terak
CORPORATION

RT-11/85A MUBASIC GRAPHICS EXTENSIONS
VERSION 1.0 RELEASE NOTES

TERAK Publication Number 60-0040-001

REV 2

14405 NORTH SCOTTSDALE ROAD • SCOTTSDALE, ARIZONA 85254 • (602) 991-1580

```
****************************************************************************
******************** RT-11/85A MUBASIC GRAPHICS EXTENSIONS ******************
 ******************       VERSION 1.0 RELEASE NOTES      **************** Rev 2
************************************************************* Pub no. 60-0040-001*
```

INTRODUCTION

This graphics extensions package provides RT-11/85 MUBASIC with graphics support for the TERAK 8510/a graphics computer system. These routines allow MUBASIC programs complete control over the 8510/a graphics and character display. Graphics are programmed by use of the 'CALL' statement (explicit or implicit) to pass parameters and control to the graphics routines. While MUBASIC can support multiple users, only one user partition may use these graphics routines.

Points, vectors, markers, and text may be displayed separately or in combination. Entire arrays of data may be graphically displayed with one CALL statement. Viewports, windows, clipping and partial screen display are also supported by this graphics package.

Minimum hardware required is a TERAK 8510/a graphics computer system. These graphics routines require approximately 3.2K of memory.

TERAK 8510/A GRAPHICS COMPUTER SYSTEM HARDWARE OVERVIEW

The Graphics display is presented on the 8532 video display as a 320 dot wide by 240 dot high matrix. The graphics display, when active, illuminates or blanks each dot according to the contents of a memory buffer. 1600 (16 bit) words are required for each third of the screen display; 4800 for an entire 320 by 240 dot display. This can be reduced, with simultaneous reduction of the graphics area of the monitor displayed, by the zone blanking feature: each third of the graphics displays may be blanked or displayed.

The Character display is presented on the video display as a matrix of characters 80 characters wide by 24 characters high. The display presented to the user is a video overlay of the graphics and character displays. Except for the conventions established by this and other system software, the two displays are independent. Like the graphics display, the character display can be blanked.

For blanking control, the display matrix is divided into three horizontal zones: each graphics zone is 80 dots high by 320 dots wide, each character zone is 8 characters high by 80 characters wide. Blanking of any one character display zone is mutually independent of the blanking of any one graphics display zone.

GRAPHICS EXTENSIONS OVERVIEW

These routines provide conventions for the description of a graphics display by the user program. Clipping and Window to Viewport translations are performed to allow the user program the simplest means of expressing the desired result. Although effectively instantaneous, the processing of graphics can be viewed as separate operations upon 'virtual' pictures. Figure 1 illustrates an analogy to the process, and should be refered to as the graphics extensions are presented.

