95

DEC/TR-~~8~~

FUNDAMENTALS OF THE REMOTE GRAPHICS INSTRUCTION SET (REGIS)

by

Charle' R. Rupp

RESEARCH AND DEVELOPMENT GROUP

REVISION 0 ... 8 April 1979
REVISION 1 ... 18 July 1979

Keywords:
Graphics
REGIS
ASCII Interface Standard

PREFACE

This document summarizes a collection of ideas concerning an instruction set for the communication of the definition of a graphic image between a computer and a peripheral graphic device. Although it is felt that this definition is unique in its ability to support a wide variety of dissimilar graphic devices in a wide variety of application environments, REGIS (which stands for the Remote Graphic Instruction Set) is felt by the author to represent a natural evolution of existing device specific interface protocols used on devices produced by several manufacturers.

The principal goals and assumptions of REGIS may be stated as follows:

1. All future graphic terminal devices will contain micro-processor computing elements and therefore are able to interpret a rich user- oriented language syntax with reasonable performance.

2. A majority of remote graphic devices communicate with a computer over serial communications lines using ASCII character codes. REGIS is therefore specifically designed to be embedded in the ASCII protocol environment and contains special features to efficiently utilize this communication media.

3. To minimize the number of different language syntaxes which users and system designers must work with, REGIS is specifically designed to be used both as a line protocol and a graphic file syntax. In addition, REGIS is sufficiently "English Level" oriented as to be useful as print string constants from a high level language which has no direct graphic device support.

4. REGIS is not meant to be a replacement for higher level application oriented graphic languages, such as the proposed SIGGRAPH CORE standard, but rather is meant to be used as the "object code" level language translation from such a higher level image definition. REGIS therefore specifically addresses the attributes of the graphic device itself and ignores such higher level language issues as viewing transformations and "real-world" coordinates.

5. REGIS's approach to instruction level code transportability is based upon the concept that the majority of the information content of a graphic image may be defined by a few common graphic instructions and that the specific attributes and capabilites of a graphic device may be transformed or ignored by a dissimilar graphic device without destroying the essential information content of the image drawn.

The purpose of this report is to present the fundamental concepts of REGIS and to discuss the underlying assumptions and tradeoffs which characterize REGIS. This report should not be considered as a definitive specification for REGIS but will hopefully serve as a common point of discussion in preparation for the development of a specification.

This report summarizes both formal and informal work carried out by a large number of individuals within the DEC organization and principally grew out of the discussions of the Common Graphics Command Set Committee. Special mention should be made of the efforts of the members in the Research and Development Department, the Computer Aided Design group and the Video Products Group which directly contributed to the content and substance of this report.

REVISION 1

This revision reflects refinements to the original document as a function of comments received from several DEC organizations and particularly the comments of Tom Powers, Don North and Bob Fleischer. The following list summarizes the areas of change:

1. Strings may be delimited by either the single quote "'" or double quote '"' character. A delimiter character may still be contained within a string by doubling.

2. Numeric constants may be delimited by the space character.

3. Macrographs may be redefined without first clearing.

4. A REGIS compatible device may have a default screen coordinate definition.

5. Pixel size adjust and multipliers now only have one parameter.

6. A guideline for the assumed size of "standard" text has been added.

7. Temporary writing options are now allowed in the vector, curve and text instructions.

8. The end point cursor position for circles and arcs has been clarified.

9. The "+" and "-" option enable/disable syntax has been changed to use numeric qualifiers instead.

10. A read instruction has been added to allow access to drawing process parameters for interactive devices.

11. The definition of color to grey-scale mapping has been clarified.

CONTENTS

FIGURES

1.0

INTRODUCTION

The Remote Graphics Instruction Set (REGIS) is a device level portable graphic image definition approach. Although meant principally to be used for the communication of graphic images between a host computer and a remote "smart" graphics device, REGIS also has applicability to tightly- coupled (bounded) graphics systems in any context in which a graphic image is stored as a sequence of characters.

This report describes the fundamental concepts of REGIS including the underlying assumptions and tradeoffs which have led to its development. The paragraphs in this section give a brief overview of REGIS in terms of the assumed system structure, the various uses of REGIS within that structure, the range of devices for which REGIS is designed to be used and illustrative examples of REGIS instruction sequences. Later sections discuss the philosophy of REGIS, the concept of a "logical graphic device" upon which REGIS is based and the specific syntax and semantics of the currently defined instruction set.

1.1

SYSTEM MODEL

Figure 1 illustrates the assumed general block diagram of the graphics system in which REGIS is meant to be used. As illustrated, the system consists of two major parts: (1) the host computer system which from the user's point of view consists of various software modules and (2) the remote graphics device. The term "remote" here refers to the assumption that the graphics device is attached to the host computer by a possibly low-speed communications link.

The host computer system is assumed to consist of four major parts (not all of which may be present in a particular system):

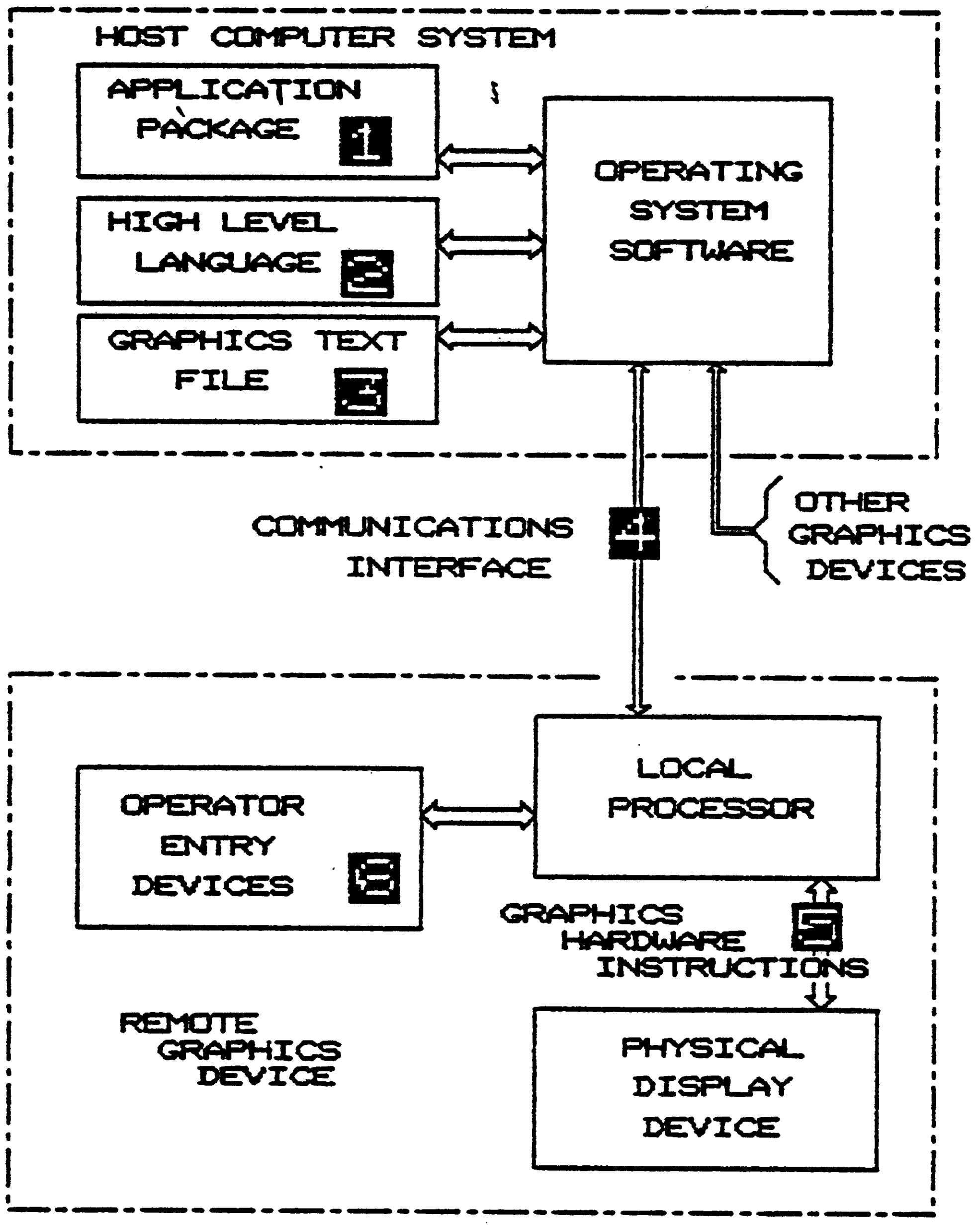
1. Graphics application package(s) - allow the user to access the graphics capabilities of the remote device in terms of syntax and semantics appropriate to the user's intended application. For example, a data plotting package which accepts tables of data and allows the user to construct plots of varying types based on that data. As a second example, a flowcharting package which allows the user to prepare a variety of flowchart diagrams for technical and managerial use.

2. High level language(s) - allow the user to generate graphic images of significant complexity requiring algorithmic definition and/or images requiring dynamic changes to the graphic image.

3. Text files - allow the user to store the definition of a graphic image generated by a high level program or application package and later cause that graphic image to be shown in the same manner as printing out a report file of standard text.

4. Operating system - may provide support to the graphics user in the general case by converting the graphic image definition provided by the other software to the required protocols necessary to drive one or more remote graphics devices.

Figure 1. GRAPHICS SYSTEM BLOCK DIAGRAM



The remote graphics devices are assumed to have the following basic structure:

1. Local processor - interprets the graphics image definition provided by the computer into a form which can be used by the hardware image generation part and transfer information concerning operator entries to the host computer.

2. Hardware Image Generator - the generally digital/ analog/ electro-mechanical portion of the graphics device which gives the visible graphic image. The pen-movement mechanism of a flat-bed plotter and the CRT tube with refresh memory circuit of a raster graphics device are examples of this.

3. Operator interface(s) - the device or devices in an interactive graphics system which allows the user to either communicate directly with the local processor for local operations or communicate with the application program(s) in the host computer.

In a bounded system, the communications interface is no longer visible to the user and the local and host processors generally collapse into a single processor. The other parts of the system remain the same as illustrated for a bounded system.

1.2

GRAPHICS LANGUAGE ACCESS POINTS

Based on the above assumed structure, the system has at least six language access points of interest to the user's and developers of such a system:

1. Graphics application languages - such as plot commands in a data plotting package or the constructors of physical images as represented by the proposed SIGGRAPH CORE graphics standard.

2. High level languages - allow access to graphics semantics in three possible ways:

    1. Direct drive of character based communication protocols by using string constants output through "PRINT" statements.

    2. Graphics sub-routines accessed by programmed calls.

    3. Embedded graphics statements accessed in the same manner as other language capabilities.

3. Graphics text files - the syntax of how the graphics image is stored as text for later ~~communication~~ communication to the graphics device.

4. Communications Interface - the syntax of the graphics commands sent sequentially to the graphics device.

5. Graphics hardware instructions - refer to the assumed digital signals generated by the local device processor to ~~the~~ control the hardware image generator.

6. User graphics entries - refer to the specialized entry sequences for local control in addition to the syntax of the user commands required to use the application software programs.

1.3

REGIS SCOPE OF APPLICATION

The natural question which arises based upon the above user language access points is "what range of application should a general purpose graphics instruction set be aimed?". It would of course be very nice if one instruction set could be found which satisfied the requirements of all points of the system but this is clearly impractical (if not impossible) considering the breadth of graphics applications and the differences in "personality" between the different language points. On the other hand, the relatively large number of different graphics language syntaxes which a user may have to be aware of also is undesirable motivating a desire to have a few standardized language approaches with special languages applied when dictated by application and hopefully based upon the existing standards. We will accept here (at least not argue against) the CORE standard for use in the application program and high level language sub-routine cases. It is the purpose of REGIS rather to attact the syntax and semantics at least of the communications interface graphics commands. By extrapolation, it is the desire of REGIS also to be used at other graphics language access points in the system which are principally based upon a character representation of the graphic image. In summary then, it is the intent of REGIS to be used for the following graphic language access points for both remote and closely-coupled graphics systems:

1. Graphics instruction string constants from high level languages whether generated by print statements or the compilation of high level language programs having embedded graphics statements.

2. The "semantic protocol" for graphcs access in a serial based communications line to a remote graphics device.

3. The syntax of graphics commands when stored in a character oriented file.

1.4

REGIS OVERVIEW

Having now establised the intended utilization of REGIS in a graphics system, the following paragraphs summarize without the detailed justification given in later sections the general syntax and semantics of REGIS instructions.

REGIS instructions are sequences of characters based upon an alphabet common to most computing systems. That is, the alphabet consists letter, digit and punctuation characters. A graphic image is drawn by sending the graphics device a sequence of INSTRUCTIONS all based on this alphabet where the general format of each instruction is a KEY-LETTER denoting the type of operation to be performed and an arbitrary number of instruction ARGUMENTS which have a prescribed meaning when associated with a KEY-LETTER. The six basic instuctions in REGIS are briefly defined as follows:

1. Screen Instruction - KEY-LETTER "S" - causes operations to be performed affecting the entire visible viewing area such as erasing the image in preparation of drawing a new image.

2. writing Attributes Instruction - KEY-LETTER "W" - causes all the following drawing instructions to costruct the image using the attributes (line pattern, color and so forth) given by the arguments of the instruction.

3. Position Instruction - KEY-LETTER "P" - causes the drawing process to commence from a specific horizontal/vertical position in the drawing area.

4. Vector Instruction - KEY-LETTER "V" - causes a straight line to be drawn between the current position and a new position specified by the instruction arguments.

