

B U G S

The SIMALE Standard Graphics Package  
Preliminary Description<sup>1</sup>

Harold Webber

Russell W. Burns

revised: December 12, 1977  
printed: January 30, 1978

---

<sup>1</sup>The research for this project was done under a grant from the National Science Foundation (GJ-28401X) and a contract from the Office of Naval Research (n00014-67-A-0191-0023). Reproduction in whole or in part is permitted for any lawful purpose of the United States Government.

## TABLE OF CONTENTS

|         |  |    |
|---------|--|----|
| 1       | Introduction.....                              | 1  |
| 2       | SIMALE Software.....                           | 2  |
| 2.1     | Display Structure Organization.....            | 2  |
| 2.1.1   | Images.....                                    | 2  |
| 2.1.2   | Image Transformations.....                     | 3  |
| 2.1.3   | Image Clipping and Projection.....             | 4  |
| 2.1.4   | Image Attributes.....                          | 6  |
| 2.1.5   | Sub-Image Linking.....                         | 7  |
| 2.1.6   | The Stack.....                                 | 9  |
| 2.2     | Modes and Communication.....                   | 10 |
| 2.2.1   | ETC and SIMALE External Registers.....         | 10 |
| 2.2.2   | Processing Modes.....                          | 12 |
| 2.2.2.1 | Display Mode.....                              | 12 |
| 2.2.2.2 | Correlate Mode.....                            | 12 |
| 2.2.2.3 | Invert Screen Coordinates Mode.....            | 13 |
| 3       | Display Structure Subblocks.....               | 14 |
| 3.1     | Initialize (Entry Point: INIT).....            | 14 |
| 3.2     | View (Entry Points: VIEW2D, VIEW3D).....       | 14 |
| 3.2.1   | Display Mode.....                              | 14 |
| 3.2.2   | Correlate Mode.....                            | 15 |
| 3.2.3   | Invert Mode.....                               | 15 |
| 3.2.4   | View Subblock Data.....                        | 15 |
| 3.2.5   | 2D Format.....                                 | 17 |
| 3.2.6   | 3D Format.....                                 | 18 |
| 3.3     | Cursor (Entry point: CURSOR).....              | 18 |
| 3.4     | Call (Entry Points: CALL2D, CALL3D).....       | 18 |
| 3.4.1   | Call Subblock Data.....                        | 19 |
| 3.4.2   | 2D Format.....                                 | 19 |
| 3.4.3   | 3D Format.....                                 | 20 |
| 3.5     | Return (Entry Points: RETURN2D, RETURN3D)..... | 20 |
| 3.6     | Extent (Entry Points: EXTENT2D, EXTENT3D)..... | 20 |
| 3.6.1   | Extent Subblock Data.....                      | 21 |
| 3.6.2   | 2D Format.....                                 | 22 |
| 3.6.3   | 3D Format.....                                 | 22 |
| 3.7     | Name (Entry Points: NAME2D, NAME3D).....       | 22 |
| 4       | Graphic Primitives.....                        | 23 |
| 4.1     | Curve (Entry Points: CURVE2D, CURVE3D).....    | 23 |
| 4.1.1   | 2D Format.....                                 | 23 |
| 4.1.2   | 3D Format.....                                 | 24 |
| 4.2     | Lines (Entry Points: LINES2D, LINES3D).....    | 24 |
| 4.2.1   | 2D Format.....                                 | 24 |
| 4.2.2   | 3D Format.....                                 | 25 |
| 4.3     | Text (Entry Point: TEXT2D, TEXT3D).....        | 26 |
| 4.3.1   | 2D Format.....                                 | 26 |