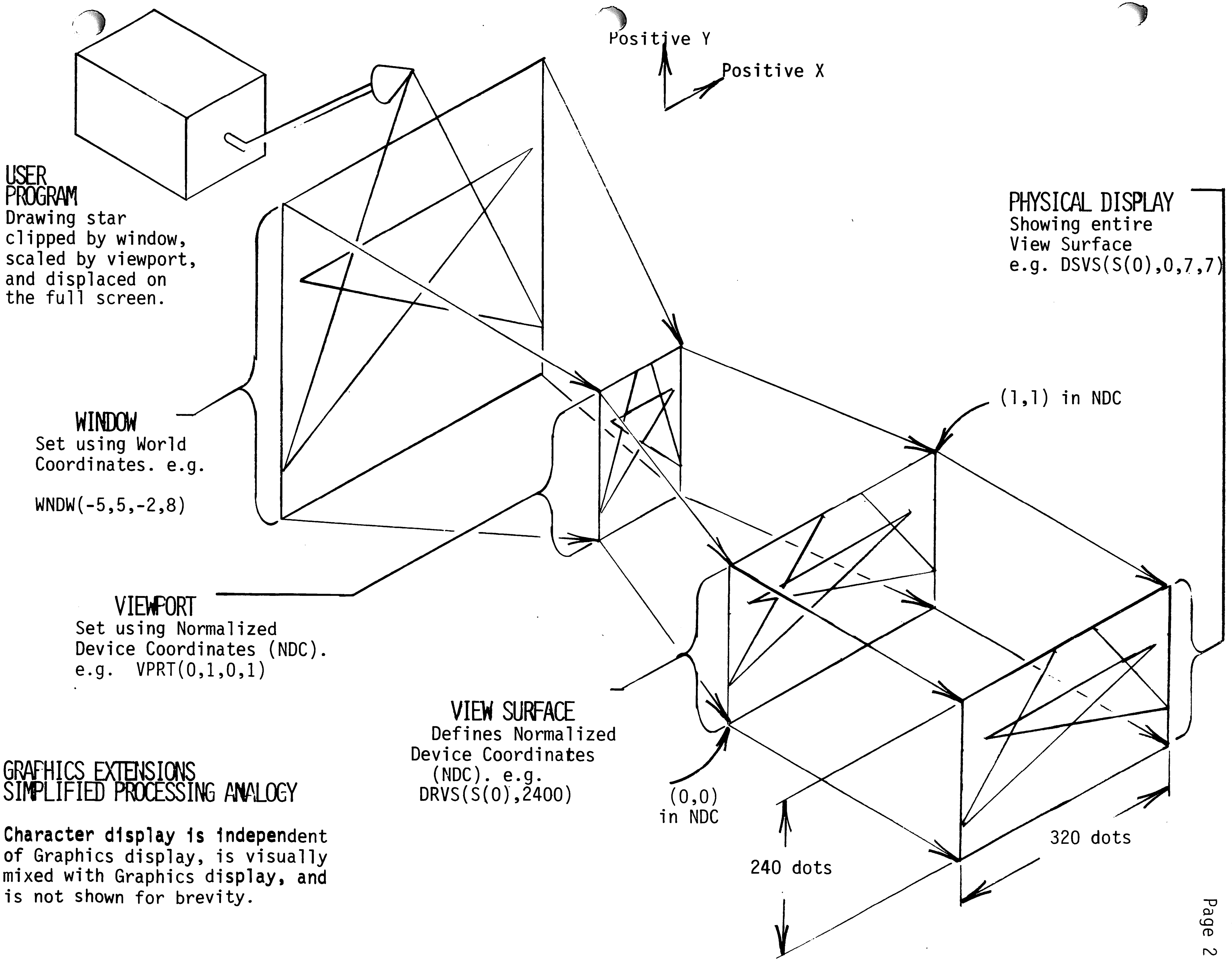
Positive Y
Positive X
USER PROGRAM
Drawing star clipped by window, scaled by viewport, and displaced on the full screen.
WINDOW
Set using World Coordinates. e.g.
WNDW(-5,5,-2,8)
VIEWPORT
Set using Normalized Device Coordinates (NDC). e.g. VPRT(0,1,0,1)
VIEW SURFACE
Defines Normalized Device Coordinates (NDC). e.g. DRVS(S(0),2400)
PHYSICAL DISPLAY
Showing entire View Surface e.g. DSVS(S(0),0,7,7)
(1,1) in NDC
(0,0) in NDC
240 dots
320 dots

Figure 1

GRAPHICS EXTENSIONS SIMPLIFIED PROCESSING ANALOGY

Character display is independent of Graphics display, is visually mixed with Graphics display, and is not shown for brevity.

WORLD COORDINATES

Graphics is used when the user program wishes to represent relationships of data pictorially. For example, one such relationship would be SALES vs TIME for a particular product produced. The graph of such a relationship would typically consist of two axes and special markers connected by lines. Such a graph is presented in Figure 2.

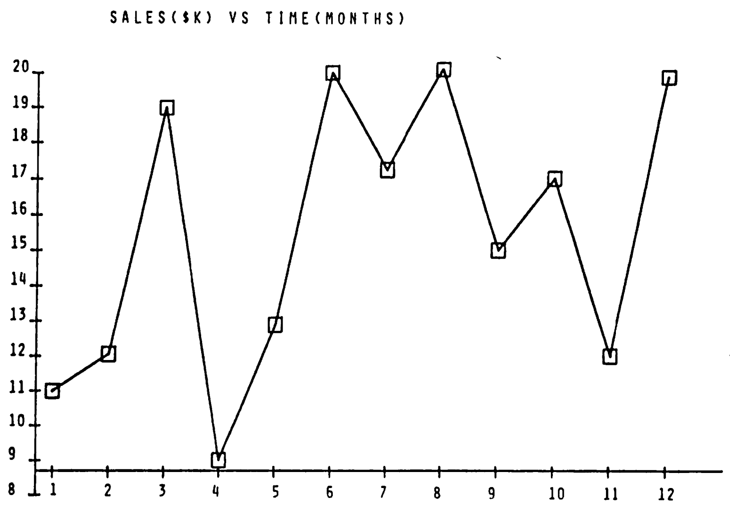

FIGURE 2

To produce a graph such as Figure 2, the user program must describe the various display elements in terms of positions and measurements in a cartesian coordinate system. The coordinates used by the program may represent any unit of measure. In Figure 2, SALES is measured in the vertical direction, in dollars, and TIME is measured in the horizontal direction, in months. The parameters passed to the graphics routines must be floating point constants or variables, and may range throughout the world of coordinate values - including values not representable within the defined window. These coordinates are called WORLD COORDINATES.

