Tektronix®
COMMITTED TO EXCELLENCE

Please Check for
CHANGE INFORMATION
at the Rear of this Manual

4054
Options 30 and 31
DYNAMIC GRAPHICS
OPERATOR'S MANUAL

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

MANUAL PART NO. 070-2289-02
PRODUCT GROUP 14

First Printing OCT 1979
Revised MAR 1982

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: 4054 Options 30 and 31 Dynamic Graphics

This manual supports the following versions of this product: Serial Numbers B010101 and up.

REV DATE	DESCRIPTION
NOV 1979	Original Issue
FEB 1980	Revised: page A-4.
JUL 1980	Revised: page 2-6.
NOV 1980	Revised: pages 1-9, 1-10, 1-11, 1-12, 2-16, 2-17, 2-20, 2-21, and D-4.
MAY 1981	Revised: pages 1-10 and A-3.
JUN 1981	Insert: pages 1-1, 1-2, and A-2 reflect addition of Option 31. Manual Part No. changed from 070-2289-00 to 070-2289-01.
OCT 1981	Reprint: manual reprinted as 070-2289-02.

CONTENTS

ILLUSTRATIONS

TABLES

Section 1

GENERAL DESCRIPTION

INTRODUCTION

The 4054 Options 30 and 31 provide the 4054 Graphic Computing System the ability to display Dynamic Graphics. Dynamic Graphics allow refreshed images to be moved, altered, or erased without erasing the entire screen. Refreshed images can be moved around the screen, either under program control or interactively using the thumbwheels or an optional graphic input device, such as the Joystick or Graphic Tablet.

Both Options 30 and 31 display both stored images and refreshed images. In addition, Option 31 produces Color Enhanced Dynamic Graphics; refreshed images appear yellow-orange and stored images appear green.

The Option 30 Circuit Board is installed in the 4054; no special ROM Packs or other add-ons are required. Installation is performed at the factory or in the field.

Option 31 must be installed in the 4054 Graphic Computing System before shipment.

Throughout this manual, all references to Option 30 also apply to Option 31 unless otherwise specified.

ABOUT THIS MANUAL

This manual describes how to program the 4054 Option 30 using the Dynamic Graphics commands. It shows you how to write simple refresh application programs. It also states the option's abilities and limitations so an experienced programmer can fully use the instrument's features.

The manual assumes the reader:

- has a working knowledge of the BASIC programming language.
- is familiar with the standard 4054 Graphic Computing System.
- has access to a 4050-Series Reference Manual, where the general 4054 commands are described.

This manual is organized as follows:

- Section 1 — Introduces product and manual, covers some Dynamic Graphics concepts and terms, provides some programmer helps, and includes an introductory Demonstration-Verification program.
- Section 2 — Lists and explains the Option 30 commands in alphabetical order.
- Appendix A — Characteristics
- Appendix B — Special Applications Program
- Appendix C — Glossary
- Appendix D — Command Summary
- Index

The companion Options 30 and 31 Dynamic Graphics Reference Guide summarizes information found in this manual.

DYNAMIC GRAPHICS CONCEPTS

This subsection gives an overview of 4054 Dynamic Graphics option operation and introduces some of the terms related to refresh graphics. Terms not previously used with 4050-Series Graphic Systems are set apart with quotes and defined in footnotes. A glossary in the back of this manual (Appendix C) also defines the terms used in this document.

Refresh vs. Store

The standard 4054 contains a 19-inch display that can present stored information as well as refreshed/dynamic information (such as blinking alpha cursor). Alphanumeric and graphic data are displayed in "store mode."[1] The vectors are written on the screen with enough intensity to store them; thus they remain until the screen is erased.

There are, however, situations where it is useful to have images that can be moved, modified, or erased — without erasing and "rewriting"[2] the entire screen. Many design and business applications find "refresh"[3] displays more useful than storage displays for graphing and interactive layout/design work. With a refresh display system the image must be drawn and continually redrawn to remain visible . This allows the image to be updated continuously each time it is redrawn. This fact coupled with a rapid refresh rate allows images to move smoothly across the screen. The effect is similar to that of the frame-by-frame motion produced by movie film.

---

[1]**Store: To draw an image on the screen that stays until erased.**

[2]**Rewrite: Erases the screen and updates it (storage mode).**

[3]**Refresh: Also referred to as "write-thru." Display writing beam intensity is limited to where the image is visible but is not strong enough to cause storage. Stored information remains on the screen.**

The usual kind of refreshed display employs a television-type scanning display tube. While such a system may be suitable for presenting alphanumeric data, it lacks in some graphing and design applications. This is because the resolution (and resulting detail) is limited by gaps between the horizontal scanning lines. However, the 4054's Option 30 uses a non-scanning refresh process. It merely writes and rewrites the image vectors — same as in store mode, but with lower writing intensity. Thus refresh images have the same detail as stored images and both can exist simultaneously on the 4054 display tube. See Figure 1-1. Refresh images can be erased or moved without affecting stored images.

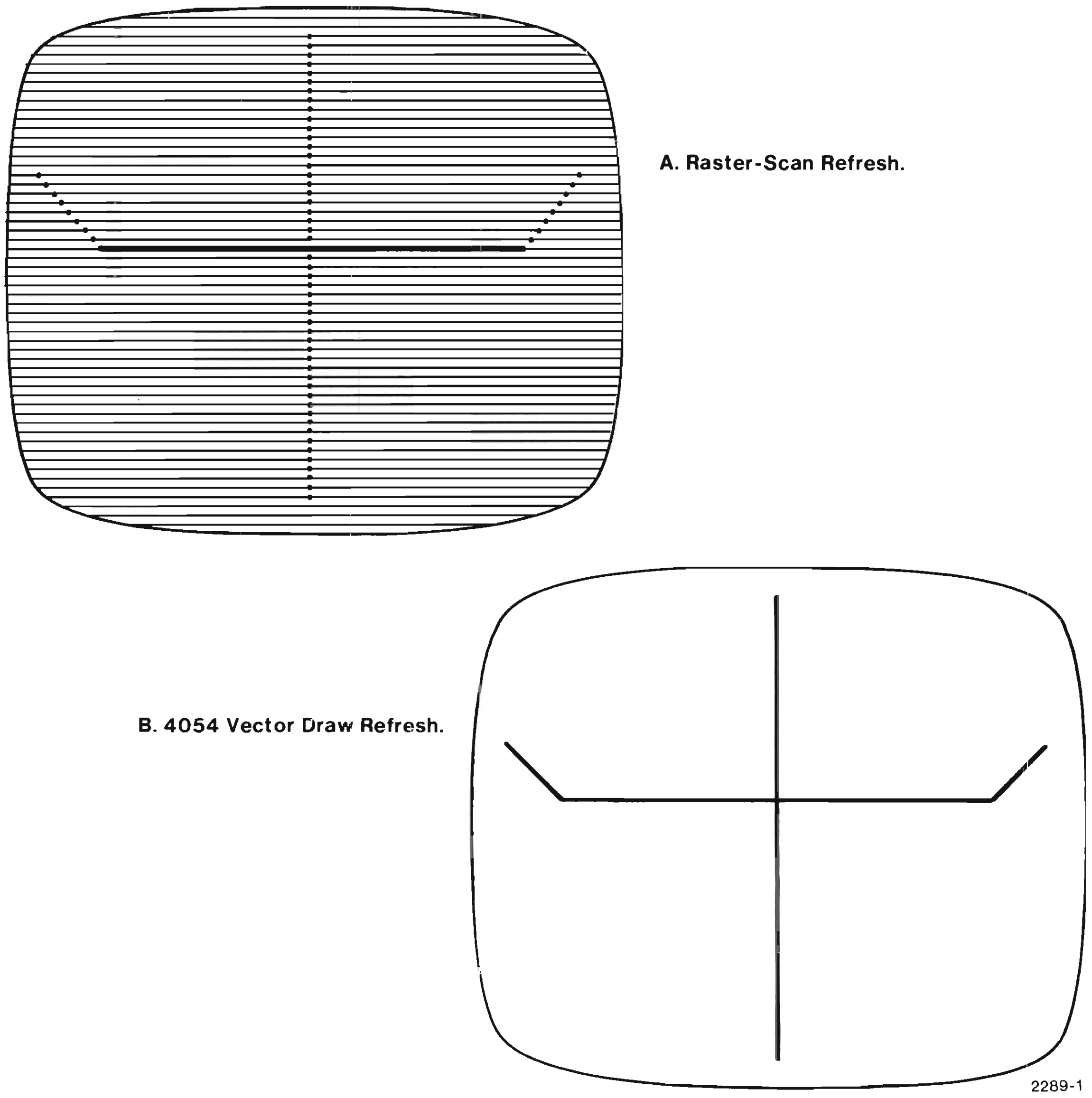


Figure 1-1. Raster-scan vs. 4054 Refresh Displays.

The standard 4054's display already has the ability to write both refreshed and stored images. Option 30, however, provides the processor and memory required to handle more complex refresh images as discrete and controllable screen entities called "objects."[4] An object may be a text string, a picture or symbol, or combination of the two. Some examples of objects are "Dynamic Graphics" text and the "axis pointer" in the Demo-Verification program, later in this section. Also the application program in Appendix B shows flow diagram symbols; each of these is an object, as are the connecting arrows. An object is created in a BASIC language program using the appropriate Dynamic Graphics commands.

The object definition consists of a number of lines in the program and starts with an "opening" (ROPEN) command. This is followed by a statement for each line (vector) and piece of text comprising the object. The definition is "closed" with the RCLOSE command. When the program is run, the Option 30 hardware recognizes these commands and stores the program's object(s) in its memory. The object is then sent to the display where it may be moved, blinked, stored, or otherwise manipulated by Dynamic Graphic commands in your program and/or by operator inputs. The kinds of object motion allowed depends on how your program is written. Refresh objects remain on the screen until an RDELETE or RINIT command is received.

NOTE

After you power up your Option 30 equipped 4054, check the adjustment of the WRITE-THRU INTENSITY Control, under the right end of the keyboard. See Figure 1-2. Set the adjustment so the cursor does not store on the screen.

---

[4]**Object: A set of vectors as text strings, pictures, etc., treated as a single screen entity. May be manipulated via Option 30.**

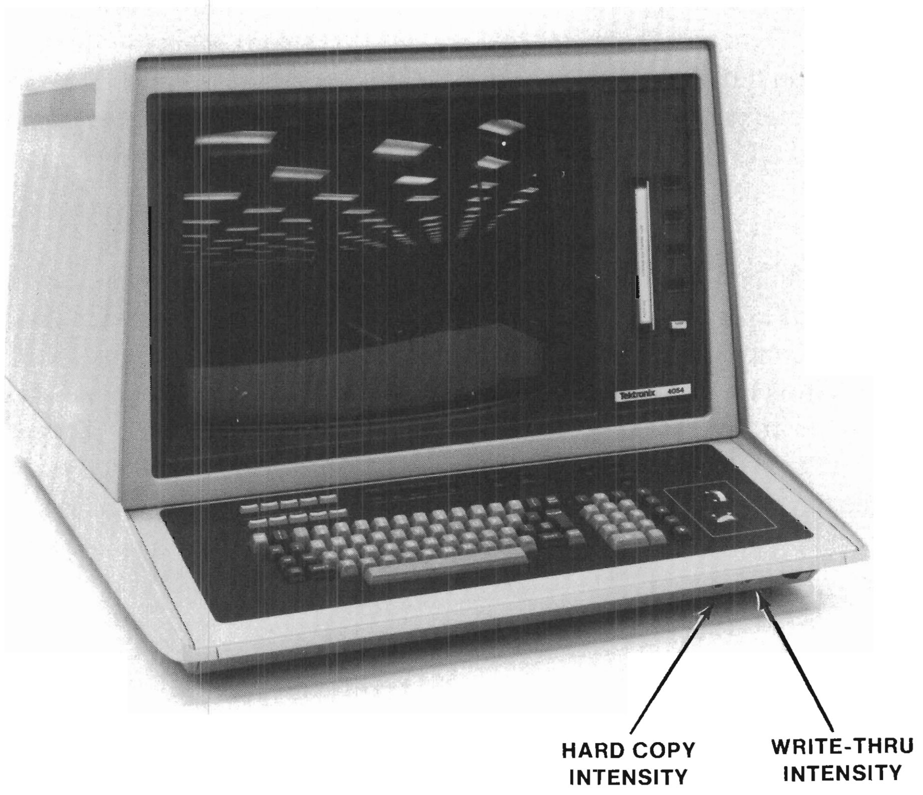


Figure 1-2. Hard Copy and Write-Thru Intensity Controls.

Moving and Fixing Dynamic Objects

Several refreshed objects can be displayed at the same time, and you can write your program so that any one of these objects can be selected to function as a "cursor."[5] As the cursor, the object can be moved around the screen, controlled by the thumbwheels. The thumbwheel set is one type of GIN (Graphic Input) device. Other GIN devices include plotter joystick and graphic tablet pen or puck. Once the object is moved to a desired screen location, you can then use the "FIX"[6] command to store the object at that location. The image of the object is copied on the screen in storage mode. The object can then be moved again and fixed at other locations, thus creating multiple images of the object on the screen. This process is used in "menu picking" applications such as the program in Appendix B. Figure 1-3 depicts a physical plant layout application of this concept.

[5]Cursor: Refreshed image pointing to writing beam location on screen. Alpha cursor (blinking box) indicates next text character position. Crosshair cursor shows graphics context location (X,Y intersection of crosshair lines).

[6]Fix: Stores the object at present screen location using FIX command.