|         |   |    |
|---------|---|----|
| 4.3.2   | 3D Format.....                                    | 26 |
| 5       | Basic Software.....                               | 27 |
| 5.1     | The Routines.....                                 | 27 |
| 5.1.1   | Initialization and Termination.....               | 27 |
| 5.1.1.1 | PROSTART.....                                     | 27 |
| 5.1.1.2 | PROTERM.....                                      | 28 |
| 5.1.2   | Image Manipulation.....                           | 28 |
| 5.1.2.1 | OPENIMAG.....                                     | 28 |
| 5.1.2.2 | CALLRDIM.....                                     | 29 |
| 5.1.2.3 | CALLDDIM.....                                     | 29 |
| 5.1.2.4 | DELIMAGE.....                                     | 30 |
| 5.1.3   | Segment Manipulation.....                         | 30 |
| 5.1.3.1 | ADDSEGMENT.....                                   | 30 |
| 5.1.3.2 | DELSEGMENT.....                                   | 31 |
| 5.1.4   | Subblock Manipulation.....                        | 31 |
| 5.1.4.1 | ADDSUBLK.....                                     | 31 |
| 5.1.4.2 | BLKTYPE.....                                      | 32 |
| 5.1.4.3 | SIMHTRAN.....                                     | 32 |
| 5.1.5   | Dynamic Subblock Manipulation.....                | 32 |
| 5.1.5.1 | ADDDYNAM.....                                     | 32 |
| 5.1.5.2 | ADDMARK.....                                      | 33 |
| 5.1.5.3 | QUERYDYN.....                                     | 33 |
| 5.1.5.4 | DELDYNAM.....                                     | 34 |
| 5.1.6   | Garbage Collection -- MRCLEAN.....                | 34 |
| 5.2     | Interrupts and Analog Devices.....                | 35 |
| 5.2.1   | Function Keys.....                                | 35 |
| 5.2.2   | Alphanumeric Keyboard.....                        | 35 |
| 5.2.3   | The Cursor.....                                   | 36 |
| 5.2.4   | Analog Input.....                                 | 37 |
| 6       | Appendix I -- Graphic Data Structure Formats..... | 38 |
| 6.1     | The Directory.....                                | 38 |
| 6.2     | The Buffer.....                                   | 39 |
| 6.3     | Image Format.....                                 | 40 |
| 6.3.1   | Unsegmented Format.....                           | 40 |
| 6.3.2   | Segmented Format.....                             | 41 |
| 6.4     | Interrupt Scanout Area.....                       | 42 |
| 6.5     | Dynamo Format.....                                | 43 |
| 7       | Appendix II -- CRUSTY Dummy Section.....          | 44 |
| 7.1     | Prefix.....                                       | 44 |
| 7.2     | Function Key Control.....                         | 45 |
| 7.3     | Cursor Control.....                               | 45 |
| 7.4     | Analog Devices Scanout Area.....                  | 45 |
| 7.5     | Alpha-numeric Keyboard Control.....               | 46 |

## 1 INTRODUCTION

This document describes the lowest level of the BUGS standard graphics package. Section 2 describes the graphic data structure organization and the functions performed by the SIMALE. SIMALE subblocks may be of two types: section 3 describes a basic set of control subblocks, while Section 4 describes a basic set of display subblocks (graphic primitives). Section 5 describes a set of subroutines, called PROCRUSTES, that provides a low level graphic access method to a program running on the Meta 4A. PROCRUSTES contains routines for building and manipulating the graphic data structure. In addition, PROCRUSTES services interrupts, polls analog input devices, and refreshes the Vector General.

The reader is assumed to be familiar with the BUGS system and with the Meta 4A Principles of Operation and the Meta 4B/SIMALE/Vector General Principles of Operation.

## 2 SIMALE SOFTWARE

The standard graphics package presents its user with a complete and general set of SIMALE programs for applications requiring hierarchical picture data structures and real-time performance. No "standard" package could ever be optimal for all applications, of course, and truly large or performance critical applications may require the hand tooling of all BUGS processors.

This package was designed to be as simple as possible and to remain extensible in the area of graphic data types. Above all, it is hoped this package will provide support for graphics extensions to Algol W.

### 2.1 DISPLAY STRUCTURE ORGANIZATION

A picture to be displayed by the standard graphics package resides in core in the form of a hierarchical data structure. A push-down stack and a directory of images are employed to aid processing of the display structure. The display file, the stack, and the directory are all maintained in BUGS core.

#### 2.1.1 IMAGES

The graphic data structure of the standard graphics package is analogous to a program structure consisting of procedures which may call other procedures. In the graphic data structure, there are images which may call other images (sub-images). An image consists of a ring of blocks containing subblocks each of which contains graphic data or calls to other images. Section 3 describes the display structure subblocks and some of the graphics data subblocks.