CURRENT POSITION

Complete displays on the TERAK 8510/a are generated by output primitives which generate lines, markers and text on the display. Intrinsic to this graphics support is the concept of the current position of a cursor at a position in the World Coordinate space. An analogy to the current position would be the position of an artist's pencil after each pencil stroke. Each output primitive affects the value of the current position in a way defined herein. The current position may be obtained by the routine CPOS.
For example the call:

```
CPOS(X,Y)
```

within a user program returns the horizontal and vertical current position of the graphic cursor into the variables X and Y respectively.

LINES, MARKERS, AND POLYLINES

A user program produces a display by invoking output primitives to specify a series of operations using positions in the World Coordinate space. For example to draw a line from the current position to the point with (X,Y) World Coordinates of (10, 20.5), the user program would issue the call:

```
LINA(10,20.5)
```

Following the execution of the LINA statement the current position then becomes the end point of the line just drawn. The name of the routine LINA means to draw a line to the 'absolute' position specified. Lines may also be drawn 'relative' to the current position via the LINR routine.

The current position may be changed without drawing lines by issuing calls the the MOVA and MOVR routines. For example the call:

```
MOVR(1.4,-0.5)
```

will move the current position 1.4 units in the positive X direction, and 0.5 units in the negative Y direction.

A routine is provided to draw predefined markers centered at a point. The size of any marker is fixed at 7 dots high by 7 dots wide. Markers are useful to distinguish certain points in the display. The marker routine moves the current postion to the point where the marker is drawn. For example:

```
MRKA(100,200,3)
```

will move the current position to the World Coordinate point (100,200) and then draw a 7 by 7 diamond shaped figure at that point. A table of all marker patterns available is presented below.

Arrays of points may be defined and then passed to a polyline routine which will then draw lines to connect the points. If the arrays X and Y were dimensioned at 20 elements, the following call would connect all the points defined by (X(n) , Y(n)) for n=1 to 20.

```
PLNA(X(1),Y(1),20)
```

TEXT

Character text may be drawn into the display by using the TEXT routine. Two variations of text may be displayed: either low quality or medium quality. Low quality text is placed into the Character Page buffer overlaying the graphics display. High quality text is placed into the graphics display. The routine SCQL is used to set the character quality for all subsequent calls to TEXT. When the quality of the text is medium, the characters drawn onto the graphics space also have the attributes of 'size' and 'spacing'. These attributes may be set from the user program using the SCSZ and SCSP routines, respectively. For example the calls:

```
SCSZ(2,1)
SCSP(.1,.15)
SCQL(1)
TEXT('STRING OF TEXT')
```

will output the characters 'STRING OF TEXT' starting at the current position. Characters are patterned after the 8 by 10 dot template of each character in the writeable character generator of the 8510/a. The characters in the example will be 20 screen dots (2*10) high by 8 screen dots (1*8) wide. After each character is drawn, the current position will change by .1 (in World Coordinates) in the X direction and by .15 in the Y direction.

WINDOWS

These routines allow the user program to locate and specify objects in the World Coordinate system, permiting the program to work in a coordinate system which is natural to the application. To produce a display, the graphics routines must convert the World Coordinate specifications to the absolute physical screen coordinates of the TERAK 8510/a display. To perform this conversion, the user program must specify which portion of the World Coordinate space is to be displayed on the View Surface. This region is called a WINDOW. It must be rectangular and is defined by a call to the WNDW routine, whose arguments specify the minumums and maximums along each axis. The generic form of the WNDW call is

```
WNDW(min_x,max_x,min_y,max_y)
```

When output primitves are executed to define objects either partially or entirely outside of the edges of the Window, clipping automatically takes place at the Window edge. The current position, however, may be outside the Window. If the current position does lie outside the Window and a line is drawn which exists partially within the Window, the line will be drawn with a slope natural to the connecting points with only that portion of the line inside the Window being visible.

VIEWPORTS

A Viewport is a rectangular portion of the View Surface onto which the Window is mapped. Most often the Viewport will occupy the entire View Surface. However, this is not always desired. For example, consider dividing the View Surface into four quadrants and mapping a different display onto each quadrant. To do this, it would be necessary to locate Viewports at different locations within the View Surface. These locations are best expressed with respect to the View Surface, rather than the World Coordinates. Normalizing these coordinates frees the user program from device dependent use of the physical display coordinates. This Viewport coordinate system is called Normalized Device Coordinates (NDC). The coordinates range from 0 to 1 in both X and Y directions, with (0,0) at the lower left of the View Surface, and (1,1) at the upper right of the View Surface, as shown in Figure 3.

```
0,1 -------------- 1,1
  |                 |
  |  View  Surface  |          FIGURE 3
  |                 |
0,0 -------------- 1,0
```

Normalized Device Coordinates locate and define the Viewport onto which the Window is mapped, as directed by a VPRT routine. The generic form of the VPRT call is

VPRT(xmin,xmax,ymin,ymax)