1. Movable object (desk) cursor in refresh.

2. Object stored in desired locations.

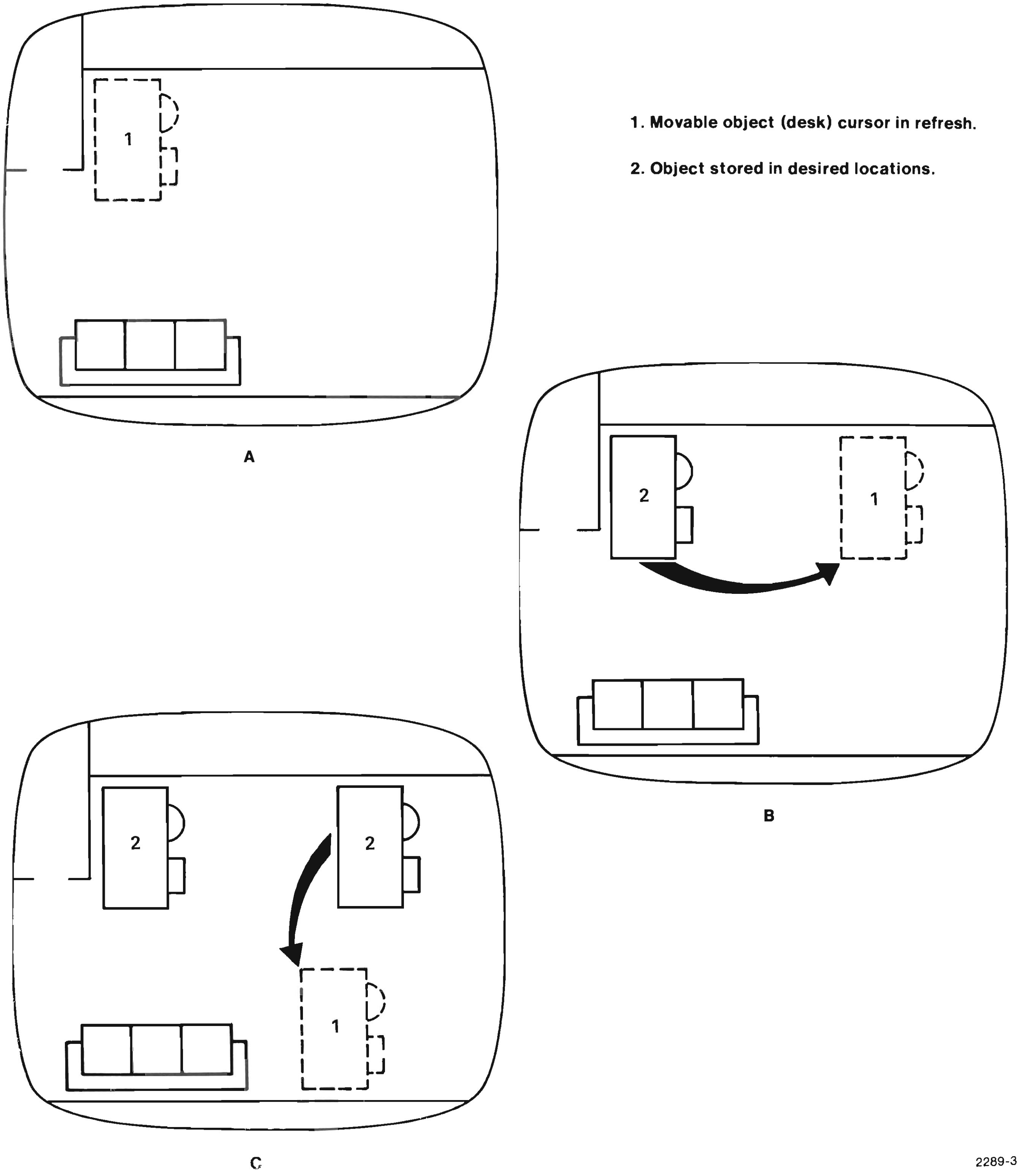


Figure 1-3. Locate/Fix/Move/Repeat.

NOTE

Since the Option 30 memory is write-only, you cannot read the FIX locations from it. You can write programs so that each time a FIX command is executed the object's screen location is sent to an "X,Y location table" in your data base; thus the program keeps a data record of that object's FIX locations.

With the CURSOR command the operator can move the object and immediately see the results. Another way to move an object is to let the program move it using the STPOINT, "setpoint,"[7] command. Setpoint takes an object and moves it to a new X,Y location specified in the STP statement. By using a loop of program statements the object can be setpointed repeatedly in small increments (steps) thus moving the object across the screen. This is one way to achieve "translational"[8] motion for limited animation effects. The STPOINT command will not allow rotational motion. One way to achieve rotation of an object is to create multiple objects, one after another, each one being rotated slightly; and then switch visibilities from one object to the next to imply motion. See Figure 1-4. Another slower method is to recalculate the new coordinates of the rotation and redefine the object repeatedly for each step.

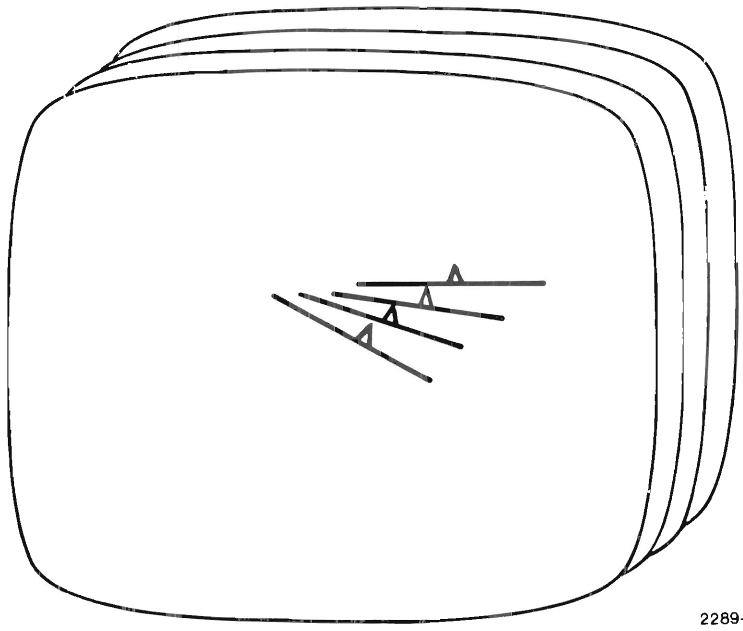


Figure 1-4. Rotation via Multiple Objects.

[7]**Setpoint: Repositions an entire object on the screen by reSETting (moving) its starting POINT location.**

[8]**Translational Motion: Object moves without being rotated.**

PROGRAMMING CONSIDERATIONS

This subsection describes programming concepts for Option 30. The first category deals with programming considerations related to moves and draws.

Moves and Draws

Before creating an object, you should move the beam to the X,Y location that will be the reference position for the object (Figure 1-5). This synchronizes the alpha and graph cursor reference positions. Also, if you are creating an object to be used as a cursor, be sure the outline of the object falls within the current viewport. Otherwise, the object will be clipped, as illustrated in Figure 1-6A.

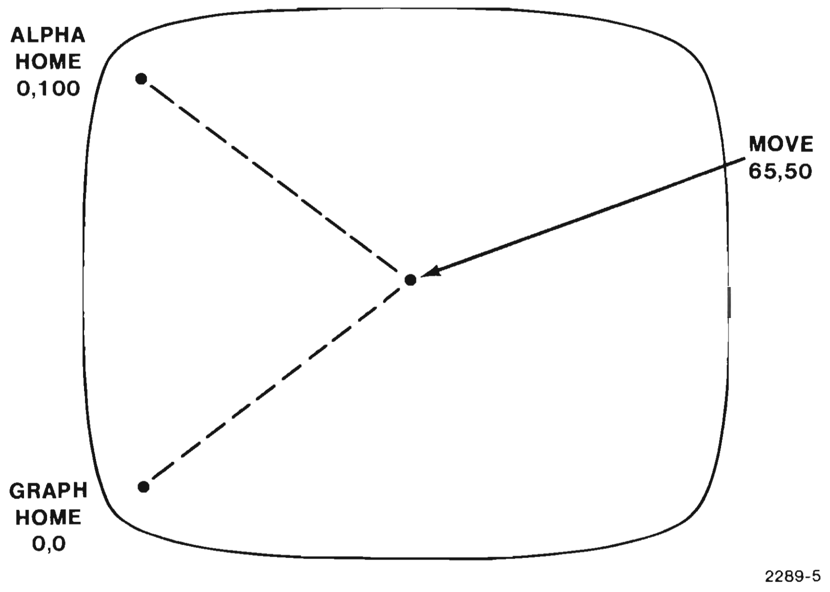


Figure 1-5. Synchronizing Alpha and Graph Reference Positions.

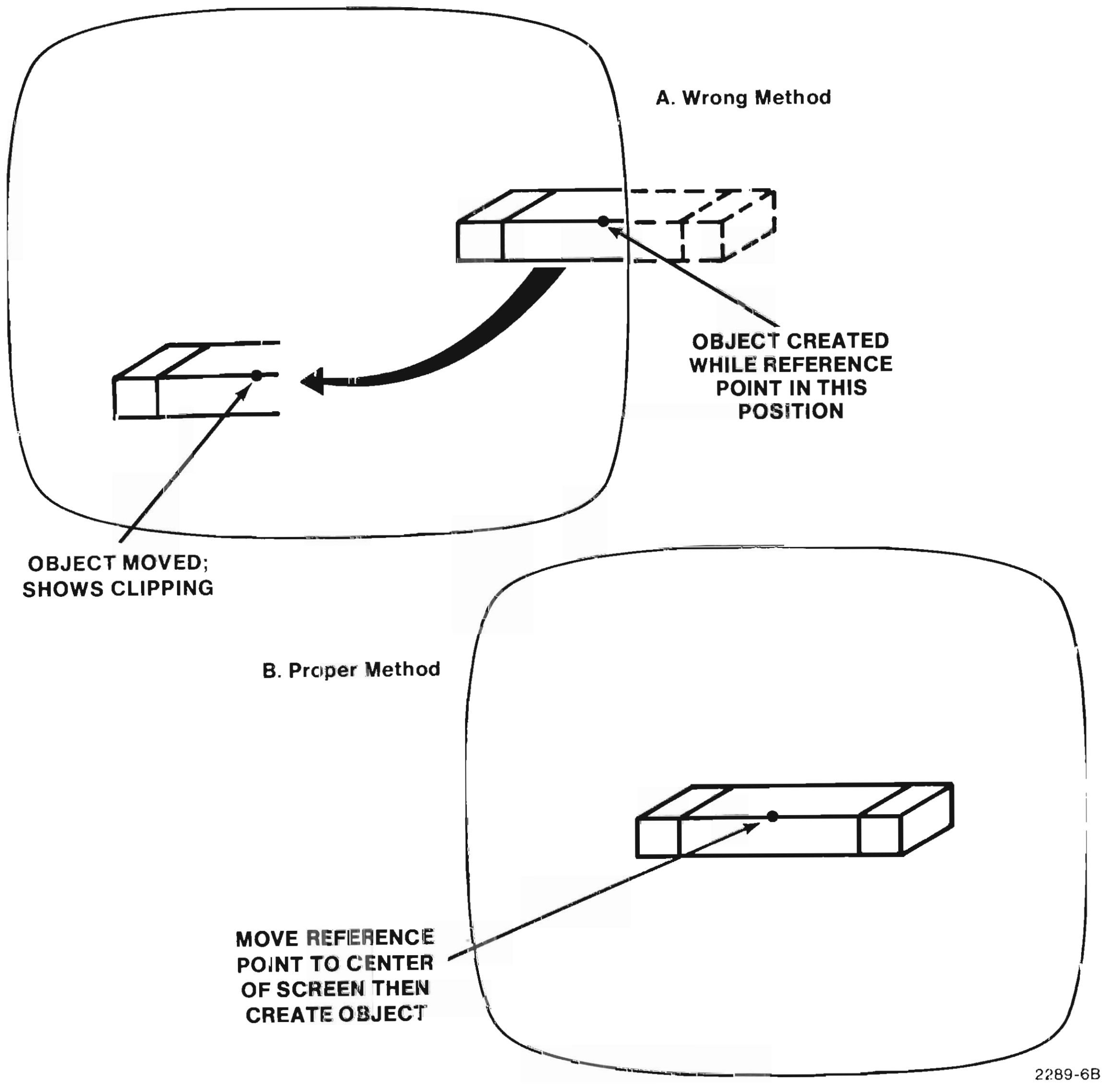

A. Wrong Method
OBJECT CREATED WHILE REFERENCE POINT IN THIS POSITION
OBJECT MOVED; SHOWS CLIPPING
B. Proper Method
MOVE REFERENCE POINT TO CENTER OF SCREEN THEN CREATE OBJECT
2289-6B

Figure 1-6. Creating Cursor Object to Avoid Clipping.

If an object is to be moved off the screen via STPOINT (setpoint), begin the object definition with an RMOVE from a point which will always remain within the viewport. Figure 1-7 illustrates this concept.