The highest level image, the root image, is the one pointed to by the ETC instruction which starts the display. The root image has a special purpose; it defines the use of the screen coordinate space and thus may not contain any graphical data subblocks. All other images are defined in

16-bit image coordinate space which is mapped into the screen space by the root image.

## 2.1.2 IMAGE TRANSFORMATIONS

Any image referenced within another image is subject to transformation by a matrix. This matrix is specified at the time of the sub-image call and can be made to rotate, scale and displace the sub-image within the calling image. Each data element of an image is post multiplied by the current transformation matrix. For two dimensional images this multiplication is of the form:

$$[X, Y, 1] \begin{bmatrix} A(1,1) & A(1,2) & 0 \\ A(2,1) & A(2,2) & 0 \\ A(3,1) & A(3,2) & 1 \end{bmatrix} = [X', Y', 1]$$

For three dimensional images it is of the form:

$$[X, Y, Z, 1] \begin{bmatrix} A(1,1) & A(1,2) & A(1,3) & 0 \\ A(2,1) & A(2,2) & A(2,3) & 0 \\ A(3,1) & A(3,2) & A(3,3) & 0 \\ A(4,1) & A(4,2) & A(4,3) & 1 \end{bmatrix} = [X', Y', Z', 1]$$

Successive sub-image calls require the ability to compose transformations. The SIMALE pushes the current transformation matrix onto a stack and accomplishes the composition of transformations by multiplying the new matrix times the current matrix during sub-image calls.

Due caution must be taken when assigning transformation matrices as it is possible for successive image calls to displace an image outside the 16 bit image co-ordinate space. This will make extents (see Section 3.6) impossible to define, and could in some cases cause image wraparound.

### 2.1.3 IMAGE CLIPPING AND PROJECTION

After transformation, all image data elements are subject to clipping and projection. Clipping is the process of finding those elements or pieces of elements which are visible within a window. In two dimensions, the window is a rectangular region in the image space. In three dimensions, the window is implicitly defined to be a pyramid whose sides are the planes  $x=+z$ ,  $x=-z$ ,  $y=+z$ ,  $y=-z$  and whose vertex is the origin.

Once clipped, the visible data elements must be mapped (or projected) from the window onto the viewport region of the screen. This is accomplished by dividing the  $x$  and  $y$  coordinates by the window scales in two dimensions, or the  $z$  coordinate in three dimensions. This scales up the coordinates to a dimensionless unit space where they are turned into display orders and scaled and displaced by the Vector General to complete the window to viewport projection.

Viewport/Window Transformation

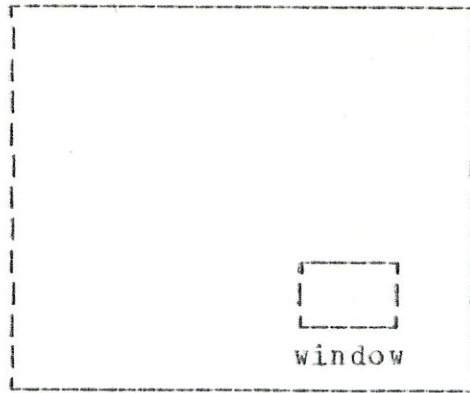
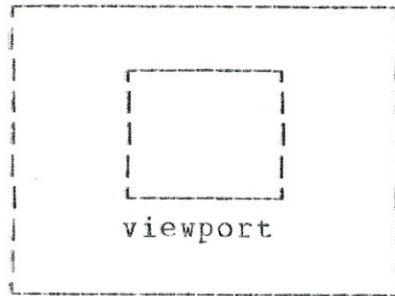


Image space (16 bits x 16 bits)



Vector General screen  
(-1,-1) to (1,1)



#### 2.1.4 IMAGE ATTRIBUTES

In addition to being subject to a transformation, any image may be modified by a halfword of attributes specified at the time of a sub-image call. These attributes allow different instances of sub-images to be displayed differently. The attribute halfword is:

|           |    |     |     |   |     |   |
|-----------|----|-----|-----|---|-----|---|
| Intensity | ** | VM  | I   | P | C   | B |
| 0         | 7  | 8 9 | A E | C | D E | F |

Intensity (bits 0-7) - This eight-bit field determines the intensity at which a sub-image is displayed (a positive number; 0 is dimmest, 7F is brightest). When attributes are composed for a sub-image call, the maximum of the two intensities is chosen as the new intensity attribute, therefore sub-images can be made brighter than their calling images.