For example:

```
VPRT(0,.5,0,.5)        MAP WINDOW ONTO THE LOWER LEFT QUADRANT
                           OF THE VIEW  SURFACE.
VPRT(.25,.75,0,.5)     MAP WINDOW ONTO LOWER HALF OF THE MIDDLE
                           2/4 OF THE VIEW  SURFACE.
VPRT(0,1,0,1)          MAP THE WINDOW ONTO THE ENTIRE VIEW
                           SURFACE.  THIS IS THE DEFAULT VIEWPORT.
```

Once a VIEWPORT and WINDOW have been defined, calls can then be made to the graphics routines to create the display. For example, consider this program fragment:

```
  .
  .
  .
100 REM set up the window for 0 <= X <= 100, 0 <= Y <= 50
110 WNDW(0,100,0,50)
120 REM set up Viewport for upper half of view surface
130 VPRT(0,1,.5,1)
  .
  .
  .
```

ALLOCATING AND DISPLAYING THE GRAPHICS SPACE

The memory used for creation of a graphics display must be allocated by the user program. This memory may be accessed by the graphics routines, or by any other mechanisms available to the program. The program must use the Draw_on_view_surface (DRVS) routine to direct the graphics routines to use an array, previously declared by the user program, as a View Surface.

A graphics display can be created without necessarily viewing it. For example, a user program may create a display for later viewing. The Display_view_surface (DSVS) routine directs the hardware to display an array as graphics on the video monitor and also controls the zone blanking of the display. The independence of the DRVS-defined View Surface and the DSVS-defined View Surface provides the capability of drawing on one View Surface, while displaying another.

The following program fragment would be used to allocate memory for a full screen of graphics and display all three zones of graphics and characters:

```
     10 REM-----ALLOCATE AND DISPLAY ARRAY S FOR GRAPHICS
     20 DIM S(2400)
     30 DRVS(S(0),2400)
     40 DSVS(S(0),0,7,7)
         .
         .
         .
```

The above program fragment defines the view surface and viewsurface-to-display mapping most commonly desired. Users desiring to control the view surface size and mapping onto the phsical surface should review the detailed descriptions of View Surfaces, DRVS and DSVS latter in this document.

GRAPHICS ROUTINES SUMMARY

The full complement of routines are summarized below. Note that underscores (_) are used to concatenate phrases from which mnemonics were extracted. Routines may be called implicitly by their appearance as a single program statement, or explicitly by being prefaced by the keyword 'CALL'. The implicit call is more efficient.

CALL	ARGUMENT LIST	EFFECT
DRVS	(array(sub),size)	Draw_on_view_surface. Defines which array to allocate as graphics memory space. The dimension of the array must be at least (size+sub). Size must be a multiple of 800.
DSVS	(array(sub),screen-displacement,graphics-zones,char-zones)	Display_view_surface. Defines which array to map onto the screen display. Screen-displacement defines how mapping is to occur, and graph-zones and char-zones define graphics and character blanking. (See below for further detail).
VPRT	(left,right,bottom,top)	Viewport defines which portion of the View Surface (defined by DRVS) is to receive the Window to Viewport mapping. All parameters are in Normalized Device Coordinates, i.e., 0 <= parameter <= 1 for all parameters. Also, 'left < right' and 'bottom < top' must be true.
WNDW	(xmin,xmax,ymin,ymax)	Window defines the X,Y World Coordinate space mapped onto the Viewport. Lines, markers, and text are clipped per the parameters in WNDW.

SLST	(linestyle)	Set_current_linestyle defines how lines text, and markers are to be drawn, where,
		linestyle = 0 means DRAW WHITE
		linestyle = 1 means DRAW BLACK
		linestyle = 2 means DRAW COMPLIMENT.
		Linestyle = 2 is valid only for lines.

NWFR		Newframe resets the View Surface defined by DRVS according to the current linestyle. If linestyle = 0 then the entire surface will become BLACK. If linestyle = 1 or 2 then the entire surface will become WHITE.
CPOS	(xpos,ypos)	Current-Position returns the current x and y position in World Coordinates into the variables xpos and ypos.

CALL	ARGUMENT LIST	EFFECT
MOVA	(x,y)	Move_to_absolute x,y moves the current position to absolute World Coordinates x,y. The View Surface is not affected.
MOVR	(x,y)	Move_to_relative x,y moves the current position to relative World Coordinates x,y. That is, the new currrent position is computed from the old current position plus the displacements x and y.
LINA	(x,y)	Line_absolute x,y draws a line per the current linestyle, from the current position to absolute World Coordinates x,y. The current position becomes x,y.
LINR	(x,y)	Line_relative x,y draws a line per the current linestyle, from the current position to relative World Coordinates x,y. That is, the new currrent position is computed from the old current position plus the displacements x and y.
PLNA	(xarray(sub),yarray(sub),n)	Polyline_absolute draws n lines connecting the coordinate points xarray(i),yarray(i) for i indexed from sub to n. Dimensions of xarray and yarray must be at least sub + n. Xarray and Yarray must be single dimension arrays.
PLNR	(xarray(sub),yarray(sub),n)	Polyline_relative draws n lines connecting the coordinate points xarray(i),yarray(i) for i indexed from sub to n. Dimensions of xarray and yarray must be at least sub + n. Xarray and Yarray must be single dimension. Note that each new current position is computed relative to the last current position.