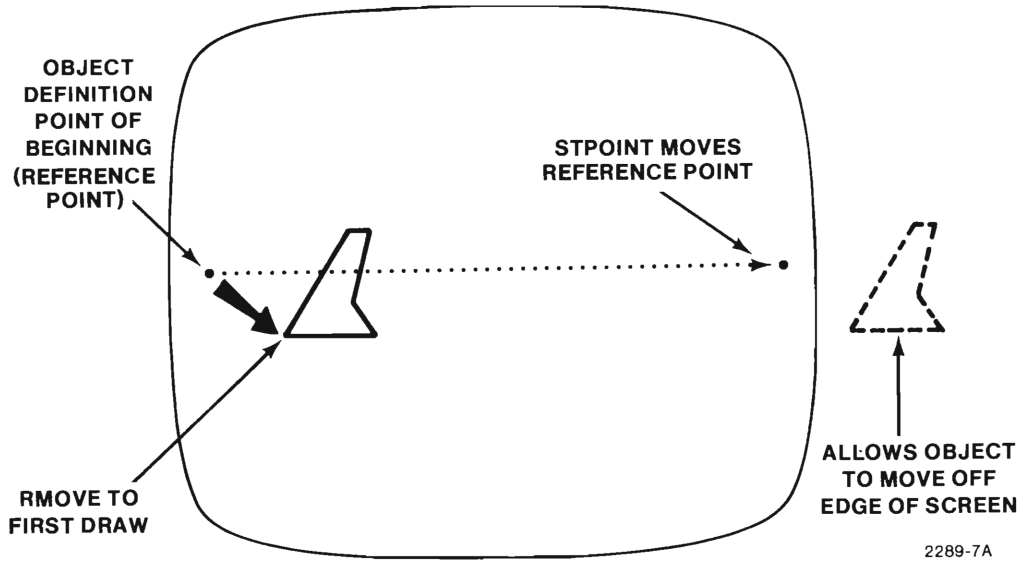

OBJECT DEFINITION POINT OF BEGINNING (REFERENCE POINT)
STPOINT MOVES REFERENCE POINT
ALLOWS OBJECT TO MOVE OFF EDGE OF SCREEN
RMOVE TO FIRST DRAW
2289-7A

Figure 1-7. Moving Object Off Screen.

Use only relative moves (RMOVE) and relative draws (RDRAW) when creating an object; otherwise the object cannot be moved on the screen. CURSOR and STPOINT cannot move an entire object containing absolute moves or draws.

You can include an absolute DRAW after the object is closed. This results in a "rubber band" [9] vector which extends from the object to the fixed DRAW point. This rubber band vector follows any object movements and may be useful in certain applications; see DEMO-VERIFICATION program. Only one rubber band can be attached to a movable object. See Figure 1-8.

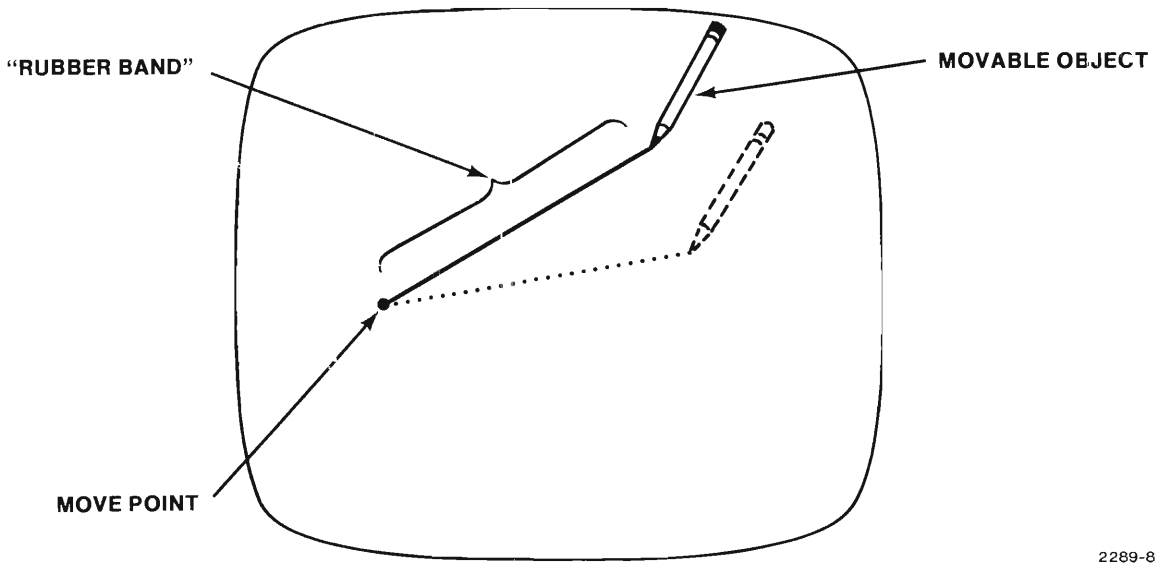


Figure 1-8. Rubber Band Effect.

[9]**Rubber band: A vector that connects a fixed point to a movable point on the screen.**

Memory Management

Option 30 contains 32K of dynamic random access memory. The following information will help you make the best use of this memory space.

Text characters consume large amounts of memory. Therefore only about two lines of text can be displayed in Refresh — as in text editing or graphics labeling.

The RMEMORY and RSPACE functions are useful monitors of Option 30 memory. RMEMORY tells how much unused or available memory remains. RSPACE tells how much memory is already used. The sum of the two should equal 32K unless part of the memory is faulty. If the total is zero, the option has been removed or disabled.

NOTE

Programs with RSP & RME will not run on a 4051 Graphic System because these commands do not have I/O default addresses (i.e., PRINT @ 3X,XX:).

If Option 30 memory fills completely, all additional commands will result in stored instead of refresh vectors.

The standard 32K of memory is more than adequate for even the longer applications programs. You can make better use of Option 30 memory by deleting (RDELETE) objects that are no longer needed. If you need an additional 1114 bytes of memory you may delete system objects numbered 65532 through 65534, if they are not needed. These are listed in Table A-2, Appendix A.

Optimizing User Interface

As a programmer, you can practice certain principles that will make the 4054 with Option 30 easier to use. These principles recognize the human factors of visual perception and also the physical/electrical limitations of the instrument.

In order to simulate rotational motion you must create multiple images, with each object being rotated slightly. The visibility command can be used to help create a smooth motion. This is done by turning on the next object in the sequence before you turn off the last one. This eliminates an obvious jerking or flutter effect.

Another use of the VIS command helps eliminate processor caused "flicker."[10] As your program is running and building objects in memory, the processor is busy displaying these objects on the screen. You will find that larger objects will flicker if they are displayed while being built. This is because the refreshed display is interrupted by processing, hence the flicker. Other refreshed objects may also flicker while the processor is busy. If such flicker is annoying to the user, you can turn off the flickering object by setting its VISIBILITY to zero. When the processor is through building objects, make the objects reappear (set VIS to 1 for each object).

You can create blink effects without using the Option 30 processor and the BLINK command. Since the processor has an implicit minimum on/off time of .02667 seconds, you can write program loops that use multiples of the blink rate to time step operations from one object to another or blink one object. Once the Option 30 processor is set in motion the program and processor can be stopped and the blinking will continue indefinitely.

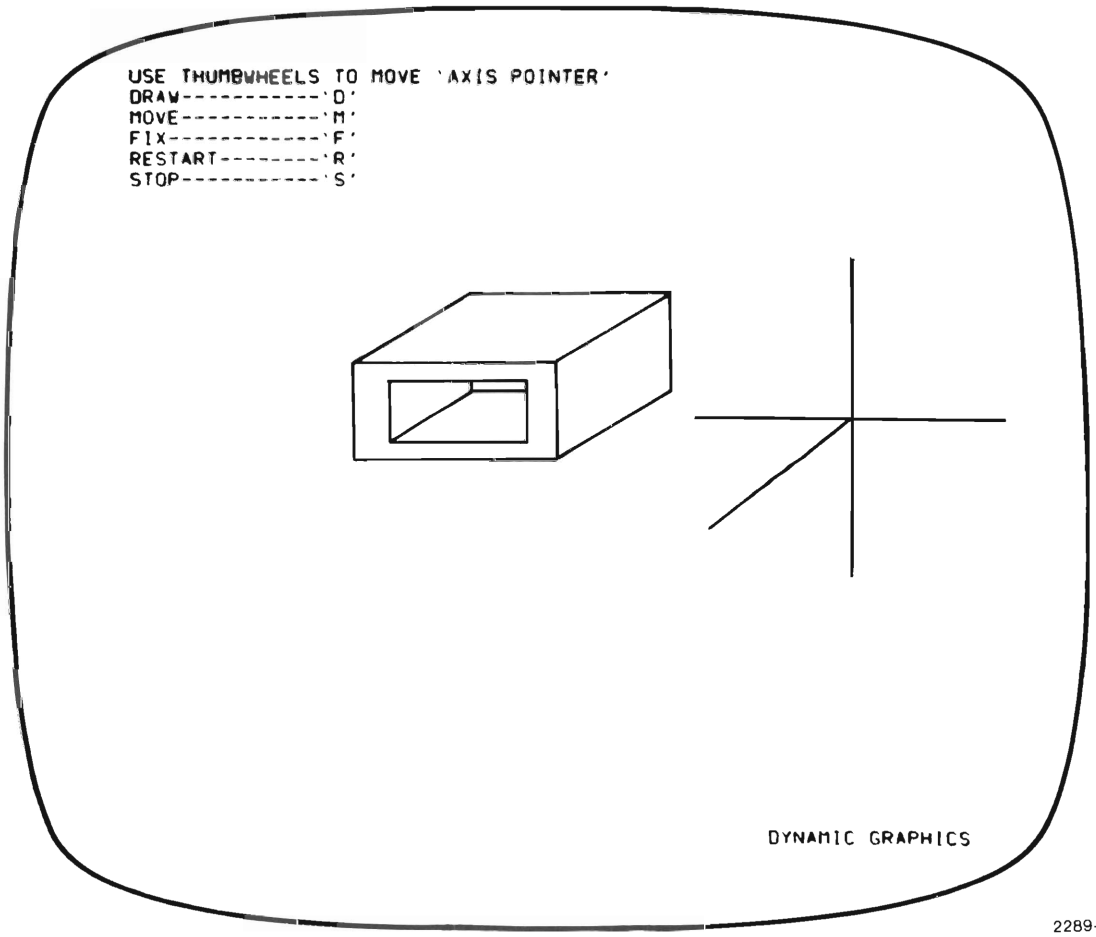
---

[10]**Flicker: Unintentional rapid blinking or twitching of an object caused by interruptions in the refresh cycle.**

DEMONSTRATION-VERIFICATION PROGRAM

In order to get a practical feel for programming the 4054 with Option 30, this subsection presents a typical and relatively simple application program. You should enter the program line-by-line while reading the accompanying text explanation. Then after you see how the commands work in this program, you can go on to develop your own programs. This program demonstrates nearly all of the Dynamic Graphics commands. The program also tells you if your Option 30 is performing properly, and thus may serve as a verification program if saved on tape or disc.

This program allows you to draw on the 4054 screen, using the thumbwheels. The program does this by drawing connecting lines from existing points to the current cursor position. The cursor in this program is a small crosshair object ("axis pointer"). Moves, draws, and fixes are accomplished by pressing certain keys that are listed in the upper left corner on the screen. See Figure 1-9.



```
USE THUMBWHEELS TO MOVE 'AXIS POINTER'
DRAW------------'D'
MOVE------------'M'
FIX-------------'F'
RESTART---------'R'
STOP------------'S'
```

```
DYNAMIC GRAPHICS
```

2289-9

Figure 1-9. DEMONSTRATION-VERIFICATION PROGRAM Screen Presentation.

Figure 1-10 shows the program listing. This discussion follows line- by-line referring to line numbers for each.

```
100 INIT
110 PAGE
120 IF RSPACE+RMEMORY=0 THEN 140
130 GO TO 160
140 PRINT "OPTION 30 NOT INSTALLED"
150 GO TO 680
160 ROPEN 1
170 PRINT "USE THUMBWHEELS TO MOVE 'AXIS POINTER'"
180 PRINT "DRAW-----------'D'"
190 PRINT "MOVE-----------'M'"
200 PRINT "FIX------------'F'"
210 PRINT "RESTART--------'R'"
220 PRINT "STOP-----------'S'"
230 RCLOSE
240 ROPEN 2
250 PRINT "DYNAMIC GRAPHICS"
260 RCLOSE
270 FOR L=0 TO 84 STEP 0.1
280 STPOINT 2,L,10
290 NEXT L
300 BLINK 2,0.8,0.2
310 X=65
320 Y=50
330 MOVE X,Y
340 ROPEN 3
350 VISIBILITY 3,0
360 RMOVE 20,0
370 RDRAW -40,0
380 RMOVE 20,-20
390 RDRAW 0,40
400 RMOVE 0,-20
410 RCLOSE
420 RAPPEND 3
430 DRAW X,Y
440 RCLOSE
450 X1=X
460 Y1=Y
470 CURSOR 3
480 POINTER X,Y,Z$
490 IF Z$="D" THEN 540
500 IF Z$="F" THEN 570
510 IF Z$="R" THEN 600
520 IF Z$="S" THEN 660
530 GO TO 330
540 MOVE X1,Y1
550 DRAW X,Y
560 GO TO 330
570 STPOINT 3,X,Y
580 FIX 3
590 GO TO 330
600 ROPEN 4
610 PRINT "AVAILABLE MEMORY=",RMEMORY
620 RCLOSE
630 RREPLACE 2,4
640 PAGE
650 GO TO 310
660 RINIT
670 PAGE
680 END
```

2289-10

Figure 1-10. DEMONSTRATION-VERIFICATION PROGRAM listing.

100 & 110	Initializes 4054 main memory. Sets alpha cursor to 0,100 and erases display screen.
120 to 150	Checks to see if Option 30 is installed. If so, goes on the 160; if not, prints "OPTION 30 not installed" and goes to 680 (END).
160	REOPEN 1 opens object 1.
170 to 220	These six lines of text comprise Object 1 — written on the screen via PRINT statements. Object appears in the upper left corner of the screen and tells which keys are used for draw, move, fix, restarting and ending the program.
230	Closes (completes) object 1.
240	Opens second object (2).
250	Object consists of a text string "DYNAMIC GRAPHICS."
260	Closes object 2.
	The next four program lines cause Object 2 to move across the lower part of the screen from left to right, then stop and blink near the right edge of the screen.
270	This line makes the variable, L, have a range of 0 to 84, with an increment of 0.1 UDUs.[11]
280 & 290	The setpoint statement includes three numeric variables: The first, 2, defines the object to be moved by the STPOINT command. The next variable is the X location to which the object is moved. "L" was defined in line 270 so the object will move in 0.1 steps for each L until it has reached 84 (UDU) on the right side of the screen. (The smaller these steps, the smoother the motion and the longer the transit time.) The last variable is the Y location, which is a constant 10 units up from the bottom of the screen. Once the object stops at X= 84, the program advances to 310.