5. Curve Instruction - KEY-LETTER "C" - causes a circle, arc, or general curve image to be drawn based upon the arguments of the instruction.

6. Text Instruction - KEY-LETTER "T" - causes a sequence of text characters to be drawn in a manner based upon the arguments of the instruction.

There are four general types of arguments used in REGIS:

1. Position arguments are strings of characters written within the bracket characters "[" and "]" which indicate changes to the horizontal and vertical writing position.

2. Option arguments are strings of characters written within the parenthesis characters "(" and ")" which in some manner modify the way in which the instruction carries out its intended purpose. The syntax of the option strings is the same as for instructions and for this reason, option arguments are also called sub-instructions.

3. Text strings are character sequences contained within the single quote character "'" and are used when a character is itself an argument of an instruction as in the case of the TEXT instruction.

4. Digit characters refer generally to the eight neighbors of a specific position.

The principle hypothesis of REGIS is that these six basic instructions and thses simple arguments are sufficient to define most of the information of most of the graphic images of interest to most of the users of graphics systems. The fact that instruction set is terse in nature is based upon the desire for a relatively efficient coomunications protocol and that it is assumed that most graphic images will prepared using application programs. REGIS as a language is still high enough in level that the graphic image definition is still not completely critical and so specific that different graphic devices have some degree of freedom in interpreting the instructions to maximize their own features and thus allow a reasonable chance of graphic image portability between one device and another.

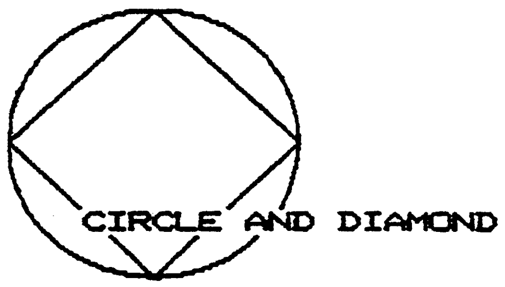
1.5

EXAMPLE IMAGE GENERATION

Figure 2 illustrates a simple graphics image consisting of a diamond ,circle and graphics text. The second part of figure 2 illustrates the REGIS instructions required to generate this image. Note that a REGIS instruction generally may have several arguments.

Figure 3 illustrates program sequences illustrating the source for the generation of these REGIS instructions based upon the three basic approaches to accessing graphics from a high level language (in this case, a language having the personality of BASIC) and the program sequences for generating the image based upon the SIGGRAPH CORE graphics standard. It is surprising that the direct use of REGIS in PRINT statements has comparable "readability" to the other approacnes. This is deceiving however due to the simplicity of the image drawn. REGIS has no viewing transformation, three-dimensional image and segment capabilites as found in the SIGGRAPH CORE and of use in higher level applications work. The assumption of REGIS is that these capabilities are translatable to sequences of REGIS instructions.

Figure 2. EXAMPLE GRAPHICS IMAGE



REGIS instructions for the above image;

```
P[100,100]
V[200,0][300,100][200,200][100,100]
P[200,100] C[+100]
P[150,150] T(S2) 'CIRCLE AND DIAMOND'
```

Figure 3. EXAMPLE APPLICATION LEVEL GRAPHICS SEQUENCES

a. Using REGIS Directly in PRINT statements

```
100 PRINT G1$, "P[100,100]"
110 PRINT "V[200,0][300,100]"
120 PRINT "V[200,200][100,100]"
130 PRINT "P[200,100]C[+100]", G0$
```

Note: The string constants G1$ and G0$ refer to character sequences which are defined by some mechanism at the begining of the program and cause the graphics process to be turned on and off in the graphics device.

b. Using Graphics Sub-routines

```
100 CALL POSITION(100,100)
110 CALL VECTOR(200,0)
120 CALL VECTOR(300,100)
130 CALL VECTOR(200,200)
140 CALL VECTOR(100,100)
150 CALL CIRCLE(200,100,100)
```

c. Embedded Graphics Commands

```
100 DRAW 100,100;200,0;
110 DRAW 300,100;200,200;100,100
120 DRAW 200,100 CIRCLE 100
```

d. SIGGRAPH CORE Statements

```
MOVE_ABS_2 (100,100)
LINE_ABS_2 (200,0)
LINE_ABS_2 (300,100)
LINE_ABS_2 (200,200)
LINE_ABS_2 (100,100)
POLYLINE_ABS_2 (X,Y)
```

Note: The arrays X and Y are assumed to contain the coordinates of closely spaced points on a circle as would be generated by some program segment.

1.6

RANGE OF INTENDED DEVICES

REGIS is ambitious in its attempt to cover a broad range of both "soft-copy" (temporary record) and "hard-copy" (permanent record) remote graphics devices. A possibly incomplete list of such devices include:

1. Flat-bed monochrome and color plotters
2. Drum monocnrome and color plotters
3. Photo-plotters including computer output microforms
4. Monocnrome and color raster CRT devices
5. Storage CRT devices
6. Stroke graphics devices
7. Flat-panel display devices such as plasma panels
8. Matrix, thermal and electrostatic hard-copy devices

The principal concept of the transportability of REGIS based images is that the essential information content of an image can be encoded in the basic set of instructions and that little or no essential information content is lost if attributes of particular device are transformed (or even ignored) by a device which does not have a particular attribute capability.

2.0

REGIS PHILOSOPHY

This section describes and discusses the rationale of REGIS in terms of (1) the approach to allowing an image originally defined for one graphics device to be drawn on a dissimilar graphic device, called the instruction set "transportability" (2) concepts of the instruction syntax and the orientation to the ASCII character code conventions and (3) concepts of handling dissimilar device attributes and capabilities, referred to as the "semantics" of REGIS.

2.1

Transportability Concepts

The desire of image definition transportability is to allow a sequence of graphic instructions to draw similar looking images on dissimilar graphic devices. The following paragraphs summarize the REGIS approach to this transportability.

2.1.1 APPLICATION FREE PRIMITIVES

REGIS implements a set ~~a~~ of drawing primitives based upon common geometric rule and compass constructions and should not contain any specific capabilities which can be interpreted in terms of the semantics of the application. For example, REGIS does not include viewing transformation, hidden line removal and other capabilities generally useful in presenting an image of real physical things. Instead REGIS assumes that these capabilities are adequately supported by higher level languages which simulate these features by a sequence of REGIS instructions. In this way REGIS does not burden an application which works with more abstract images such as a data plotting package. In a similar manner, REGIS does not have built in axis and bar-graph primitives to support data plotting for the same reason.

2.1.2 MANDATORY BASE IMPLEMENTATION

REGIS instructions are based on a general instruction syntax which all devices are required to "parse". That is, every device using the REGIS instruction set must completely interpret all possible character sequences defined by this general instruction grammar, whether those character sequences mean anything or not. Semantically, REGIS instructions and parameters are separated into a REGIS BASE category, a STANDARD EXTENSIONS category and a REGIS OPEN category. In concept, most of the essential information content of a graphic image is definable in terms of the REGIS BASE instructions and parameters. All REGIS based graphic devices are required to interpret these BASE capabilities as faithfully as possible. In addition, users who know ahead of time that the image they are defining is to be used on several different devices are encouraged to use only the BASE REGIS capabilities to ensure maximum transportability. For example, color is not a common attribute of every graphic device, and for this reason the REGIS BASE is "color-blind". This works out well in practice because color does not normally carry essential information content. Put differently, users should avoid using color to relate essential information in the context that that same information is not also implied by monochrome boundaries or patterns.

There are several categories of STANDARD EXTENSION capabilities, including very high resolution devices (such as photo plotters) and raster scan display devices (including color CRT displays). The intent here is the same as for the REGIS BASE. That is, all CRT GRAPHIC devices of approximation the same resolution are expected to implement the full range of the STANDARD RASTER EXTENSIONS.

This concept works because all devices are required to fully parse the general instruction syntax. Therefore, a capability in one of the extension sets is simply ignored or transformed to a simpler attribute in a device for which that capability does not exist.

2.1.3 FIDELITY OF INFORMATION PRESENTATION

Devices are allowed varying levels of fidelity in the presentation of information implied by REGIS instructions on the basis of maintaining reasonable performance while maximizing the translation of an instruction sequence. The general rule for determining the allowed degree of loss of fidelity are as follows:

1. Low resolution /monochrome devices are allowed the greatest degree of latitude in approximation the information content. In the limit of a drawing a very detailed image with a large information content (exceeding the information content of the simple device) essential information may be lost.

2. Higher resolution (and presumably more expensive or more specialized) devices are expected to implement the highest degree of fidelity of information presentation.

One of the reasons that this approach should work is the assumption that images are prepared on lower resolution for later output on higher resolution (possibly slower, usually more expensive) devices or that a high information content image is prepared directly for the high resolution device. Rarely would an image prepared on a high resolution device be drawn on a substantially lower resolution device except perhaps for "quick-checking" or otherwise the user must be willing to accept the possible misinterpretation of information as a result of the lower fidelity of reproduction.

2.1.4 CRITICAL DEVICE PARAMETERS

The third major concept in allowing maximal image transportability is the use of critical setup parameters defined once for each image and placed at the beginning of the instruction sequence for that image definition. Thus; in the worst case the user might have to adjust these parameters to achieve maximum fidelity in the reproduction of a graphic image. As an example of such a critical device parameter, REGIS allows the user to set the range of X-Y coordinates to be used for an image definition. These presumbly would be set to the actual physical range values of device which the user mostly uses. These range parameters could be adjusted to maximally utilize a dissimilar low resolution device but presumably would not have to adjusted for playin the same image on a high resolution device.

2.2

SYNTAX CONSIDERATIONS

2.2.1 Character Codes

REGIS is based on the common ASCII-96 or ASCII-128 character alphabet. To support environments which have only the ASCII-96 character set (or have a character set such as BCD which is not- translatable to the ASCII-128 character set) REGIS does not differentiate between upper and lower case letters. As a matter of convention, REGIS will be defined in terms of upper-case letters with the understanding that lower case letters will be converted to upper case for interpretation. The exception to this rule is that lower case letters will remain lower case in a quoted string for environments which do support the ASCII-128 alphabet. The selection of ASCII for REGIS instructions is based first on the desirability that REGIS instructions be human- readable (at least to the degree that assembly language code is readable) and secondly on the presumption that a majority of systems which use graphics either use ASCII directly as their base text code or provide some easily accessible means for the translation of their internal code (such as EBCDIC or BCD) to ASCII.

2.2.2 ASCII Control Codes

To avoid as many problems as possible when using REGIS in an ASCII based environment, REGIS does not use any ASCII control codes. All REGIS instructions, parameters, and extensions can be written using only the printable character sets from ASCII-96 or ASCII-128.

2.2.3 Syntactic Relation to Higher Level Languages

The REGIS general syntax is based on a simple easily parsed grammar which should make it easily generated by common high level languages and user application packages. In particular, the REGIS instruction types have been kept very close to the SIGGRAPH CORE to allow simple and straightforward implementation.

2.2.4 Use in "PRINT" strings

The level of REGIS instructions and parameters has been set high enough that the use of REGIS directly in the PRINT strings of such languages as BASIC, FORTRAN, and PASCAL on systems which do not directly support graphic devices allows reasonably clear (although admittedly terse) coding. In particular, number arguments are encoded as the usual decimal integer, fixed-point, and floating- point (scientific notation) strings generated by these languages.

2.2.5 Communications Line Efficiency

The level of REGIS instructions has been set low enough so that reasonable performance may be obtained even at low transmission rates. It is for this reason that the "macrograph string" has been included in the REGIS base to allow~~ing~~ repetitive string sequences to be defined once and then referred to by name any number of times. It is for this rason also that REGIS is generally free format and requires no~~t~~ instruction separators and allows multiple instructions per "line" (in fact, REGIS has no awareness of line boundaries, since ASCII control characters are ignored).

2.2.6 High-level Language "Personality"

Although it is not possible to define a completely compatible personality amongst all higher level languages, REGIS has attempted to capture the essence of many higher level language personilizations to allow the most rapid learning by users with minimal interchanges of syntactic constructs. Common numeric and string contant constructs , the ignoring of blanks and auto-conversion of lower case to upper case are examples of the implemetation of this concept.

2.3

Semantic Considerations

2.3.1 Range of devices

REGIS is intended for use on the following range of graphic display devices:

1. Low to high resolution flat-bed plotters
2. Low to high resolution rotary plotters
3. Above plotters with binary or full-tone color
4. monochrome binary raster CRT divices
5. Full-tone and color raster CRT devices
6. Storage CRT devices
7. Stroke graphics devices
8. monochrome Plasma devices
9. "Future" x-y addressable display devices such as LED arrays, LCD displays and Electro-luminescent displays.

It is felt that the REGIS BASE instructions adequately defines a common denominator of these devices, and that the EXTENSIONS and OPEN categories adequately access the unique capabilities of current and future devices.

2.3.2 Position Addressing

A specific point in the viewing area of a graphic device is assumed to be representable by an X-Y (horizontal-vertical) pair of numbers and that the viewing area is rectangular in shape for the purpose of code transportability. The size of an X-Y increment is assumed to be defined by the device itself in the REGIS BASE (if indeed the device is able to define it at all!) and otherwise definable by user supplied parameters in the REGIS extensions.

2.3.3 Semantic Defaults

There are generally no default parameters in the REGIS BASE. In fact, all parameter ranges required by a device are derivable from user supplied screen coordinates which are transformed to physical coordinates by the device. An exception to this is the size of "standard characters" in the device which will naturally vary as a function of physical design decision . To accommodate such variances, REGIS allows the user to "adjust" these parameters so that the actual visual result is as close as possible to what the user expected.

