	Maybe	No	No	Yes	No	Yes	No
GPRINT	Maybe	Maybe	No	No	Yes	No	Yes	No
JPRINT	Maybe	Maybe	No	Yes	Yes	Yes	Yes	Yes

Table 3-8 (cont)

COMMAND CAPABILITY MATRIX

Name	Refresh	Store	Image	Cursor	String	Key	#Input	#Output
AINPUT	Maybe	Maybe	No	No	Yes	Yes	Yes	No
RINPUT	Maybe	Maybe	No	No	Yes	Yes	Yes	No
GINPUT	Maybe	Maybe	No	No	Yes	Yes	Yes	No
JINPUT	Maybe	Maybe	No	Yes	Yes	Yes	Yes	Yes
IMAGES	No	No	Yes	No	Yes	No	No	No
CHANGE	No	No	Yes	No	Yes	No	Maybe	Maybe
DEFINE	No	No	Yes	No	Yes	No	Yes	Yes
POINTS	No	No	Yes	No	Yes	No	Yes	Yes
TOGGLE	No	No	Yes	No	Yes	No	Yes	Yes
AMOVE	No	No	Yes	No	Yes	No	Yes	No
RMOVE	No	No	Yes	No	Yes	No	Yes	No
GMOVE	No	No	Yes	No	Yes	No	Yes	No
JMOVE	Maybe	Maybe	Yes	Yes	Yes	Yes	Yes	Yes
ASCALE	No	No	Yes	No	Yes	No	Yes	No
RSCALE	No	No	Yes	No	Yes	No	Yes	No
GSCALE	No	No	Yes	No	Yes	No	Yes	No
JSCALE	Maybe	Maybe	Yes	Yes	Yes	Yes	Yes	Yes
ASHEAR	No	No	Yes	No	Yes	No	Yes	No
RSHEAR	No	No	Yes	No	Yes	No	Yes	No
GSHEAR	No	No	Yes	No	Yes	No	Yes	No
JSHEAR	Maybe	Maybe	Yes	Yes	Yes	Yes	Yes	Yes
ATAPER	No	No	Yes	No	Yes	No	Yes	No
RTAPER	No	No	Yes	No	Yes	No	Yes	No
GTAPER	No	No	Yes	No	Yes	No	Yes	No
JTAPER	Maybe	Maybe	Yes	Yes	Yes	Yes	Yes	Yes
AROTATE	No	No	Yes	No	Yes	No	Yes	No
RROTATE	No	No	Yes	No	Yes	No	Yes	No
GROTATE	No	No	Yes	No	Yes	No	Yes	No
JROTATE	Maybe	Maybe	Yes	Yes	Yes	Yes	Yes	Yes
PRINTS	No	Yes	No	No	Yes	No	No	No
INPUTS	No	Yes	No	No	Yes	No?	No	No
STRING	No	No	No	No	Yes	No	Maybe	Maybe
SOUNDS	No	No	No	No	Yes	No	No	No
MUSIC	No	No	No	No	Yes	No	No	No

Section 4
REPLACEABLE PARTS

PARTS ORDERING INFORMATION

Replacement parts are available from or through your local Tektronix, Inc. Field Office or representative.

Changes to Tektronix instruments are sometimes made to accommodate improved components as they become available, and to give you the benefit of the latest circuit improvements developed in our engineering department. It is therefore important, when ordering parts, to include the following information in your order Part number, instrument type or number serial number, and modification number if applicable.

If a part you have ordered has been replaced with a new or improved part, your local Tektronix Inc Field Office or representative will contact you concerning any change in part number

Change information if any, is located at the rear of this manual

SPECIAL NOTES AND SYMBOLS

X000	Part first added at this serial number
00X	Part removed after this serial number

FIGURE AND INDEX NUMBERS

Items in this section are referenced by figure and index numbers to the illustrations

INDENTATION SYSTEM

This mechanical parts list is indented to indicate item relationships. Following is an example of the indentation system used in the description column

1 2 3 4 5 *Name & Description*

Assembly and/or Component
Attaching parts for Assembly and/or Component
\- - - * - - -
Detail Part of Assembly and/or Component
Attaching parts for Detail Part
\- - - * - - -
Parts of Detail Part
Attaching parts for Parts of Detail Part
\- - - * - - -

Attaching Parts always appear in the same indentation as the item it mounts, while the detail parts are indented to the right. Indented items are part of, and included with, the next higher indentation The separation symbol - - - * - - - indicates the end of attaching parts.

Attaching parts must be purchased separately, unless otherwise specified.

ITEM NAME

In the Parts List, an Item Name is separated from the description by a colon (:) Because of space limitations, an Item Name may sometimes appear as incomplete. For further Item Name identification, the U.S. Federal Cataloging Handbook H6-1 can be utilized where possible.

ABBREVIATIONS

	INCH
#	NUMBER SIZE
ACTR	ACTUATOR
ADPTR	ADAPTER
ALIGN	ALIGNMENT
AL	ALUMINUM
ASSEM	ASSEMBLED
ASSY	ASSEMBLY
ATTEN	ATTENUATOR
AWG	AMERICAN WIRE GAGE
BD	BOARD
BRKT	BRACKET
BRS	BRASS
BRZ	BRONZE
BSHG	BUSHING
CAB	CABINET
CAP	CAPACITOR
CER	CERAMIC
CHAS	CHASSIS
CKT	CIRCUIT
COMP	COMPOSITION
CONN	CONNECTOR
COV	COVER
CPLG	COUPLING
CRT	CATHODE RAY TUBE
DEG	DEGREE
DWR	DRAWER
ELCTRN	ELECTRON
ELEC	ELECTRICAL
ELCTLT	ELECTROLYTIC
ELEM	ELEMENT
EPL	ELECTRICAL PARTS LIST
EQPT	EQUIPMENT
EXT	EXTERNAL
FIL	FILLISTER HEAD
FLEX	FLEXIBLE
FLH	FLAT HEAD
FLTR	FILTER
FR	FRAME or FRONT
FSTNR	FASTENER
FT	FOOT
FXD	FIXED
GSKT	GASKET
HDL	HANDLE
HEX	HEXAGON
HEX HD	HEXAGONAL HEAD
HEX SOC	HEXAGONAL SOCKET
HLCPS	HELICAL COMPRESSION
HLEXT	HELICAL EXTENSION
HV	HIGH VOLTAGE
IC	INTEGRATED CIRCUIT
ID	INSIDE DIAMETER
IDENT	IDENTIFICATION
IMPLR	IMPELLER
IN	INCH
INCAND	INCANDESCENT
INSUL	INSULATOR
INTL	INTERNAL
LPHLDR	LAMPHOLDER
MACH	MACHINE
MECH	MECHANICAL
MTG	MOUNTING
NIP	NIPPLE
NON WIRE	NOT WIRE WOUND
OBD	ORDER BY DESCRIPTION
OD	OUTSIDE DIAMETER
OVH	OVAL HEAD
PH BRZ	PHOSPHOR BRONZE
PL	PLAIN or PLATE
PLSTC	PLASTIC
PN	PART NUMBER
PNH	PAN HEAD
PWR	POWER
RCPT	RECEPTACLE
RES	RESISTOR
RGD	RIGID
RLF	RELIEF
RTNR	RETAINER
SCH	SOCKET HEAD
SCOPE	OSCILLOSCOPE
SCR	SCREW
SE	SINGLE END
SECT	SECTION
SEMICOND	SEMICONDUCTOR
SHLD	SHIELD
SHLDR	SHOULDERED
SKT	SOCKET
SL	SLIDE
SLFLKG	SELF-LOCKING
SLVG	SLEEVING
SPR	SPRING
SQ	SQUARE
SST	STAINLESS STEEL
STL	STEEL
SW	SWITCH
T	TUBE
TERM	TERMINAL
THD	THREAD
THK	THICK
TNSN	TENSION
TPG	TAPPING
TRH	TRUSS HEAD
V	VOLTAGE
VAR	VARIABLE
W/	WITH
WSHR	WASHER
XFMR	TRANSFORMER
XSTR	TRANSISTOR

CROSS INDEX—MFR. CODE NUMBER TO MANUFACTURER

Mfr Code	Manufacturer	Address	City, State, Zip
01295	TEXAS INSTRUMENTS, INC., SEMICONDUCTOR GROUP	P O BOX 5012, 13500 N CENTRAL EXPRESSWAY	DALLAS, TX 75222
04222	AVX CERAMICS, DIVISION OF AVX CORP.	P O BOX 867, 19TH AVE. SOUTH	MYRTLE BEACH, SC 29577
27014	NATIONAL SEMICONDUCTOR CORP.	2900 SEMICONDUCTOR DR.	SANTA CLARA, CA 95051
55680	NICHICON/AMERICA/CORP.	6435 N PROESEL AVENUE	CHICAGO, IL 60645
80009	TEKTRONIX, INC.	P O BOX 500	BEAVERTON, OR 97077

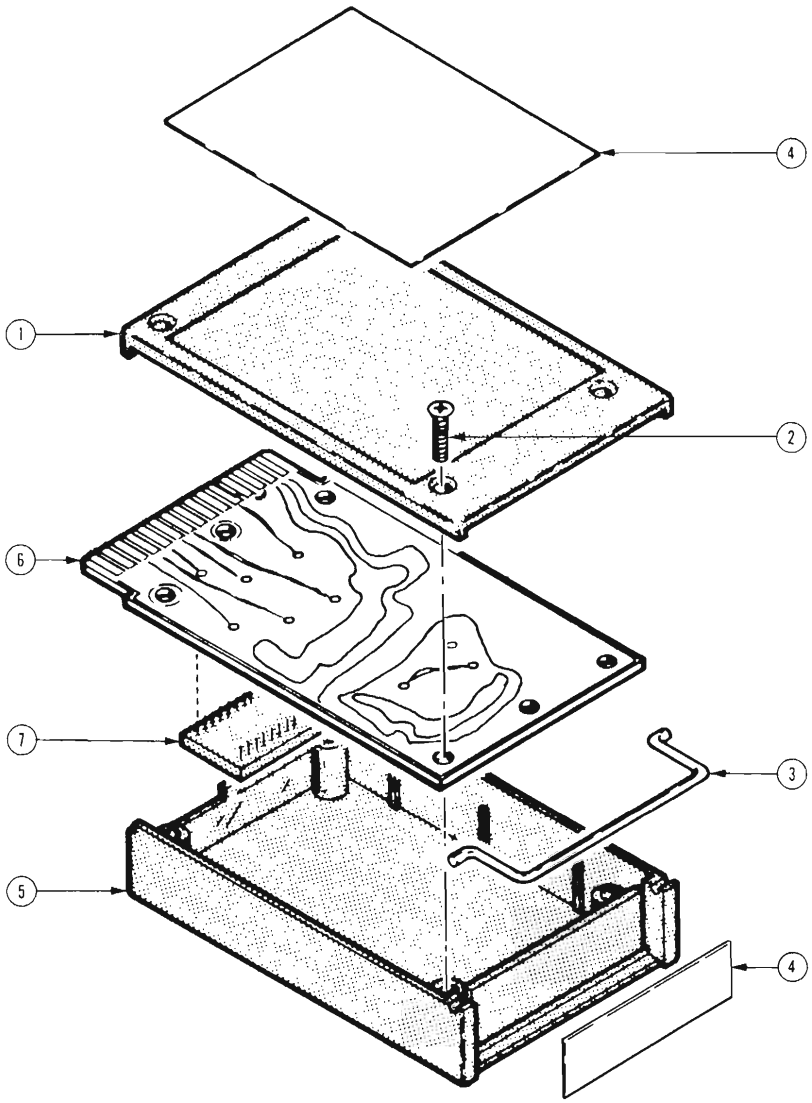
Component No.	Tektronix Part No.	Serial/Model No. Eff	Dscont	Name & Description	Mfr Code	Mfr Part Number
				4051R12		
A1	670-8200-00			CKT BOARD ASSY:GRAPHICS ENHANCEMENT	80009	670-8200-00
A1C115	283-0422-00			CAP.,FXD,CER DI:0.047UF,+80-20%,50V	04222	DG015E473Z
A1C205	283-0422-00			CAP.,FXD,CER DI:0.047UF,+80-20%,50V	04222	DG015E473Z
A1C305	283-0422-00			CAP.,FXD,CER DI:0.047UF,+80-20%,50V	04222	DG015E473Z
A1C401	290-0804-00			CAP.,FXD,ELCTLT:10UF,+50-10%,25V	55680	25ULA10V-T
A1U105	156-0916-02			MICROCIRCUIT,DI:8-2 INP 3-STATE BFR,BURN	27014	DM81LS97
A1U115	160-2041-00			MICROCIRCUIT,DI:4096 X 8 EPROM	80009	160-2041-00
A1U205	156-0916-02			MICROCIRCUIT,DI:8-2 INP 3-STATE BFR,BURN	27014	DM81LS97
A1U305	156-0480-02			MICROCIRCUIT,DI:QUAD 2 INP & GATE	01295	SN74LS08NP3
A1U315	160-2042-00			MICROCIRCUIT,DI:4096 X 8 EPROM	80009	160-2042-00
A1U405	156-0469-02			MICROCIRCUIT,DI:3/8 LINE DCDR	01295	SN74LS138NP3
				4052R12		
A2	670-8201-00			CKT BOARD ASSY:GRAPHICS ENHANCEMNT	80009	670-8201-00
A2C1	283-0422-00			CAP.,FXD,CER DI:0.047UF,+80-20%,50V	04222	DG015E473Z
A2C13	283-0422-00			CAP.,FXD,CER DI:0.047UF,+80-20%,50V	04222	DG015E473Z
A2C101	290-0804-00			CAP.,FXD,ELCTLT:10UF,+50-10%,25V	55680	25ULA10V-T
A2C111	283-0422-00			CAP.,FXD,CER DI:0.047UF,+80-20%,50V	04222	DG015E473Z
A2U1	160-2039-00			MICROCIRCUIT,DI:4096 X 8 EPROM	80009	160-2039-00
A2U11	160-2040-00			MICROCIRCUIT,DI:4096 X 8 EPROM	80009	160-2040-00
A2U111	156-0469-02			MICROCIRCUIT,DI:3/8 LINE DCDR	01295	SN74LS138NP3
A2U113	156-0480-02			MICROCIRCUIT,DI:QUAD 2 INP & GATE	01295	SN74LS08NP3

CROSS INDEX—MFR. CODE NUMBER TO MANUFACTURER

Mfr Code	Manufacturer	Address	City, State, Zip
09922	BURNDY CORPORATION	RICHARDS AVENUE	NORWALK, CT 06852
80009	TEKTRONIX, INC.	P O BOX 500	BEAVERTON, OR 97077
83385	CENTRAL SCREW CO.	2530 CRESCENT DR.	BROADVIEW, IL 60153
91260	CONNOR SPRING AND MFG. CO.	1729 JUNCTION AVE.	SAN JOSE, CA 95112

Fig & Index No	Tektronix Part No	Serial/Model No. Eff	Dscont	Qty	1 2 3 4 5	Name & Description	Mfr Code	Mfr Part Number
1-1	380-0384-01			1		HSG HALF,ROM PK:LID,ABS	80009	380-0384-01
						(ATTACHING PARTS)		
-2	211-0102-00			4		SCREW,MACHINE:4-40 X 0.500",FLH,STL	83385	OBD
						- - - * - - -		
-3	367-0189-00			1		HANDLE,BOW:2.62 L,SST	91260	OBD
-4	----- -----			1		MKR SET,IDENT:		
-5	380-0343-01			1		HSG HALF,PTR:INNER,ABS	80009	380-0343-01
-6	----- -----			1		CKT BOARD ASSY:(SEE A1 OR A2 REPL)		
-7	136-0751-00			2		. SKT,PL-IN ELEK:MICROCKT,24 PIN	09922	DILB24P108
						STANDARD ACCESSORIES		
	070-4639-00			1		MANUAL,TECH:INSTRUCTION	80009	070-4639-00

FIG. 1 EXPLODED VIEW



Section 5
DIAGRAMS AND SCHEMATICS

Symbols and Reference Designators

Electrical components shown on the diagrams are in the following units unless noted otherwise:

Capacitors = Values one or greater are in picofarads (pF).
Values less than one are in microfarads (μF).

Resistors = Ohms (Ω).

Graphic symbols and class designation letters are based on ANSI Standard Y32.2-1975.

Logic symbology is based on ANSI Y32.14-1973 in terms of positive logic. Logic symbols depict the logic function performed and may differ from the manufacturer's data.