---

[1]**UDU — User defined units.**

300	This line blinks Object 2. The second and third variables in the statement set the on time to 0.8 seconds and the off time to 0.2 seconds. This blinking continues until the blink rate is reset or the object is RDELETEd.
310 & 320	Sets X and Y starting values for future moves.
330	Moves the graphic cursor to 65,50.
340	Opens object 3, the movable crosshair/axis.
350	Sets the visibility of object 3 to invisible.
360 to 400	These moves and draws (all relative) create the crosshair/"axis pointer," Object 3.
410	Closes (completes definition of) object 3.
420 to 430	Draw X,Y (rubber band to 65,50) is appended to object 3.
440	Closes the appended object.
450 to 460	Make X1 and Y1 equal to X and Y.
470	Redefines the graphic cursor to be object 3 ("axis pointer").
480	Pointer statement places the graphic cursor (now Object 3) on the screen. The object can be moved around the screen using the thumbwheels. Notice, it is rubber banded to the fixed point 65, 50. The Z$ is a variable which is set equal to any key pressed while the POINTER statement is being executed. Thus operator input can control the next program step. The POINTER statement also sets the variables X and Y equal to the coordinates of the cursor object at the time the key was pressed.
490	Works with the pointer statement to draw a line when key, D, is pressed.
500	Works with the pointer statement to record X,Y location and FIX the object when the F key is pressed.
510	Restarts the program via lines 600 — 650 where available dynamic memory is displayed.

520	Ends program by jumping to line 660.
530	If any key is pressed, other than those listed, the program assumes a move and returns to 330 for next move.
540 to 560	Moves the writing beam to location X1, Y1 and draws from there to the current cursor location at X, Y; then returns to 330 for next move.
570	Setpoints (moves) the crosshair object to X, Y.
580 & 590	Fixes (stores) the crosshair at that screen location; then returns to 330 for next move.
600 & 610	Opens Object 4, which prints the available number of bytes in Option 30 memory. This object is created and displayed only, following a program restart (press key "R").
620	Closes object 4.
630	Replaces object 2 (DYNAMIC GRAPHICS) with object 4 (AVAILABLE MEMORY = XXXXX) upon restart.
640 & 650	As part of restart, the screen is erased (PAGE) and program returns to line 310, where you begin drawing again.
660 to 680	Coming from line 520, the program ends by initializing Option 30 memory (RINIT), and erasing the screen (PAGE).

Section 2

COMMAND DESCRIPTIONS

INTRODUCTION

This section describes the 13 commands and 2 functions unique to Option 30 (arranged alphabetically). The explanations follow the same general format found in the 4050-Series Graphic System Reference Manual. The syntactic and descriptive forms of the commands are enclosed in a box for easy recognition. A brief purpose statement is followed by a more detailed explanation. Examples are included in some of the explanations. The sample Demonstration-Verification program in Section 1 also shows how these commands may be used.

THE BLINK COMMAND

SYNTAX FORM

<Line number> BLI numeric expression, numeric expression, numeric expression

DESCRIPTIVE FORM

<Line number> BLINK object number, on time, off time

Purpose

The BLINK command alternately displays the specified object for a period of time and stops displaying it during the off time.

Explanation

First make the object visible (VIS= 1), then set the on and off blink times by entering the number of seconds for each.[1] The maximum on or off time is approximately 6.8 seconds. The maximum blink rate (shortest allowed on/off time) is 0.02667 seconds/cycle.

NOTE

Large objects (800 vectors or more) will blink at a slightly slower rate than set by the on/off times in the BLINK statement. Also, setting blink 'on' time to less than .02667 seconds will cause the object to appear 'on' all the time (instead of 'off' as one might expect).

Blink is cancelled by: BLI n,0,0.

---

[1]**A VIS=0 over-rides the BLINK command, so you will not see the blinking object until VIS=1.**

THE BRIGHTNESS COMMAND

SYNTAX FORM

<Line number> BRI numeric expression

DESCRIPTIVE FORM

<Line number> BRIGHTNESS display code

Purpose

The BRIGHTNESS command sets one of two brightness levels for all refreshed objects appearing on the display.

Explanation

The BRIGHTNESS command used with Dynamic Graphics Option actually specifies the normal brightness level in either the focused or non-focused state. The codes used with this command are as follows:

Display Code	Intensity	Focus	Affected Displays
0	Normal	Defocused	Refreshed/Stored
1	Normal	Focused	Refreshed/Stored
2	Bright	Defocused	Stored only
3	Bright	Focused	Stored only

The defocused lines in characters and vectors appear wider than focused lines and seem brighter. The default setting, after INIT statement, is 1 (focused).

THE CURSOR COMMAND

SYNTAX FORM

<Line number> CUR numeric expression

DESCRIPTIVE FORM

<Line number> CURSOR object number

Purpose

The CURSOR command causes the specified object to replace the crosshair cursor. As the cursor, this object can be moved about the screen with the thumbwheels or other GIN devices.

Explanation

The CURSOR command causes the specified object to be used as the graphic input cursor. The POINTER command displays the GIN cursor. If a CURSOR command does not precede the POINTER command, a CURSOR 0 is automatically issued and the GIN cursor is the default full screen crosshair.

Objects with absolute DRAW and MOVE statements have limited cursor movement. If one DRAW or MOVE is present, all previous parts of the object move as the cursor with a "rubber band" vector extending to that DRAW or MOVE position. All vectors in the object after the absolute DRAW or MOVE are not repositioned by the GIN device.

NOTE

Initial point of object should be centered on screen if you intend to use object as a cursor. The cursor cannot move off screen; so if object must go off screen, use an RMOVE from the initial point in the definition to adjust object toward that edge.

THE DASH COMMAND

SYNTAX FORM

<Line number> DAS numeric expression

DESCRIPTIVE FORM

<Line number> DASH dash pattern

Purpose

The DASH command specifies how refreshed object vectors (RDRAW) are displayed. These vectors are displayed as: continuous lines, dashed lines, or dotted lines according to the dash pattern entered with this command.

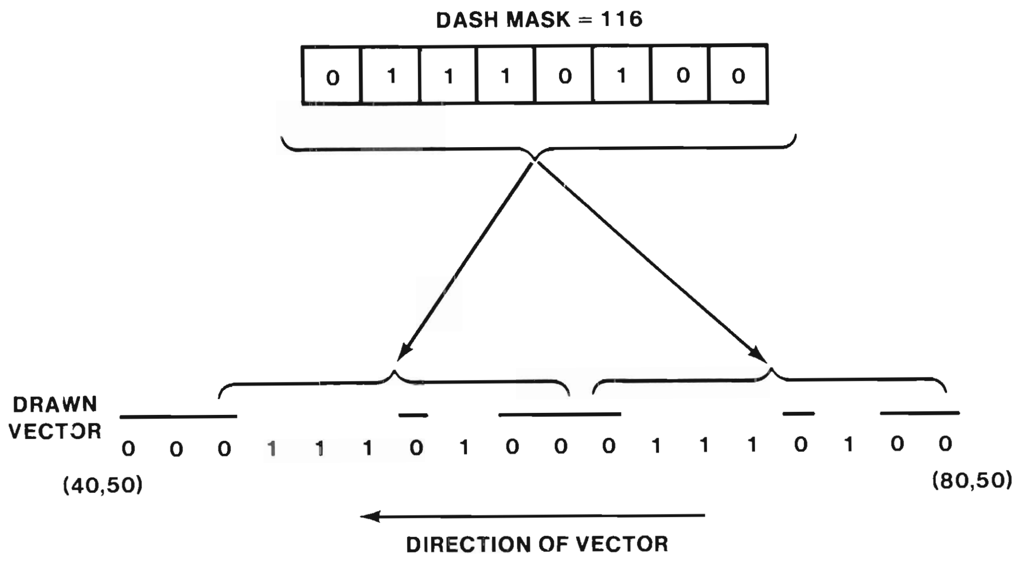
Explanation

The DASH command controls all objects drawn after the command is issued. The dash pattern is also called a "dash mask" and is an integer between 0 and 255. The binary equivalent of this number/mask is an 8-bit number whose pattern determines which parts of a vector are displayed and which are not visible. A bit 0 corresponds to visible vector segment, while a bit 1 means that the segment is not visible.

NOTE

Do not confuse visibility of dash line segments (0=visible) with VIS command (1=visible).

The following example shows how this mask is interpreted. Suppose the dash mask is 116 (= 01110100 in binary), and a vector is drawn from 80,50 to 40,50. The vector is drawn as follows:



PATTERN	DASH MASK NUMBER	BINARY EQUIVALENT
————————	3	00000011
— — — —	170	10101010
— — — —	85	01010101
—————	15	00001111
(no vector is drawn)	255	11111111
—— ————	200	11001000

2289-11A

Figure 2-1. Dash Mask Examples.

If a bit is 0 in the dash mask, the corresponding segment is drawn (visible); if a bit is 1, that segment is not drawn. The pattern is repeated as many times as needed to draw the vector. A DRAW or RDRAW does not reset the dash pattern to the beginning. Reissuing the DASH command will reset its starting point. See the 4050-Series Graphic System Reference Manual for more examples of this command.

THE FIX COMMAND

SYNTAX FORM

<Line number> FIX numeric expression

DESCRIPTIVE FORM

<Line number> FIX object number

Purpose

The FIX command draws the specified refreshed object in storage mode at its present screen location. The refreshed object remains superimposed and may be moved to a new screen location; the stored image of that object does not move.

Explanation

By using the FIX command objects can be stored on the screen. Since the refreshed image remains, multiple FIXs can be performed on the same object as it is moved from one location to another, using the POINTER command.

If needed, your programs can be written to send the GIN coordinates of a fixed object to a "store matrix" in your data file. The Option 30 memory is write-only so FIX locations cannot be read from it.

The object is fixed irrespective of its current visibility. You can FIX an invisible object, which will make it visible as a stored image, but the refreshed image remains invisible.

THE RAPPEND COMMAND

SYNTAX FORM

< Line number> RAP numeric expression

DESCRIPTIVE FORM

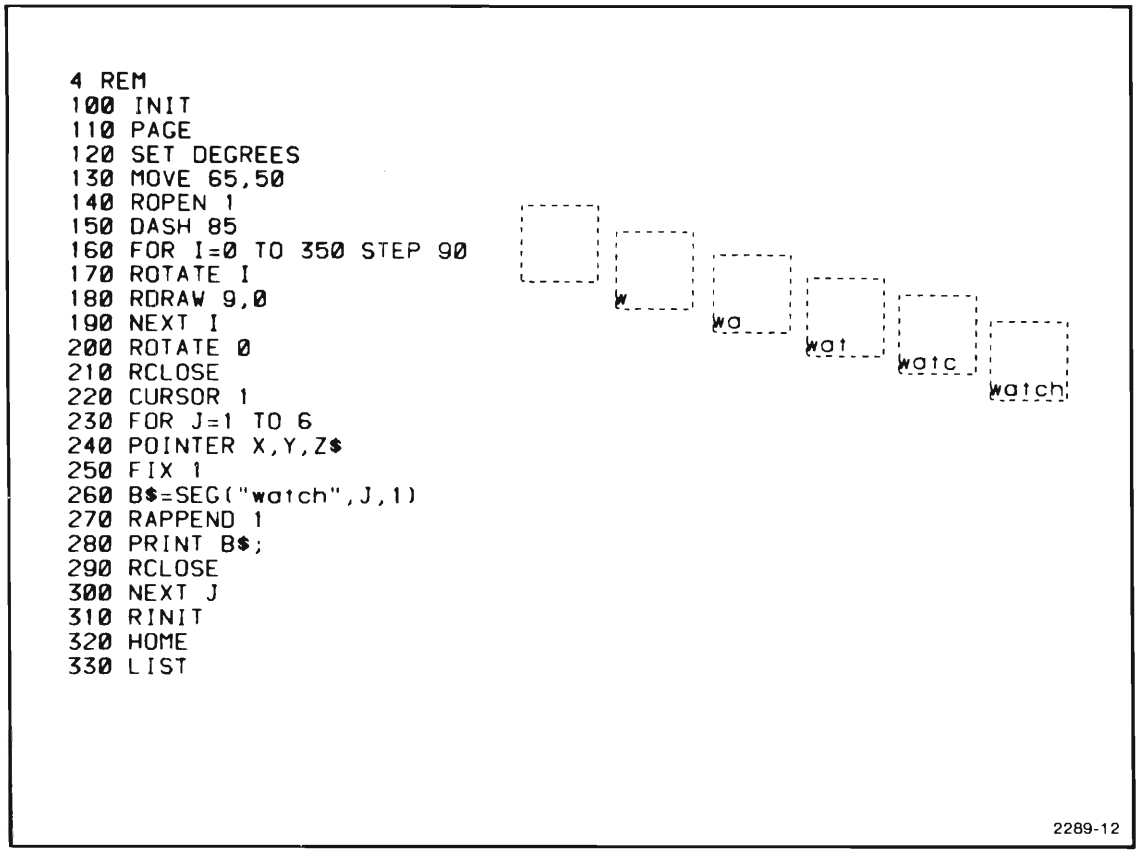
< Line number> RAPPEND object number

Purpose

The RAPPEND (Refresh Append) command allows you to add vectors to an existing object. Unlike the ROPEN command, RAPPEND allows you to append vectors to an existing object without destroying its contents first.