Devices which implement REGIS extensions are required to implement default values for all such semantic features which affect the visible display. These defaults will approximate as much as possible the visual impression of the image on a device which does not have these visual enhancements and these defaults must be assumed each and every time the screen clear operation is performed. For example, the default writing pen for a plotter is black (although it is white or green for a CRT device!) as opposed to some hue capability (unless of course the divice naturally only has a single hue writing capability .. green for storage tubes and orange for plasma displays).

2.3.4 Range of application

For the purpose of this report, graphic images will be arbitrarily broken down into three categories:

1. Data plotting

2. Presentation graphics

3. Image processing

The first two categories are sometimes referred to as "data representation graphics" and the later category referred to as "inherently graphic". The intent here is to establish a framework for the range of applicability of REGIS. Contrary to any pre-conceived (or real!) notions of relative complexity of the above categories, we will assume that the list represents increasing complexity in the order given and presumably increased cost. Although REGIS may adequately handle the first category, its principal target is the second category which includes mainly abstract images such as flow-charts and block-diagrams. This does not mean that REGIS can not be used for data-plotting but only that it may lack the performance necessary for this more restricted category. Similarly, REGIS can be used as the protocol base for complex CAD and real-world imagery (ERTS photo processing, cartooning and so forth) but also would be found to be less than desired in these cases particularly if no higher level language support is provided.

3.0

THE BASE LOGICAL GRAPHIC DEVICE

REGIS is used to define an image for an abstract graphic device called the REGIS Logical Graphic Device. In principle, this abstract device is a composite of a wide range of physical devices. Features unique to a specific type of physical device are part of the extended REGIS logical device and are discusse in later sections. In concept, since an image is defined in terms of this abstract device, an image defined for one physical device is transportable to a different type of physical device with the exception that feature extensions are ignored on one device or the other or approximated in a different way on one device versus the other. It is not expected that any one specific device have the parameters of the logical device, but rather each REGIS based physical device maps (transforms) the parameters given in terms of the logical device to its specific physical parameters.

The REGIS logical device is defined in terms of:

1. Parameters of the viewing area

2. Attributes of viewing points in the viewing area

3. The general process of defining an image in the viewing area by modifying the attributes of the viewing points (REGIS by nature is a serial drawing process .. the visible portions of the image are drawn in the order in which the instructions are received).

4. The allowed range of parameters and attributes

The phrase "implementation dependent" will be used to refer to a parameter or attribute value that has a semantic meaning which varies with the type of physical device or with personal preference. The allowed range of variance for an implementation dependent feature will be defined in more detailed later.

3.1

VIEWING AREA DEFINITION

The REGIS Logical Device viewing area consists of a generally rectangular grid of a finite (although possibly large) number of viewing-point. Each viewing point is of finite area and bounded on four sides by the invisible lines of grid. Usually, a viewing point is the smallest physical area with homogeneous structure that can be seen by the viewer and normally corresponds to the smallest area which can be modified by the physical device. Because the grid lines break up the image in the same manner that a window screen breaks up an outdoor scene, the viewing area will also be called a "screen". Mathematically, the screen area may be thought of as a set of viewing- points such that the union of all viewing-points completely covers the viewing area. A viewing point may also be referred to as a picture element or "pixel" for short (sometimes referred to by the yet shorter nickname "pel").

When the screen is rotated in such a manner that the image has its "normal" orientation, then the four edges of the rectangular screen are called the left, right, top and bottom edges in accordance with the normal meaning of these words and the terms horizontal and vertical are used to refer to pixel positions relative to these four sides also in accordance with the normal meaning of these words. The position of each pixel on the screen is uniquely identified by the combination of two numbers which refer to the horizontal (or X) position and the vertical (or Y) position of the pixel. Using this the pixel (viewing-point) VP(X,Y) refers to the viewing point at the horizontal value denoted by the number X and at vertical value denoted by the number Y.

Using REGIS instructions, the user may define the possible range of values of X and Y by identifying the numeric values which are taken on by X and Y at the four edges. Without loss of generality, these parameters may be assumed to be non-negative and directly define the following screen area parameters:

SYT	Value of the Y-position at the top of the screen area
SYB	Value of the Y-position at the bottom of the screen area
SXL	Value of X at the left edge
SXR	Value of X at the right edge

Assuming for the moment that these parameters are all of integer value, these parameters indirectly define the total number of horizontal and vertical positions to be used as follows:

```
SXN = abs(SXR-SXL)+1
    = the number of horizontal screen
        positions

SYN = abs(SYT-SYB)+1
    = the number of vertical
        screen positions

STN = SXN * SYN
    = the total number of unique screen
        positions (number of pixels)
```

These parameters can now be used to define three ranges of graphic device "resolution" (here used in the context of referring to quantity of information capability and not the quality of the individual pixels) as follows:

1. LOW RESOLUTION
   MIN(SXN,SYN) <= 256

2. MEDIUM RESOLUTION
   256 <= MIN(SXN,SYN) <= 1024

3. HIGH RESOLUTION
   MIN(SXN,SYN) > 1024

REGIS differentiates between the low/medium class of devices and the high resolution devices in a special way. Basically, screen positions for low resolution devices and definable entirely by using integer numbers or the whole parts of generally fractional numbers. The additional resolution of high resolution devices is required to be accessed by fractional parts of the X and Y values. In this manner, the simpler low resolution devices may easily "approximate" the positions on high resolution devices by simply taking the integer parts of all numbers and similarly, a high resolution device may depict an image with better fidelity by making use of the fractional parts of the values. Medium resolution devices may use some part of the fractional values in addition to the whole parts on an implementation dependent basis.

We have now established a reasonable frame-work for rules governing the degree of fidelity of representing a graphic image:

1. LOW RESOLUTION DEVICES are given the greatest lee way in the degree of fidelity of representation

2. MEDIUM RESOLUTION devices should attempt to get as close as possible to the ideal, recognizing that there still may be "jagged edges" occasionally

3. HIGH RESOLUTION devices are expected to represent the image with the maximum possible degree of fidelity.

Note that the REGIS BASE logical device is ignorant of the real physical size of a pixel, consistent with the fact that most low resolution graphic devices have no controllable pixel size (as in the case of raster CRT, stroke CRT, and storage CRT devices). Other devices must therefore define default values of physical pixel sizes from the given data in conjunction with a determination of the resolution of the original device as determined by the above paramters.

Note that the parameter SXR is not necess rily greater than SXL and similarly, SYT is not necessarily larger than SYB. To define the REGIS instructions in terms of the logical device, it will be necessary to be able to identify the pixels which are above, below, to the left and to the right of a given pixel, in terms of the user supplied screen coordinate definition. This is accomplished by the X,Y screen increments SXINC and SYINC which are defined as follows:

SXINC = SGN(SXR,SXL)

SYINC = SGN(SYB,SYT)

where SGN( ) is the signum function. By this convention:

```
VP(X+SXINC,Y) is to the right of VP(X,Y)
VP(X-SXINC,Y) is to the left of VP(X,Y)
VP(X,Y+SYINC) is below VP(X,Y)
VP(X,Y-SYINC) is above VP(X,Y)
```

3.2

VIEWING POINT ATTRIBUTES

Each viewing point generally has a foreground intensity attribute, a background intensity attribute and a foreground-background selector attribute as defined in the following paragraphs.

Each pixel has a background "intensity" value called the BI(X,Y). The semantic meaning of the word intensity is implementation dependent in the sense fo psychophysical and photometric measurement units. Intensity in the BASE logical device is defined simply as being some visually discernible property of a viewing point such that a pixel have one intensity value is separable from another pixel having a different intensity value. The BASE REGIS LOGICAL DEVICE IS BOTH COLOR-BLIND AND BINARY in terms of intensity attributes, so without loss of generality, we may assume that the possible values BI(X,Y) are either "off" or "on".

The foreground intensity attribute associated with the pixel at position X,Y is denoted by FI(X,Y) and similarly in the Base logical device may have values of "on" or "off". Because of the binary nature of the base device, the FI and BI quantities are redundant, but are included here for completeness since they will be needed in the definition of the extended logical device.

Each viewing point has a foreground-background selector function denoted by FBS(X,Y,t) which has a binary value which is generally a function of time. Without loss of generality, the values of FBS will be taken to be "F" (for foreground selection also denoted by the binary value "1") and "B" (for background selection also denoted by the binary value "0"). The FBS funtion for each VP(X,Y) is used to select whether the currently defined foreground or the currently defined background attribute is to be seen. In the base logical device, we may take the value of BI(X,Y) to off for all (X,Y) and the value of FI(X,Y) to be on for all (X,Y). Then the set of all VP(X,Y) for which FBS(X,Y) has value "F" ("1") will be visually discernible as an impsosed on a uniform bacground intensity.

3.3

GENERAL DRAWING PROCESS

The generation of an image on the logical device (as distinguished from the definition of the image) consists of executing a sequence of instructions which modify the FBS values for a selected subset of X,Y values. Note that the sequence of instruction execution is important in REGIS whereas the order of image definition constructs in a higher level graphic language may not be important. The general~~ly~~ process of generating a REGIS logical image is as follows:

STEP 1: Define the viewing area parameters.

STEP 2: Erase the viewing area

STEP 3: Identify a sequence of X,Y positions and change the FBS for each of these pixels.

An image may be MODIFIED by repeating step 3. A new image may be drawn by repeating from step 2.

In terms of the different REGIS instructions, the above steps are accomplished as follows:

The SCREEN instruction is used to define the viewing area parameters, set the background attributes and erase the screen image.

The WRITING attributes instruction is used to set the foreground attributes, particularly line drawing patterns.

The POSITION instruction is used to select an X,Y pixel position to ~~be~~ begin writing points.

The VECTOR, CURVE, and TEXT instructions are used to modify a sequence of FBS values for pixels from the current X,Y position and generally ending up at a different position of the screen.

4.0

REGIS GENERAL SYNTAX

This section discusses the general syntax and semantics of REGIS. This discussion applies not only to the BASE REGIS instructions discussed in the next section but also provides the general framework for extending REGIS by way of the REGIS general grammar. That is, REGIS can support for the purpose of transportability any extension as long as it conforms to the general grammar rules.

Since REGIS is meant to be used in conjunction with high level languages, extensions to REGIS should not include the more traditional "program structure" features such as conditional control and computations .. REGIS in itself is not a programming language.

4.1

Alphabet

REGIS is based principally on the ASCII-96 and ASCII-128 alphabets but for the purpose o maximum utilization is restricted to use only those characte which are common to BCD and EBCDIC alphabet systems as well. The REGIS alphabet therefore consists of the following characters:

<letter>	A,B,...,Z
<digit>	0,1,...,9
<punctuation>	[ ] , ; . ( ) ' " : @ + -

It is hoped that extensions to REGIS conform to this alphabet as much as possible. In addition, the following alphabet conventions apply:

1. Lower case letters a, b, ... z may be used but are treated as upper case letters outside of the range of a quoted string. Devices which do not have lower case support may also convert lower to upper case in quoted strings.

2. The parenthesis characters "(" and ")" should be reserved for option sequences outside of quoted strings.

3. The bracket characters "[" and "]" should be reserved for position arguments outside of quoted strings.

4. The semi-colon character ";" is reserved for use as an instruction separator. That is, the occurrence of the ";" character will always terminate and instruction whether the instruction parameters are complete. The ";" character does not terminate a quoted string construction.

5. The "at" character "@" is reserved for use by macrograph strings and may not be used in any other context outside of a quoted string. Within quoted strings the "@" character is just another printing character .. that is, there is no ability to expand macrographs within quoted strings.

6. The blank character " " is generally ignored outside of quoted strings. The exception to this is that blank characters are not allowed within the extent of numeric constant strings. Thus, the blank character can be used to delimit successive numeric parameters.

7. The comma character "," serves generally as an argument separator and is generally ignored unless the adjoining characters are otherwise indistiguishable as in the case of two adjoining digit sequences which refer to two different numbers.

8. ASCII control characters and similar communications line control and device control characters are generally ignored in REGIS and in concept can be inserted anywhere without affecting the interpretation process.

4.2

General Grammar

Figure 4 illustrates and summarizes the meta-language notation which will be used to describe the syntax of REGIS instructions. This meta-notation is the commonly employed extension to the Backus Naur Form or Backus Normal Form meta-notation originally developed for use in defining the ALGOL language.

Figure 5 summarizes the REGIS general grammar. The following paragraphs state in words what this figure more succinctly defines in mathematical terms.

1. A REGIS image definition consists of an arbitrary length sequence of intsructions possibly (but no necessarily) separated by the semi-colon character. Any character which does not begin an instruction (a <letter> character) is ignored.

2. A REGIS instruction consists of a KEY-LETTER followed by a parameter list. REGIS is therefore limited to having only 26 different types of instructions. Since there are only six key-letters used in the REGIS BASE this is not a significant restriction and leaves considerable room for extension.

3. An instruction parameter list consists of an arbitrary number of four basic types of parameters arranged in an arbitrary order syntactically. Semantically, all instructions and instruction parameters are assumed to be

executed in the order ("left-to-right") of the reception of those instructions and parameters.

4. A sequence of characters enclosed in the bracket characters generally refers to an X/Y position argument.

5. A sequence of characters which begins with the apostrophe character "'" and ends with an apostrophe character is a text string and treated as a single argument. The apostrophe characters themselves are not part of the text string. If an apostrophe is to be included in a text string then it is doubled. The full quote character '"' may also be used as a string delimiter in the same way as the single quote character. A string which begins with a " character is not terminated by the ' character and similarly a string which begins with the " character is not terminated with the ' character. In this way the, the " character can be included in a string without being doubled by simply using the single quote character as the string delimiter and similarly the single quote character can be included in a string delimited by the full quote character. For example:

   'a''C' refers to the string a'C

   '''' refers to the string '

   "A" refers to the string A

   '"' refers to the string "

   "'" refers to the string '

   "A'""B" refers to the string A'"B

   REGIS instructions may have multiple string parameters. In the this case the following ambiguity may occurr. If 'ABC' and 'DEF' are two consecutive string parameters, then the resulting structure is:

   'ABC''DEF'

   which will be treated by REGIS as the single string ABC'DEF. This ambiquity may be avoided by separating possible consecutive string paramters by the comma character. This may arise particularly in the context that macrographs are used to supply the string parameters.