CALL	ARGUMENT LIST	EFFECT
MRKA	(x,y,n)	Marker_absolute x,y moves the current position to World Coordinates x,y and draws marker n centered at x,y. Valid n range is Ø <= n <= 7. See below for details on marker paterns.
MRKR	(x,y,n)	Marker_relative x,y moves the current position to relative World Coordinates x,y, and draws marker n centered there. That is, the new currrent position is computed from the old current position plus the displacements x and y. Valid n range is Ø <= n <= 7. See below for details on marker paterns.
SCQL	(charquality)	Set_char_quality specifies the type of text output by the TEXT routine. Charquality = Ø will cause TEXT to produce low quality text output into the character display, starting at a row and column closest to the current position. Charquality = 1 or 2 will cause TEXT to produce medium quality text output into the graphics display, starting with the lower left corner of the first character at the current position. Charquality =1 will cause TEXT to generate characters at a slow rate, for visual impact.
SCSZ	(xsize,ysize)	Set_current_char_size specifies the size of characters output by the TEXT routine when charquality = 1 or 2. The character height is xsize*1Ø , and the character width is ysize*8. Height and width are in physical screen dots.
SCSP	(xspace,yspace)	Set_current_char_space defines the movement of the current position after each character is drawn by TEXT when charquality is 1 or 2 (graphics text). The spacing is in relative World Coordinates, That is, after each character, the new currrent position is computed from the old current position plus the displacements xspace and yspace.
TEXT	(string expression)	Text outputs the string defined by string expression per the current values of charquality, charspace, and charsize.

INITIAL VALUES

When the graphics extensions package is initially run, the default values of the control attributes are set as if the following calls had been made.

```
VPRT(Ø,1,Ø,1)
WNDW(Ø,1,Ø,1)
MOVA(Ø,Ø)
SLST(Ø)
SCQL(1)
SCSP(Ø,Ø)
SCSZ(1,1)
```

MARKERS AVAILABLE

For the marker absolute (MRKA) and marker relative (MRKR) routines, the following eight markers are available.

N	MARKER PATTERN
Ø	A single dot
1	A vertical bar 5 dots high
2	A horizontal bar 5 dots wide
3	A diamond 7 dots wide by 7 dots wide
4	A square 7 dots by 7 dots
5	A square 5 dots by 5 dots
6	A cross (X) 7 dots by 7 dots
7	A block (filled square) 7 dots by 7 dots

VIEW SURFACES

A View Surface is that portion of the graphics array which is manipulated (drawn on) by calls to graphics routines. Typically, the 'portion' is the entire array. A View Surface is always the same width as the physical display, but the height of a View Surface may be less than, equal to, or greater than the physical display height.

The Draw_on_view_surface (DRVS) routine defines the View Surface which is to be drawn on by the graphics routines. The 'size' parameter of DRVS defines its View Surface height, which may be one or more zones. (A zone corresponds to 8ØØ array elements or 1/3 of the physical screen.)

The Display_view_surface (DSVS) routine directs the Hardware to the View Surface which is to be displayed on the physical display. The 'displacement' parameter defines the View Surface to physical display zone mapping of the View Surface on the display. The zone blanking parameters define the zones in the character and graphics display which are visible.

The View Surface defined by DRVS and the View Surface displayed by DSVS are completely independent. Normally, they are coincident.

EXAMPLES OF DRVS AND DSVS

```
10 DIM S(24ØØ)
20 REM-----DEFINE 3 ZONES OF VIEW SURFACE TO DRAW ON.  ONE FULL SCREEN
30 DRVS(S(Ø),24ØØ)
40 REM-----DISPLAY 3 ZONES OF GRAPHICS AND CHARACTERS
50 REM-----DISPLACEMENT AT PHYSICAL ZONE Ø.
60 DSVS(S(Ø),Ø,7,7)

10 DIM S(8ØØ)
20 REM-----DEFINE A SINGLE ZONE OF VIEW SURFACE
30 DRVS(S(Ø),8ØØ)
40 REM-----DISPLAY GRAPHICS IN PHYSICAL ZONE 1, BLANK GRAPHICS IN Ø AND 2
50 REM-----DISPLAY CHARACTERS IN PHYSICAL ZONE Ø AND 2, BLANK CHARS IN 1
60 DSVS(S(Ø),1,2,5)

10 DIM S(32ØØ)
20 REM-----DEFINE VIEW SURFACE LARGER THAN PHYSICAL SURFACE
30 DRVS(S(Ø),32ØØ)
40 REM-----DISPLAY LOWER HALF OF VIEW SURFACE DISPLACMENT ZONE 1
50 REM-----BLANK GRAPHICS IN PHYSICAL ZONE Ø, BLANK CHAR IN 1 AND 2.
60 DSVS(S(16ØØ),1,3,4)
       .
       .
       .
400 REM-----NOW DISPLAY LAST THREE ZONES OF GRAPHICS ON SCREEN
410 REM-----BLANK ALL CHARACTER DISPLAY
420 DSVS(S(Ø),-1,7,Ø)
```

DRVS

For reference, the generic DRVS call is: DRVS(array(sub),size) .