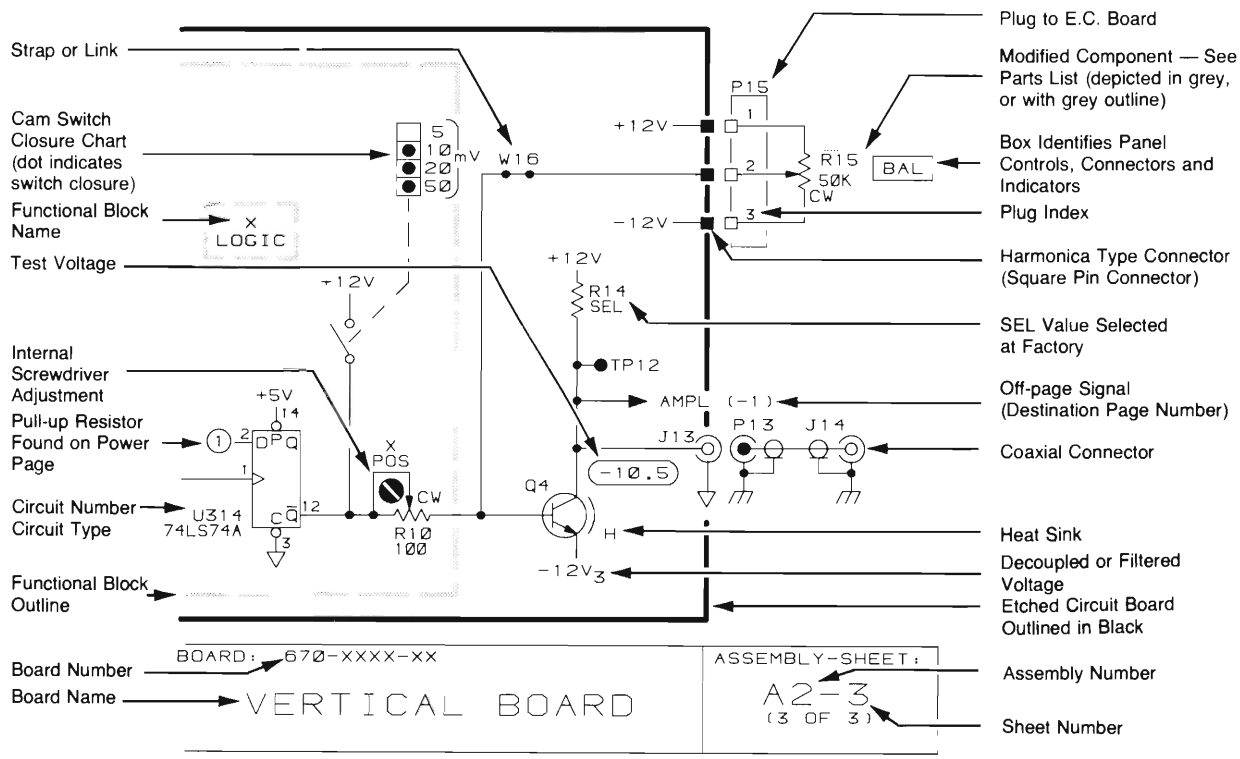
Abbreviations are based on ANSI Y1.1-1972. Other ANSI standards that are used in the preparation of diagrams by Tektronix, Inc., are:

Y14.15, 1966	Drafting Practices.
Y14.2, 1973	Line Conventions and Lettering.
Y10.5, 1968	Letter Symbols for Quantities Used in Electrical Science and Electrical Engineering.

The following prefix letters are used as reference designators to identify components or assemblies on the diagrams.

A	Assembly, separable or repairable (circuit board, etc.)	H	Heat dissipating device (heat sink, heat radiator, etc.)	S	Switch or contactor
AT	Attenuator, fixed or variable	HR	Heater	T	Transformer
B	Motor	HY	Hybrid circuit	TC	Thermocouple
BT	Battery	J	Connector, stationary portion	TP	Test point
C	Capacitor, fixed or variable	K	Relay	U	Assembly, inseparable or non-repairable (integrated circuit, etc.)
CB	Circuit breaker	L	Inductor, fixed or variable	V	Electron tube
CR	Diode, signal or rectifier	M	Meter	VR	Voltage regulator (zener diode, etc.)
DL	Delay line	P	Connector, movable portion	W	Wirestrap or cable
DS	Indicating device (lamp)	Q	Transistor or silicon-controlled rectifier	Y	Crystal
E	Spark Gap, Ferrite bead	R	Resistor, fixed or variable	Z	Phase shifter
F	Fuse	RT	Thermistor		
FL	Filter				

The following special symbols may appear on the diagrams:



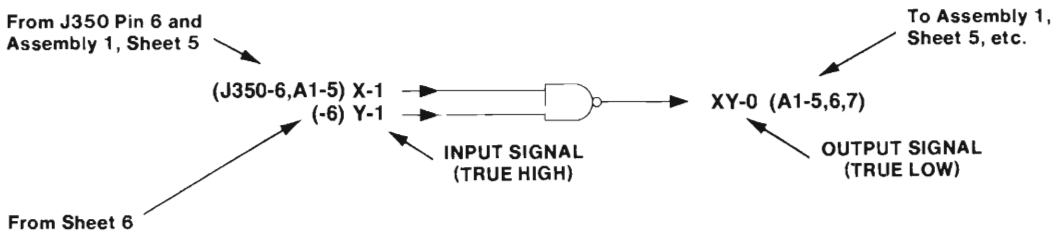
1. True High and True Low Signals

Signal names on the schematics are followed by -1 or a -0. A TRUE HIGH signal is indicated by -1, and a TRUE LOW signal is indicated by -0.

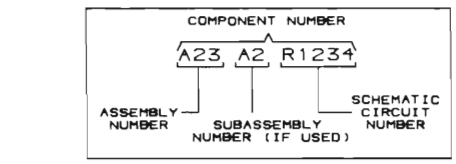
SIGNAL -1 = TRUE HIGH
SIGNAL -0 = TRUE LOW

2. Cross-References

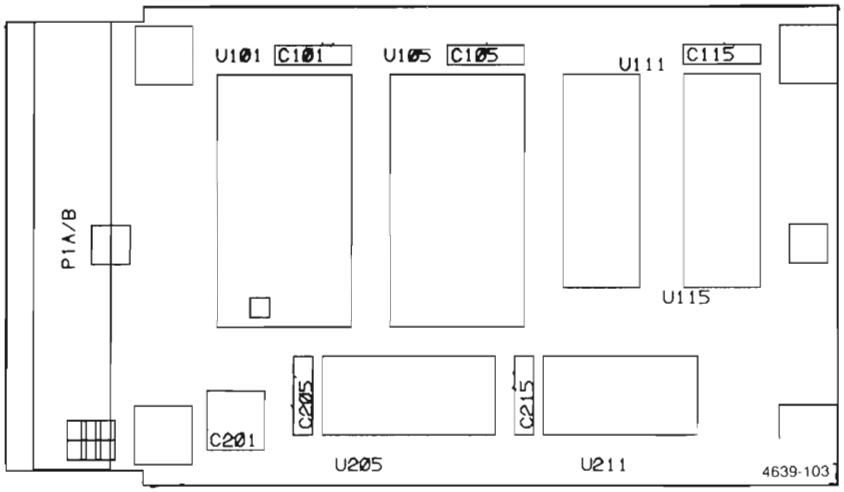
Schematic cross-references (from/to information) are included on the schematics. The "from" reference only indicates the signal "source," and the "to" reference lists all loads where the signal is used. All from/to information will be enclosed in parentheses.



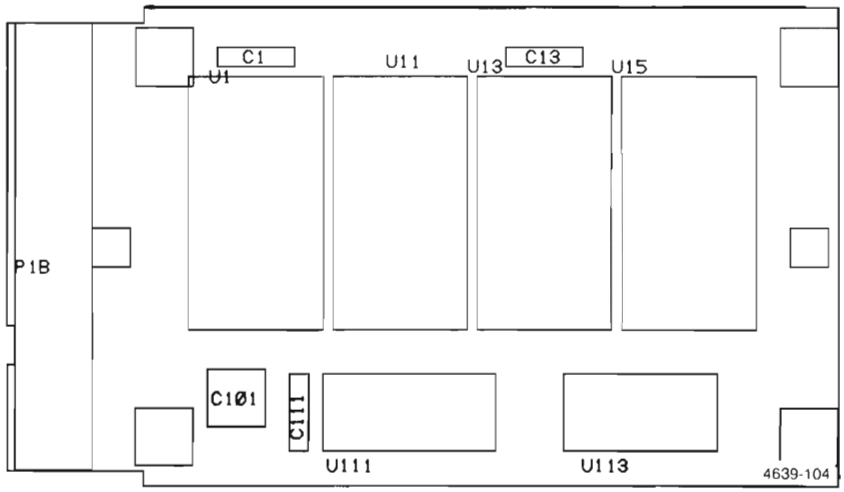
3. Component Number Example



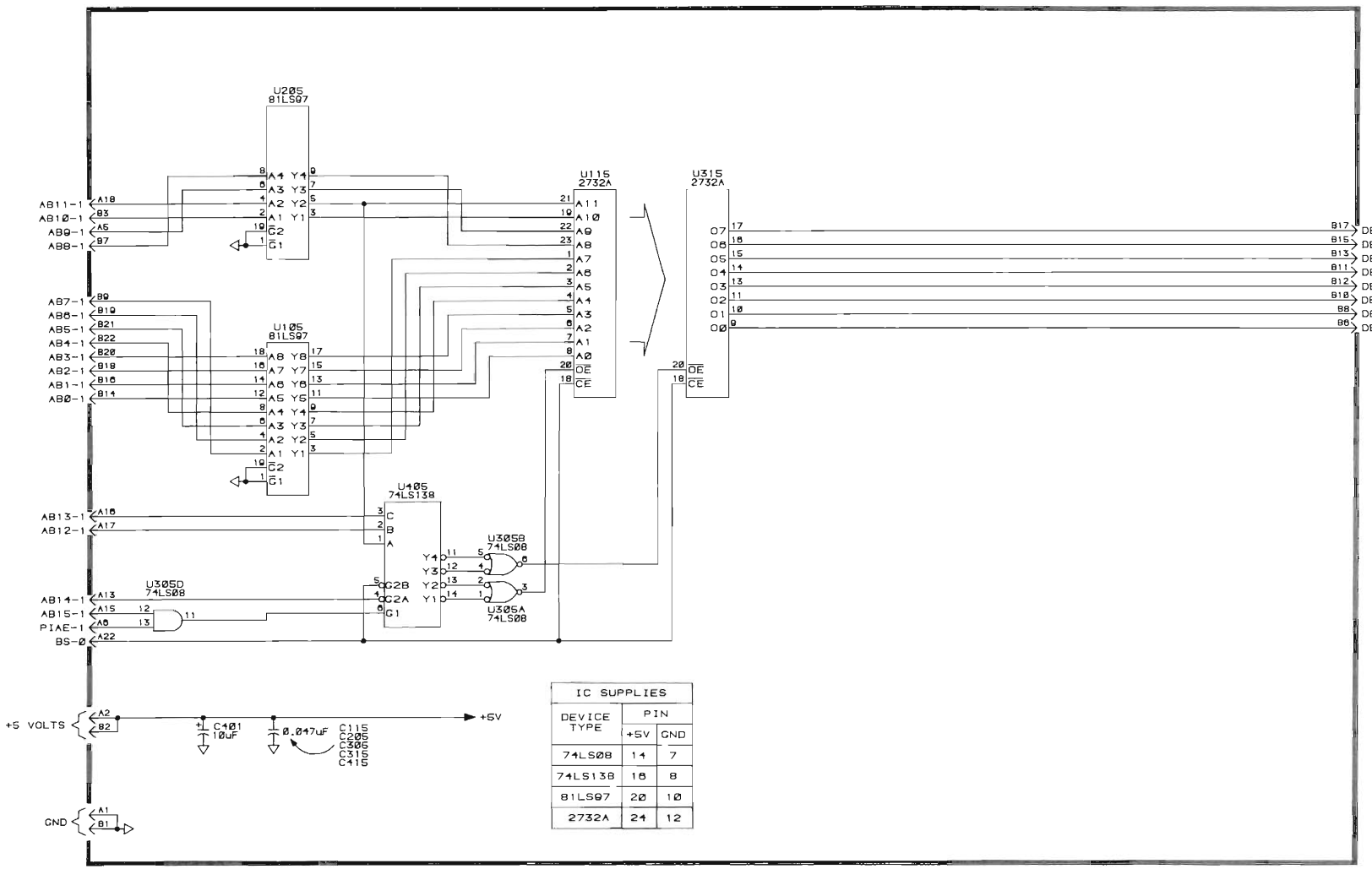
CHASSIS-MOUNTED COMPONENTS HAVE NO ASSEMBLY NUMBER PREFIX— SEE END OF REPLACEABLE ELECTRICAL PARTS LIST



4051R12 8K ROM Pack (670–8200-00) Component Locations.

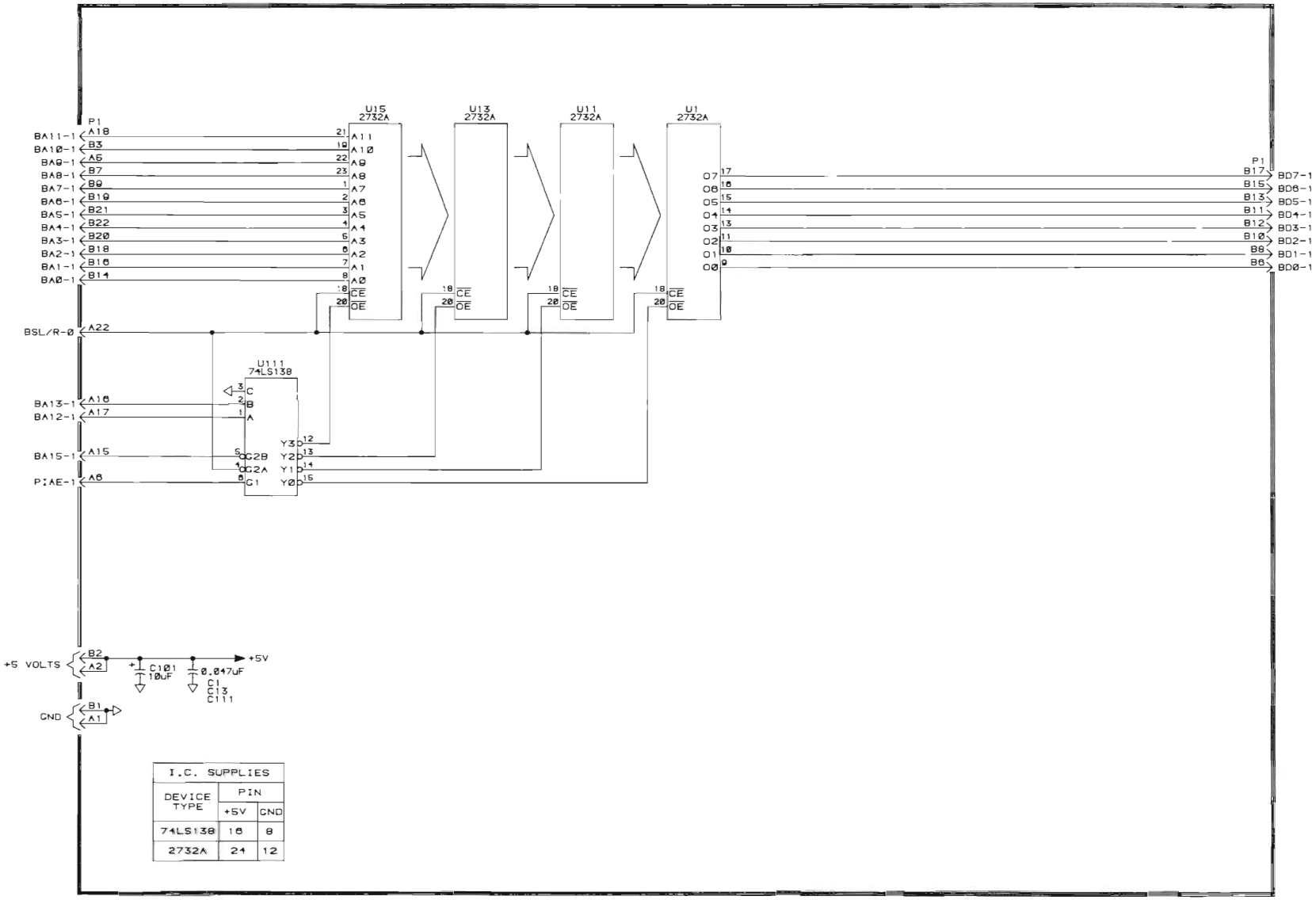


4052R12 16K ROM Pack (670-8201-00) Component Locations.