6. Digit characters represent either numbers or sequences of arguments where each argument is a digit character is an argument.

7. Sequences of characters enclosed within the parenthesis characters "(" and ")" refer to instruction options. The syntax of each option is the same as the syntax of a REGIS instruction including parameters of the four basic types. For readability, options (also called sub-instructions) may be separated by comma characters but not by the semicolon

character which would terminate the instruction.

Figure 5 also summarizes the semantic meaning of the six REGIS BASE instructions.

Figure 4. META-LANGUAGE NOTATION

The syntax of REGIS is defined in terms of the extended Backus Normal Form (BNF) meta-syntax, summarized in the following:

< >	Enclose the name of syntactic variables (a quantity to be replaced by a string of ASCII characters)
-->	means that the syntax variable of the left is defined in terms of one of the string forms on the right
{ }	Means that the enclosed string may be repeated any number of times, or a selection is made from the enclosed string forms
|	Select one of the alternative forms of the left or right of this symbol
. .	Enclose the name of a single character which other wise has no printing representation

All other characters represent themselves.

For example, tne the following syntax equation:

```
<A> --> A<B>
    --> .cr.{B}
```

means that the syntax variable <A> is defined ("can be replaced by") either a string of characters beginning with "A" and followed by a string of characters definable by the syntax variable <B> or that <A> is defined by a string starting with the carriage-return character (.cr.) and followed by any number of "B" characters.

Figure 5. GENERAL SYNTAX

```
<REGIS>          --> <INSTRUCTION> <REGIS>
       --> ; <REGIS>
       --> <any character> <REGIS>

<INSTRUCTION>  --> <KEY> <PAR-LIST>

<KEY>  --> <letter>

<PAR-LIST> --> '<text> <PAR-LIST>
          --> "<text2> <PAR-LIST>
          --> (<OPTION-LIST> <PAR-LIST>
          --> [<x-y argument>] <PAR-LIST>
          --> <digit> <PAR-LIST>

<OPTION-LIST>  --> <KEY> <PAR-LIST> <OPTION-LIST>
               --> , <OPTION-LIST>
               --> )

<text> --> '
       --> <any other character> <text>

<text2>--> "
       --> <any other character> <text>
```

4.3

MACROGRAPH STRINGS

Macrograph strings provide the ability for the user to define a commonly used character sequence to be stored temporarily in the graphics device and then refer to this string anyplace in the sequence of instruction execution. Although macrograph strings allow the user to collect together several complete instructions and then execute these instructions several times during the generation of a single image or to be used in the generation of each of several separate images, it is not the intent of macrograph strings to serve as the commonly employed sub-picture structure or SIGGRAPH CORE segment structure. Rather, macrographs strings are syntactic mechanism for reducing the number of characters that have to be transmitted over the communications interface. For example, if the image being drawn is a sheet of music, each of the different types of notes could be defined as a macrograph string and then referred to by its name resulting in as much as a 5 to 1 reduction of communications overhead without substantially reducing the readability of the image definition (and possibly improving the readability).

Macrograph strings conform to a grammar built on top of the REGIS general grammar. That is, in the process of parsing REGIS instructions, the detection of a macrograph string reference causes the characters previously defined for that macrograph to be substituted for the string reference characters. Thus, in the general case a macrograph string may be just an argument of an instruction (such as a position argument) or may even be a piece of an argument (such as an argument to a sub-instruction).

All operations relating to macrographs are initiated by the "at" character "@". The syntax of macrograph operations is defined in figure 5 and described in words in the following:

1. There are 26 macrograph strings altogether which can be defined. Each macrograph string is identified by a single letter (lower case being converted to upper case for the purpose of macrograph string identification).

2. The macrograph strings are cleared by the character sequence "@." ("at" followed by period). The operation of clearing a macrograph string is interpreted as meaning that the string is defined to consist of no characters (the empty string). Macrograph strings must be cleared before any macrograph string may be defined and used. Macrographs do not have to be cleared if they are not to be used.

3. A macrograph string is defined by the sequence:

   ```
   @:<letter><STRING>@;
   ```

   where <letter> refers to any alphabet character, <string> is any sequence of characters not containing "@;" or "@:". The sequence "@:" is called the string definition initiator and the sequence "@;" is called the string definition terminator.

4. A macrograph string is referred to by the sequence

   @<letter>

   where <letter> is one the names given a previously defined macrograph. This sequence may appear anyplace in a sequence of REGIS istructions.

The following notes refer to conventions to be applied to achieve maximum transportability of images which use this capability:

1. A macrograph string can not contain the definition of another macrograph string (no "conditional" definition).

2. A macrograph string definition may refer to other macrograph strings in a nested manner. For maximum transportability, this capability should not be used for more than one level. A macrograph string may not refer to itself in its definition either directly or indirectly (no recursive string definition capability).

3. If a macrograph is defined which already is defined, then the old definition is replaced by the new definition and the old definition is lost.

4. Macrograph string may be of any length (including no characters at all .. the null string) but for maximum transportability, the sum of the character lengths used for all macrographs should not exceed 2000 characters. Some systems may have difficulty transmitting character sequences longer than 80 characters.

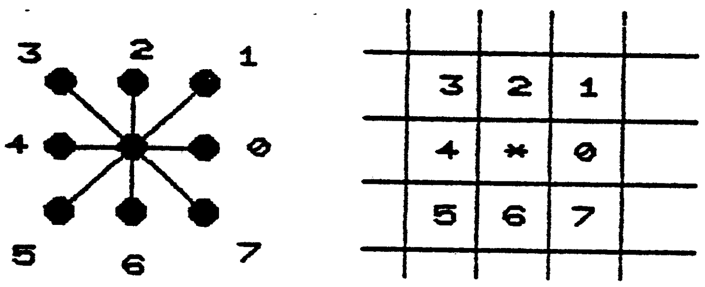
4.4

POSITION ARGUMENTS

REGIS provides two syntactic structures for the definition of drawing positions: (1) X/Y coordinates and (2) pixel-vextor (pv) relative movement. Pixel-vectors correspond roughly to the old (but still very useful) "chain-encoding" technique which is still used today to drive many incremental plotting devices. The X/Y coordinate structure is the somewhat tradition system of defining positions using two numbers separated by a comma (or space) character with a new twist which allows both absolute and relative coordinates to be syntactically distinguishable and thus eliminating the need to have separate relative and absolute instructions or options.

4.4.1 Pixel Vectors

Syntactically, a pixel-vector is denoted by a single digit character in the range 0 to 7. Each of these numbers refers to one of the eight pixel neighbors relative to the current position. By convention, pv number 0 refers to the right, pv number 1 refers to the pixel up and to the right and so forth around a circle in a counter-clockwise direction. This convention is summarized in the following diagram:



The following notes apply the use of pixel-vectors:

1. The character sequence:

   0226

   refers to a sequence of four pixel vectors and not to the number two-hundred and twenty-six.

2. Note that pixel vectors allow only relative positioning and no absolute position.

3. The size of a pixel vector is by default the unit size defined by the screen coordinate setup operation (defined in the next section) and is modifiable by a multiplicative factor by the writing attributes instruction to be a multiple of the base unit pixel vector.

4. The physical length (linear measure) of the pixel vectors may vary in one direction versus another on a device dependent basis particulary the diagonal directions.

5. Depending on the granularity of the unit pixel size defined by the screen coordinate setup operation, the exact position after executing a sequence of pv's is subject to roundoff error and thus pv positioning sequences should be broken occasionally by absolute positioning.

4.4.2 PARAMETER POSITIONING

The general syntax of a position parameter argument for a REGIS instruction is as follows:

```
<PA> --> [<xpart><ypart>]

<xpart> --> <coordinate>
<ypart> --> ,<coordinate>
        --> (nil)

<coordinate> --> <n>
             --> +<n>
             --> -<n>
             --> (nil)
```

The meaning of these forms is as follows:

1. A position argument generally consists of an x-position part and a y-position part, either or both of which may be missing.

2. Each coordinate part consists of a number possibly preceeded by a plus or minus sign ("+" or "-").

3. If a coordinate part is not preceeded by a sign character, then that argument is an absolute coordinate meaning that the value of that part is changed to be this new number independent of the old value of that coordinate.

4. If a coordinate part is preceeded by a sign character, then that part is changed relative to its current value. That is, the coordinate part is increased or decreased by the amount given by the number in the direction indicated by the sign.

5. Note that one coordinate part can be change on a relative basis while the other coordinate part is changed on an absolute basis.

6. If a coordinate part has a nil definition (no characters) then that coordinate part is left unchanged.

Examples of using position arguments are given in the next section.

4.4.3 Position Blocks

REGIS allows a form of block structure (borrowed from such high level languages as ALGOL, PL1 and PASCAL) to be applied to drawing positions. This means that a current drawing position can be saved by a "begin" operation and later after possibly several intervening position changes the position can be restored to its old position by an "end" of block operation. This feature is important in REGIS for two reasons. First of all, it reduces in many cases the number of characters which have to be sent to the remote point. Secondly, it allows the graphics device to syntactically identify closed polygon and curve sections and distinguish these structures from just a sequence of vectors or curve arcs. The begin and end points are identified by option characters "B" and "E" in the option arguments of the drawing instruction which uses the block structure. The following notes apply to block structure positioning:

1. Only one mechanism records block begin points. Thus, an "end" reference in a vector instruction can cause a position saved by a "begin" operation in a position instruction to be returned. This should be avoided for clarity if possible.

2. The block structure may be nested but for the purpose of maximum transportability only 1 level of block structure should be used (only one level is needed to achieve most of the benefits of block structuring).

4.5

EXTENSIBILITY GUIDELINES

The following notes give general guidelines for extending REGIS to take advantage of device specific characteristics:

1. When possible, a new feature should be added to REGIS by adding options to existing instructions.

2. A new instruction type should only be added to REGIS if the new operation can not be performed by a sequence of existing REGIS instructions and the designer can ensure a minimal loss of information content when the new instruction is ignored by a REGIS device which does not implement the instruction.

3. Avoid using a syntax for extension which has already been used as a standard extension.

4. Do not try to extend REGIS beyond its intended range of applicability as outlined in the philosophy section.

Syntactic transportability is ensured by requiring the complete implementation of all arguments and instruction forms whether those arguments are used or not. In particular, each device should implement the following "skipping" syntactic elements which allow arguments which are not used or instructions which are not implemented to be ignored:

Skip over a bracket parameter:

```
<SKIPB> --> ]
        --> ;
```

```
         --> <any-other> <SKIPB>
```

Skip over a text string:

```
<SKIPQ> --> '
        --> <any-other> <SKIPQ>

<SKIPQ2>--> "
        --> <any-other> <SKIPQ2>
```

Skip over option sequence:

```
<SKIPP> --> )
        --> <letter><SKIPI><SKIPP>
        --> , <SKIPP>
```

Skip over an entire instruction:

```
<SKIPI> --> [<SKIPB><SKIPI>
        --> (<SKIPP><SKIPI>
        --> '<SKIPQ><SKIPI>
        --> "<SKIPQ2><SKIPI>
        --> <digit><SKIPI>
```

Having implemented these skipping variables a device would then invoke the appropriate variable for each of the instructions or arguments for which no semantic meaning has been implemented. To aid in detecting errors in graphic definition sequences, a device might choose to set a flag which causes an error message to be shown anytime one of these skipping elements were accessed. This would also be helpful in ensuring that a particular graphic definition meets some predefined level of REGIS implementation.

5.0

BASE REGIS INSTRUCTIONS

This section details the syntax and semantics of the six BASE REGIS instructions. Figure 6 summarizes the primitive syntactic elements of instructions. Note that REGIS allows numeric parameters to be given in integer, floating point and scientific notation. The following rules should be used in determining the fidelity to which different devices must implement the interpretation of numeric formats:

1. Low resolution devices are assumed to use integer numeric quantities only. However, low resolution devices must extract the integer portions of floating point and scientic notation number constants.

2. Medium resolution devices at least reduce numeric quantities to integer form but may also use some portion of the increased resolution provided by floating point numbers.

3. High resolution devices are expected to use the entire resolution provided by floating point numbers.

For reference purposes, figure 7 summarizes the syntax and semantics for the base instructions in a form suitable for quick reference.