\*\* (bits 8 and 9) - Unused.

VM:Vector Mode (bits 10-11) - This field determines the type of lines drawn by the Vector General for all lines drawn in the sub-image. These are the same vector mode bits found on Vector General orders and include solid vectors (x'0'), dashed vectors (x'1'), dotted vectors (x'2'), and end points (x'3'). When composed, the max is taken meaning solid vectors can be turned into dashed, dashed into dotted, etc.

I:Invert Enable (bit 12) - This bit, when set, allows a viewport to be processed by invert mode.

P:Pick Disable (bit 13) - This bit, when set, prevents a sub-image from being scanned during correlation mode. This is useful to prevent an image from being picked when the user so wishes. During composition the "OR" of these bits is chosen, so sub-images can be forced to be unpickable.

C:Clip Enable (bit 14) - This bit is maintained by the SIMALE to tell if a given image is totally visible and therefore requires no clipping (clipping disabled) or is only partially visible and requires clipping (clipping enabled). The user will most likely always wish to enable clipping and leave

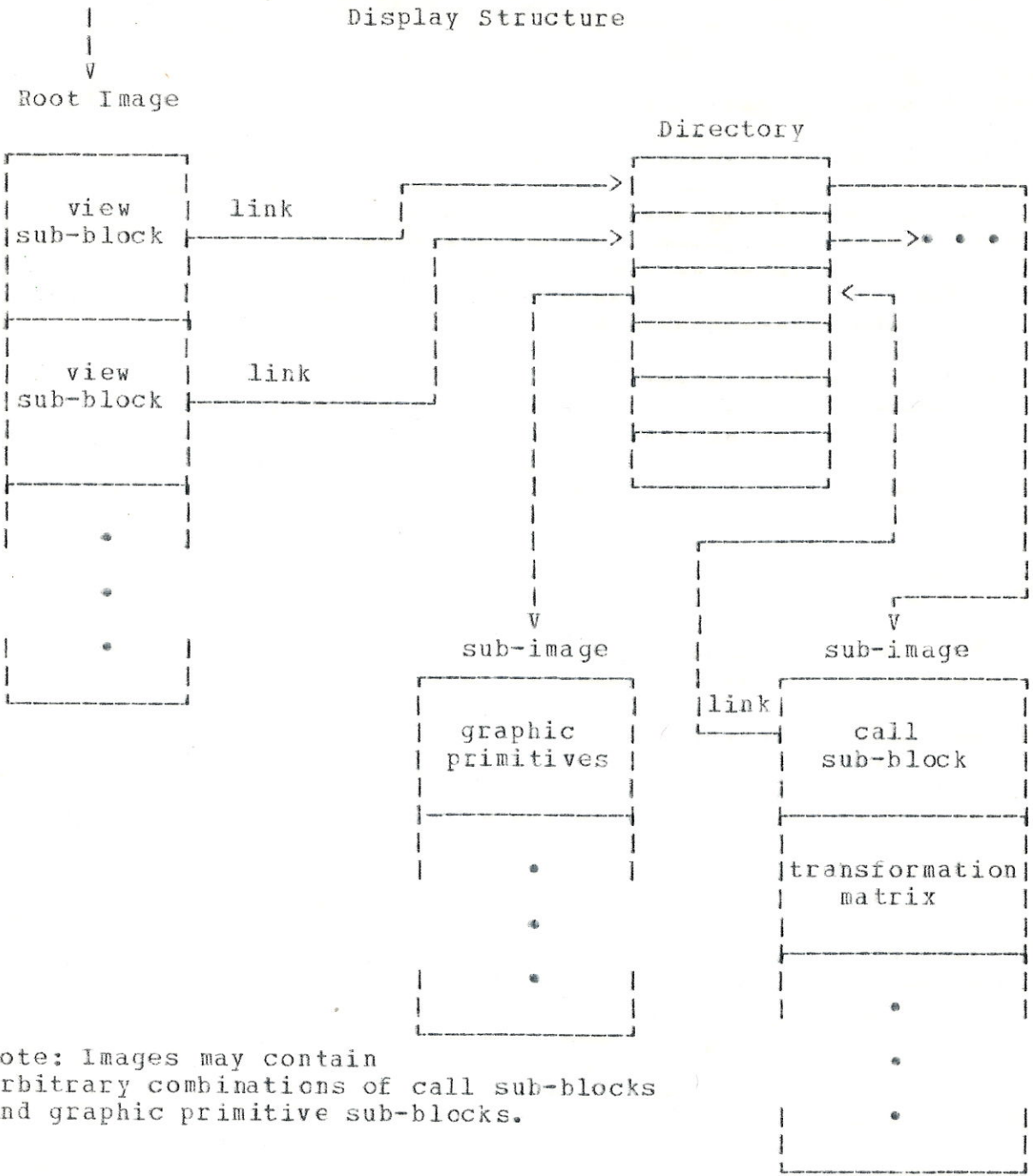
maintenance of this bit to extent subblocks (see 3.6).