Memory space for graphics is allocated in increments of 800 array elements, corresponding to the size of one graphics blanking zone (1/3 of graphics display). This allows the user program to define a View Surface which is smaller, equal to, or larger than the physical display. Normalized Device Coordinates of the View Surface always remain from 0 to 1 in both X and Y directions, although the physical mapping of the device coordinates may vary. Thus, the Normalized Device Coordinate range 0..1 in the Y direction may correspond to one, two, three, etc zones of physical display, as defined by the 'size' parameter. The Normalized Device Coordinate range 0..1 in the X direction always corresponds to the full width of the physical display. The size parameter in DRVS must be a multiple of 800.

DSVS

For reference, the generic DSVS call is:
DSVS(array(sub),displacement,graphics-zones,char-zones) .

A View Surface may be displayed on the physical display of the video monitor at a displacement corresponding any of the physical display zones. The 'displacement' parameter in DSVS specifies the number of zones of offset required by the user program. The normal value is zero, which will display the View Surface starting at array(sub) in the upper zone of the physical display. A displacement of -1 will start the display in the upper zone at Array(sub+800), -2 at Array(sub+1600), etc. Decreasing the displacement has the visual effect of moving the display upward on the physical display. The displacement may also be positive. Increasing the displacement has the visual effect of moving the display downward on the screen.

Blanking of the graphics and character displays is controlled by the DSVS parameters 'graphics-zones' and 'char-zones'. These zone parameters are computed by assigning the value 1, 2, and 4 to the upper, middle, and lower physical display zones (either graphics or character), and then adding up the values of the zones to be unblanked. The following table summarizes the eight combinations:

PARAMETER VALUE	EFFECT ON GRAPHICS OR CHARACTER BLANKING
0	BLANK ALL THREE ZONES
1	DISPLAY LOWER, BLANK MIDDLE AND UPPER ZONES
2	DISPLAY MIDDLE, BLANK LOWER AND UPPER ZONES
3	DISPLAY LOWER AND MIDDLE, BLANK UPPER ZONE
4	DISPLAY UPPER, BLANK LOWER AND MIDDLE ZONES
5	DISPLAY UPPER AND LOWER, BLANK MIDDLE ZONE
6	DISPLAY UPPER AND MIDDLE, BLANK LOWER ZONE
7	DISPLAY ALL ZONES

Note that the DSVS displacement parameter may cause the display of memory outside of the View Surface. Any such superfluous memory in the physical display may be blanked by setting the 'graphics-zone' blanking parameter to correspond to the 'displacement' parameter. A generalized analogy to the graphics display process is presented in Figure 4.

EXAMPLES OF DRVS AND DSVS

```
10 DIM S(2400)
20 REM-----DEFINE 3 ZONES OF VIEW SURFACE TO DRAW ON.  ONE FULL SCREEN
30 DRVS(S(Ø),2400)
40 REM-----DISPLAY 3 ZONES OF GRAPHICS AND CHARACTERS
50 REM-----DISPLACEMENT AT PHYSICAL ZONE Ø.
60 DSVS(S(Ø),Ø,7,7)

10 DIM S(800)
20 REM-----DEFINE A SINGLE ZONE OF VIEW SURFACE
30 DRVS(S(Ø),800)
40 REM-----DISPLAY GRAPHICS IN PHYSICAL ZONE 1, BLANK GRAPHICS IN Ø AND 2
50 REM-----DISPLAY CHARACTERS IN PHYSICAL ZONE Ø AND 2, BLANK CHARS IN 1
60 DSVS(S(Ø),1,2,5)

10 DIM S(3200)
20 REM-----DEFINE VIEW SURFACE LARGER THAN PHYSICAL SURFACE
30 DRVS(S(Ø),3200)
40 REM-----DISPLAY LOWER HALF OF VIEW SURFACE DISPLACMENT ZONE 1
50 REM-----BLANK GRAPHICS IN PHYSICAL ZONE Ø, BLANK CHAR IN 1 AND 2.
60 DSVS(S(1600),1,3,4)
         .
         .
         .
400 REM-----NOW DISPLAY LAST THREE ZONES OF GRAPHICS ON SCREEN
410 REM-----BLANK ALL CHARACTER DISPLAY
420 DSVS(S(Ø),-1,7,Ø)
```