Figure 6. REGIS BASIC SYNTAX ELEMENTS

SYNTAX ITEM	SYNTAX VARIATION	MEANING
<char>		Any ASCII printing character
<letter>		Any ASCII letter (lower case conversion)
<digit>		Any of the characters 0, 1, .. 9
<pv>		Pixel vector characters 0,1,2,3,4,5,6,7
<bit>		The characters '0' and '1'
<n>		NUMERIC CONSTANT
	<ni>	Integer constant
	<digit>{<digit>}	
	<nf>	Fixed point number
	<ni>.{<digit>}	
	<ne>	Floating point number
	<ni>E{+|-}<ni> or <nf>E{+|-}<ni>	
<ang>	{+|-}<n>	Angle argument
	0<= <n> <=360	
<pcnt>	<n>	PERCENTAGE ARGUMENT
	such that 0<= <n> <= 100	
<pa>	<pax><pay>	POSITION ARGUMENT
	<pax>	X-position part
	<nil>	no X-position change
	<n>	set x absolute to <n>
	+<n>	increase x by <n>
	-<n>	decrease x by <n>
	<pay>	Y-position part
	<nil>	no Y-position change
	,<n>	set Y absolute to <n>
	,+<n>	increase y by <n>
	,-<n>	decrease y by <n>
<aa>		ABSOLUTE X,Y ARGUMENTS
	same as <pa> except only absolute X and Y arguments	
<ra>		RELATIVE X,Y ARGUMENTS
	same as <pa> except only relative X and Y arguments	

Figure 7. REGIS BASE INSTRUCTIONS

A	INST. KEY	PARAM- ETER	PARAMETER VALUES	MEANING
X	S			SCREEN INSTRUCTION
X		( )		Screen Options
X			E	erase screen
			S[<LT>][<RB>]	define screen coordinates
			T(S<n>)	adjust standard text size
			W[<aa>]	adjust pixel vector size
X	W			WRITING ATTRIBUTES INSTRUCTION
X		( )		Writing Options
X			P0	invisible line writing pattern
X			P1	full line writing pattern
			P2 to P9	predefined line patterns
X			P<bit>...	user defined line pattern
X		[<aa>]		Set pixel vector multiplier
X	P			POSITION CURSOR INSTRUCTION
X		[<pa>]		Set current cursor to <pa>
X		<digit>		Pixel vector change to current cursor
		( )		Position options
			B	bounded position sequence
			E	end position sequence
X	V			VECTOR INSTRUCTION
X		[<pa>]		Draw vector from current cursor to <pa>
X		<digit>		Draw pixel vector
		( )		Vector options
			B	begin bounded vector sequence
			E	end vector sequence
X	C			CURVE INSTRUCTION
X		[<pa>]		Draw circle (arc) through <pa>
X		( )		Curve options
X	X			C	draw circle (arc) centered on <pa>
X			A<ang>	draw circular arc
X			S	start curve sequence
X			B	begin bounded curve sequence
X			E	end of curve sequence
X	T			TEXT INSTRUCTION
X		'<text>'		Draw <text> string
		"<text2>"		Draw <text2> string
X		( )		Graphic text options
X			S<n>	set standard text size multiple
X	@			MACROGRAPH STRING
X		.		Initialize (clear) macrograph strings
X		:<letter>		Begin definition of string <letter>
X		;		End macrograph string definition
X		<letter>		Substitute macrograph string

5.1

The SCREEN Instruction

1. PURPOSE
   The screen instruction, key-letter "S", is used to control screen coordinate parameters and attributes which affect the entire viewing area. In the REGIS BASE this includes clearing the screen area, setting the coordinate system to be used and making adjustments to device specific parameters.

2. SYNTAX

```
     <REGIS> --> S <SCREEN>

     <SCREEN> --> [<SKIPB> <SCREEN>
              --> '<SKIPQ> <SCREEN>
              --> "<SKIPQ2><SCREEN>
              --> <digit> <SCREEN>
              --> (<SOPS>

     <SOPS>   --> E <SOPS>
              --> S[<pa>][<pa>] <SOPS>
              --> T(S<n>) <SOPS>
              --> W[<aa>] <SOPS>
              --> , <SOPS>
              --> ) <SCREEN>
              --> <any-other> <SKIPI> <SOPS>
```

   The forms <SKIPB> and <SKIPQ> are syntactic states used to skip over (ignore) bracket and quoted string arguments respectively. The form <SKIPI> is a syntactic state used to skip over an entire instruction, including all parameters.

5.1.1 Screen erase - S(E) The screen erase option, option letter "E" is used roughly to initialize the viewing area in preparation to drawing a new image. The precise meaning of "erase" is device dependent but generally has the connotation of creating a homogeneous visual image. For example, the sequence of characters:

```
        S(E)
```

will cause (or request to be caused) a new sheet of paper to be loaded into a flat-bed plotter and will cause the entire screen to have no visible structure (all black, all white etc.) in a raster display device. For devices which have REGIS extensions, all parameters and attributes (except possibly for screen attributes themselves) should take on their default values upon the execution of the screen erase function.

5.1.2 Screen Coordinate definition S(S[ ][ ]) The screen coordinate definition option is a mandatory implementation sequence which allows the user to define the coordinate ranges which are to be used in position arguments. A specific device may define defaults for screen positioning.

The screen definition option has two position arguments. The first gives the user defined coordinates of the upper-left corner of the viewing area rectangle and the second gives the user defined coordinates of the lower right corner of the viewing area. As described in the "Logical Device" section, this information is sufficient for the device to set up appropriate scaling functions to accommodate a wided range of coordinate parameter settings. It is assumed that the coordinates will normally be set to the actual physical coordinates of the device that the application is to use the most often. In this case, the device can determine that no scaling is necessary and set the transformation functions to unary operators, thus reducing the overhead required in transforming position arguments.

EXAMPLES

S(S[0,0][383,239]) - Sets the "origin" point to the upper left corner with a range approximately equivalent to standard broadcast TV.

S(S[0,511][511,0]) - Sets the origin point to be the lower left corner with a range approximately equivalent to a medium resolution graphics raster terminal display.

S(S[1,1][1000.00,500.00])
- Sets the coordinate ranges for a high resolution plotting device.

Note that the number of pixels actually accessible is the integral part of the screen parameters. Accessiblity to a larger number of pixels than can be addressed in this manner for high resolution devices may be accomplished by using the S(M ..) option described below.

FIDELITY OF IMPLEMENTATION

1. Low resolution devices have the greatest flexibility in implementing the coordiante transformation feature, but must implement some form of transformation. The simplest form would be to round-off the coordinate range in each direction to be the nearest multiple of the devices inherent physical positioning capability. The transformations then become simple shift operations. Using this approach as much as three-fourths of the viewing area would not be usable.

2. Medium resolution devices are expected to use a transformation scheme which still allows the approximation of the coordinates but has fine enough granularity to

achieve less than 20 percent loss in usable viewing area in the worst case.

3. High resolution devices are expected to use a nearly exact coordinate transformation algorithm which looses little if any of the usable viewing area.

```
************
*  NOTE    *
************
```

The screen coordinate option is not meant to be used as a viewing or image transformation (that is, define a coordinate system which "makes sense" for his or her problem .. this should be done using higher level software). It is meant rather to provide a mechanism for transporting image definitions from one device to another while ensuring that the maximum capability of a device is usable in the simplest cases. To ensure that this feature is not mis-used, specific implementations should perform some type of destructive operation along with the screen coordinate setup function, such as an automatic screen clear operation. Such an operation should not be performed in lieu of a screen clear.

For the remainder of this report, it will be assumed that the screen is defined to have 400 horizontal pixels by 300 vertical pixels with the origin at the upper left corner. That is, it will be assumed that the following screen instruction has been performed:

```
S(S[0,0][399,299], T(S1), W[1,1])
```

This also sets the text and pixel size adjustment parameters to there default values.

5.1.3 Text adjust option - S(T(S<n>)) The parameter <n> is a generally floating point number which is to be used to uniformly increase or decrease the size of the "standard" device character size on a multiplicative basis. As a guideline, REGIS assumes that the height of the standard character size is approximately one-tenth to one-thirtieth the the physical size of the smaller of the horizontal or vertical direction. For example:

```
S(T(S1.5))
```

will uniformly cause the size of each of the standard sizes to be increased by 50 percent. Like the screen coordinate option, this feature allows a transported image definition to be "tuned" to the specific device characteristics. The option should appear only once for each image definition and has a default parameter of 1.0.

FIDELITY OF IMPLEMENTATION

1. Low resolution devices are not expected to implement this feature at all since such devices normally have little or not capability of changing the incremental size of characters.

2. Medium resolution devices may have some capability in adjusting character sizes.

3. High resolution devices are expected to implement a reasonable range of character adjustment, at least to the extent of publication "point" sizes.

5.1.4 Pixel Size Adjust - S(W[<n>]) The argument <n> is a generally floating point number used to adjust the horizontal and vertical pixel sizes on a multiplicative basis. Like the text adjust feature, pixel size adjustment should only be performed once for a specific image and is normally ignored by low-resolution devices which normally do not have the ability to adjust pixel sizes on an incremental basis. This option should not be used in lieu of the writing options pixel multiplier option. Significant digits after the decimal point may be interpreted by high resolution devices as an extension of the number of available pixels. For example:

```
        S(W[0.01])
```

would be recognized as a request to use 100 pixels between each integral pixel.

5.2

The POSITION Instruction

1. PURPOSE

REGIS assumes and uses what is referred to as the "current writing position" convention for determining where a drawing instruction starts its operation in the viewing area. This means that a REGIS device is required to maintain the value of an X and Y set of numbers, and these saved values are assumed always to be the starting point for a drawing operation. These values are up-dated to new values after each drawing operation. A simple example of the current value concept is the position of a drawing pen on a fat-bed plotter device. The cuurent value of X and Y are also commonly referred to as the "writing cursor" or "cursor" for short. The purpose of the position instruction is to move the cursor (change the value of the current X and Y values) without drawing any visible image (that is, move the plotter pen witn the pen "up"). REGIS allows the cursor to be moved on both an absolute and relative basis using both numeric X/Y parameters as well as pixel vectors.

2. SYNTAX

```
<REGIS> --> P<PINST>

<PINST> --> [<pa>] <PINST>
        --> <pv> <PINST>
        --> (<POPS>
        --> '<SKIPQ> <PINST>
        --> "<SKIPQ2> <PINST>

<POPS>  --> B <POPS>
        --> E <POPS>
        --> , <POPS>
        --> ) <PINST>
        --> <any-other> <SKIPI> <POPS>
```

3. GENERAL SEMANTICS

The position instruction allows the cursor to be changed by direct X/Y parameters by using the form P[<pa>]. The cursor oosition may be changed by relative pixel vectors using the form P{<pv>}. A begin/end position block may be defined by using the position options P(B) and P(E).

5.2.1 Position Change Arguments - P[<pa>] The form <pa> allows the the cursor to be changed on a relative or absolute basis as described earlier. For example, assume the current values of X and Y are 100 and 50 respectively. Then:

P-Instruction	New X/Y value		
P[0,0]	0	0	
P[30,23]	30	23	
P[200]	200	50	(only X changed)
P[,42]	100	42	(only Y changed)
P[+10,-25]	110	25	(relative change)
P[0,+10]	0	60	(combination relative and absolute change as for "carriage return")
P[15,20]	15	20	
P[10,20][+5]	15	0	(multiple <pa>'s)

Although the syntax allows several <pa> arguments to be used in a single P-Instruction, this generally has the same effect as a single <pa> argument as illustrated in the last example above.

5.2.2 Pixel Vector Positioning - P{<pv>} To support programs which use "chain encoding" techniques and to allow small relative changes in the cursor, REGIS allows the user direct acces to pixel vectors. Generally, a sequence of <pv> digits are used. The number of unit pixels actually moved for each digit is determined by the current multiplier value as set by the writing options instruction defined in the next section. Assuming as above that the current value of X and Y are 100 and 50:

Multiplier	Instruction	New X/Y Values	
1	P0	101	50
1	P1	101	49
1	P0002	103	49
(ANY)	P01234567	100	50
5	P667	105	65
23	P5	77	73
1	P0000000000	110	50
1	P[+10]	110	50

As illustrated in the last example, direct positioning is usually more efficient than pixel chains for large movements. One very effective use of pixel-vector movement is to achieve a form of superscript and sub-script operation in the context of graphics text.

5.2.3 Position Blocks - P(B) and P(E) The position begin and end options allow a simple means for recording a current cursor value and then returning to that value at some later point in the sequence. That is, the action of the P(B) option is to save the current value of X and Y. After generally several intervening drawing instructions which change the cursor value, the execution of the P(E) option causes the cursor to be restored to its original value. This is particularly useful in defining images which are to have maximum transportability, since it allows accumulative roundoff errors to be occasionally eliminated, particularly in the case of text drawing which may be very roughly approximated on low resolution devices.

5.3

Writing Attributes Instruction

1. PURPOSE

The writing attributes instruction, key-letter "W" allows the user to control the manner in which the pixel image is to be drawn at the pixel level. In the REGIS BASE this means the selection of line drawing patterns and the adjustment to pixel sizes.

2. SYNTAX

```
<REGIS> --> W <WINST>

<WINST> --> ( <WOPS>
        --> [<n>] <WOPS>
        --> ' <SKIPQ> <WINST>
        --> " <SKIPQ2> <WINST>
        --> <digit> <WINST>

<WOPS>  --> P {<digit>}<WOPS>
        --> , <WOPS>
        --> ) <WINST>
        --> <any-other> <SKIPI>
```

3. GENERAL SEMANTICS

In the REGIS BASE there are only two significant W-Instruction structures. The W[<aa>] allows the user to define a multiplicative factor to be applied to pixel vectors used in the P and V instructions. The W(P<digit>) form allows the user to select a line drawing pattern such as a solid line or a dot-dash pattern.

5.3.1 Pixel Multipliers - W[<n>] The argument <n> represents a numeric constant which serves the purpose of causing each pixel vector referred to by a P or V instruction to be repeated the number of times given by that number. The pixel multiplier also has affect on the repetition length of line drawing patterns. Devices will change a pixel multiplier value of 0 to the default value of 1. In general, the pixel multiplier may be a non-integer number, but this capability is normally used only in high-resolution devices.

The following are examples of this feature:

```
W[1]     - same as W[1], the default value
W[10]    - each <pv> refers to 10 unit pixels
W[0.6]   - fine adjustment to pixel
```

vector size on a high resolution device, interpreted as W[1] on low resolution devices.