IC SUPPLIES		
DEVICE TYPE	PIN	
	+5V	GND
74LS08	14	7
74LS138	16	8
81LS97	20	10
2732A	24	12

INSTRUMENT:	NOTES:	BOARD: 870-8200-00 BK ROM PACK	ASSEMBLY-SHEET:
4051R12	4839-S1	GRAPHICS ENHANCEMENT ROM PACK	A1-1 (1 OF 1)



I.C. SUPPLIES		
DEVICE TYPE	PIN	
	+5V	GND
74LS138	16	8
2732A	24	12

INSTRUMENT:	NOTES: KEYSLOT LOCATED BETWEEN PINS A8 & A7.	BOARD: 670-8201-00 16K ROM PACK	ASSEMBLY-SHEET:
4052R12	1838-S2	GRAPHICS ENHANCEMENT ROM PACK	A2-1 (1 OF 1)

Appendix A

COMMAND SUMMARY

Function	Example	Purpose
ACROSS	CALL"ACROSS",C,X,Y	Draws crosshairs C times at position X,Y.
ADOTS	CALL"ADOTS",I$,C,X,Y	Draws image I$ dots C times at position X,Y.
ADRAW	CALL"ADRAW",I$,C,X,Y	Draws image I$ vectors C times at position X,Y.
AGIN	CALL"AGIN",X,Y	Finds graphic cursor X,Y position.
AINPUT	CALL"AINPUT",S$,C,X,Y,K$	Prints S$ and keys K$ C times at position X,Y.
AMOVE	CALL"AMOVE",I$,X,Y	Moves image I$ to position X,Y.
APOINT	CALL"APOINT",I$,N,X,Y	Find image I$ point N nearest position X,Y.
APRINT	CALL"APRINT",S$,C,X,Y	Print S$ C times at position X,Y.
AROTATE	CALL"AROTATE",I$,A,X,Y	Rotate image I$ by angle A around position X,Y.
ASCALE	CALL"ASCALE",I$,H,V,X,Y	Scale image I$ by factors H,V around position X,Y.
ASHEAR	CALL"ASHEAR",I$,H,V,X,Y	Shear image I$ by angles H,V around position X,Y.
ATAPER	CALL"ATAPER",I$,H,V,X,Y	Taper image I$ by factors H,V around position X,Y.
BOUNDS	CALL"BOUNDS",I$,X0,Y0,X1,Y1	Find image I$ minimum X,Y and maximum X,Y positions.

Function	Example	Purpose
CHANGE	CALL"CHANGE",F,I$	Change numeric array F to image I$.
	CALL"CHANGE",I$,F	Change image I$ to numeric array F.
DASHED	CALL"DASHED",H,V,X0,Y0,X1,Y1	Draw dashed grid H,V from X0,Y0 to X1,Y1.
DEFINE	CALL"DEFINE",I$,N,X,Y	Define image I$ point N X,Y position.
DOTTED	CALL"DOTTED",H,V,X0,Y0,X1,Y1	Draw dotted grid steps H,V from X0,Y0 to X1,Y1.
GCROSS	CALL"GCROSS",C,X,Y	Draw crosshairs C times at graphic + X,Y.
GDOTS	CALL"GDOTS",I$,C,X,Y	Draw image I$ dots C times at graphic + X,Y.
GDRAW	CALL"GDRAW",I$,C,X,Y	Draw image I$ vectors C times at graphic + X,Y.
GGIN	CALL"GGIN",X,Y	Find X,Y position at graphic + X,Y.
GINPUT	CALL"GINPUT",C$,C,X,Y,K$	Print C$ and keys K$ C times at graphic + X,Y.
GMOVE	CALL"GMOVE",I$,X,Y	Move image I$ to graphic + X,Y.
GPOINT	CALL"GPOINT",I$,N,X,Y	Find image I$ point N nearest graphic + X,Y.
GPRINT	CALL"GPRINT",C$,C,X,Y	Print C$ C times at graphic + X,Y.
GROTATE	CALL"GROTATE",I$,A,X,Y	Rotate image I$ by angle A around graphic + X,Y.
GSCALE	CALL"GSCALE",I$,H,V,X,Y	Scale image I$ by factors H,V around graphic + X,Y.
GSHEAR	CALL"GSHEAR",I$,H,V,X,Y	Shear image I$ by angles H,V around graphic + X,Y.
GTAPER	CALL"GTAPER",I$,H,V,X,Y	Taper image I$ by factors H,V around graphic + X,Y.

Function	Example	Purpose
IMAGES	CALL"IMAGES",S$	Enter Mag Tape ASCII file into S$.
INPUTS	CALL"INPUTS",C$	Enter C$ into keyboard buffer (28 characters minimum).
JCROSS	CALL"JCROSS",C,X,Y,K$	Draw crosshairs C times at joystick.
JDOTS	CALL"JDOTS",I$,C,X,Y,K$	Draw image I$ dots C times at joystick.
JDRAW	CALL"JDRAW",I$,C,X,Y,K$	Draw image I$ vectors C times at joystick.
JGIN	CALL"JGIN",C,X,Y,K$	After T milliseconds find joystick X,Y position and keys K$
JINPUT	CALL"JINPUT",C$,C,X,Y,K$	Print C$ and keys K$ C times at joystick.
JMOVE	CALL"JMOVE",I$,C,X,Y,K$	Move image I$ to joystick.
JPOINT	CALL"JPOINT",I$,C,N,X,Y,K$	Find image I$ point nearest joystick.
JPRINT	CALL"JPRINT",C$,C,X,Y,K$	Print C$ C times at joystick.
JROTATE	CALL"JROTATE",I$,C,A,X,Y,K$	Rotate image I$ by angle A around joystick.
JSCALE	CALL"JSCALE",I$,C,H,V,X,Y,K$	Scale image I$ by factors H,V around joystick.
JSHEAR	CALL"JSHEAR",I$,C,H,V,X,Y,K$	Shear image I$ by angles H,V around joystick.
JTAPER	CALL"JTAPER",I$,C,H,V,X,Y,K$	Taper image I$ by factors H,V around joystick.
LOCATE	CALL"LOCATE",C,X,Y,K$	Print joystick X,Y position at graphic.
MUSIC	CALL"MUSIC",S$ [S$ example: T4E316G332B4EGB5EG58.G516G54 F#R]	Sound musical notes in S$ at speaker.
POINTS	CALL"POINTS",I$,N,X,Y	Find image I$ point N X,Y position.

Function	Example	Purpose
PRINTS	CALL"PRINTS",C$	Print C$ (printable control characters).
RCROSS	CALL"RCROSS",C,X,Y	Draw crosshairs C times at center + X,Y.
RDOTS	CALL"RDOTS",I$,C,X,Y	Draw image I$ dots C times at relative X,Y.
RDRAW	CALL"RDRAW",I$,C,X,Y	Draw image I$ vectors C times at relative X,Y.
RGIN	CALL"RGIN",X,Y	Find graphic cursor X,Y position relative to X,Y.
RINPUT	CALL"RINPUT",C$,C,X,Y,K$	Print C$ and keys K$ C times at center + X,Y.
RMOVE	CALL"RMOVE",I$,X,Y	Move image I$ to relative position X,Y.
RPOINT	CALL"RPOINT",I$,N,X,Y	Find image I$ point N nearest relative X,Y.
RPRINT	CALL"RPRINT",C$,C,X,Y	Print C$ C times at center + X,Y.
RROTATE	CALL"RROTATE",I$,A,X,Y	Rotate image I$ by angle A around relative X,Y.
RSCALE	CALL"RSCALE",I$,H,V,X,Y	Scale image I$ by factors H,V around relative X,Y.
RSHEAR	CALL"RSHEAR",I$,H,V,X,Y	Shear image I$ by angles H,V around relative X,Y.
RTAPER	CALL"RTAPER",I$,H,V,X,Y	Taper image I$ by factors H,V around relative X,Y.
RUBBER	CALL"RUBBER",C,X,Y,K$	Draw rubber band from joystick to graphic.
R12LVL	CALL"R12LVL"	Print ROM pack ID, version, and copyright.
SOUNDS	CALL"SOUNDS",S$	Turn internal speaker on and off at rates dependent on value of characters in string.

Function	Example	Purpose
STRING	CALL"STRING",F,S$	Change numeric array F into string S$.
	CALL"STRING",S$,F	Change string S$ into numeric array F.
TOGGLE	CALL"TOGGLE",I$,N	Toggle image I$ point N move flag.
VERTEX	CALL"VERTEX",I$,C,X,Y,K$	Draw rubber bands from joystick to image I$.

Appendix B

UNDERSTANDING ERRORS

The following error messages may occur while using the Graphics Enhancement ROM Pack. This abbreviated list of the more frequent errors is here for convenience. See a more complete list of error messages in the 4050 Series Graphic System Reference Manual or the Plot 50 Introduction to Graphic Programming in BASIC Instruction Manual.

ERROR MESSAGES

Message Number	Error Message	Meaning
12	INVALID COMMAND ARGUMENT IN IMMEDIATE LINE; or, INVALID COMMAND ARGUMENT IN LINE XX	There is a parameter error in the CALL statement to a ROM pack: the syntax is improper, or the routine argument is invalid.
32	CALL NAME INVALID IN IMMEDIATE LINE; or, CALL NAME INVALID IN LINE XX	The routine name specified in the CALL statement can not be found. The name may be misspelled; or the ROM pack may be missing, defective, or incorrectly installed. Example: CALL "FIX IT" where the routine "FIX IT" resides in a ROM pack that is not plugged into the system.
36	UNDEFINED VARIABLE IN IMMEDIATE LINE; or, UNDEFINED VARIABLE IN LINE XX	There is an unidentified variable in the specified line. Either a variable or an input array is not defined or contains undefined elements (i.e., not assigned a value). You may also get this error if a null string is in the ROM pack call. Example: INIT DIM A(2,2) A(1,2) = 4 PRINT A

Appendix C

THREE-BYTE STRING FORMAT

The R12 ROM Pack gets much of its speed from the way it stores graphic data points. MOVEs and DRAWs are stored as three character strings (substrings). These three characters take three bytes of storage in an ordinary 4050 BASIC string. The graphic data in a three-byte string is packed with absolute display coordinates, containing the exact resolution that the display requires (10 bits X, 10 bits Y). The data does not provide or require substantial conversions to screen space or user coordinates. These factors contribute to faster graphics.

To convert these 10-bit coordinates to 12-bit coordinates for the 4054 Graphic System, multiply by four. This causes the display to fill the screen as you would expect, but the actual resolution is still 10 bits.

The format of the three-byte string data items is as follows:

MSB LSB

FIRST BYTE

0	M/D	x	x	x	y	y	y

0: This bit always zero
M/D: MOVE/DRAW Flag (0 = DRAW)
xxx: Three high bits of 10-bit X coordinate
yyy: Three low bits of 10-bit Y coordinate

SECOND BYTE

0	x	x	x	x	x	x	x

0: This bit always zero
xxxxxxx: Seven low bits of 10-bit X coordinate

THIRD BYTE

0	y	y	y	y	y	y	y

0: This bit always zero
yyyyyyy: Seven low bits of 10-bit Y coordinate

Appendix D

ASCII CODE CHART

B7 B6 B5 / BITS / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0 0	20 DLE 10 16	40 SP 20 32	60 0 30 48	100 @ 40 64	120 P 50 80	140 ` 60 96	160 p 70 112
0 0 0 1	1 GTL SOH 1 1	21 LLO DC1 11 17	41 ! 21 33	61 1 31 49	101 A 41 65	121 Q 51 81	141 a 61 97	161 q 71 113
0 0 1 0	2 STX 2 2	22 DC2 12 18	42 " 22 34	62 2 32 50	102 B 42 66	122 R 52 82	142 b 62 98	162 r 72 114
0 0 1 1	3 ETX 3 3	23 DC3 13 19	43 # 23 35	63 3 33 51	103 C 43 67	123 S 53 83	143 c 63 99	163 s 73 115
0 1 0 0	4 SDC EOT 4 4	24 DCL DC4 14 20	44 $ 24 36	64 4 34 52	104 D 44 68	124 T 54 84	144 d 64 100	164 t 74 116
0 1 0 1	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 % 25 37	65 5 35 53	105 E 45 69	125 U 55 85	145 e 65 101	165 u 75 117
0 1 1 0	6 ACK 6 6	26 SYN 16 22	46 & 26 38	66 6 36 54	106 F 46 70	126 V 56 86	146 f 66 102	166 v 76 118
0 1 1 1	7 BEL 7 7	27 ETB 17 23	47 ' 27 39	67 7 37 55	107 G 47 71	127 W 57 87	147 g 67 103	167 w 77 119
1 0 0 0	10 GET BS 8 8	30 SPE CAN 18 24	50 (28 40	70 8 38 56	110 H 48 72	130 X 58 88	150 h 68 104	170 x 78 120
1 0 0 1	11 TCT HT 9 9	31 SPD EM 19 25	51) 29 41	71 9 39 57	111 I 49 73	131 Y 59 89	151 i 69 105	171 y 79 121
1 0 1 0	12 LF A 10	32 SUB 1A 26	52 * 2A 42	72 : 3A 58	112 J 4A 74	132 Z 5A 90	152 j 6A 106	172 z 7A 122
1 0 1 1	13 VT B 11	33 ESC 1B 27	53 + 2B 43	73 ; 3B 59	113 K 4B 75	133 [5B 91	153 k 6B 107	173 { 7B 123
1 1 0 0	14 FF C 12	34 FS 1C 28	54 , 2C 44	74 < 3C 60	114 L 4C 76	134 \ 5C 92	154 l 6C 108	174 | 7C 124
1 1 0 1	15 CR D 13	35 GS 1D 29	55 - 2D 45	75 = 3D 61	115 M 4D 77	135] 5D 93	155 m 6D 109	175 } 7D 125
1 1 1 0	16 SO E 14	36 RS 1E 30	56 . 2E 46	76 > 3E 62	116 N 4E 78	136 ^ 5E 4	156 n 6E 110	176 ~ 7E 126
1 1 1 1	17 SI F 15	37 US 1F 31	57 / 2F 47	77 UNL ? 3F 63	117 O 4F 79	137 UNT _ 5F 95	157 o 6F 111	177 RUBOUT (DEL) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY

octal 25 PPU — GPIB code
NAK — ASCII character
hex 15 21 — decimal

Appendix E

TRANSFORMATION EXAMPLES

The transformations appear in this order:

X DOTS
Y DOTS
XY DOTS

X CROSS
Y CROS
XY CROSS

X DRAW
Y DRAW
XY DRAW

X PRINT
Y PRINT
XY PRINT

X DASHED
Y DASHED
XY DASHED (3)

X DOTTED
Y DOTTED
XY DOTTED (3)