B:Blink Enable (bit 15) - This bit will cause the display of the sub-image to blink. Composition of blink bits is accomplished by "OR"ing so it is possible to force sub-images to blink.

#### 2.1.5 SUB-IMAGE LINKING

References within images to other images are made via a directory in core. This directory is maintained by PROCURSTES. Each image in the display structure has its own entry in the directory which contains the data base pointer for that image, and any other information the user wishes. Sub-image linking is accomplished by taking a halfword link from the calling image, and adding it to the directory base register (address 2C), and using the result to find the correct entry in the directory. The entry should be the data base pointer for the called image. If the pointer in the directory is zero, the reference is treated as a call to a null image; if the pointer is odd, a SIMALE interrupt will occur with an interrupt code of 16.

Display Structure



Note: Images may contain arbitrary combinations of call sub-blocks and graphic primitive sub-blocks.

### 2.1.6 THE STACK

Because the display structure can have many images, some of which are referenced within other images, a stack is necessary to save the processing environment when sub-image calls are made. This stack is maintained by the Meta 4B firmware using local storage registers 23-27 for this purpose.

When a call is made to a sub-image, a new stack element is created using the the stack base pointer (local address 23), the stack element size (address 24), and the stack element offset (address 26) to find the starting address and to check for possible stack overflow. This starting address is used to set the stack data pointer (address 27) which is incremented each time something is pushed into the stack.

The firmware automatically pushes local storage addresses 20-22 and 30-33 into the stack to save the current image's link, attributes, and name (see Section 4.2). The SIMALE then pushes in the current transformation matrix to fill the stack element. For 2-dimensional sub-image calls, 13 halfwords are required for each stack element, and for 3-dimensional calls, 19 halfwords are required.

## 2.2 MODES AND COMMUNICATION

In the standard package the SIMALE has been programmed to produce pictures while interpreting a hierarchical data structure, or structured display file. This display file is continuously scanned by the SIMALE to maintain a display on the Vector General.

### 2.2.1 ETC AND SIMALE EXTERNAL REGISTERS

All communication with the SIMALE standard package is accomplished via a register file in the Meta 4B's local storage. The thirty-two registers of this file encompass the ETC registers (local store addresses 20-2F), and the SIMALE External registers (local store addresses 30-3F). In general, the ETC registers allow the Meta 4B firmware to chain through ETC blocks and subblocks and to maintain the stack and image directory. The SIMALE external registers are used as a communication area between the SIMALE's programs and the Meta 4B. A description of those ETC registers dealing with initial and final data and block pointers can be found in the Meta 4B/SIMALE/Vector General manual. All others are described in this manual. The allocation of registers is as follows:

### ETC Registers

| <u>Name</u>            | <u>Addr.</u> | <u>Function</u>   |
|------------------------|--------------|---|
| Stack Base Pointer     | 23           | Pointer to the beginning of the SIMALE Call stack in core.              |
| Stack Element Size     | 24           | The length of each stack element.                                       |
| Stack Maximum Offset   | 25           | The length of the stack in bytes.                                       |
| Stack Element Offset   | 26           | Offset in bytes from the top of the stack of the current stack element. |
| Stack Data Pointer     | 27           | Address of the next word in the stack to be read or written.            |
| Directory Base Pointer | 2C           | Pointer to the start of the directory.                                  |

### SIMALE External Registers

| <u>Name</u>            | <u>Addr.</u> | <u>Function</u>   |
|------------------------|--------------|---|
| Current Image Link     | 30           | Directory offset of image currently being processed.                                |
| Current Attributes     | 31           | Composed attributes of that image.  |
| Element Name           | 32           | Name of the current element in image (set by NAME, section 3.7).                    |
| ?                      | 33           | Reserved for user; this register will be pushed on stack automatically during CALL. |
| SIMALE Processing Mode | 34           | Mode of operation desired (see section 2.2.2).                                      |
| Correlate Center X,Y   | 35-36        | X,Y co-ordinates of the center of the pick viewport.                                |
| Correlate Scale X,Y    | 37-38        | X,Y scale of the pick viewport.   |
| Inverted Center X,Y    | 39-3A        | Inverted X,Y center position, returned by Invert mode (section 2.2.2.3).            |
| Inverted Scale X,Y     | 3B-3C        | Inverted X,Y scale, returned by Invert mode.  |

## 2.2.2 PROCESSING MODES

The SIMALE is currently programmed to process the display structure in three different ways. A mode is selected by the SIMALE Processing Mode Register (local store address 34). To start processing of the display structure (regardless of mode) the appropriate ETC registers are pointed at the root block (see section 2.1.1) and the ETC instruction is executed.

### 2.2.2.1 Display Mode

(Processing Mode = 0)

Display mode is the most common and is the only one to produce pictures on the Vector General. In this mode the SIMALE runs through the display structure, transforming picture elements, determining which are visible, and outputting instructions to the Vector General.

### 2.2.2.2 Correlate Mode

(Processing Mode = 1)

In this mode, the SIMALE is passed, in registers 35-38, the center and scale in screen coordinates of a correlation viewport. The SIMALE then scans the display structure for the first picture element (line, character, etc.) in an image enabled for correlation and lies entirely or partially inside this viewport.

Once found, an interrupt is generated (interrupt code=1), and processing stops. By looking at the ETC and SIMALE external registers and at the stack, it is possible to know which element caused the interrupt, what image, block, and subblock it was in, and the history of sub-image calls. Portions of this information are found in the interrupt scanout area maintained by PROCUSTES (see Section 6.4).

### 2.2.2.3 Invert Screen Coordinates Mode

(Processing mode = 2)

In invert mode, the SIMALE takes the correlation viewport center and scale in registers 35-38 and searches the display for the first viewport with invert enabled that contains the correlation center. The correlation viewport is then projected back from screen space to image co-ordinate space and the result is returned in registers 39-3C. Finally an interrupt (interrupt code=2) is generated, and processing stops.



### 3 DISPLAY STRUCTURE SUBBLOCKS

Subblocks are the fundamental components of images. Each subblock type is interpreted by a different SIMALE program or "control store load". The entry point name of the SIMALE program that interprets a subblock is used to identify that subblock type. An arbitrary number of subblocks can be placed consecutively to form blocks, which are linked together to form an image. There are two broad categories of subblocks, those which control the flow of interpretation of the display structure, and those which actually produce display information. More emphasis will be given now to the control types, as they determine the form and capabilities of the display structure.

#### 3.1 INITIALIZE (ENTRY POINT: INIT)

The initialize subblock must be executed once before use of the standard package to ensure initialization of SIMALE registers and constants used throughout the package. This subblock is executed as part of program initialization and is not part of any image. This subblock is created during PROCRUSTES initialization.

#### 3.2 VIEW (ENTRY POINTS: VIEW2D, VIEW3D)

The view subblock may only be used in a root image. Its function is to manage the mapping of images onto the screen space. Two types of view subblocks exist, one for display of two-dimensional images and one for three-dimensional images. The next three sections describe the operation of the view subblock in each SIMALE mode.