Explanation

The sample program in Figure 2-2 illustrates the use of the RAPPEND command. Here repeated appends are made to an object (dashed box). Each successive RAPPEND adds a letter from the word "watch" to the object, which is moved and redrawn. Notice that the object is not recreated on each pass, since RAPPEND does not destroy it. Also notice that the object is updated on each pass and retains the already appended letters.



```
4 REM
100 INIT
110 PAGE
120 SET DEGREES
130 MOVE 65,50
140 ROPEN 1
150 DASH 85
160 FOR I=0 TO 350 STEP 90
170 ROTATE I
180 RDRAW 9,0
190 NEXT I
200 ROTATE 0
210 RCLOSE
220 CURSOR 1
230 FOR J=1 TO 6
240 POINTER X,Y,Z$
250 FIX 1
260 B$=SEG("watch",J,1)
270 RAPPEND 1
280 PRINT B$;
290 RCLOSE
300 NEXT J
310 RINIT
320 HOME
330 LIST
```

Figure 2-2. RAPPEND Example.

RAPPEND operates only in BASIC program (run) mode.

THE RCLOSE COMMAND

SYNTAX FORM

<Line number> RCL

DESCRIPTIVE FORM

<Line number> RCLOSE

Purpose

The RCLOSE (Refresh Close) command closes (completes the definition of) the object which is currently open.

Explanation

The RCLOSE command tells the Dynamic Graphics Option that you are through defining an object. Those vectors listed between an ROPEN (or RAPPEND) and the next RCLOSE become the definition of the specified object. No object number is needed for an RCLOSE because it always refers to the current open object.

An ROPEN, RAPPEND, END, STOP, and pressing the BREAK key, do an implied RCLOSE if an object is open. This means that only one object can be open at a time.

THE RDELETE COMMAND

SYNTAX FORM

<Line number> RDE numeric expression

DESCRIPTIVE FORM

<Line number> RDELETE object number

Purpose

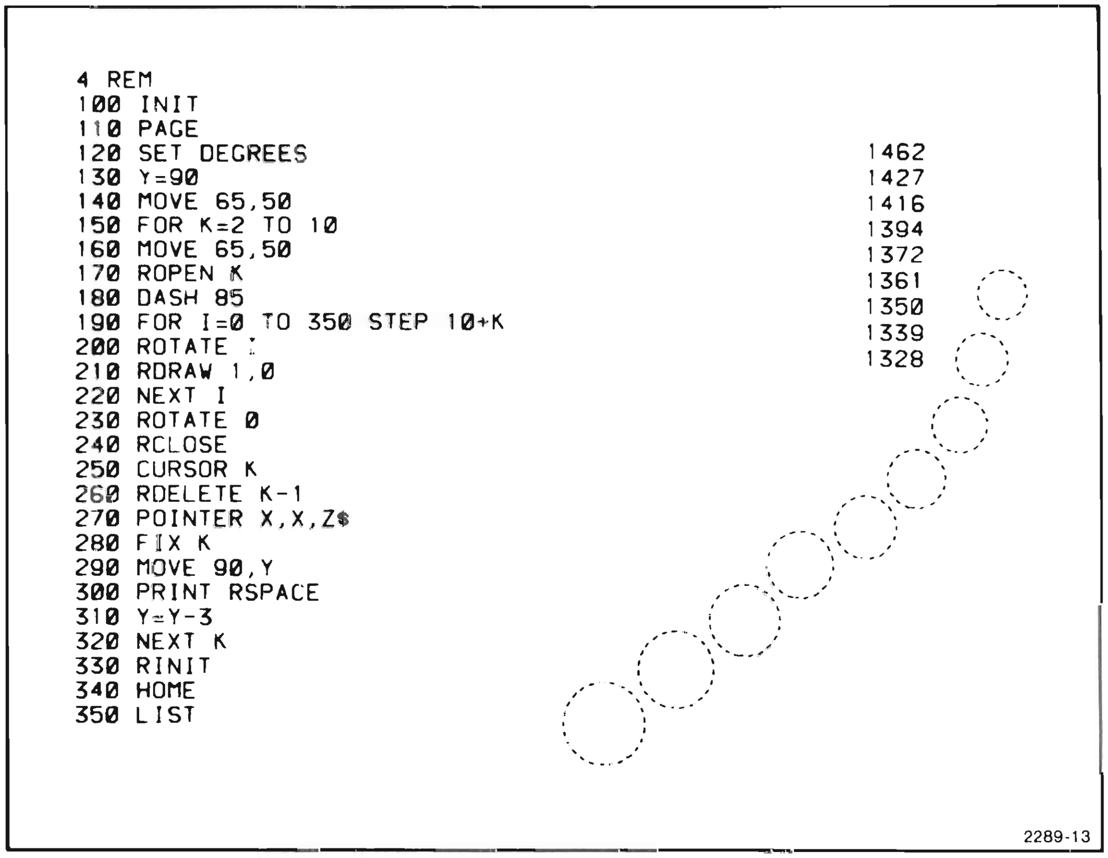
The RDELETE (Refresh Delete) command deletes the specified object from the Dynamic Graphics Memory. The object no longer exists and the Option 30 memory is freed for other objects.

Explanation

The RDELETE command is used to remove an object from the display screen and from the Option 30 memory only.

Your applications program using refresh is first stored in the 4054's main memory. When the program is run, objects are created and loaded into Option 30 memory and displayed on the screen. When you are through with an object, you (or the program) can RDELETE it, making room for other objects. Later, when you rerun the program these objects remain because they were retained by the stored program.

Line 260 of the sample program in Figure 2-3 shows an interesting use of the RDELETE command. The program creates a new object, K, beginning with line 170. Line 180 sets the dash pattern. Lines 190 to 230 draw the circle. Line 190 shows the size of the circular object depends on K (STEP 10+ K). Line 260, RDELETE K-1, deletes the previous object when it is no longer needed. K-1 makes each new object smaller. Line 30 prints the memory space occupied by the present object; it shows the space getting smaller for each new object.



```
4 REM
1ØØ INIT
11Ø PAGE
12Ø SET DEGREES
13Ø Y=9Ø
14Ø MOVE 65,5Ø
15Ø FOR K=2 TO 1Ø
16Ø MOVE 65,5Ø
17Ø ROPEN K
18Ø DASH 85
19Ø FOR I=Ø TO 35Ø STEP 1Ø+K
2ØØ ROTATE I
21Ø RDRAW 1,Ø
22Ø NEXT I
23Ø ROTATE Ø
24Ø RCLOSE
25Ø CURSOR K
26Ø RDELETE K-1
27Ø POINTER X,X,Z$
28Ø FIX K
29Ø MOVE 9Ø,Y
3ØØ PRINT RSPACE
31Ø Y=Y-3
32Ø NEXT K
33Ø RINIT
34Ø HOME
35Ø LIST
```

```
1462
1427
1416
1394
1372
1361
135Ø
1339
1328
```

2289-13

Figure 2-3. RDELETE Example.

THE RINIT COMMAND

SYNTAX FORM

<Line number> RIN

DESCRIPTIVE FORM

<Line number> RINIT

Purpose

The RINIT (Refresh Initialize) command erases (deletes) ALL objects from Option 30 memory, and recreates the system objects.

Explanation

The RINIT command initializes the Option 30 memory, only. The command deletes all system objects except those not created by the user (i.e., crosshair — 65532, "?" — 65533, and refresh print character — 65534). 31590 bytes of memory are made available.

RINIT does not reset the dash pattern or brightness.

The INIT command also performs an implied RINIT.

THE RMEMORY FUNCTION

SYNTAX FORM

< numeric variable= > RME

DESCRIPTIVE FORM

< numeric variable= > RMEMORY

Purpose

The RMEMORY (Refresh Memory) function returns the number of bytes of unused or AVAILABLE memory in the Option 30.

Explanation

The RME function is simply used to monitor the Option 30 memory.

The RME may be treated as a variable, sending its information to the program instead of the display screen. The following program line function illustrates this feature:

```
120 IF RME > 1000 THEN 170

        or

120 A= RME

130 IF A > 1000 THEN 170
```

Here the program checks for available memory. If there is enough room to create a given object (1000 bytes), it proceeds to line 170 and does so.

RME has no I/O default address (such as 3X,XX:), and will not work in a 4051 BASIC program.

THE ROPEN COMMAND

SYNTAX FORM

<Line number> ROP numeric expression

DESCRIPTIVE FORM

<Line number> ROPEN object number

Purpose

The ROPEN (Refresh object Open) command opens an 'object file' and allows you to start creating an object with a given identification ("object") number.

Explanation

This command opens an object file in Dynamic memory under the specified object number, n. If an object with the same number already exists, an ROPEN n will destroy the existing object. All statements following the ROPEN command and preceding RCLOSE (such as: CHA 1; RDR 10,20; PRINT "X") become part of the object.

Construction of an object must occur in BASIC program mode, not immediate mode.

Option 30 allows the following object numbers. Some are user definable while others are system defined upon power-up or RINIT.

- User-defined — 1 through 65531
- System objects — 0 and 65532 through 65535

See Table A-2, SYSTEM DEFINED OBJECTS (Appendix A).

NOTE

Reference to an illegal object number (i.e., 4.1) results in an error message number 88.

THE RREPLACE COMMAND

SYNTAX FORM

<Line number> RRE numeric expression, numeric expression

DESCRIPTIVE FORM

<Line number> RREPLACE object number 1, object number 2

Purpose

RREPLACE (Refresh Replace) deletes existing object 1 and old object 2 becomes new object 1. Original object 1's display parameters (location blink rate, brightness, etc.) remain in effect for the new object.

Explanation

When the new object replaces the old, the result is an object with the definition of the new and the environmental/display parameters of the old. The definition of the old is deleted from Option 30 memory.

The effects of an RREPLACE command are depicted in Figure 2-4. The location and blinking attributes of the old object 1 are transferred to the new object.

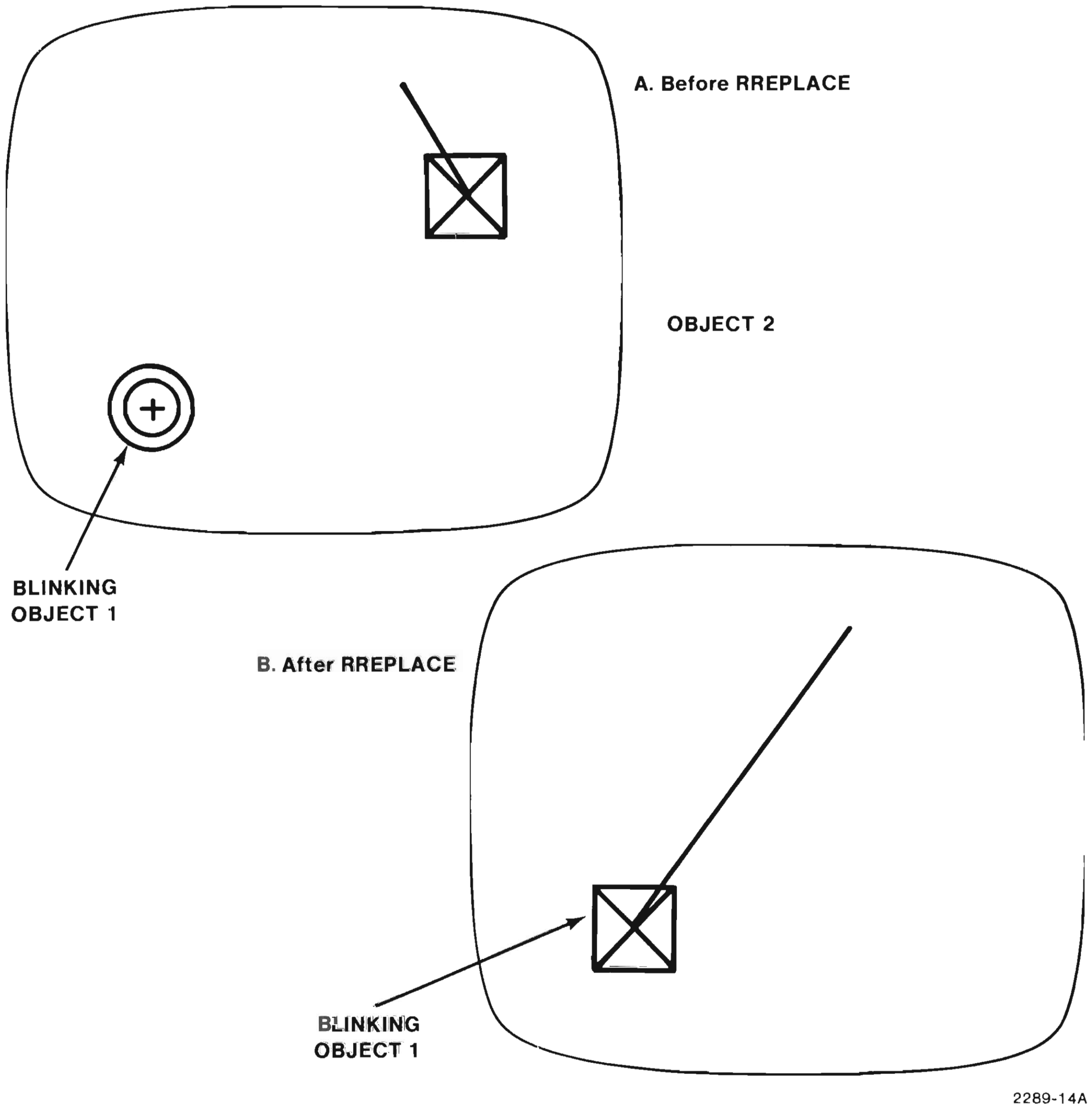


Figure 2-4. Effects of RREPLACE Command on Displayed Objects.

THE RSPACE COMMAND/FUNCTION

SYNTAX FORM

<numeric variable= > RSP

DESCRIPTIVE FORM

<numeric variable= > RSPACE

Purpose

RSPACE (Refresh Space) function returns the number of bytes of OCCUPIED Option 30 memory space.