5.3.2 Line Drawing Patterns - W(P<digit>) Allows the user to select from one of 10 predefined line drawing patterns or define a new pattern by a sequence of on/off pixel vectors. The possible values of this option are as follows:

P0	Draw image as if the pen is "up" (that is, no visible image)
P1	Draw image with solid lines
P2	Dash pattern
P3	Dash Dot pattern
P4	Dot Dot pattern
P5	Dash Dot Dot pattern
P6	(to be defined)
P7	(to be defined)
P8	(to be defined)
P9	(to be defined)
P<bit><bit>...	user defined pattern

The user defined pattern is interpreted as a sequence of <pv> size line segments based upon an alternating pattern of P0 and P1 line types. For example, the sequence:

```
P111010
```

constructs a Dash Dot pattern similar to the P3 line type.

Each device is assumed to have a certain repetition length for the line drawing patterns. This length is not required to be standardized since it is assumed that a basic dot pattern will carry essentially the same information content independently of this repetition period. Devices may use the current pixel vector multiplier value to expand the pattern spacing, and thus achieve additional discernible line pattern types.

5.4

The VECTOR Instruction

1. PURPOSE

The Vector instruction, key-letter "V" is used to draw straight line segments of arbitrary angle between the the current cursor location and one or more new positions, using numeric position arguments or pixel vectors.

2. SYNTAX

```
<REGIS> --> V <VECTOR>

<VECTOR>--> [<pa>] <VECTOR>
        --> <pv> <VECTOR>
        --> ' <SKIPQ> <VECTOR>
        --> " <SKIPQ2> <VECTOR>
        --> ( <VOPS>

<VOPS>  --> B <VOPS>
        --> E <VOPS>
        --> W <WINST> <VOPS>
        --> , <VOPS>
        --> ) <VECTOR>
        --> <any-other> <SKIPI> <VOPS>
```

3. GENERAL SEMANTICS

Semantically, the vector instruction works in the same manner as the P-instruction with the exception that the pen is "down". That is, the v-instruction causes a visible image to be generated. The lines drawn by this instruction are subject to the current line drawing pattern selected by the W-instruction and the pixel-vectors drawn are subject to the current pixel vector multipliers selected by the W-instruction.

The B and E options allow the user to define a closed polygon. That is, at the time that an E option is interpreted, a line will be drawn back to the previously defined B (begin point) position.

The W option allows the writing attributes to be set on a temporary basis. After completion of the vector instruction, the value of the writeing attributes will be returned to the values which they had before the execution of the V-instruction.

Figure 8 gives examples of the kind of images drawn using the V-instruction including the use of line patterns .

FIDELITY OF IMPLEMENTATION

1. Low resolution devices will normally approximate straight lines by a jagged sequence of unit size pixel vectors.

2. Medium resolution devices have varying degrees of quality when drawing vectors.

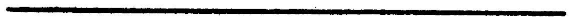
3. High resolution devices are expected to draw lines of sufficient smoothness that no discernible jagged edges appear to the naked eye.

Figure 8. VECTOR INSTRUCTION EXAMPLES

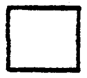
Assume that the writing mode is set with:

```
W(P1)
```

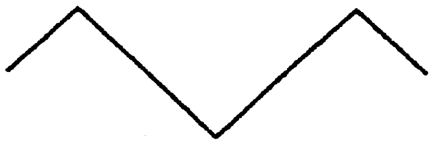
then: P[100]V[+400] gives:



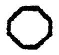
The sequence V[+50][,+50][-50][,-50] gives:



V[+50,-50][+100,+100][+100,-100][+50,+50] gives:



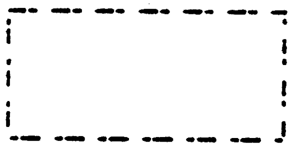
With W[10] (pixel size of 10) then V01234567 gives:



with the pattern defined by:

```
W[4](P11101000)
```

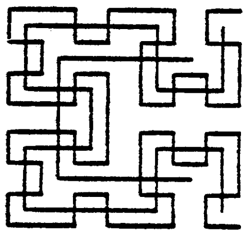
Then V[+200][,+100][-200][,-100] gives:



The sequence of instructions:

```
P[432,72]
W[96]V460P[456,48]
W[48]V642446064600206P[468,36]
W[24]V46066424642202446424460646002066642446064
V6002060024220602066460
```

gives the chain encoded image:



5.5

The CURVE Instruction

Simple graphics devices do not usually have a built in curve generation capability. This feature is built into REGIS for at least four reasons;

1. With the assumption that all devices will have local processors, there is no longer any significant complexity to having a local curve capability.

2. The availability of a local curve capability can substantially reduce the number of characters transmitted to the device.

3. The inclusion of a curve capability gives REGIS a degree of completeness relative to the idea of incorporating the "rule and compass" drawing primitives.

4. Each graphic device knows its abilities better than the software driving the device and thus is better able to select an optimal quality versus performance tradeoff during the design phase. This avoids the common problem which arises when a circle is approximated by an 18 sided polygon on a low resolution device which looks exactly like an 18 sided polygon on a high resolution device (instead of a circle!).

1. PURPOSE

   The curve instruction, key-letter "C" is used to draw circles, arcs of circles and curve interpolation sequences. By a curve interpolation sequence is meant a curved line image of varying radius of curvature such as would be drawn by a draftsman using "french curves" or similar aids.

2. SYNTAX

```
     <REGIS> --> C <CURVE>

     <CURVE> --> [<pa>] <CURVE>
             --> '<SKIPQ> <CURVE>
             --> "<SKIPQ2> <CURVE>
             --> <digit> <CURVE>
             --> ( <COPS>

     <COPS>  --> C <COPS>
             --> A<ang> <COPS>
             --> S <COPS>
             --> B <COPS>
             --> E <COPS>
             --> W <WINST> <COPS>
             --> , <COPS>
             --> ) <CURVE>
             --> <any-other> <SKIP1> <COPS>
```

3. GENERAL SEMANTICS

Pixel vectors normally have no meaning for the curve command .. all position arguments are given using the form [<pa>]. The meaning of a position argument is dependent upon whether or not a curve position block has been selected. Outside of a position block, position arguments mean "draw a circle" or if the arc-angle option has been selected "draw an arc of a circle". Within a position block, position arguments refer to points on a curve through which an interpolated curve is to be drawn. There are two types of curve interpolation sequences as determined by whether or not the the operation is initiated by the sequence option "S" or the bounded begin option "B". In the first case, an open end point curve (or a curve with discontinuities) is drawn. In the second case, the end point of the interpolated curve is drawn back to the begin point with a continuous derivative (and therefore looks "smooth"). The W option allows the definition of temporary writing attributes using the W instruction parameters.

5.5.1 Circles and Arcs - C[<pa>] forms

If no curve position block has been selected, then the instruction form:

```
C[<pa>]
```

will draw a circle with the current cursor position as the center and [<pa>] a point on the circumference of that circle. The cursor is left at the center after drawing the circle. This same instruction with the center option enabled:

```
C(C)[<pa>]
```

will the circle with the current cursor as a point on the circumference and [<pa>] as the center. In this case, the cursor is left on the circumference of the circle at the the end point (equal begin point) of the circle drawing.

If the arc-angle option (A<ang>) is used, then an arc of a circle is drawn starting at the point on the circumference arc ending <ang> degrees from that point. The cursor is left at the end point of the arc if the (C) option was used, otherwise the cursor is left at the center of the arc. If the angle is positive, then the arc will be drawn in the counter-clockwise direction and if the angle is negative, the arc will be drawn in the clockwise direction. Figure 9 illustrates examples of the circle commands.

5.5.2 General Curve Interp lation

Either the option "S" (for curve sequence) or "B" (for begin or bounded) start a general curve interpolation sequence. The sequence is ended by the "E" end option. There can be no intervening circle or block structured P or V instructions within the range of a curve begin-end sequence. Within these option points, the position arguments (including the current cursor position) are points on a curve through which a smooth curve image is to be drawn. The problem of determing the slope of the curve at the end points is handled in the following manner:

1. The positions at the start and end points are not implicitly drawn but are used to define the slope of the curve at the end points.

2. The begin and end points may be visually included in the curve by using the "null" point syntax [] to indicate that the point is to be repeated in the sequence. In this case, the visual appearance of the curve at the end points may not be realistic.

3. For a bounded curve, option "B", the point information at the beginning of the sequence is retained so that the closed curve will have a continuous first derivative at the end point.

As a result of this convention, at least three position arguments in addition to the current cursor position must be given to result in a visible curve segment between the second and third positions given. One additional curve segment is then drawn for each additional position argument. The current cursor position is always maintained at the last position argument given and thus leads the drawing process by one curve segment. Figure 9 also illustrates examples of the curve interpolation instructions.

5.5.3 Fidelity of implementation

1. Low resolution devices will normally approximate circles by a polygon of a certain number of sides which will generally vary depending upon the radius of the circle. Circular arcs will normally be drawn by a polygon approximation with a portion of the polygon removed. Incremental algorithms, such as Bresenham's circle algorithm might also be used for low resolution devices. The user is warned that on low resolution devices, the accuracy of a circular arc may be poor, so that the image definition should not depend upon the end of arc positioning.

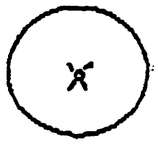
2. In medium and high resolution devices, it is expected that an arc or circle be represented with high accuracy and little or no discernible jaggedness. In high resolution devices, the accuracy of the circular arc positioning is expected to be at the pixel level.

3. REGIS does not specify the algorithm by which devices implement the curve interpolation algorithm but it is assumed that it will be similar in nature to the commonly used spline algorithms. The exact curve sequence drawn varies depending upon the algorithm used, but this is not considered a problem since the essential information content is shown in spite of minor variations in the curve path. Devices may implement more than one curve interpolation algorithm which are user accessible by option extensions.

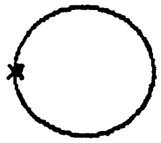
Figure 9. CURVE INSTRUCTION EXAMPLES

o = cursor start position
x = cursor end position

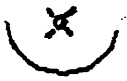
C[+50] gives:



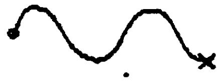
C(C)[+50] gives:



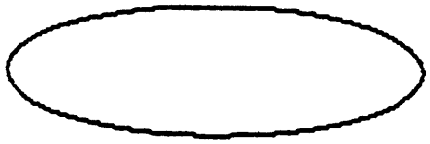
C(A180)[-40] gives:



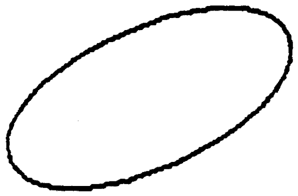
C(S)[][+20,-20][+40,+40][+40,-40][+40,+40][](E) gives:



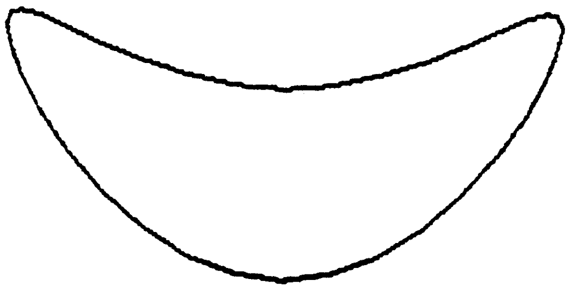
C(B)[+150,-50][+150,+50][-150,+50](E) gives:



C(B)[+80,-100][+120][-80,+100](E) gives:



C(B)[+200,+150][+200,-150][-200,+200](E) gives:



5.6

The TEXT Instruction

1. PURPOSE

The graphic text instruction, key-letter "T" is used to draw strings of characters starting at the current cursor position.

2. SYNTAX

```
<REGIS> --> T <GTEXT>

<GTEXT> --> [ <SKIPB> <GTEXT>
        --> <digit> <GTEXT>
        --> '<text> <GTEXT>
        --> "<text2> <GTEXT>
        --> ( <TOPS>

<TOPS>  --> S<n> <TOPS>
        --> W <WINST> <TOPS>
        --> , <TOPS>
        --> ) <GTEXT>
        --> <any-other> <SKIPI> <TOPS>
```

3. GENERAL SEMANTICS

Each REGIS device is assumed to have a built in character definition and writing capability. The alphabet for writing is assumed to have at least 64 printing characters and in general will implement the full ASCII printing set. Each character of the <text> string is written by placing the visual image of the referenced character at the current cursor location. After writing the character, the cursor position is changed to a point which would be the logical position for writing another character (normally horizontally a distance a little more than the visual width of a character). The character size option allows characters to be written with varying sizes and with varying spacing. The W option allows the setting of temporary attributes which are in effect only during the duration of a single T instruction.

5.6.1 Text Size option - T(S<n>)

The parameter <n> is generally floating point number which allows the user to increase the size of the characters drawn (and proportionately the spacing between characters) by a multiplicative factor of the devices "standard" size. Generally, the range of <n> should be limited to 0 to 10. A value of 0 will be taken to refer to the default value 1.

5.6.2 Fidelity of implementation

1. Low resolution devices generally will have a standard size character which will vary considerably in size from one device to another. Thus the user should not depend upon the cursor value at the end of drawing a sequence of characters and should reposition the cursor by some other means. The quality of character representation at this level is also assumed to be poor and typically drawn as a sequence of short vectors or as a pixel pattern.

2. Medium resolution devices are expected to allow reasonably good quality characters and deterministic end point cursor values and must implement a full range of integral character sizes.

3. High resolution devices are expected to draw "letter quality" characters and give fractional as well as integral character sizes. Since the standard (smallest) character size on a high resolution device will generally be very small in proportion to low resolution devices, the screen instruction text adjust feature may have to be set when transporting a graphic image definition.