X MOVE
Y MOVE
XY MOVE

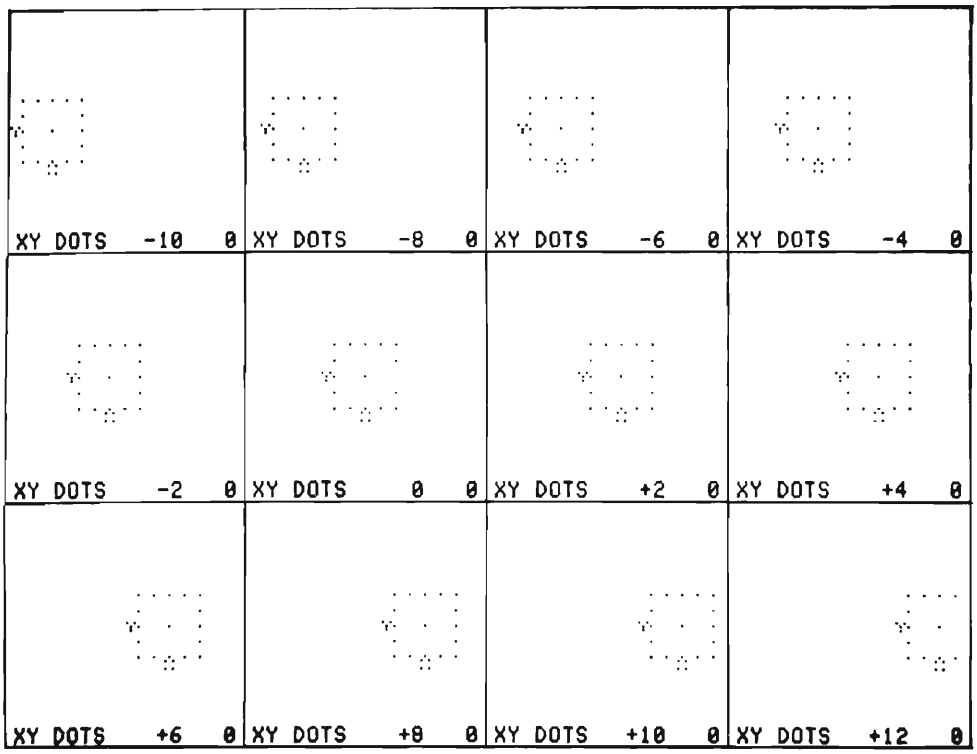
X SCALE
Y SCALE
XY SCALE

X SHEAR
Y SHEAR
XY SHEAR

X TAPER
Y TAPER
XY TAPER

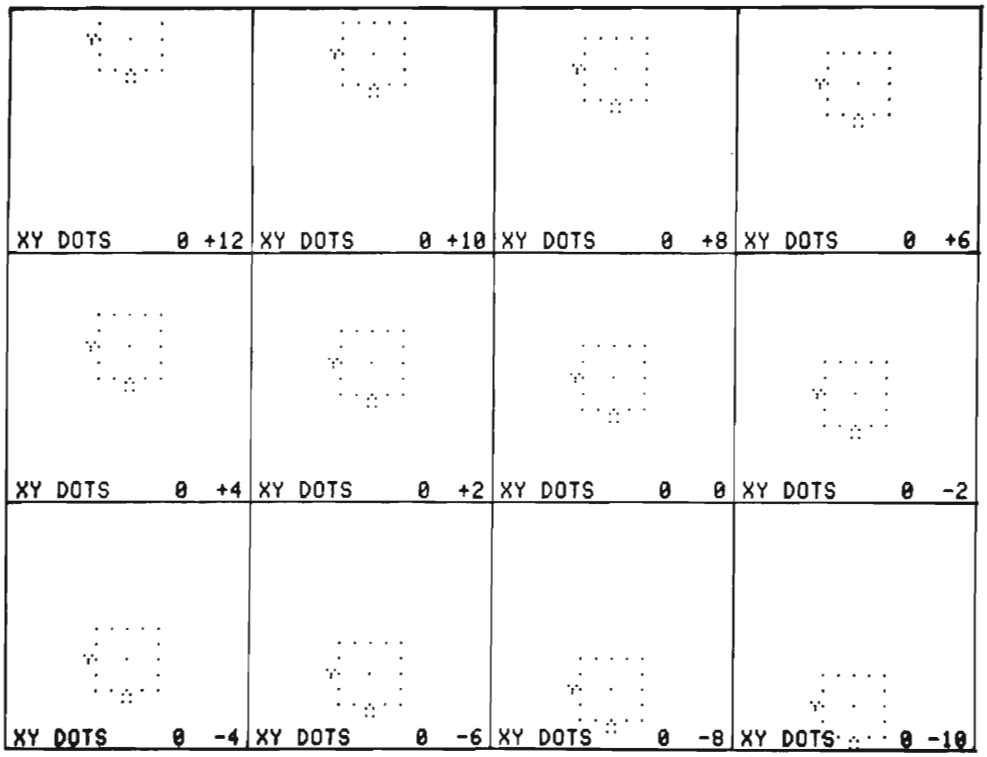
ROTATION

X DOTS TRANSFORMATIONS



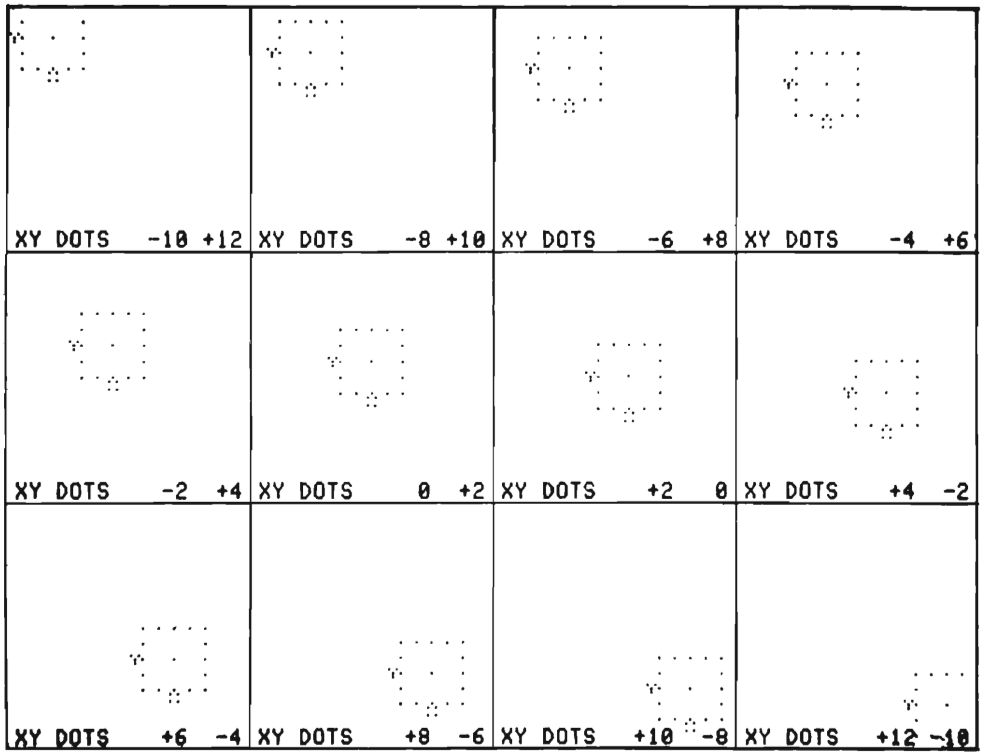
4639-68

Y DOTS TRANSFORMATIONS



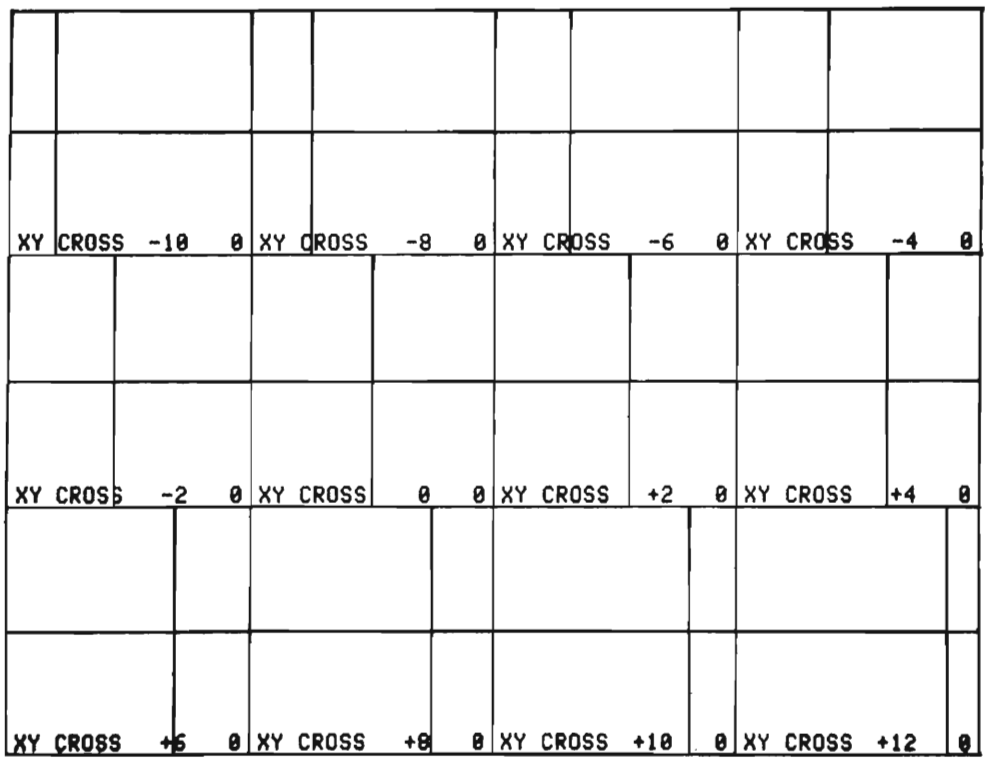
4639-69

XY DOTS TRANSFORMATIONS



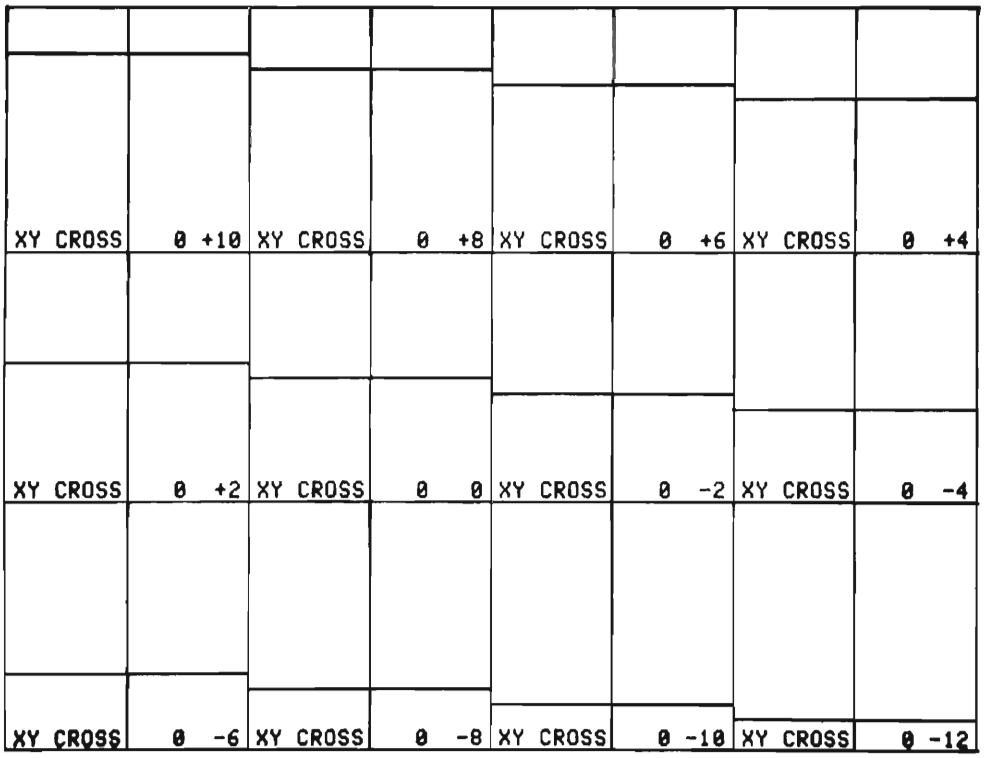
4639-70

X CROSS TRANSFORMATIONS



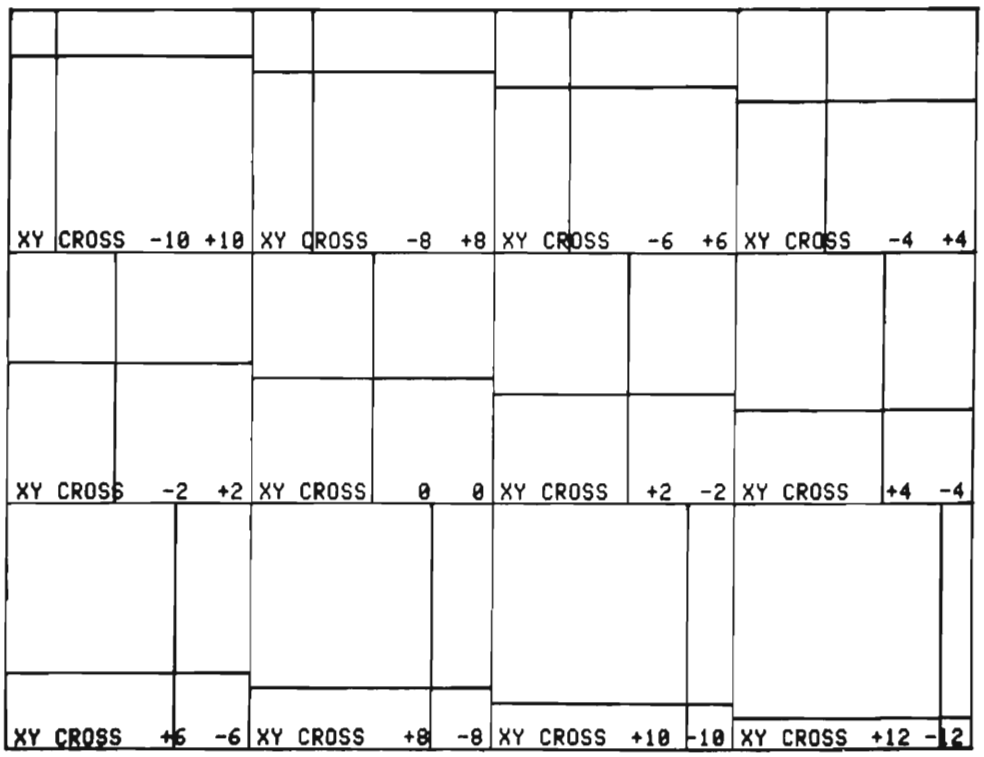
4639-71

Y CROSS TRANSFORMATIONS



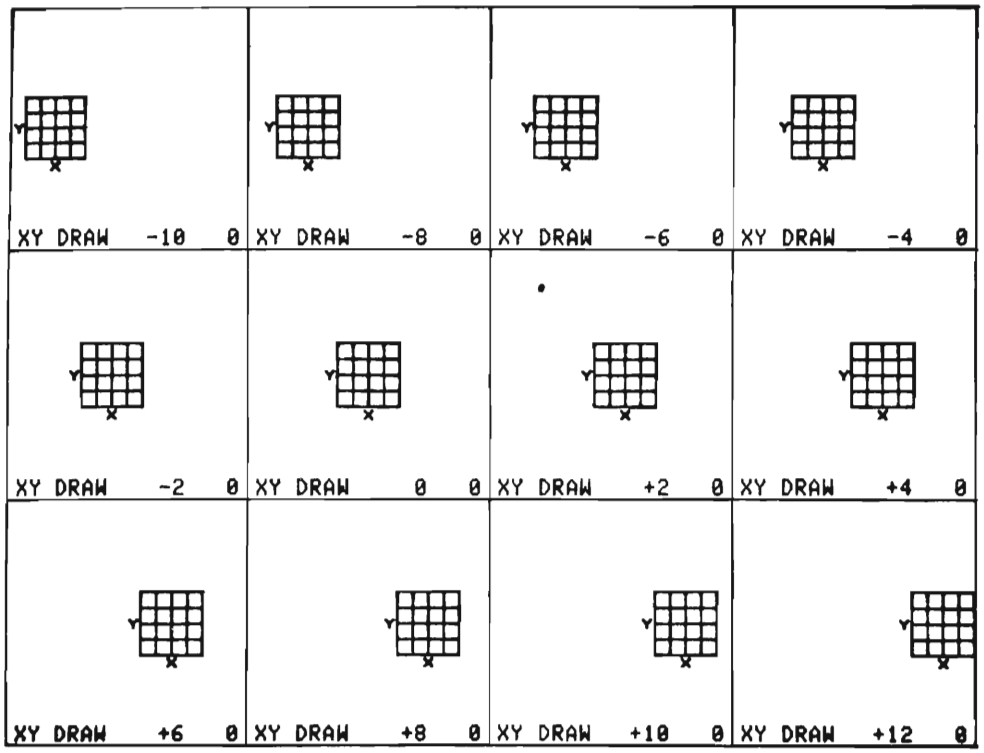
4639-72

XY CROSS TRANSFORMATIONS



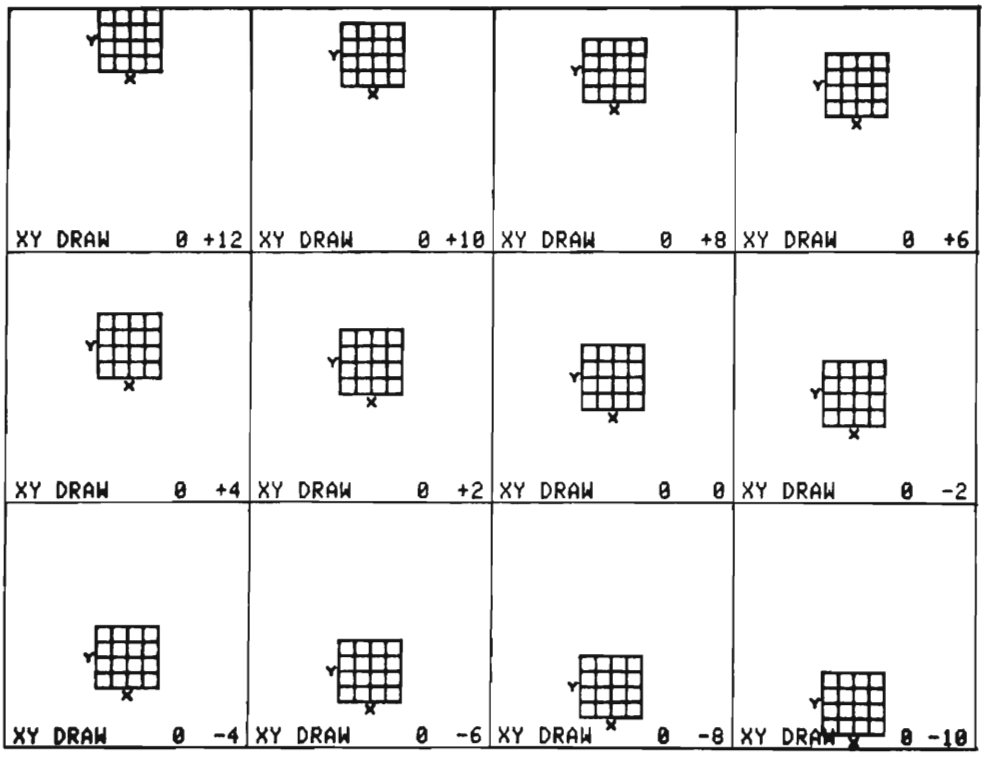
4639-73

X DRAW TRANSFORMATIONS



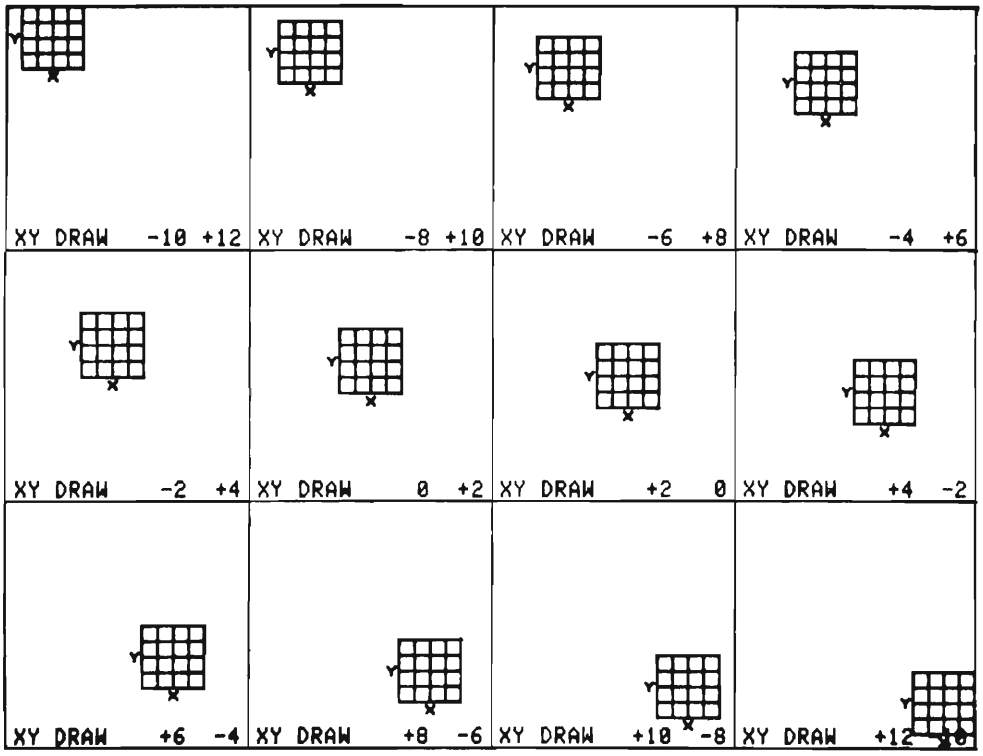
4639-74

Y DRAW TRANSFORMATIONS



4639-75

XY DRAW TRANSFORMATIONS



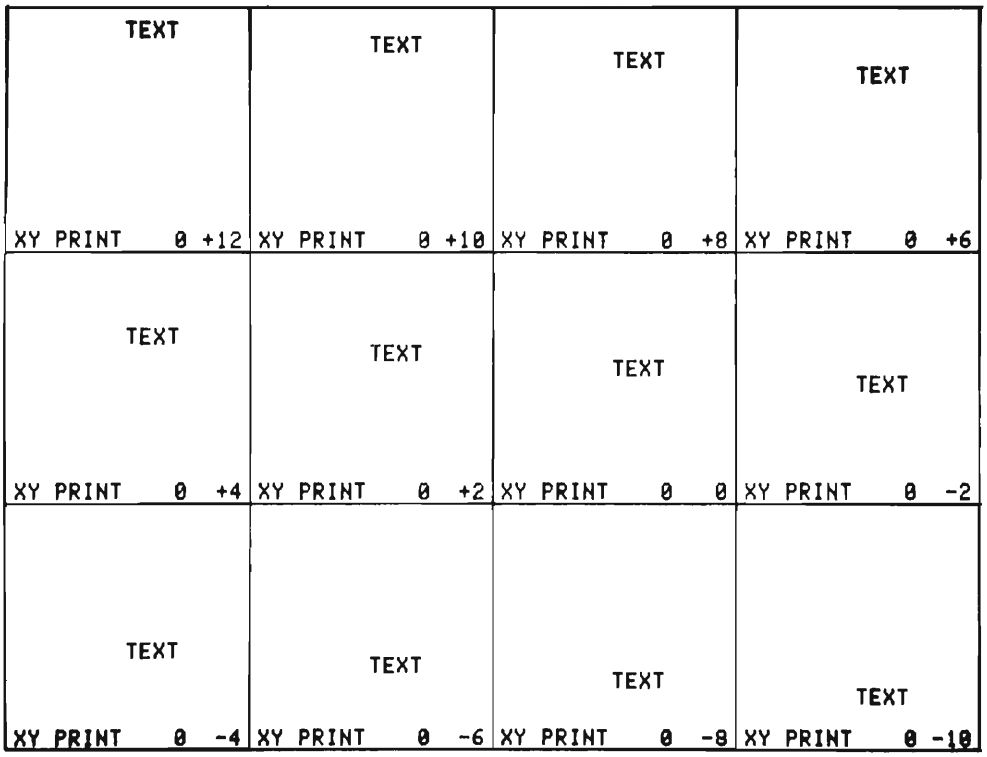
4639-76

X PRINT TRANSFORMATIONS

TEXT XY PRINT -14 0	TEXT XY PRINT -12 0	TEXT XY PRINT -10 0	TEXT XY PRINT -8 0
TEXT XY PRINT -6 0	TEXT XY PRINT -4 0	TEXT XY PRINT -2 0	TEXT XY PRINT 0 0
TEXT XY PRINT +2 0	TEXT XY PRINT +4 0	TEXT XY PRINT +6 0	TEXT XY PRINT +8 0