Explanation

This function, like RMEMORY, is used to monitor Option 30 memory. The command can be entered in immediate mode and the amount of occupied dynamic memory will be displayed on the screen.

The RSPACE function may also be treated as a variable, sending its information to the program instead of the display screen. The following program steps illustrate this.

```
100 IF RSP = 32768 THEN 530

530 PRINT "DYNAMIC MEMORY FULL"
```

Here the program monitors the occupied memory, and when it is full, the program informs the operator of this condition by using a PRINT statement.

If no objects are added to Option 30 memory after power-up, the occupied space will consist of objects 0, 65532, 65533, 65534, and 65535. An RSPACE should then indicate 1114 bytes of occupied space.

NOTE

RME + RSP + 64 should always add up to 32768 bytes of memory in a properly functioning, current model, Option 30. The 64 extra bytes are due to Option 30 overhead.

RSPACE has no I/O default address (such as 3X,XX:) and will not work in 4051 BASIC programs.

THE STPOINT COMMAND

SYNTAX FORM

<Line number> STP numeric expression, numeric expression, numeric expression

DESCRIPTIVE FORM

<Line number> STPOINT object number, X location, Y location

Purpose

The STP (Set Point) command moves the reference point (starting point) of the specified object to a new screen location given by the X,Y part of the STP statement.

Explanation

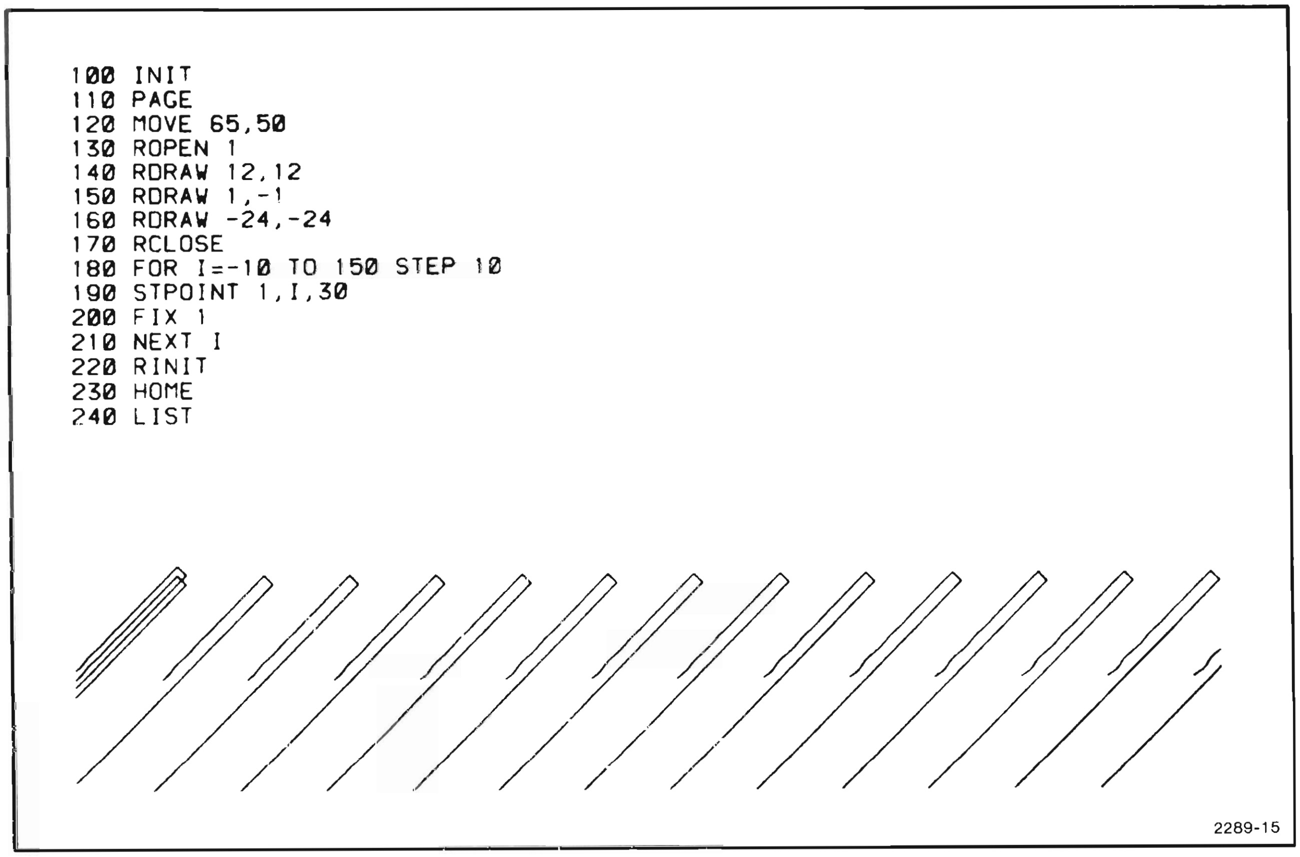
This command allows a program to move a previously defined object. The object must be made up only of relative moves or draws (RMOVE, RDRAW). If the object definition contains absolute moves or draws, STP cannot operate without distorting the object

The STP command operates by moving an object's reference point, which is the first point in the object's definition, to the new location specified in the command. The rest of the object of course moves with the reference point. Since you specify the new location in absolute coordinates, the command cannot move an object's reference point off the screen; therefore begin an object definition with an RMOVE from a point that can remain on the screen if the complete object must be moved off the screen. Refer back to Figure 1-7.

You can achieve a rubber band effect between a fixed point and an object relocated with the STP command. Although an object used with STP should not be defined initially with absolute moves or draws, an absolute draw can be appended to the object. This is illustrated in the DEMONSTRATION-VERIFICATION program earlier in this manual. The appended draw to a fixed point produces a line (rubber band) that stretches to the last defined point in the object.

To effectively use the STP command, you may make use of repeating variables such as in the sample program in Figure 2-5. Line 180 defines the X-axis location variable, I. Here −10 to 150 specifies the X-axis range of setpoint motion, and STEP 10 specifies the space between repeated objects by incrementing the value of I each time STP moves the object.

To show a rapid, smooth, non-stored setpoint motion of an object, refer to lines 270 through 290 of the DEMONSTRATION-VERIFICATION program earlier in this manual.



```
100 INIT
110 PAGE
120 MOVE 65,50
130 ROPEN 1
140 RDRAW 12,12
150 RDRAW 1,-1
160 RDRAW -24,-24
170 RCLOSE
180 FOR I=-10 TO 150 STEP 10
190 STPOINT 1,I,30
200 FIX 1
210 NEXT I
220 RINIT
230 HOME
240 LIST
```

Figure 2-5. STPOINT Example.

THE VISIBILITY COMMAND

SYNTAX FORM

<Line number> VIS numeric expression, numeric expression

DESCRIPTIVE FORM

<Line number> VISIBILITY object number, display argument (0/non0)

Purpose

The VISIBILITY command lets you control the visibility of the specified objects on the display screen.

Explanation

An object's visibility can easily be switched on or off using this command. The display argument (parameter) following the object number is either zero or non-zero:

- 0 — sets the object NOT visible.
- any non-0 — sets this object visible.

There are no gradations of visibility; only on and off.

NOTE

The VISIBILITY command, and any command using an object number, assumes that the object has been closed.

The visibility of an object can also be controlled by the BLINK command. This command may also be used to make an object visible during the object build (eliminating flicker); include after ROPEN.

Appendix A

CHARACTERISTICS

PERFORMANCE

Option 30 operates in conjunction with the 4054 central processor and display circuits. The number of vectors displayable without flicker depends on the length of the vectors. Text displayable is related to the number of strokes in each character. This means the instrument's performance can be specified only in terms of user-specified tasks. See Figure A-1.

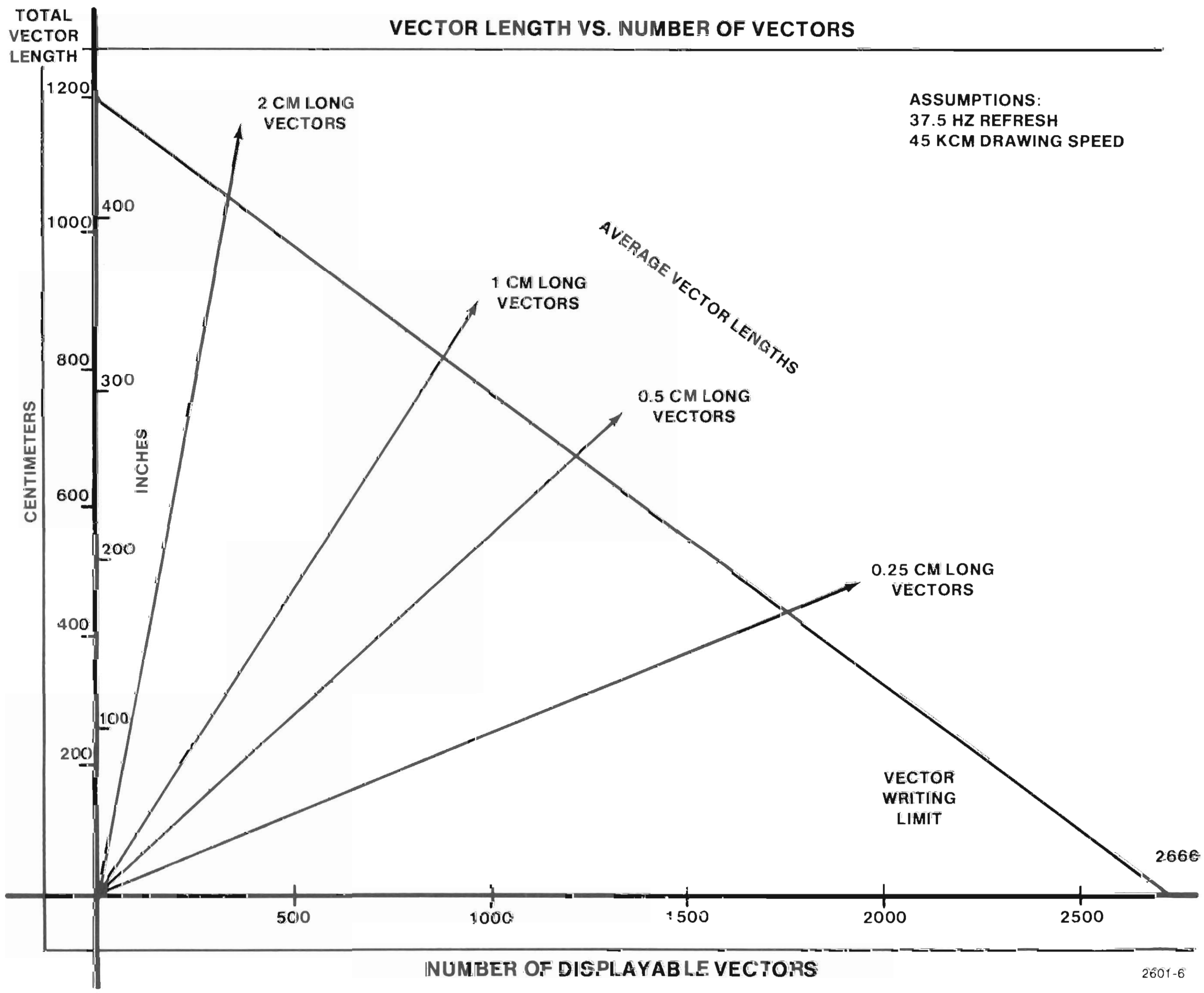


Figure A-1. Object Display/Flicker Characteristics.

Microprocessor

8X300 clock frequency	7.4 MHz
External registers	16 — 8 bit registers
Addressable RAM	32K bytes

General

Storage requirement/vector	11 bytes
Refresh rate	37.5 Hz
Vector generator writing rate (function of Vector Generator in 4054 only).	45K cm/second
Addressable area	4096 by 4096

ENVIRONMENTAL SPECIFICATIONS

The environmental limits for the Option 30 are the same as those for the standard 4054 Graphic System.

Heat dissipation — Option 30 adds 15 watts (51 BTU) to the 4054's heat output.

ACCESSORIES

Standard:

- 4054 (Options 30 and 31) Dynamic Graphics User's Manual — 070-2287-00 and up
- Ring binder
- 4054 Dynamic Graphics Reference Guide — 070-2586-00 and up

Optional:

- 4054 (Options 30 and 31) Dynamic Graphics Service Manual — 070-2601-00 and up

Table A-1

OPTION 30 COMMANDS DEFAULT I/O ADDRESSES

Use with PRINT and FIND statements. See "I/O Addresses" in 4050-Series BASIC Language Reference Manual.

Command	I/O Address
BLINK	32,8:
BRIGHTNESS	32,30:
CURSOR	32,6:
DASH	32,31:
FIX	32,5:
RAPPEND	32,4:
RCLOSE	32,2:[a]
RDELETE	32,7:
RINIT	32,29:[a]
RMEMORY	none
ROPEN	32,3:
RREPLACE	32,0:
RSPACE	none
STPOINT	32,16:
VISIBILITY	32,1:

[a]To use default I/O addressing with the RCLOSE and RINIT commands, a "dummy" parameter must be included. The parameter can be any number or defined variable. Example: RINIT =PRINT @ 32,29: 0.

Table A-2

OPTION 30 SYSTEM DEFINED OBJECTS

The following objects are built by the system at each power-up or RINIT. Accessible characters may be redefined by the user/programmer.

Object Number	Explanation
Object 0	Standard crosshair, NOT accessible to user.
Object 65532	Input prompt (blinking '?'). Accessible.
Object 65533	'Full Page' message. Accessible.
Object 65534	Blinking alpha cursor. Accessible.
Object 65535	Prints refresh character (used by PRINT @ 32,24:). NOT accessible.