6.0

THE EXTENDED REGIS LOGICAL DEVICE

This section describes extensions to the Logical REGIS Graphic Device required to accomodate the specific features of devices which are not common to all graphics devices. This is not meant to be an exhaustive discussion on the subject, but rather an illustrative presentation which can be followed for the extension of REGIS to device capabilities not described here. The case of raster CRT devices is covered in some detail to fully illustrate the process of extending REGIS.

6.1

Dimensional Screens

Dimensional screens refers generally to the class of graphics devices in which the physical size of pixels has a repeatable meaning and in particular allows the user to choose from a variety of pixel sizes. Examples of such devices include both paper and photographic plotting devices. REGIS accomodates such devices by allowing the user to define the size of pixels in physical dimensions. The logical device is therefore extended by the following parameters:

SDX -	is the physical size of a pixel in the horizontal direction
SDY -	is the physical size of a pixel in the vertical direction

By knowing the number of pixels selected by the screen coordinate definiton operation, the device is then able to determine the size of paper needed to draw the image.

The units of the physical measure should be optionally selectable by the user with the metric system being the default value.

These devices also will generally allow (and need) the width of line segments to be user controllable. This type of feature would be included in REGIS by using a width option in the writing attributes instruction.

6.2

Gray Scale and Color

The simplest form of color capability refers to the the ability of a plotter device to use different colored pens to draw on different colors of paper. In this case, the paper represents the background color and the pens represent the foreground color. Black and white raster graphics devices inherently have the ability to show varying intensity levels of light, called gray-scale. More complex devices, such as high resolution color raster devices have the ability to present a full range of color hues in addition to gray scale (intensity) capability.

To accomodate this broad range of capabilities, the extended REGIS logical device adopts the following scheme which allows the selection of color and gray-scale in a simple manner for simple devices but still has full access to the broad spectrum of attribute capabilities of the most capable of the devices:

1. Each pixel has associated with it a foreground and a background "intensity" attribute which is user controllable. The differentiation between foreground and background is most clear in the case of the plotter but has little meaning for the high resolution raster color device.

2. The simplest form of intensity attribute is a scale of 8 numbers representing lowest to highest brightness on a gray scale raster device.

3. For simple color devices, the scale of eight intensity values is interpreted as the three primary colors, the three complementary colors and the black and white intensity values. The mapping between gray-scale and color follows the conventions which have been established for photographic and broadcast television work as summarized in the foloowing table:

Gray-Scale	Color Value
0	Black (dark)
1	Blue
2	Red
3	Magenta
4	Green
5	Cyan
6	Yellow
7	White

This is based on the standard RGB to luminance level transformation used in the television industry (NTSC color standard). A more accurate transformation for devices having a broad grey scale range is (normalized to the range 0 to 7):

$$I = 4.1*G + 2.1*R + 0.8*B$$

4. For devices having a full range of color capability, the Hue-Lightness-Saturation system of color definition is adopted. In summary, this system requires three generally floating point numbers to be used in defining a specific intensity parameter value:

Hue (H) - The hue of the color expressed as an angle on the color wheel.

Lightness (L) - The relative brightness of the color expressed as a percentage of full brightness.

Saturation (S) - Expressed as a percentage of the fully saturated hue.

For the purpose of converting between the low-resolution RGB notation and the higher resolution HLS system, the following table should be used (generally, lightness and saturation are ignored and only the hue angle has meaning):

TO CONVERT HLS TO RGB:

Hue	Lightness	RGB	Intensity
331 to 30	15 to 85	B	I1
31 to 90	15 to 85	M	I2
91 to 150	15 to 85	R	I3
151 to 210	15 to 85	Y	I6
211 to 270	15 to 85	G	I4
271 to 330	15 to 85	C	I5
(any)	0 to 15	D	I0
(any)	85 to 100	W	I7

TO CONVERT RGB TO HLS: (Saturation = 100)

Intensity	RGB	Hue	Lightness
I0	D	0	0
I1	B	0	50
I2	M	60	50
I3	R	120	50
I4	G	240	50
I5	C	300	50
I6	Y	180	50
I7	W	180	100

In concept, the intensity parameter can be applied independently to the foreground and to the background of the graphic image. That is, we can select to use a blue pen to draw on a yellow piece of paper. Obviously, devices will have extreme ranges of implementation of the intensity attributes.

6.3

Text Attributes

Text attributes include a wide variety of features associated with the presentation of textual characters, including (but not limited to) any combination of the following:

1. Variable character size including independent width and height adjustment.

2. Variable character spacing including proportional spacing and letter-spacing.

3. Angular orientation of characters and independent angular character spacing.

4. Foreign language fonts.

5. Alternate representation fonts (Gothic, Futura, and so forth).

6. Superscript and subscript capability.

7. Overstrike (underline, APL characters, and so forth).

It is the intent of REGIS to support such features in an extension set. The user should recognize that the degree to which a specific device supports these charateristics is extremely broad and is therefore one of the areas of image generation with the least portability capabilities.

To support the majority of these features, an extended REGIS device may support the definition of the following extended logical device parameters:

1. WIDTH - The width of the hypothetical parrallelogram in which a character is written.

2. HEIGHT - The height of the parallelogram measured in pixel-vector units.

3. DANG - The direction of the character width line measured as an angle in degrees relative to the horizontal axis.

4. HANG - The direction of the height side of the character parallelogram measured relative to the width direction (relative slanting of characters, as for an Italic representation).

5. TDX - The relative horizontal spacing of two characters measured in pixel units.

6. TDY - The relative vertical spacing of two characters measured in pixel units.

7. FONT - A scalar parameter used to identify which of several possible fonts is currently in use.

6.4

Area Attributes

This class of extensions refers to the general capability of many devices to associate attributes to areas bounded by REGIS primitive line images. The visual attributes may include combinations of the following:

1. Shading patterns (similar in concept to line patterns).

2. Color and intensity variation.

These capabilities syntactically are covered by other REGIS extensions. REGIS distinguishes two approaches to identitfying the boundaries of an area:

1. FILLed areas - An area defined by and bounded by either a bounded sequence of vectors or a bounded (closed end point) curve.

2. SHADed areas - An area defined to be the difference between two not-necessarily bounded vector or curve sequences.

The first case corresponds to the common concept of bounded surface. The second case corresponds to the typical approach used to represent data as a "histogram" image.

6.5

Dynamic Attributes

The area of dynamic attributes covers a broad spectrum of features which are illustrated at one extreme by "blinking" attributes and at the other extreme by a fully animated cartoon. The common point in REGIS for such capabilities is the notion that the foreground/background selector function varies as a function of time. Using this model the typical blink attribute is interpreted as an alternation of foreground and background visual attributes.

6.6

User Interaction

User interaction refers to the general capability of some devices to send information to the controlling computer as a result of computer queery and/or operator entry using one of several possible devices. Generally, REGIS expects such devices to return character sequences in response to such computer or operator request.

7.0

STANDARD RASTER EXTENSIONS

This section describes a standard set of REGIS instruction extensions to be used with raster CRT graphics devices. The additional capabilities accessible using these extensions include:

1. Background and foreground gray-scale and color intensity attributes.

2. Negative image attributes (white on black, black on white and so forth).

3. Screen area scrolling (screen motion).

4. The "alternate" (blink) dynamic attribute.

5. Memory value dependent writing attributes (replace writing in addition to overlay writing).

6. Area attributes for bounded and unbounded areas.

7. Text attributes and user definable characters.

The syntax of the instruction extensions is summarized in figure 10. Figure 11 summarizes the syntax of the intensity attributes.

Figure 10. STANDARD RASTER DISPLAY EXTENSIONS

A	INST. KEY	PARAM-ETER	PARAMETER VALUES	MEANING
X	S	<pv>		Move screen image by pixel vector amount
		[<pa>]		Move screen image by <pa>
X		( )		Extended options
X			N<n>	negate (reverse) the image
X			N0	disable negative image (default)
X			I<intens>	set background intensity
X	W	( )		Extended writing options
X			A<n>	enable Alternation (blink)
X			A0	disable alternation (default)
X			N<n>	enable negative image writing
X			N0	disable negative image writing (default)
X			E	erase writing
X			C	complement writing
X			R	replace writing
X			V	overlay writing (default)
X			S	enable area shading
Y			<digit>	use <digit> line pattern as fill pattern
			'<char>'	use character as shading pattern
X			S0	disable area shading (default)
X			I<intens>	set writing intensity
X	T	[<ra>]		Set relative text spacing
X		( )		Extended text options
X			A<digit>	alphabet (font) select
X			S[<aa>]	set absolute character row/column size
X			M[<aa>]	set row/column pixel multipliers
Y			I<ang>	set Italic slant
Y			D<ang>	set character row direction
X	L			LOAD ALPHABET CHARACTER INSTRUCTION
X		'<char>'		Select character to be loaded
X		<text>;		Character pattern definition
		( )		Load character options
			D	select decimal number base
			H	select hexa-decimal number base
			S[<aa>]	character definition size
			I<intens>	background intensity select
	R			READ GRAPHICS PARAMETERS
		( )		Identify parameter to be read
			P	read current cursor position

Figure 11. RASTER EXTENSIONS - INTENSITY PARAMETER

		MEANING
<intens>		INTENSITY ARGUMENT
	<digit>	Set one of 8 predefined intensities
	( )	Intensity options
	D	dark (default I0)
	B	blue (default I1)
	R	red (default I2)
	G	green (default I3)
	M	magenta (default I4)
	C	cyan (default I5)
	Y	yellow (default I6)
	W	white (default I7)
	H<ang>	HLS hue angle
	L<pcnt>	HLS lightness in percent
	S<pcnt>	HLS saturation in percent

7.1

Screen Instruction Extensions

1. PURPOSE

Raster scan extensions to the screen instruction allow setting of the background intensity (color and/or gray-scale value), reversal of the image intensities and screen "scrolling".

2. SYNTAX

```
<SCREEN>--> [<aa>] <SCREEN>
        --> <pv> <SCREEN>
        --> ( <SOPS>

<SOPS>  --> N<n> <SOPS>
        --> NO <SOPS>
        --> I <intens>

<intens>--> <digit>
        --> ( <ITOP>

<ITOP>  --> D|B|R|M|G|C|Y|W
        --> H<n>
        --> L<n>
        --> S<n>
        --> )
```

3. GENERAL SEMANTICS

The forms [<aa>] and <pv> allow the screen image to "move" in the viewing area in the same way that most raster CRT devices allow text scrolling. The forms N<n> (<n> is any positive no-zero number,but usually 1) and N0 reverse the screen image (black on white or vice versa) and cause the "normal" image to be shown respectively. The intensity parameter allows the gray-scale level to be set on a scale of 0.00000 (minimum) to 7.00000, the selection of 1 of the eight primary colors, secondary colors or dark and white, or a more precise setting of the color value by using the HLS system of color definition.

7.1.1 Screen motion - [<aa>] and <pv>

These forms are included to allow devices the dynamic capability of performing screen motion as for the case of text scrolling or "strip- chart" generation. Since this is a purely dynamic feature, the range of implementation is very broad ranging from no implementation at all to the full scale "panning" operation of allowing the operator to move the viewing window through a much larger image definition. The intent of this feature is summarized in the folling examples.

```
        S[,+20]
```

move the screen image up the equivalent of "one" text line;

```
        S0
```

move the screen image right one pixel to make room for the next data item in a strip-chart presentation.

7.1.2 Screen image reversal N<n> and N0

These options allow access to the inherent capability of many raster scan CRT devices to "reverse" their image. That is, if the device normally presents characters and lines as white written on a dark background then invocation of the S(N1) option will show the image as a dark line on white background. The option sequence S(N0) then would return the image rendition to its normal impression (N0 is the default value). In the context of color, this feature may have no meaning (ignored) or be used to reverse the color values (blue becomes yellow, red becomes cyan and so forth). Screen reversal should not be used to carry significant information.

7.1.3 Background intensity - I<ITOP>

The inensity parameter in the screen instruction is used to uniformly change the background intensity of the graphic image. The intension is the same as loading a certain colored piece of paper into a plotting device. This parameter may be changed during the process of drawing the image to define the color value of the "off" pixels in a line pattern, or the background color to the rectangle which encloses a character, but no essential information content should be placed in such a dynamically defined background attribute to ensure maximum transportability. In either case, the action of a screen erase operation is to use the then current background intensity to uniformly define the background color and/or gray-scale.

1. Low resolution devices generally will only implement at most the integral gray-scale values and/or the 6 primary/secondary color values but must convert the HLS values to these integral values if they support gray-scale or color.

2. High resolution devices are required to support at least 64 levels of gray-scale or 64 levels on each of the primary colors.

7.2

Writing Attributes Extensions

1. SYNTAX

Raster extensions to the writing attributes instruction allow access to a wide range of capabilities of CRT devices, including color, gray-scale, the simple blink dynamic attribute, image "memory" modification, and area filling.

2. SYNTAX

```
<WOPS>  --> A<n> <WOPS>
        --> A0 <WOPS>
        --> N<n> <WOPS>
        --> N0 <WOPS>
        --> E
        --> C
        --> R
        --> V
        --> S<FILL>
        --> S0
        --> I<intens>

<FILL>  --> <digit>
        --> '<char>'
        --> nil
```

3. GENERAL SEMANTICS

The forms A<n> and A0 are used to enable and disable writing of an image portion using a form of blinking attribute. The options E, C, R, and V are used to modify the image memory. The S<n> and S0 options are used to enable and disable area filling and the I<intens> form is used to identify the foreground writing intensity (value of the "on" pixels in a line pattern or character pattern).