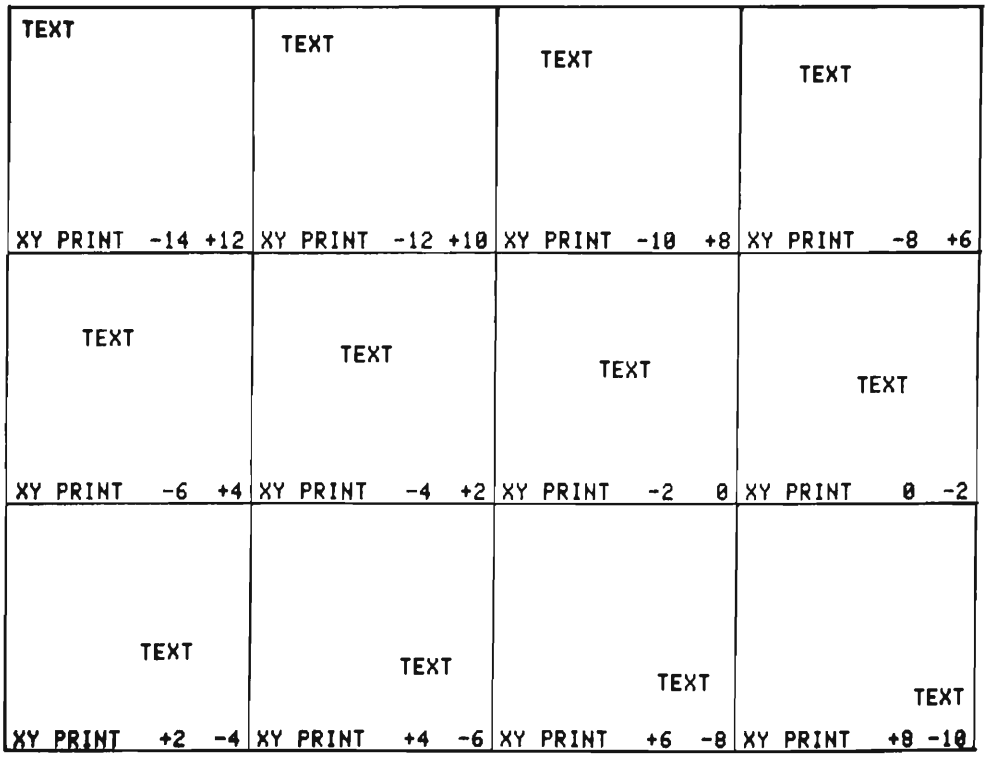
4639-77

Y PRINT TRANSFORMATIONS



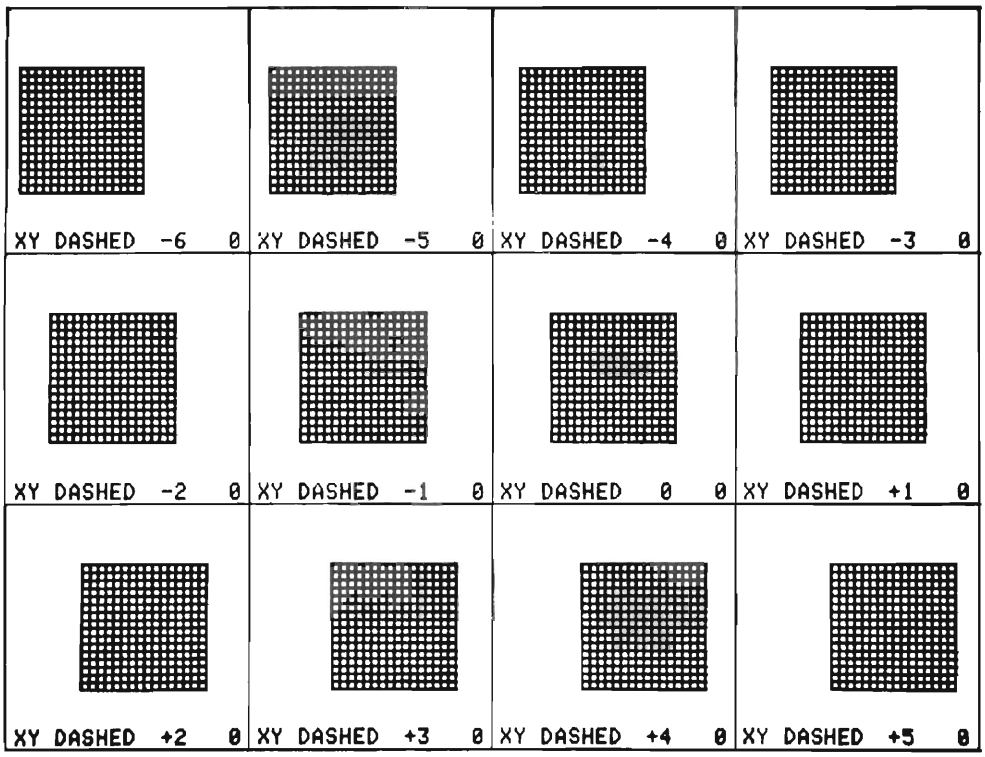
4639-78

XY PRINT TRANSFORMATIONS



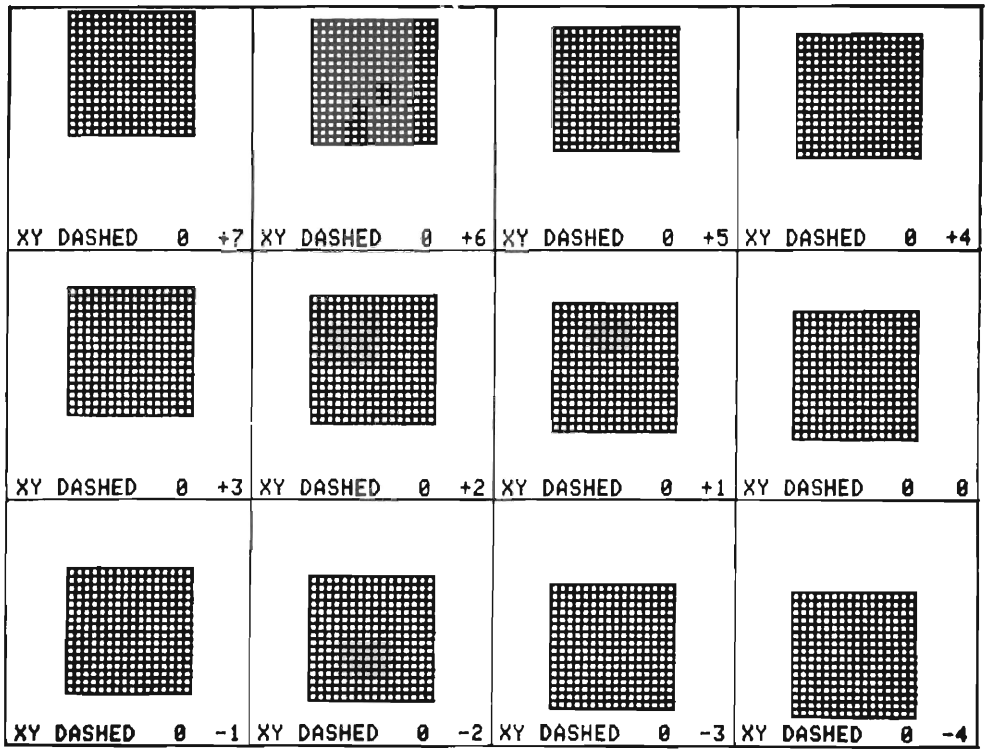
4639-79

X DASHED TRANSFORMATIONS



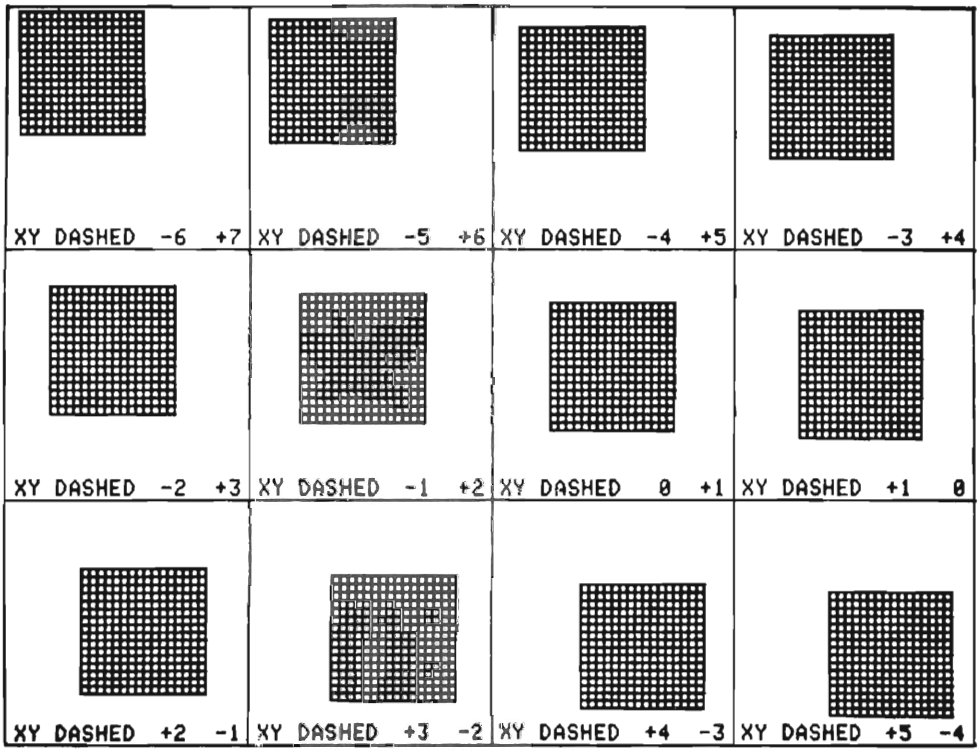
4639-80

Y DASHED TRANSFORMATIONS



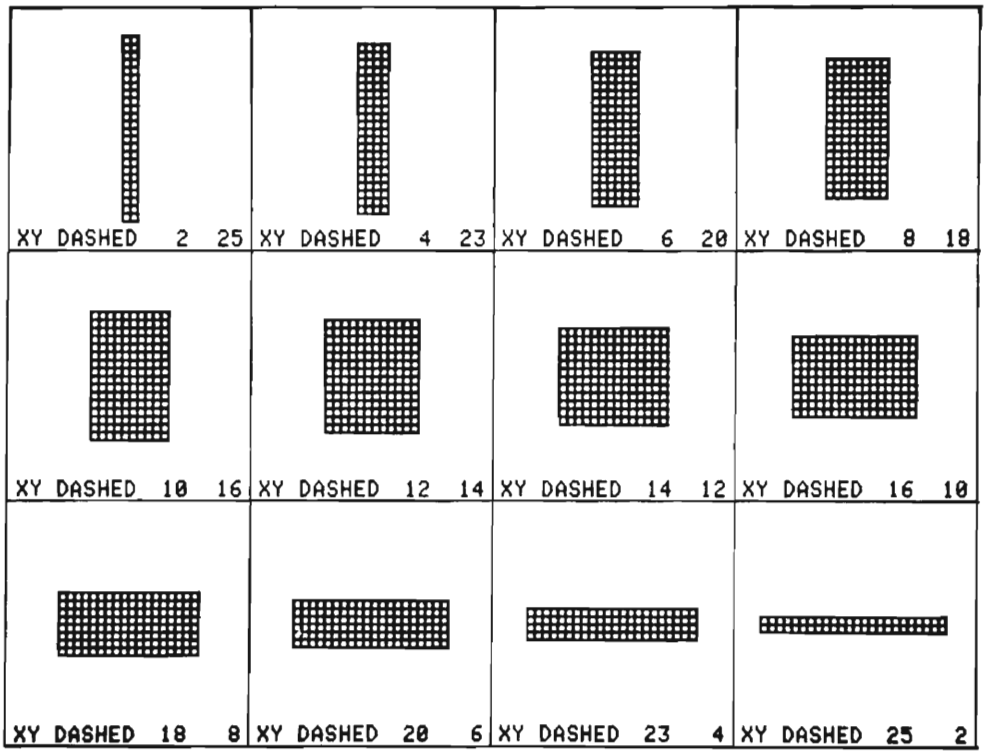
4639-81

XY DASHED TRANSFORMATIONS (1 OF 3)



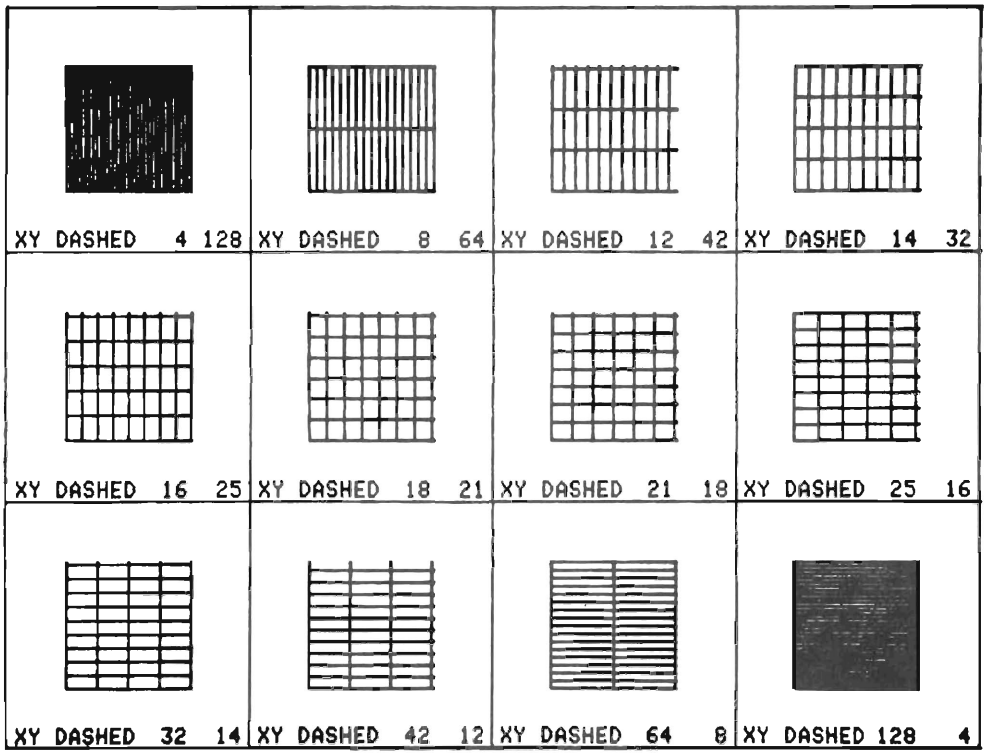
4639-82

XY DASHED TRANSFORMATIONS (2 OF 3)



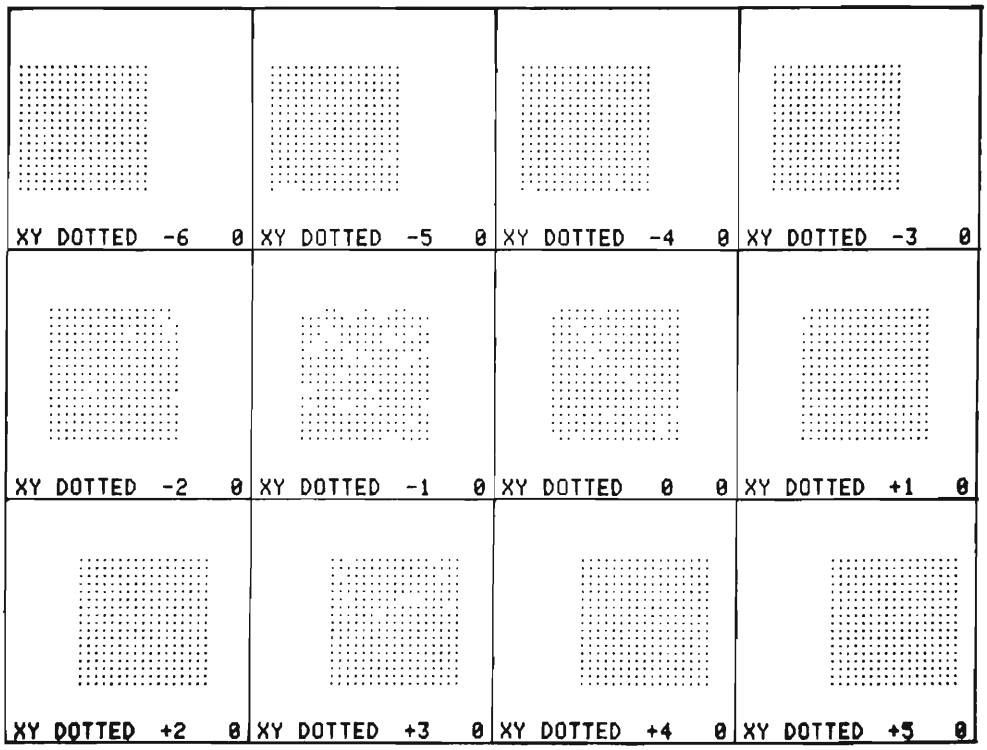
4639-83

XY DASHED TRANSFORMATIONS (3 OF 3)



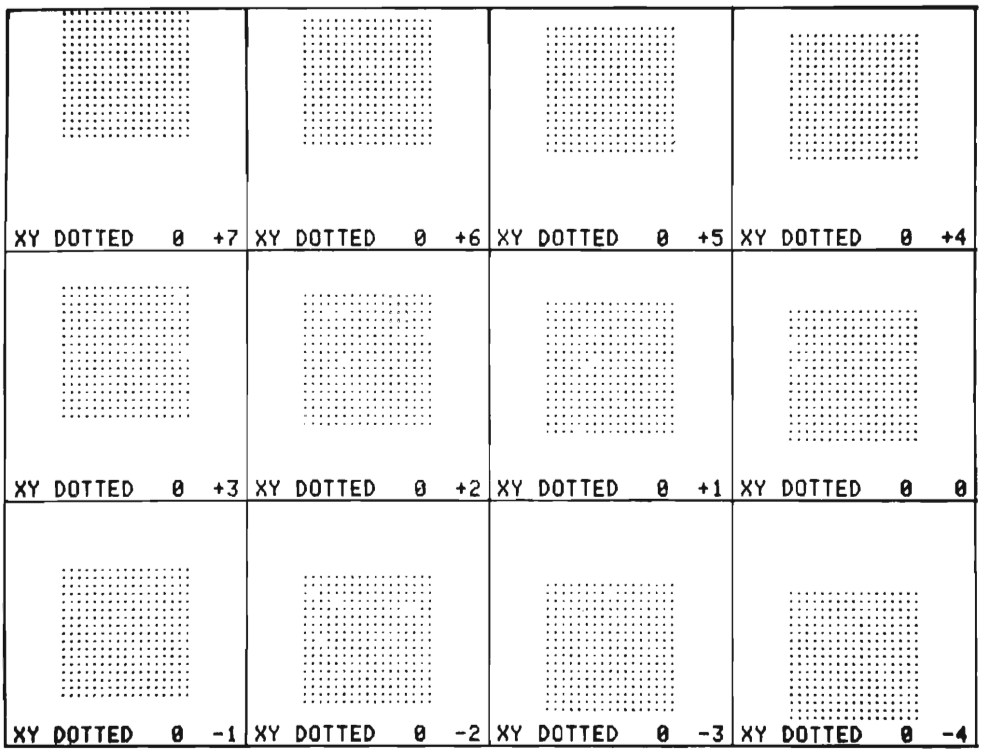
4639-84

X DOTTED TRANSFORMATIONS



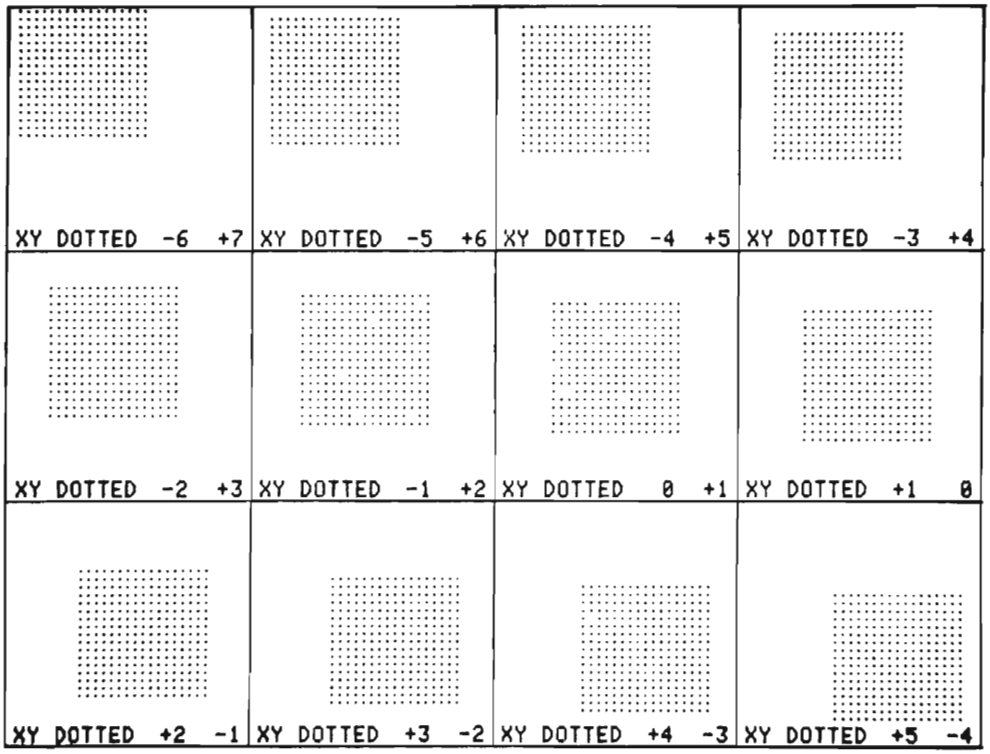
4639-85

Y DOTTED TRANSFORMATIONS



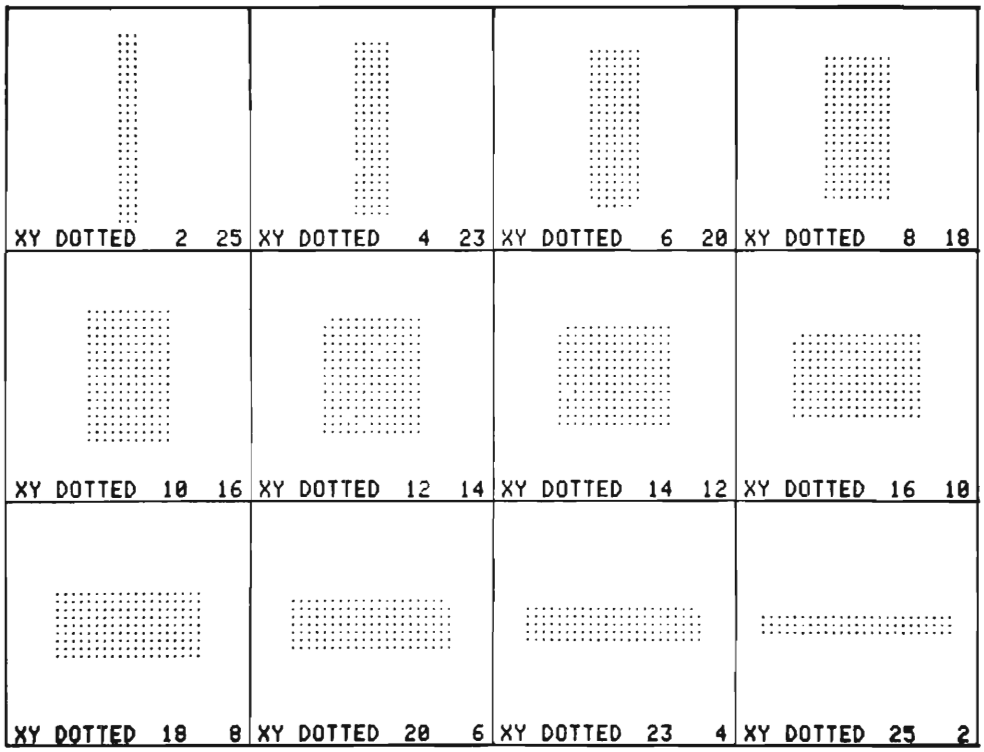
4639-86

XY DOTTED TRANSFORMATIONS (1 OF 3)



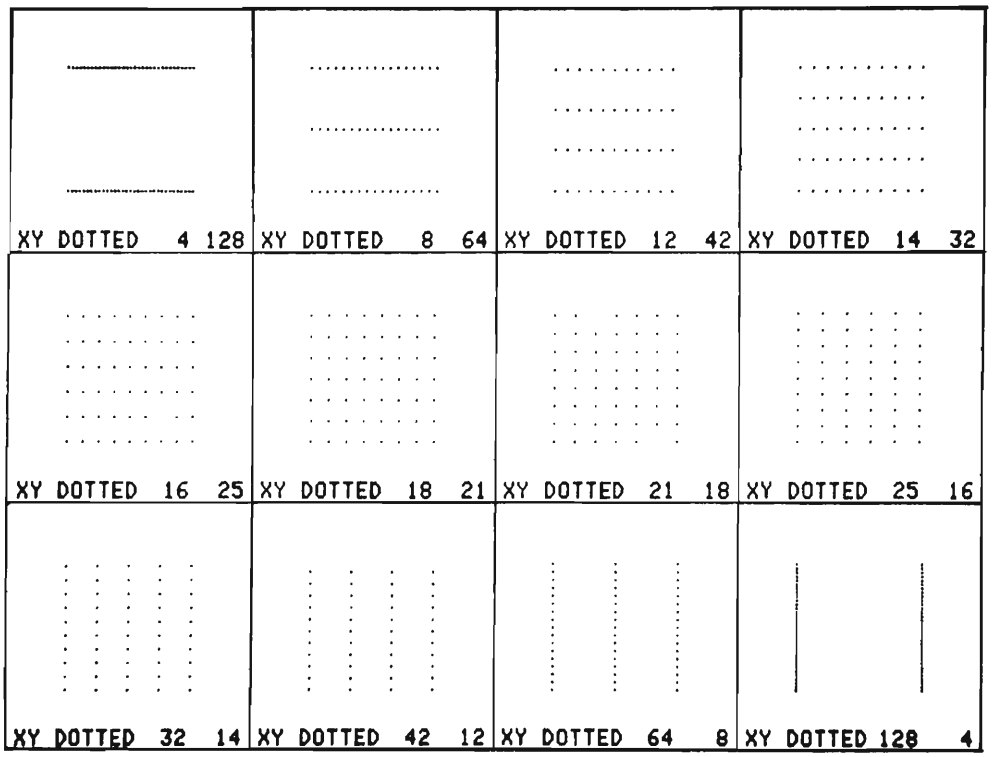
4639-87

XY DOTTED TRANSFORMATIONS (2 OF 3)