EXAMPLE PROGRAMS AND EXERCISES

As an exercise, enter the following program, run it, and then perform the suggested changes which follow.

```
5 REM --- DRAW A SIMPLE SINE CURVE
10 DIM S(2400)
20 DRVS(S,2400)
30 DSVS(S,0,7,7)
40 WNDW(-10,10,-1.2,1.2)
50 SLST(0)
60 NWFR
70 MOVA(0,-1.5)
80 LINA(0,1.5)
90 MOVA(-10,0)
100 LINA(10,0)
110 MOVA(-10,SIN(-10))
120 FOR X=-10 TO 10 STEP .2
130 LINA(X,SIN(X))
140 NEXT X
150 STOP
```

a. To get an understanding of viewports, run the above program as is, and then insert this line: 45 VPPT(.2,.6,.5,1)

b. Change line 20 in the original program to:
```
20 DRVS(S,1600)
```
Run the program to see the effect of a different size View Surface.

c. With line 20 changed per (b), change line 30 to:
```
30 DSVS(S,1,3,7)
```
to see the effect of a different displacement of the View Surface.

d. Change line 50 to draw black on white, as:
```
50 SLST(1)
```

e. Change line 40 to WNDW(-10,10,-.8,.8) and obsrve how the graphics routines clip the display.

f. Insert the following lines to draw text into the graphics space
```
141 MOVA(-10,0)
142 SCSP(.5,0)
144 SCQL(1)
146 TEXT('THIS IS TEXT IN THE GRAPHICS SPACE')
```

g. With the changes in (f), add the following line:
```
145 SCSZ(1,3)
```

When writing programs which use the graphics routines in MUBASIC, it is occasionally convenient to enter the graphics calls in immediate mode, rather than in a program. This allows interactive selection of points corresponding to World Coordinates and display arrangement. As an exercise, enter various DRVS and DSVS calls in immediate mode, and observe their effect upon the graphics display.

EXAMPLE PROGRAM

This program will create a display similar to Figure 2.

```
3 REM---BLANK THE DISPLAY
4 PRINT CHR$(12)
5 DATA 11000,12053,18987,9000,12889,20000
6 DATA 17234,20100,15002,17000,12000,19876
10 REM---DEMONSTRATION PROGRAM    SALES  VS  TIME
20 REM---DIMENSION S FOR USE AS GRAPHICS DISPLAY MEMORY
30 DIM S(2400)
40 REM---DRAW ON S, USE 3 FULL ZONES
50 DRVS(S(0),2400)
60 REM---DISPLAY ARRAY S AT THE USUAL DISPLACEMENT
70 REM---TURN ON ALL GRAPHICS AND CHARACTER DISPLAYS
80 DSVS(S(0),0,7,7)
90 REM---VIEWPORT DEFAULTS TO ENTIRE SCREEN
100 REM---WINDOW IS 0 TO 13 IN X FOR 12 MONTHS, AND
110 REM---10,000 TO 20,000 FOR DOLLAR SALES
120 WNDW(0,13,8000,23000)
130 REM---SET LINESTYLE TO DRAW WHITE ON BLACK
140 SLST(0)
150 REM---CLEAR THE GRAPHICS DISPLAY
160 NWFR
170 REM---DRAW THE AXES
180 MOVA(.7,20000)
190 LINA(.7,8000)
195 MOVA(.7,8700)
200 LINA(13,8700)
203 REM---SET CHARACTER QUALITY FOR CHARACTER DISPLAY
205 SCQL(0)
210 REM---PLACE TICK MARKS FOR THE MONTHS
220 FOR I=1 TO 12
230 MRKA(I,8700,1)
232 MOVA(I,8000)
235 TEXT(STR$(I))
240 NEXT I
242 REM---SET CHAR QUALITY FOR GRAPHICS DISPLAY
244 SCQL(1)
245 REM---SET CHAR SPACE
247 SCSP(.3,0)
250 REM---PLACE TICK MARKS FOR SALES
260 FOR I=8000 TO 20000 STEP 1000
270 MRKA(.7,I,2)
271 MOVA(0,I)
275 TEXT(STR$(I/1000))
280 NEXT I
290 REM--- SET CHAR SPACE, USE DEFAULT CHAR SIZE, TO DRAW TITLE
300 SCSP(.3,0)
320 MOVA(1,21000)
330 TEXT('SALES ($K) VS TIME (MONTHS)')
340 REM---DRAW THE INFORMATION
350 FOR I=1 TO 12
360 READ A
370 IF I=1 THEN MOVA(1,A)
380 LINA(I,A)
390 MRKA(I,A,4)
400 NEXT I
410 STOP
```