7.2.1 Alternate attribute - A<n> and A0

When enabled by the alternate attribute A<n> (<n> a positive integer usually 1) all image generating instructions cause the pixels to visually blink at device dependent rate. The concept of alternation is that the image section so drawn will for some period of time be visually shown with the foreground intensity and then for a period of time show the underlying background intensity value and then repeat this sequence. The form A0 does not turn off blinking for pixels already written using the A<n> option but only disables the aternate attribute from being associated with pixels in subsequent drawing instructions.

7.2.2 Memory modification - R,C,E,V

For devices which save the pixel image in a memory device of some sort, these options allow the dynamic modification of the memory elements on a pixel basis as defined by the following rules:

1. V option - overlay the new line pattern or characters onto the existing memory pixels ... that is an "OR" operation of the "ON" pattern or character pixels. This is the default REGIS option and means that lines written over text characters will appear visually to strike through the character.

2. R option - Replace writing causes the ON and OFF pixels in the line pattern or character to be written into the memory independent of the current memory values at the addressed locations.

3. C option - Complement writing causes the current line pattern or character pattern to be "XORed" with the current image in such a way that if the same sub-image is written twice at the same postions, the result would appear as if no writing had been performed at all.

4. E option - Erase causes all pixel values addressed in subsequent drawing instructions to revert to their background selector values. Thus, if every pixel were written with the E option the same effect as a screen clear would be achieved independent of the current value of the memory or the line pattern.

Note that these four options are mutually exclusive and generally can not be combined. That is, the invocation of the "R" option will over-ride the previous invocation of the "V" option.

7.2.3 Area shading - S<n>, S'<char>' and S0

These option forms allow devices with built in area filling capabilities to draw an image uniformly filled with the current intensity values. The S<digit> option invokes area filling for all subsequent vector and curve instructions (using the line pattern given by <digit> until the execution of the S0 option (disables filling). The form S'<char>' allows a character pattern in the currently selected alphabet to be used as the filling pattern. The actual operation performed by the device is dependent upon whether the subsequent instructions are defined to be bounded line/curve sequences are open sequences.

1. OPEN SEQUENCES

In this case, the V or C instructions are not given begin (B) and end (E) options. The action taken is as follows. The Y value recorded at the time that the S<n>

option is invoked is used as a vertical reference line. All subsequent V instructions are then drawn as four sided figures with this reference line as the base, the vector drawn as the top (bottom if the vector is below the reference line) and the vector start and end points are taken to be the coordinates of vertical lines which complete the four sided area. Several vectors drawn in this way will then represent a form of polygon which generally has several closed sections. Curve instructions defined by the (S) and (E) options are drawn as if the curve sequnce consisted of infinitesimal vectors. Note that the vector and curve sequences may cross the Y reference line any number of times.

2. BOUNDED SEQUENCES

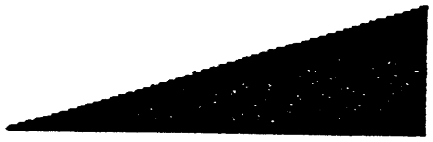
This case is distinguished from the open case by the appearance of begin and end options in the vector or curve instructions. In the case of the vector instruction, the positions specified between (B) and (E) options (including the initial current value) are taken to be points on a closed polygon and the entire bounded area is filled even if there are enclosed image segments (which will get overlayed or replaced as a function of the current memory modification option value). Similarly, the bounded curve sequence will cause a the complete area defined by the interpolation sequence to be filled. The circle curve option is always considered a closed curve and therefore does not require the begin end option specifications. The special case of a circular arc is handled as if the arc so defined is a piece of a "pie" chart.

The shading enable option S<n> will use the selected line pattern to fill in the vertical direction. Additional options of the shading option allow the use of a predefined line pattern or user defined pattern for the horizontal direction as well. Alternately, the user may refer to a previously defined character as the filling pattern in which case the character pattern is replicated sufficiently to fill the area with appropriate clipping of the boundaries of the characters at the edges of the polygon or curve. Hard copy devices may choose to simulate the area filling operation by appropriate cross-hatching patterns. To achieve maximum transport, and efficiency, area filling should not be simulated as a sequnce of vectors.

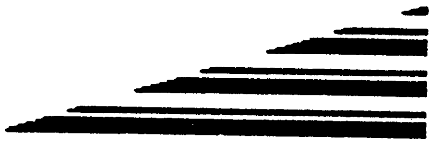
Figure 12 illustrates several examples of area filling for both open and bounded vector and curve sequences.

Figure 12. Vector and Curve Extensions Examples

The shading sequence W(S1)V[+300,-100] gives:

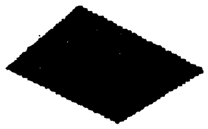


If this same shading sequence is used with the pattern S11111010 then the following pattern shaded triangle is drawn:



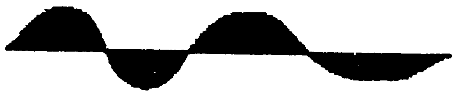
Bounded polygons can be filled using the (B) and (E) options:

V(B)[+50,-50][+90,+50][-60,+40](E)



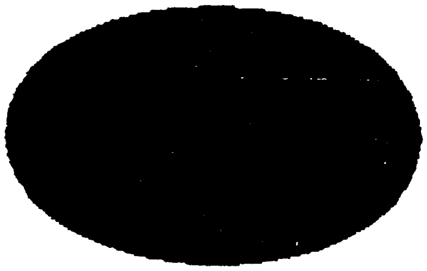
Curves may shaded relative to a y position as in the case of vectors:

C(S)[][+50,-30][+50,+60][+70,-60] ... [](E)



or shaded as a bounded filled area, as for the instruction:

C(B)[+150,-100][+150,+100][-150,+100](E)



7.3

Text Instruction Extensions

1. Extensions for graphics text include user definable character parameters and user definable alphabets.

2. SYNTAX

```
    <TEXT>  --> [<ra>]

    <TOPS>  --> A <digit> <TOPS>
            --> S [<aa>] <TOPS>
            --> M [<aa>] <TOPS>
            --> D <angle> <TOPS>
            --> I <angel> <TOPS>

    <REGIS> --> L <LOAD>

    <LOAD>  --> [ <SKIPB> <LOAD>
            --> ( <LOPS>
            --> '<text> <LOAD>
            --> <text>

    <LOPS>  --> D <LOPS>
            --> H <LOPS>
            --> S [<aa>] <LOPS>
            --> I <intens>
            --> )
```

7.3.1 Text instruction extensions

Figure 13 summarizes the user selectable text character parameter controls. These parameters are summarized in the following.

T[<ra>] - The relative position argument <ra> is used to define the relative X and Y changes to be applied after each character is written. These explicit parameters are used even though consecutive characters may overlap.

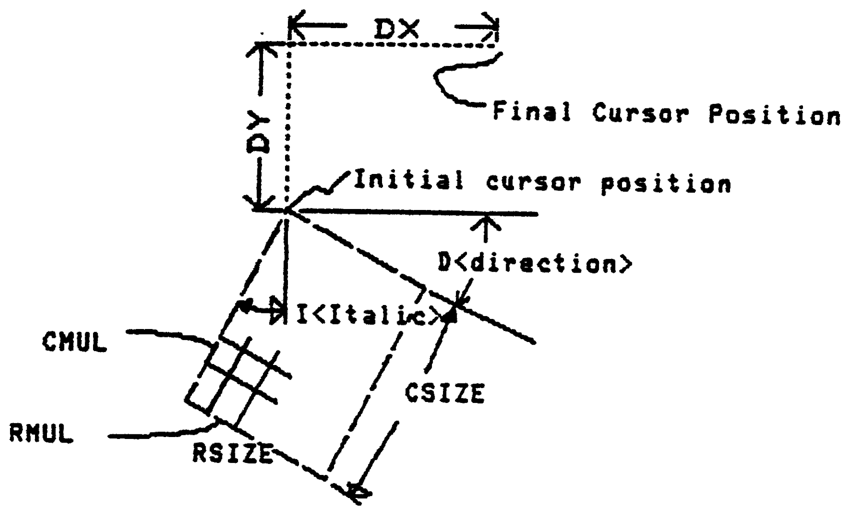
S[<aa>] - The X and Y portions of the absolute argument <aa> are used to define the width and height of the character rectangle to be drawn (row size and column size).

M[<aa>] - The X and Y portions of the absolute argument <aa> are used to define the pixel multiplier to be applied to the width and height. Generally, the width multiplier is selected so that the row size divided by the row multiplier is the same as the character width measured in pixel vectors but may be different from this value for special purposes. In particular, fewer than a multiple number of pixels may be drawn so that not all of the character is drawn or more than an integral number of pixels are drawn in which case the character pattern is repeated.

D<angle> - Direction selects the angle which the width of the character makes relative to the horizontal axis and is given in degrees.

I<angle> - Italic angle specifies the angle that the height (columns) of the characters are written relative to the direction of the columns as defined by the then current character direction.

Figure 13. Text Parameter Extensions



DEFINITION OF PARAMETERS

DX = change in X position after writing
DY = change in Y position after writing

CSIZE = column size

RSIZE = row size

CMUL = column dot multiplier

RMUL = row dot multiplier

D<angle> = row direction, range of:
    0 <= <angle> <= 360

I<angle> = Italic offset

7.3.2 User defined characters - A option and L instruction

A REGIS device may implement several alphabets including font variation as well as foreign language and graphic symbol characters. These alternate alphabets are selected using the "A" option in the text instruction. Up to 10 different alphabets are syntactically distinguishable. The alphabet numbered 0 is always the national character set and may not be modified in content except for font representation.

REGIS allows one or more of the alphabets 1, 2, ... 9 to be memory loadable using the LOAD instruction. The general syntax of a load instruction is the letter "L" followed by the ASCII character index which will be used to refer to the character and this is then followed by a sequence of numeric parameters each of which specifies a single row in the the character pattern. The number of row patterns given generally implies the character size. By default, the row patterns (most significant bit is the left most displayed pixel and the first parameter is the top of the character) are defined by decimal integers separated by commas.

The load instruction options allow selection of character size on an alphabet basis and number base for definition. In particular, the option H selects the hexa-decimal number base for row definition (the most efficient base for line communication), the S[<aa>] allow the user to define the size of the characters to be loaded in a manner similar to the text instruction size parameter, and the I<inters> option allows a background color to be associated with a specific character definition.

Figure 14 illustrates examples of the extended T instruction and the L instruction.

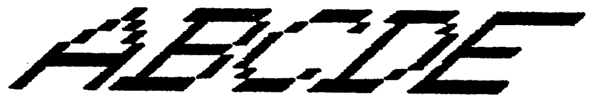
Figure 14. Extended TEXT Instruction Examples

Using T[+10,+0](A0,S1,D0,I0)

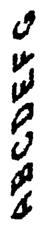
Using T[+20,+0](S2)'ABCDEFG' gives:

Using T[+30,+0](S3)'ABCDEFG' gives:

Using T[+100,+30](S9,I-45)'ABCDE' gives:



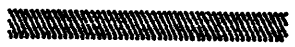
Using D90 (direction up) and I45 gives:



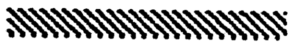
Using the alphabet character defined by:

L'A'8844221188442211884 4;

then T(A1,S[200,20],M[1,2])'A' gives:



The same character with T(M[2,2]) gives:



Note that this type of character can be used as the primitive of a bar chart drawing program.

7.4 Read Parameters Instruction - R

1. PURPOSE

The read parameters instruction, key-letter "R" allows the user to read back to the "host" computer (source of REGIS instructions) parameters of the drawing process in interactive environments.

2. SYNTAX

```
    <REGIS> --> <RINST>

    <RINST> --> ( <ROPS>
            --> [ <SKIPB> <RINST>
            --> ' <SKIPQ> <RINST>
            --> " <SKIPQ2> <RINST>
            --> <digit> <RINST>

    <ROPS>  --> P <ROPS>
            --> , <ROPS>
            --> ) <RINST>
            --> <any-other> <SKIPI>
```

3. GENERAL SEMANTICS

The option R(P) causes the current graphics cursor position to be transmitted to the host source in an interactive terminal environment. The general syntax of the response is:

<header> <X-part>,<Y-part> <suffix>

where <X-part> and <Y-part> are numeric strings which define the current writing position in terms of the current screen coordinates definition. The <header>and <sufix> are implementation sequences which allow the host to recognize the response string in the possible context of other character strings.

8.0

INSTALLATION ENVIRONMENTS

For the purposes of this report, REGIS addresses two environments for the application of the graphics instructions: (1) embedding in ANSI escape sequences and (2) use in a bounded graphics system.

8.1

ANSI Encoding

At the time of this report, the most logical approach to embedding REGIS in ANSI escape sequence environments appears to be the use of the application program selection sequences. That is, a graphic definiton sequence is initiated by the invocation of the escape sequnce:

```
        .esc._
```

and terminated by the sequence:

```
        .esc.\
```

All characters within this sequence are interpreted as REGIS instructions. Although an entire screen image does not have to be included in one invocation of these sequences, individual instructions should not be broken up accross escape sequence boundaries.

8.2

Bounded Systems

As discussed earlier, REGIS is also intended to be used in bounded systems even if no remote communications link is involed. This is to allow maximum transportability of images defined in a bounded system to be used in distributed systems and vice versa.

In the realm of such bounded systems, REGIS should be used for the following:

1. Graphics text file definition of a graphic image.

2. Direct use of REGIS instructions in high level lenaguages when no embedded graphics statements are provided.

3. Communication of a graphics image to a separate graphics device.

It is assumed that such a device would have a REGIS interface "driver" to convert the character strings originating at any of these sources to the native graphics hardware instructions of the system.