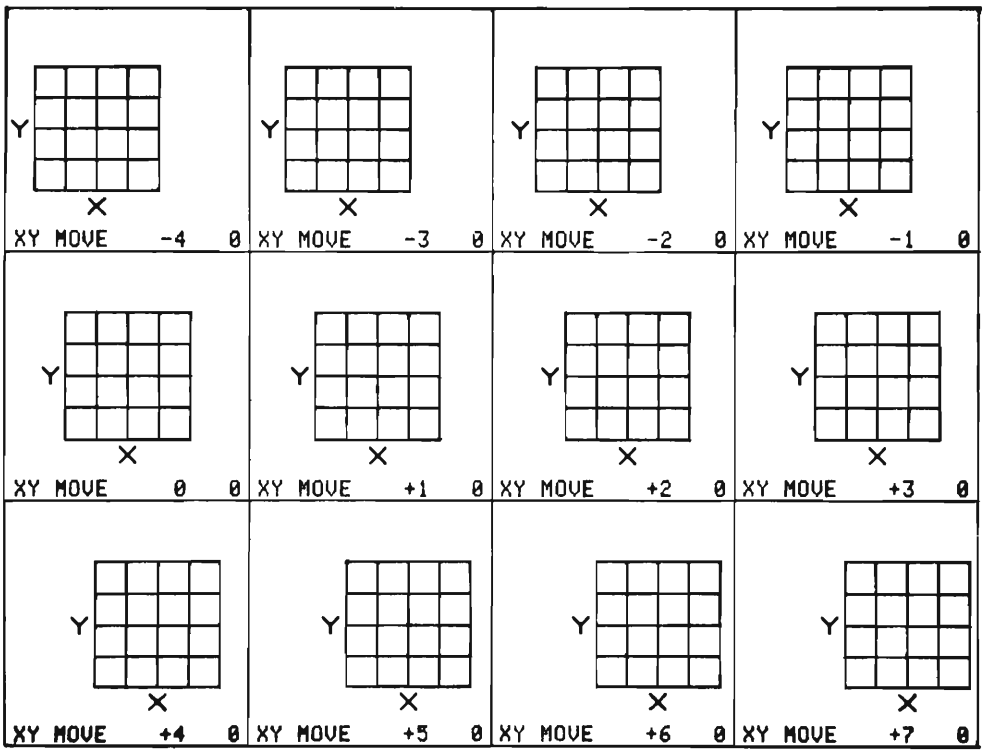
4639-88

XY DOTTED TRANSFORMATIONS (3 OF 3)