##### 3.2.1 DISPLAY MODE

In this mode VIEW sets the scale and displacement registers of the Vector General to accomplish the window to viewport transformation, resets the current transformation

matrix to unity and causes display of the referenced image to begin.

### 3.2.2 CORRELATE MODE

To accomplish the correlation function, VIEW compares the screen coordinates of the pick viewport registers (35-38) with the viewport defined by each view subblock in the root image. If the pick viewport is within the area of the screen enclosed by the view subblock viewport, correlation on the image referenced by the view subblock will be performed, otherwise processing of the referenced image is skipped and execution is resumed at the next view subblock. If correlation is to be performed and if correlation is enabled in the attributes of the referenced image, the pick viewport is transformed into unit space to produce a special pick window. The image is then processed to see if any elements fall within the pick window.

### 3.2.3 INVERT MODE

This mode requires VIEW2D to process the pick viewport registers (35-38). If the pick center is found to be within the specified viewport, the pick viewport is translated into the image coordinate space, these values are returned in registers 39-3C, and an interrupt is generated. If these values are not in the viewport, processing continues at the next subblock in the root image. Any transformation matrix specified on the viewport will be ignored in the inverse calculation.

### 3.2.4 VIEW SUBBLOCK DATA

Link: This is the link to the directory entry of the image to be displayed or correlation processed.

Attr.: These are the image's attributes.

Viewport: These are the center x,y position and scale of a region on the screen into which the visible portion of the displayed image is mapped. If not specified, the viewport bounds of the previous view subblock will be used.

Window: For VIEW2D only this is the center x,y position and scale of the region in the image space to which display primitives are clipped and mapped into the viewport screen space. If not specified, the window bounds of the previous view subblock will be used.

For VIEW3D, the window is implicitly defined as the planes  $x = -z$ ,  $y = -z$ ,  $x = +z$ , and  $y = +z$ ; the "pyramid of vision".

Matrix - This is an optional transformation matrix which, if specified, is applied to the viewed image. If no matrix is specified, the unit matrix transformation is applied to the viewed image.

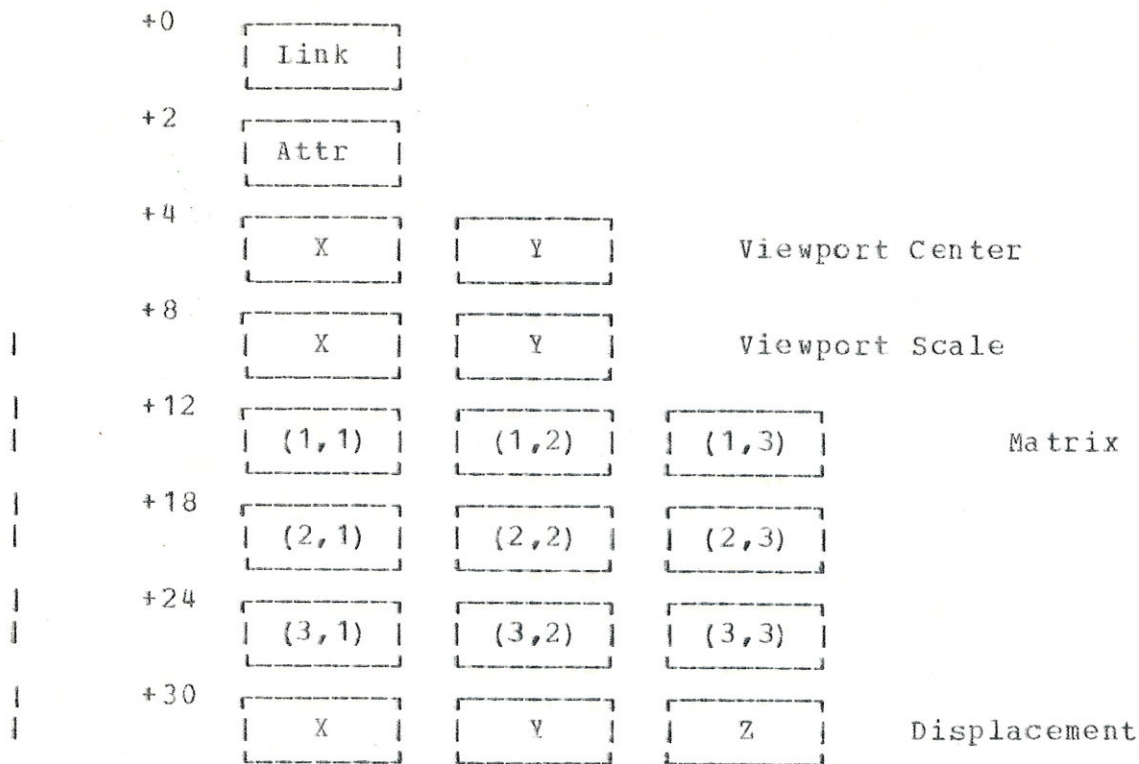
For VIEW2D the matrix consists of six halfwords; a 2 x 2 rotation matrix plus an x and y displacement.