A program can internally determine whether the Graphic System is a 4054, Option 30 by use of RND (Ø). Execution of an RND (Ø) command on a 4054, Option 30, will produce a result of 0.505007490939.

Appendix B

APPLICATIONS PROGRAM

This program shows how to use Option 30 for a typical flow diagram application. The display presents user instructions in the upper left corner. The seven boxes along the right margin contain symbols that may be selected and moved around the screen via the thumbwheels.

Move the crosshairs over any desired symbol ("menu item") and press the space bar. The crosshairs are then replaced by this symbol. Now move the symbol to the desired screen location and store it there by pressing the space bar.

Next, move the cursor object (still in refresh) back on top of next symbol to be used; press the space bar again. The new symbol now appears, replacing the first one. Move this symbol into position and press space bar to store it.

To provide a connecting arrow, move cursor symbol over top of arrow (menu item 6) and press space bar. Bring tail of arrow into position next to first box. Then rotate arrow to face in proper direction. Rotate clockwise by pressing the R key for 10 degree increments. (The L key rotates counter-clockwise.) When the arrow is pointing in the proper direction and the tail is in the right position, press space bar. This establishes the start of the arrow line. Then move the thumbwheel until the arrowhead extends to the next symbol. The arrow will now connect those two end points. Press space bar to store this arrow.

Lines may also be used instead of arrows. Select 'DRAW' symbol (menu item 7) and notice the crosshair reappears. Move the crosshair to the start of a draw line and press space bar. Then move the crosshair to the other end of the line segment and press space bar again.

Now, move to the next line's starting point and press space bar. This second line is drawn from this point. Press space bar again at the end of this line, and repeat process for any other lines.

To place text on the diagram, first press key 'T'. Now enter your text string. This text becomes the cursor, replacing the old cursor object. Notice single underline. End this text string by entering <CR>, carriage return. The text string can now be moved around the screen using the thumbwheels. Position the text and store with space bar.

This program can be used to create diagrams such as are shown in Figures B-1 and B-2. Also you can go to lines 1400, 1470, 1540, 1620, 1690, and 1840 and change the corresponding object definitions to provide the symbols you need for a particular application. In a physical plant layout, the symbols might be machinery or office furniture (desks, chairs, files, etc.). In a landscape architecture application the symbols might be various trees, shrubs, boulders, stepping stones, etc. For electrical engineering applications your menu items might be schematic symbols of circuit components.

If you decide to make diagrams that will be stored in a data base and outputted to graphic plotter, you should include a GIN after each FIX statement, which determines the X,Y location. This data will then be sent to an "X,Y FIX location table" on your tape or disc.

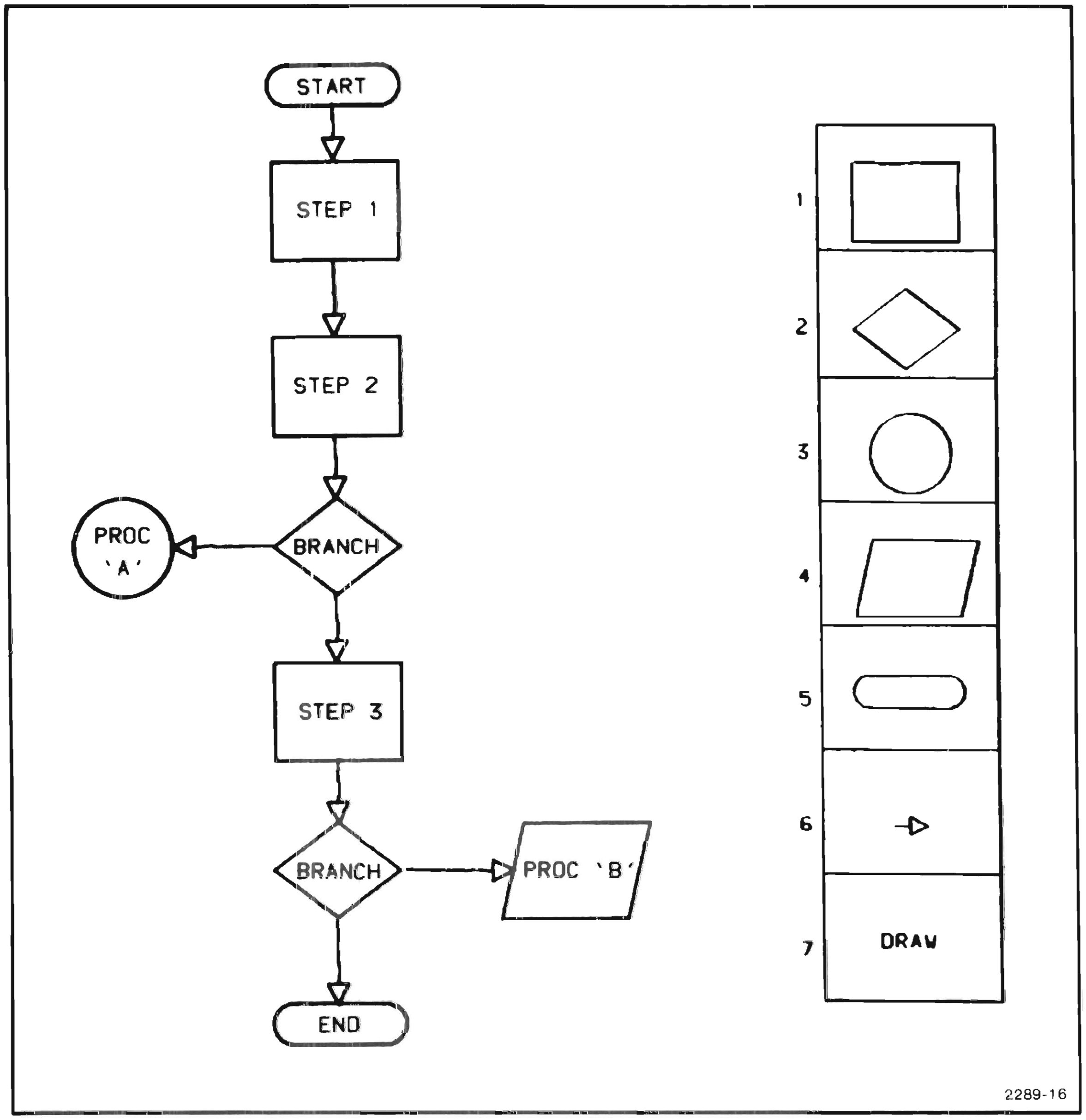


Figure B-1. Flow Diagram Application.