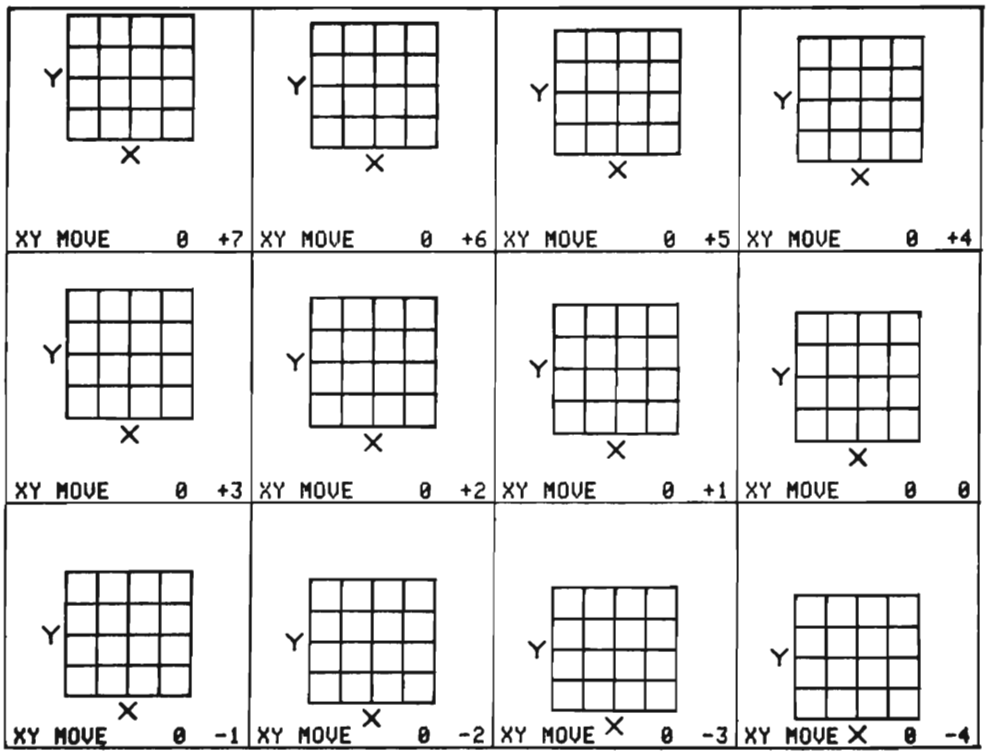
4639-89

X MOVE TRANSFORMATIONS



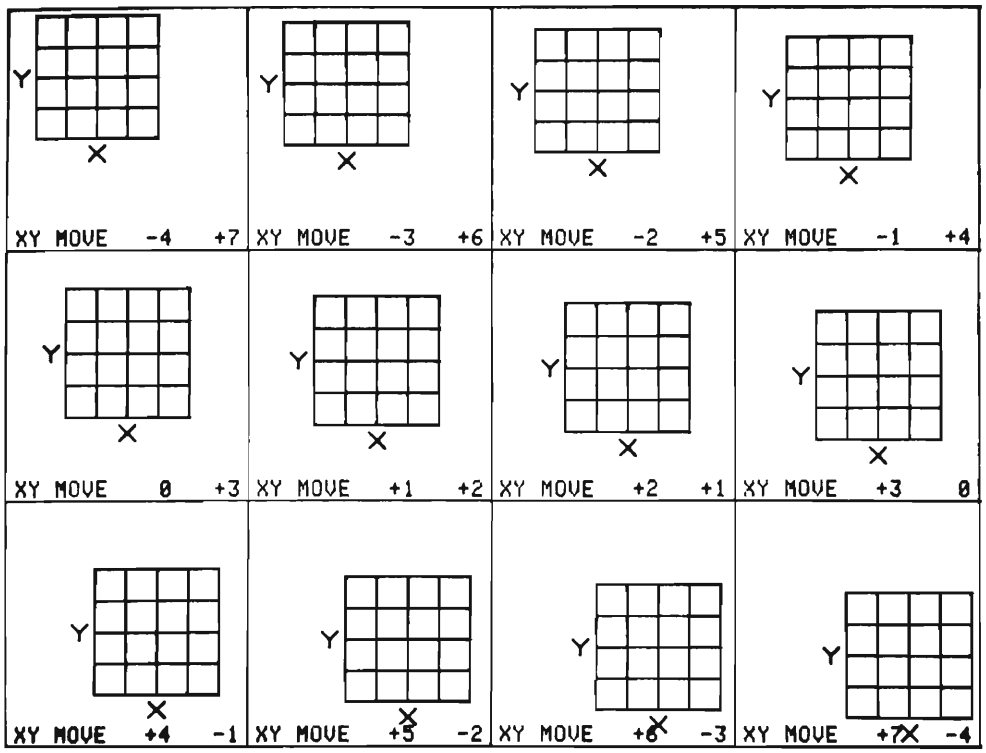
4639-90

Y MOVE TRANSFORMATIONS



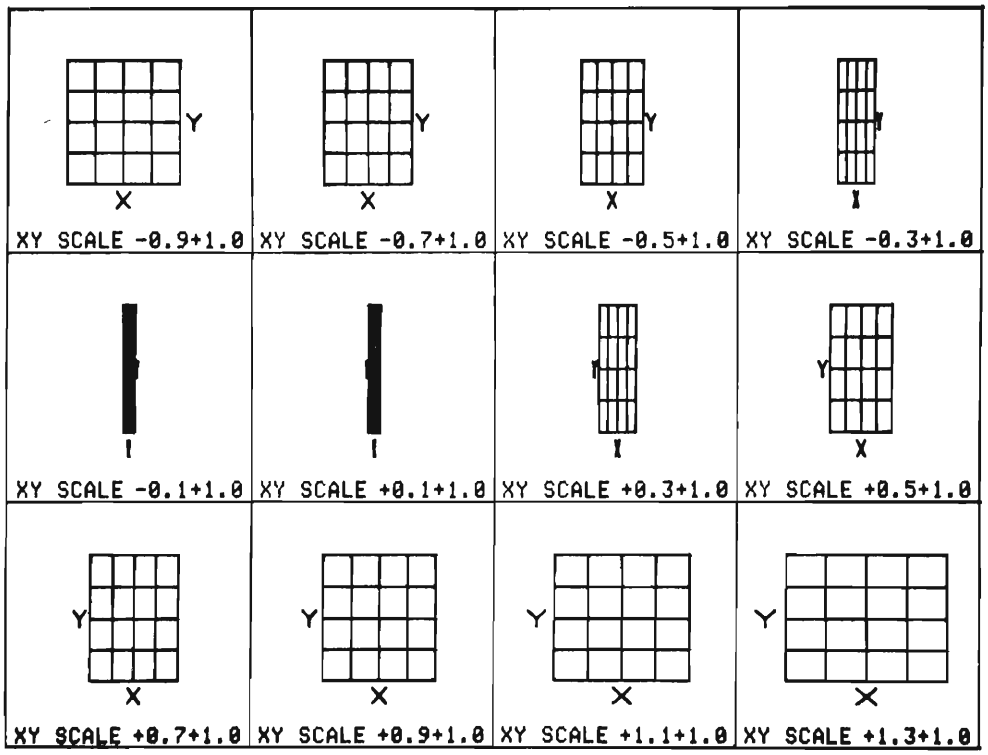
4639-91

XY MOVE TRANSFORMATIONS



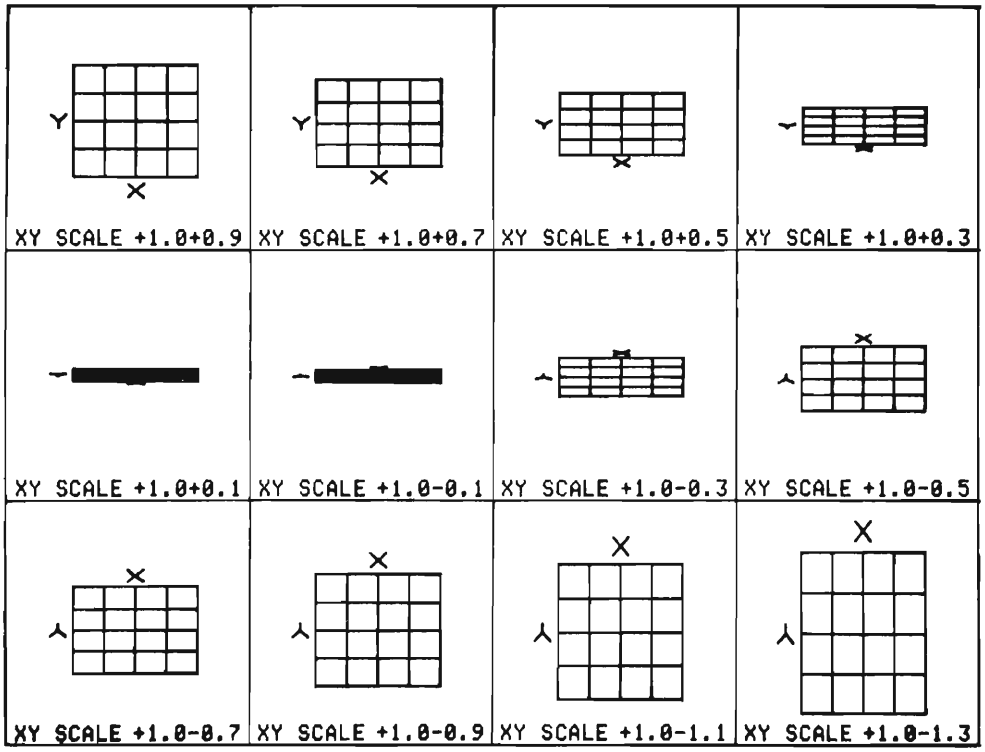
4639-92

X SCALE TRANSFORMATIONS



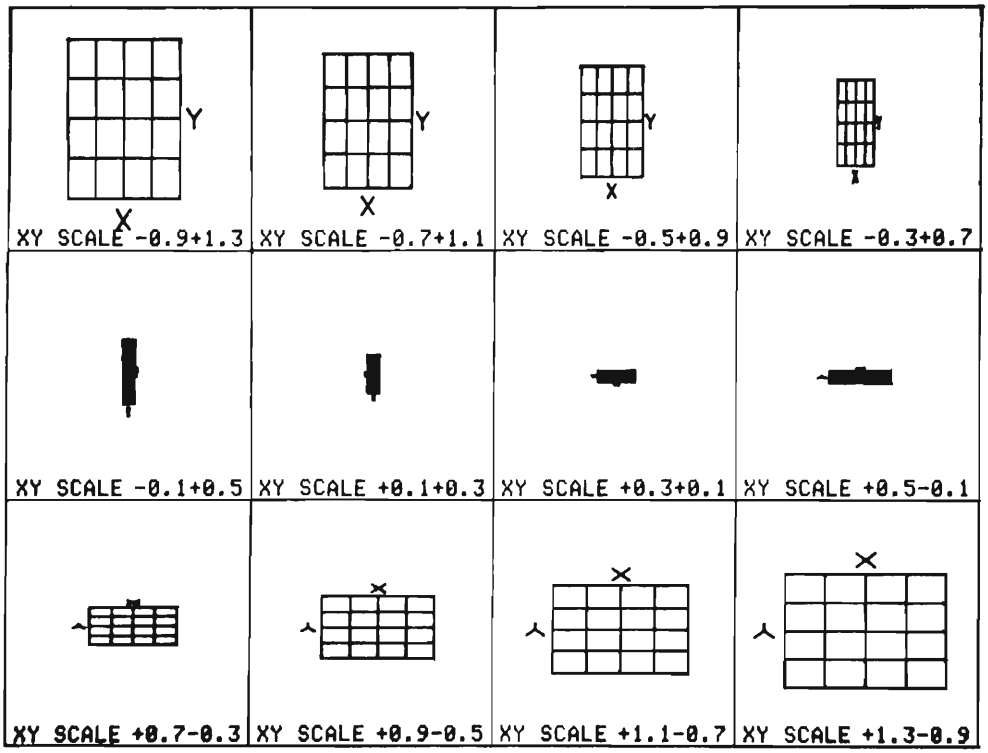
4639-93

Y SCALE TRANSFORMATIONS



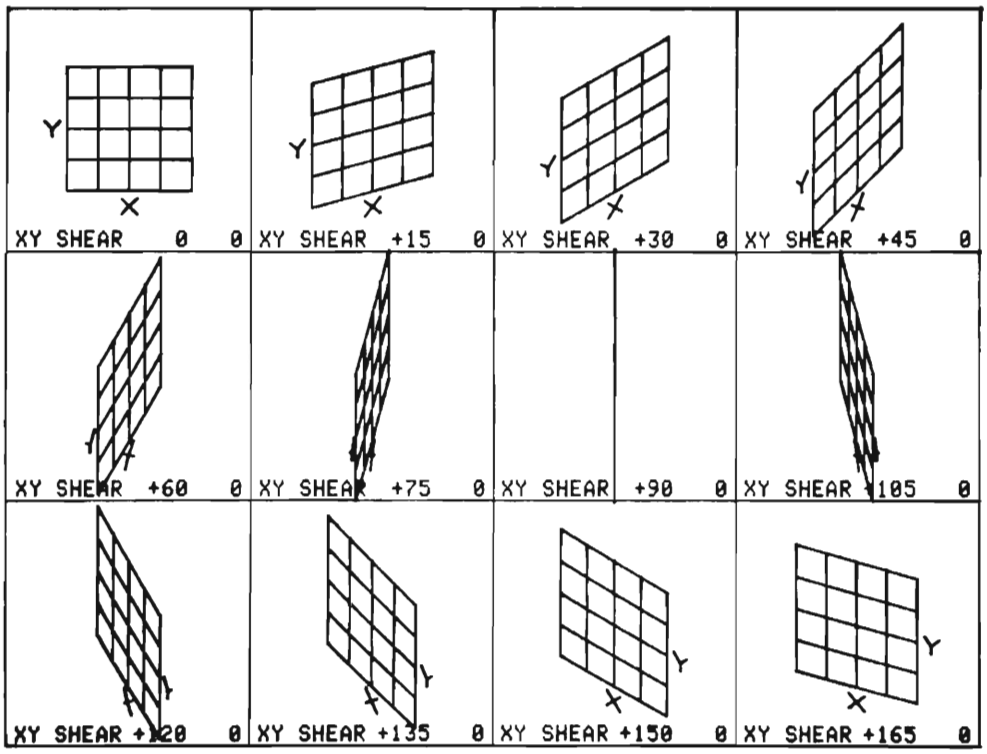
4639-94

XY SCALE TRANSFORMATIONS



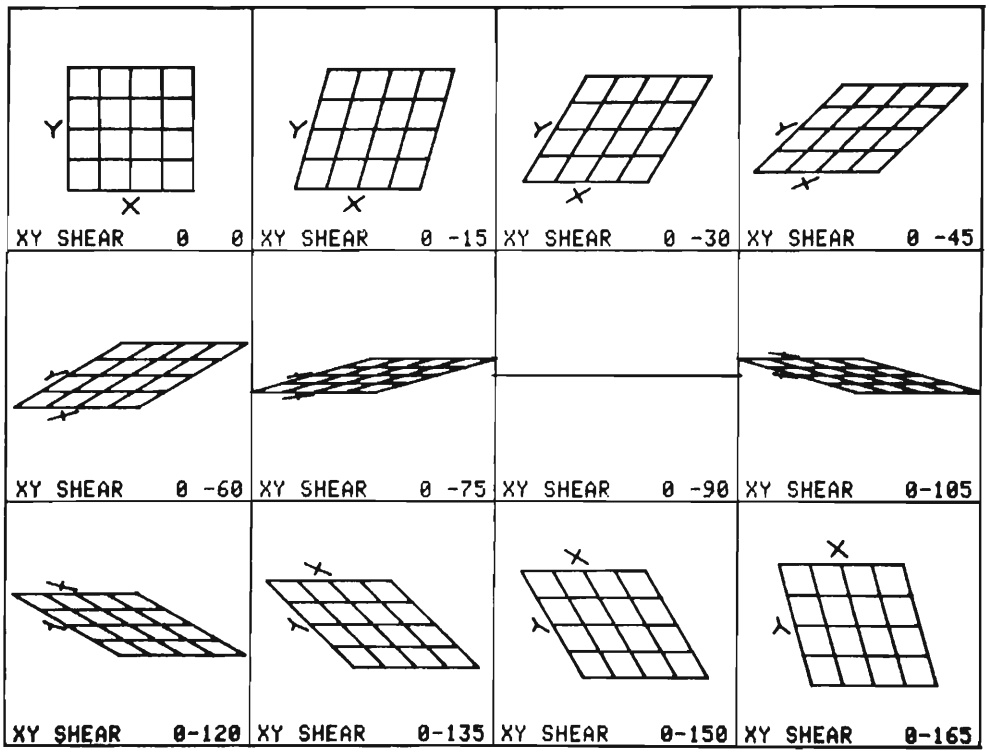