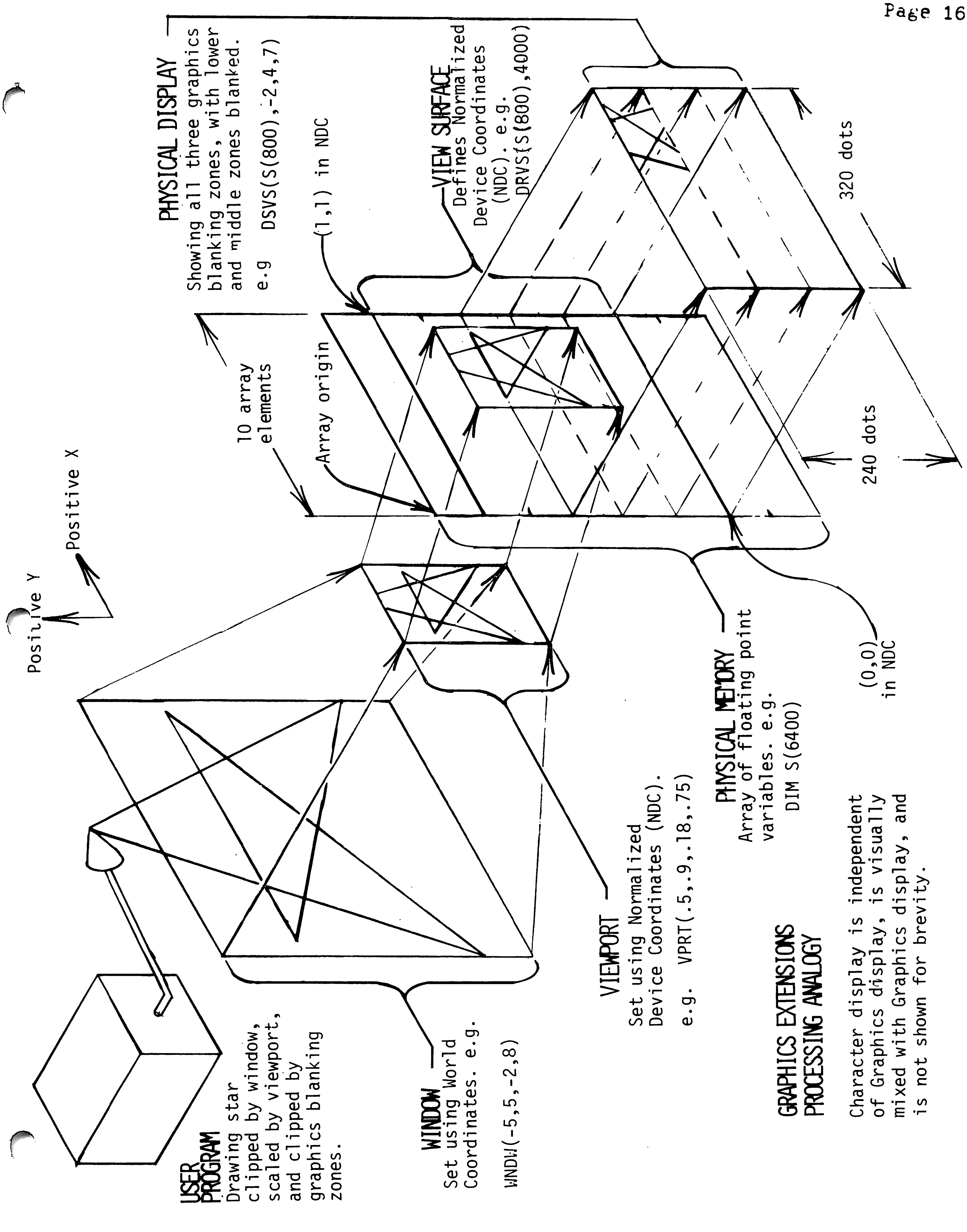
USER PROGRAM
Drawing star clipped by window, scaled by viewport, and clipped by graphics blanking zones.
WINDOW —
Set using World Coordinates. e.g
WNDW(-5,5,-2,8)
VIEWPORT —
Set using Normalized Device Coordinates (NDC).
e.g. VPRT(.5,.9,.18,.75)
GRAPHICS EXTENSIONS PROCESSING ANALOGY
Character display is independent of Graphics display, is visually mixed with Graphics display, and is not shown for brevity.
Positive Y
Positive X
10 array elements
Array origin
PHYSICAL MEMORY —
Array of floating point variables. e.g
DIM S(6400)
(0,0) in NDC
(1,1) in NDC
PHYSICAL DISPLAY
Showing all three graphics blanking zones, with lower and middle zones blanked.
e.g DSVS(S(800),-2,4,7)
VIEW SURFACE
Defines Normalized Device Coordinates (NDC). e.g.
DRVS(S(800),4000)
320 dots
240 dots

FIGURE 4