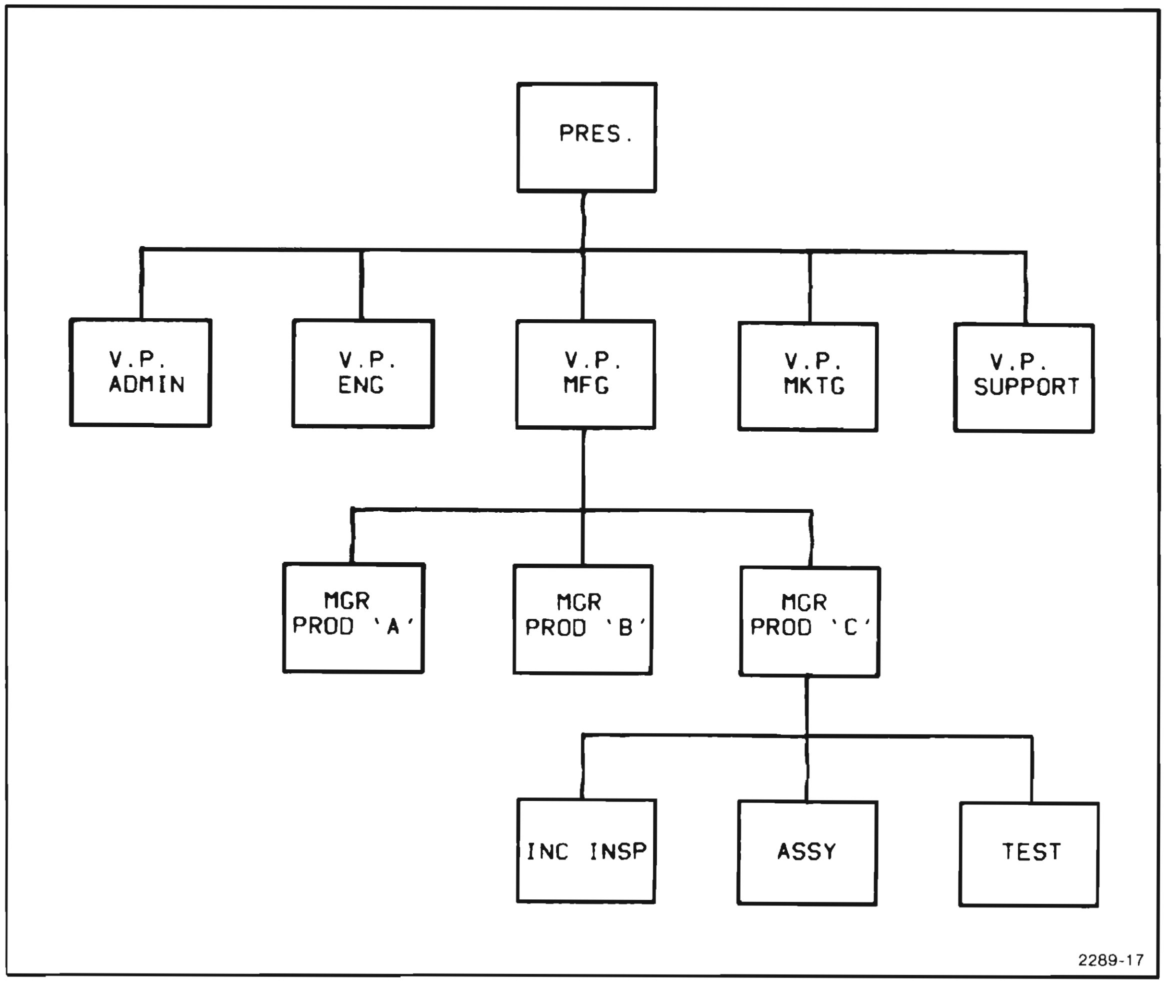


Figure B-2. Organization Chart Application.

```
        100 INIT
        110 PAGE
        120 SET DEGREES
        130 REMARK INITIAL VALUES FOR:OUTPUT DEVICE,ROTATION,ARROW SW.,
        140 REMARK                   NUMBER OF SYMBOLS,HEIGHT & WIDTH
        150 DATA 32,0,0,7,9,12
        160 READ T,A,K,N,H,W
        170 H1=H/2
        180 W1=W/2
        190 BRIGHTNESS 3
        200 REMARK BUILD  REFRESH MENU AND SYMBOLS
        210 ROPEN 1000
        220 CHARSIZE 3
        230 PRINT """S"" TO STOP"
        240 PRINT """R"" TO ROTATE SYMBOL RIGHT"
        250 PRINT """L"" TO ROTATE SYMBOL LEFT"
        260 PRINT """T"" TO ENTER TEXT"
        270 REMARK DRAW SYMBOL MENU FRAME
        280 FOR I=0 TO N
        290  H2=100/N
        300  W2=130-8-W
        310  PRINT @T,21:W2,100-H2*I
        320  PRINT @T,20:130,100-H2*I
        330 NEXT I
        340 PRINT @T,21:130,100
        350 PRINT @T,20:130,100-H2*(I-1)
        360 PRINT @T,21:W2,100-H2*(I-1)
        370 PRINT @T,20:W2,100
        380 REMARK DRAW SYMBOLS IN FRAME
        390 FOR I=1 TO N
        400  MOVE @T:130-4-W1,100-H2*I+H1+1
        410  GIN @T:X,Y
        420  GOSUB 1380
        430  MOVE @T:X-(W1+8),Y-1
        440  PRINT @T:I;
        450 NEXT I
        460 RCLOSE
        470 CHARSIZE 1
        480 K=1
        490 REMARK FOLLOWING IS POINTER COMMAND FOR CROSSHAIR INTERACTION
[1160]  [1220]  [1340]
+++3+++ 500 POINTER X5,Y5,Z$
        510 IF Z$="S" THEN 1350
        520 IF Z$="L" THEN 820
        530 IF Z$="R" THEN 850
        540 IF Z$="T" THEN 880
        550 REMARK IF X LOCATION IS IN MENU AREA--SELECT NEW MENU ITEM
        560 IF X5=>W2 THEN 1230
        570 REMARK FIX IMAGE IF SYMBOL # LESS THAN 5 OR
        580 REMARK           IF ODD NUMBERED SEQUENCE
        590 IF K1 OR I<6 THEN 1170
        600 IF I=7 THEN 670
        610 REMARK APPEND PREVIOUS POINT TO EXISTING SYMBOL
        620 RAPPEND 1
```

```
[690]
+++1+++    630 REMARK DRAW FROM PREVIOUS POINT==RUBBERBANDING==
           640 DRAW X5,Y5
           650 RCLOSE
           660 GO TO 1190
[600]
+++1+++    670 REMARK REPLACE CROSSHAIR WITH RUBBERBAND DRAW
           680 ROPEN 1
           690 GO TO 630

----------------------------------------------------------------------
-----------------
[1300]
---1---    700 REMARK BUILD CROSSHAIR FOR MOVE/DRAW SYMBOL
           710 MOVE 65,50
           720 ROPEN 2
           730 VISIBILITY 2,0
           740 RMOVE -20,0
           750 RDRAW 40,0
           760 RMOVE -20,20
           770 RDRAW 0,-40
           780 RMOVE 0,20
           790 RCLOSE
           800 RREPLACE 1,2
           810 RETURN
[520]
+++1+++    820 REMARK INCREMENT ANGLE OF ROTATION BY 15 DEGREES
           830 A=A+15
           840 GO TO 1240
[530]
+++1+++    850 REMARK DECREMENT ANGLE OF ROTATION BY 15 DEGREES
           860 A=A-15
           870 GO TO 1240
[540]
+++1+++    880 REM TEXT ENTRY PROCESSING
           890 I=1
           900 A$="_"
[1040]     [1070]
+++2+++    910 ROPEN 1
           920 PRINT @32,21:X5,Y5
           930 PRINT A$;
           940 RCLOSE
           950 CURSOR 1
           960 REMARK SINGLE CHARACTER INPUT ROUTINE
           970 POINTER X4,Y4,Z$
           980 REMARK LENGTH OF ZERO MEANS END OF INPUT
           990 IF LEN(Z$)=0 THEN 1080
           1000 REMARK RUBOUT MEANS TO DELETE LAST CHARACTER
           1010 IF ASC(Z$)=127 THEN 1050
           1020 REMARK INSERT CHARACTER INTO STRING
           1030 A$=REP(Z$,LEN(A$),0)
           1040 GO TO 910
```

```
[1010]
+++1+++   1050 REMARK "RUBOUT" LAST CHARACTER
          1060 A$=REP("",LEN(A$)-1,1)
          1070 GO TO 910
[990]
+++1+++   1080 REMARK DELETE UNDERLINE CHARACTER
          1090 A$=SEG(A$,1,LEN(A$)-1)
          1100 REMARK BUILD STRING IMAGE IN REFRESH
          1110 ROPEN 1
          1120 PRINT @32,21:X5,Y5
          1130 PRINT A$;
          1140 RCLOSE
          1150 CURSOR 1
          1160 GO TO 500
[590]
+++1+++   1170 REMARK FIX THE CURRENT OBJECT
          1180 FIX 1
[660]
+++1+++   1190 REMARK CYCLE SYNCRONIZATION ROUTINE
          1200 K1=NOT(K1)
          1210 IF NOT(K1) AND I>5 THEN 1240
          1220 GO TO 500
[560]
+++1+++   1230 I=N+1-INT(N*Y5/100+1)
[840]   [870]   [1210]
+++3+++   1240 REMARK RE-BUILD THE SYMBOL ROUTINE
          1250 K1=0
          1260 MOVE 65,50
          1270 ROTATE A
          1280 ROPEN 1
          1290 VISIBILITY 1,0
          1300 GOSUB (I=7)+1 OF 1380,700
          1310 RCLOSE
          1320 VISIBILITY 1,1
          1330 CURSOR 1
          1340 GO TO 500
[510]
+++1+++   1350 REMARK STOP-PROCESSING AND CLEAR REFRESH IMAGES
          1360 RINIT
          1370 END

-----------------------------------------------------------------------
----------------
[420]   [1300]
---2---   1380 REM SYMBOL DRAW ROUTINE
          1390 GO TO I OF 1400,1470,1540,1620,1690,1840,1940
[1390]
+++1+++   1400 REM box
          1410 RMOVE @T:-W1,H1
          1420 RDRAW @T:W,0
          1430 RDRAW @T:0,-H
          1440 RDRAW @T:-W,0
          1450 RDRAW @T:0,H
          1460 RETURN
```

```
[1390]
+++1+++  1470 REM diamond
         1480 RMOVE @T:0,H1
         1490 RDRAW @T:W1,-H1
         1500 RDRAW @T:-W1,-H1
         1510 RDRAW @T:-W1,H1
         1520 RDRAW @T:+W1,H1
         1530 RETURN
[1390]
+++1+++  1540 REM circle
         1550 RMOVE @T:0,-H1
         1560 FOR V0=0 TO 360 STEP 10
         1570  ROTATE V0+A
         1580  RDRAW @T:0.8,0
         1590 NEXT V0
         1600 ROTATE A
         1610 RETURN
[1390]
+++1+++  1620 REM parallelogram
         1630 RMOVE @T:-W1/3*2,H1
         1640 RDRAW @T:W,0
         1650 RDRAW @T:-W1/3,-H
         1660 RDRAW @T:-W,0
         1670 RDRAW @T:W1/3,H
         1680 RETURN
[1390]
+++1+++  1690 REM TERMINAL
         1700 RMOVE @T:-W1/3*2,H1/3*2
         1710 RDRAW @T:W1/3*4,0
         1720 FOR V0=90 TO -90 STEP -22.5
         1730  ROTATE V0+A
         1740  RDRAW @T:0,-H1/6
         1750 NEXT V0
         1760 ROTATE A
         1770 RDRAW @T:-W1/3*4,0
         1780 FOR V0=90 TO -90 STEP -22.5
         1790  ROTATE V0+A
         1800  RDRAW @T:0,H1/6
         1810 NEXT V0
         1820 ROTATE A
         1830 RETURN
[1390]
+++1+++  1840 REM ARROWS
         1850 IF K THEN 1870
         1860 RMOVE @T:-2,0
[1850]
+++1+++  1870 RDRAW @T:2,0
         1880 RDRAW @T:0,1
         1890 RDRAW @T:2,-1
         1900 RDRAW @T:-2,-1
         1910 RDRAW @T:0,1
         1920 RMOVE -2,0
         1930 RETURN
[1390]
+++1+++  1940 REM DRAW
         1950 PRINT "HHDRAW";
         1960 RETURN
```

Appendix C

GLOSSARY

This glossary pertains to this manual and is not intended to be a universal reference. It supplements the terms in the 4054 Operator's Manual Glossary.

Alpha Cursor	A blinking, non-storing rectangular symbol which indicates the next character writing position.
Alphanumeric	Refers to letters and numbers.
Argument	A value, device, or file that is operated on by a program or resident command (also called a parameter). The term also refers to the part of the program or command syntax that specifies the value, device, or file.
Beam, writing	The stream of electrons (within the cathode-ray-tube) which causes a display image to appear on the screen.
Close	Refers to completing (closing) the definition of a refreshed object.
Cursor	A pointer or object (alpha cursor, crosshair, or other refreshed object) that may be moved around the screen via thumbwheels (GIN device).
Dash mask	A parameter used with the DASH command to set either a dash, solid, or dotted line pattern for a given set of vectors.
Default	A predetermined value or condition that is assumed if no other value or condition is specified.
Display	The cathode-ray-tube screen where the operator views data (graphic and/or text).
Dynamic Graphics	The product name for Option 30 that denotes movable or changeable (dynamic) picture images (graphics).
Environmental parameter	A parameter or argument that applies to the entire screen and all objects displayed, rather than to a single object only.

Fix	Stores the object on the screen at its present location via FIX command. Object may then appear on a Hard Copy.
Flicker	Unintentional rapid blinking or twitching of an object, caused by interruptions of the refresh cycle.
GIN device	A device such as the thumbwheels or joystick, that is used for inputting graphic data (X,Y coordinates) to the Graphic System.
Menu item	One item from a menu list.
Menu list	A list of operations or symbols displayed on the screen, usually in refresh.
Menu picking	Selecting a menu item from a menu list.
Object	A set of vectors as text strings, pictures, etc., treated as a single screen entity. May be manipulated via Option 30.
Object file	The part of Dynamic Memory occupied by an object definition (between an ROPEN and RCLOSE).
Open	Begins an object definition in a BASIC program (see ROPEN command).
Parameter	See Argument.
Refreshed image	An image on the display that is continually redrawn in non-storage or Write-thru mode.
Repaint	See Rewrite.
Rewrite	Display is erased and updated images are written on the screen. Occurs in Storage mode.
Rubber band	A vector that connects a fixed point to a movable point (usually in object) on screen.
Statement	A line of program that includes a command and its qualifying parameters.

Storage mode	The display mode where data is written on the screen with enough intensity to store until a manual or program-controlled erase.
Store	To retain an image on the screen as a result of writing with sufficient writing-beam intensity.
Store Matrix	A table or matrix, part of a program on a tape or disc memory file, so arranged as to store: the object number, the FIX location, rotation, etc., for each store of a refreshed image.
Set Point	Repositions an entire object (or part of it) on the display by reSETing (moving) its starting POINT location.
Thumbwheels	Operator knobs with partially exposed surfaces; a pair of these by the lower right keyboard controls the location of the cursor.
Translational motion	Object moves without rotating.
Vector	Writing beam writes from one screen location to another, creating a straight line.
Viewport	The user definable portion of the display that lets the display image show. Surrounding the viewport is a masked area (like a frame) which hides corresponding parts of the image.
Visibility	An on/off visibility control of an object's refreshed screen image.
Write-thru	Also called Refresh. A display mode that prevents information from storing when it is written on the screen. Previously stored information remains in view.
Writing beam	See Beam, writing.
X,Y Location Table	See Store Matrix.

Appendix D

COMMAND SUMMARY

This appendix is a quick reference for programming or operating the 4054 with Dynamic Graphics option. Each command is described briefly in terms of a simple and typical example.

The following command summary provides:

- command name and syntax,
- an example,
- explanation of example,
- range of arguments, where applicable,
- default values for optional arguments,
- whether the command/function may be assigned to variables,
- whether the command operates only in BASIC program run (vs. immediate) mode. All commands operate in both modes unless otherwise indicated.

BLINK object number, on time, off time

BLI 1, 0.5, 0.2

Blinks object 1 on (visible) for 0.5 seconds and off (invisible) for 0.2 seconds, repeatedly.

The minimum on/off time is 0.03 seconds. You may use variables for on/off times.

BRIGHTNESS display intensity code

BRI 0

Sets the brightness of all subsequent object vectors to normal and defocused.

Code	Intensity	Focus	Affected Displays
0	Normal	Defocused	Refreshed/Stored
1	Normal	Focused	Refreshed/Stored
2	Bright	Defocused	Stored only
3	Bright	Focused	Stored only

CURSOR object number

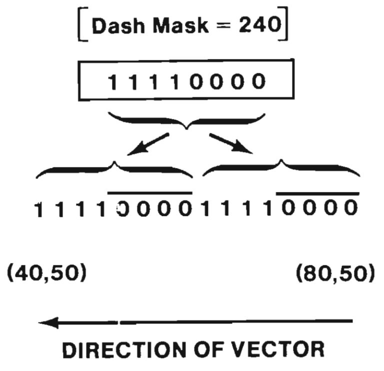
CUR 1

Causes object 1 to become the graphic cursor displayed by the POINTER statement. Cursor position is controlled by the thumbwheels or optional joystick. (Replaces the standard crosshair cursor.)

DASH dash mask

DAS 240

Sets the dash pattern for displayed vectors. This decimal value 240 is converted to binary ($240_{10} = 11110000_2$), and this binary number determines the dash pattern, as shown below.



Here are some other common dash patterns with corresponding dash masks.

Pattern	Dash Mask Number	Binary Equivalent
———— ————	3	00000011
-- — — -	170	10101010
— — -- —	85	01010101
————	15	00001111
(no vector is drawn)	255	11111111
—— ————	200	11001000

If the mask is greater than 255, 256 is subtracted from the mask until it is between 0 and 255.

The default value for the dash mask is 0 (solid vectors), and is reset to 0 by RINIT command.

FIX object number

FIX 3

Stores object 3 at its present location on the screen. Object 3 is still displayed in refresh at the same location.

RAPPEND object number

RAP 3

Appends information to object 3. All vectors and alphanumerics between this RAPPEND and the next RCLOSE are included in object 3.

RAPPEND operates only in BASIC program mode.

RCLOSE

RCL

Closes (completes) the object being defined. The object number is given in the corresponding ROPEN or RAPPEND statement.

An ROPEN, RAPPEND, END, STOP, and BREAK do an implied RCLOSE if an object is open.

RDELETE object number

RDE 2

Deletes object 2 from Dynamic Graphics memory.

RINIT

RIN

Deletes all user-defined objects from Dynamic Graphics memory and redefines the system objects to their defaults. (See the SYSTEM OBJECTS list.)

[numeric variable=] RMEMORY

RME

Returns the number of bytes of available (unused) Dynamic Graphics memory.

May be assigned to a variable, sending this information to the program.

A= RME

IF A > 1000 THEN 170

If 1000 bytes of memory are available, go to line 170.

ROPEN object number

ROP 1

Opens object 1 file. Subsequent vectors and/or text strings until RCLOSE become "object 1." ROP 1 deletes any existing definition of object 1.

Construction of object must occur in BASIC program mode, not immediate mode.

Allowed object numbers:

- user-defined — 1 through 65531
- system objects — 0 and 65532 through 65535

(See object list at the end of this appendix.)

RREPLACE first object number, second object number

RRE 1, 2

Object 1 is deleted and object 2 becomes the new object 1. Original object 1's display parameters (blink rate, brightness, etc.) remain intact.

[numeric variable=] RSPACE

RSP

Returns (displays) the number of bytes of occupied Dynamic Graphics memory space.

May be assigned to a variable, sending this information to the program.

```
100 B = RSP

110 IF B = 32768 THEN 530

530 PRINT "DYNAMIC MEMORY FULL"
```

STPOINT object number, X location, Y location

STP 1, 10, 20

Positions starting point of object 1 to a new screen location 10, 20 UDUs*, thus moving entire object.

*UDU — user defined units

VISIBILITY object number, display argument

VIS 3, 0

Object 3 is no longer visible on the screen.

When the display argument is non-zero, the object will be visible (in refresh).

0 — invisible

non0 — displayed but not stored.

SYSTEM OBJECTS

The following objects are built by the system at each power-up or RINIT. Accessible characters may be redefined by user.

Object 0 — standard crosshair cursor. Not accessible to the user.

Object 65532 — Input prompt (blinking '?'). Accessible.

Object 65533 — 'Full Page' message. Accessible.

Object 65534 — Blinking alpha cursor. Accessible.

Object 65535 — Prints refresh character (used by PRI @32,24:). Not accessible.

ERROR MESSAGE

Number 88

Prompted by reference to an illegal object number (such as a decimal number).

INDEX