4639-95

X SHEAR TRANSFORMATIONS



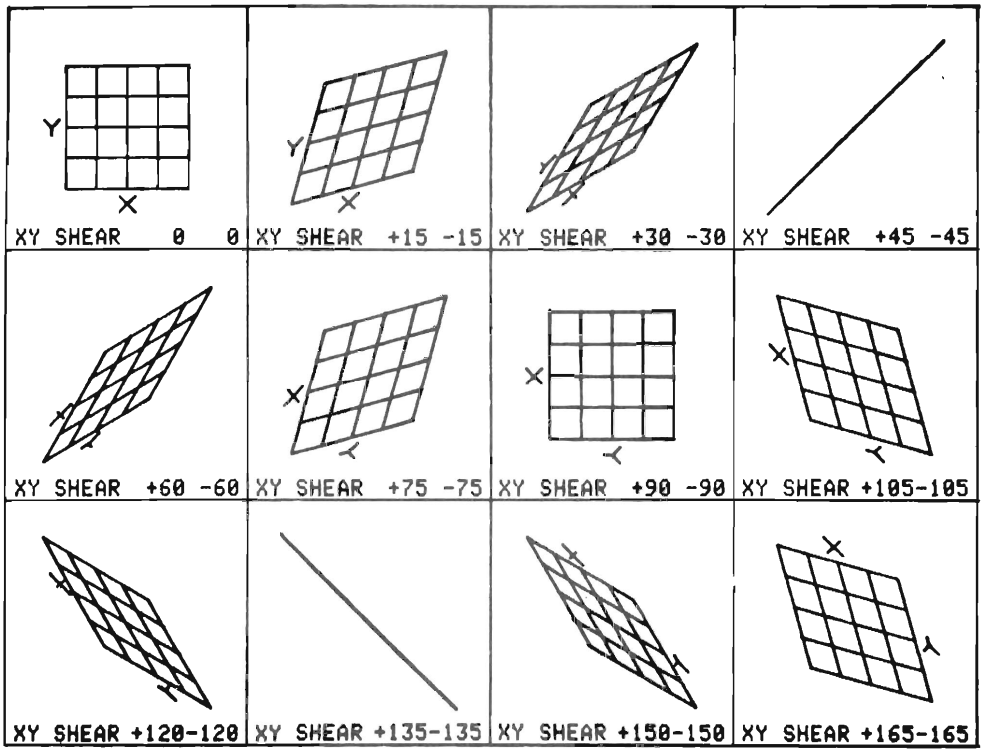
4639-96

Y SHEAR TRANSFORMATIONS



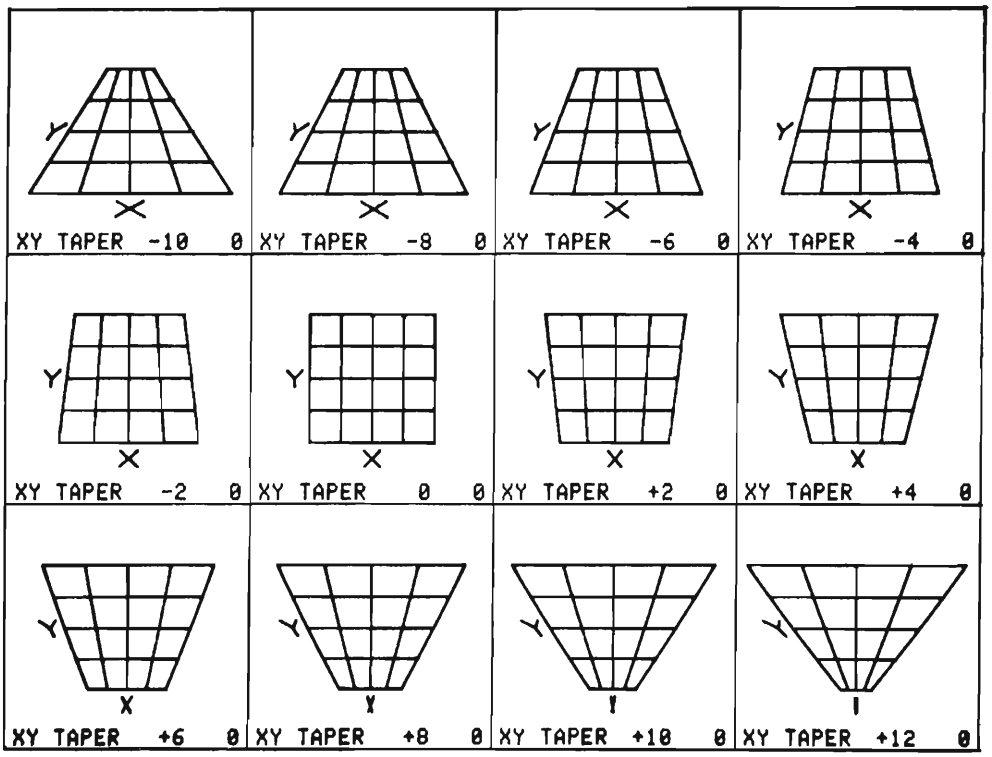
4639-97

XY SHEAR TRANSFORMATIONS



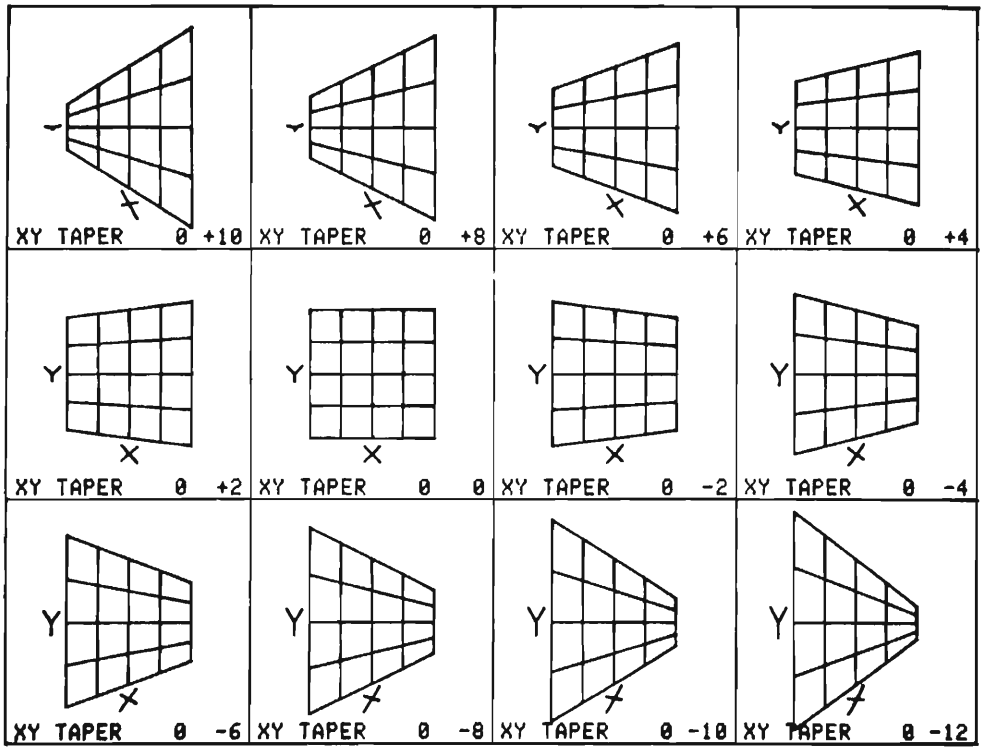
4639-98

X TAPER TRANSFORMATIONS



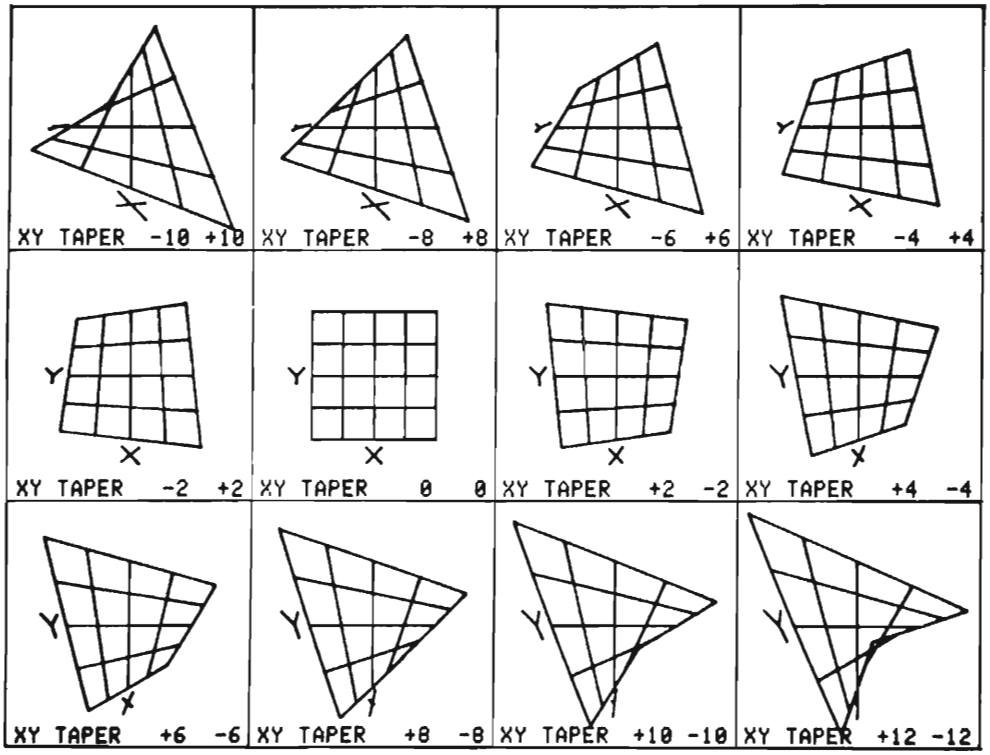
4639-99

Y TAPER TRANSFORMATIONS



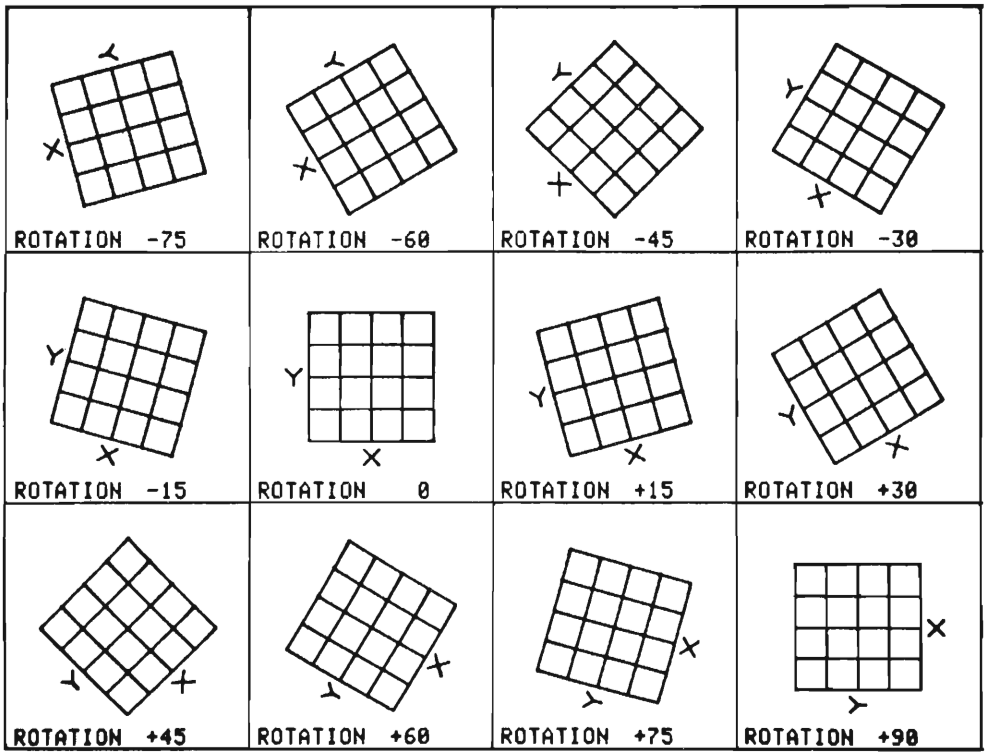
4639-100

XY TAPER TRANSFORMATIONS



4639-101

ROTATION TRANSFORMATIONS



4639-102

INDEX