For VIEW3D the matrix consists of twelve halfwords, a 3 x 3 rotation matrix plus an x, y, and z displacement.

### 3.2.5 2D FORMAT

|       |   |       |   |       |                 |
|-------|---|-------|---|-------|-----------------|
| +0    | <table border="1"><tr><td>Link</td></tr></table>  | Link  |   |       |                 |
| Link  |   |       |   |       |                 |
| +2    | <table border="1"><tr><td>Attr</td></tr></table>  | Attr  |   |       |                 |
| Attr  |   |       |   |       |                 |
| +4    | <table border="1"><tr><td>X</td></tr></table>     | X     | <table border="1"><tr><td>Y</td></tr></table>     | Y     | Viewport Center |
| X     |   |       |   |       |                 |
| Y     |   |       |   |       |                 |
| +8    | <table border="1"><tr><td>X</td></tr></table>     | X     | <table border="1"><tr><td>Y</td></tr></table>     | Y     | Viewport Scale  |
| X     |   |       |   |       |                 |
| Y     |   |       |   |       |                 |
| +12   | <table border="1"><tr><td>X</td></tr></table>     | X     | <table border="1"><tr><td>Y</td></tr></table>     | Y     | Window Center   |
| X     |   |       |   |       |                 |
| Y     |   |       |   |       |                 |
| +16   | <table border="1"><tr><td>X</td></tr></table>     | X     | <table border="1"><tr><td>Y</td></tr></table>     | Y     | Window Scale    |
| X     |   |       |   |       |                 |
| Y     |   |       |   |       |                 |
| +20   | <table border="1"><tr><td>(1,1)</td></tr></table> | (1,1) | <table border="1"><tr><td>(1,2)</td></tr></table> | (1,2) | Matrix          |
| (1,1) |   |       |   |       |                 |
| (1,2) |   |       |   |       |                 |
| +24   | <table border="1"><tr><td>(2,1)</td></tr></table> | (2,1) | <table border="1"><tr><td>(2,2)</td></tr></table> | (2,2) |                 |
| (2,1) |   |       |   |       |                 |
| (2,2) |   |       |   |       |                 |
| +28   | <table border="1"><tr><td>X</td></tr></table>     | X     | <table border="1"><tr><td>Y</td></tr></table>     | Y     | Displacement    |
| X     |   |       |   |       |                 |
| Y     |   |       |   |       |                 |

### 3.2.6 3D FORMAT



### 3.3 CURSOR (ENTRY POINT: CURSOR)

The cursor subblock may be referenced only in a root image. Its purpose is to draw the center and the borders of the pick viewport whose screen coordinates are in registers 35-38.

*definition?* →

### 3.4 CALL (ENTRY POINTS: CALL2D, CALL3D)

The call subblock accomplishes sub-image calls within images. The referenced sub-image will have attributes assigned and may be subjected to a transformation by a matrix. The current image environment (i.e. attributes, transformation matrix, name, etc.) is automatically saved on a push down stack.

### 3.4.1 CALL SUBBLOCK DATA

Link - This is the link to the directory entry of the image to be called.

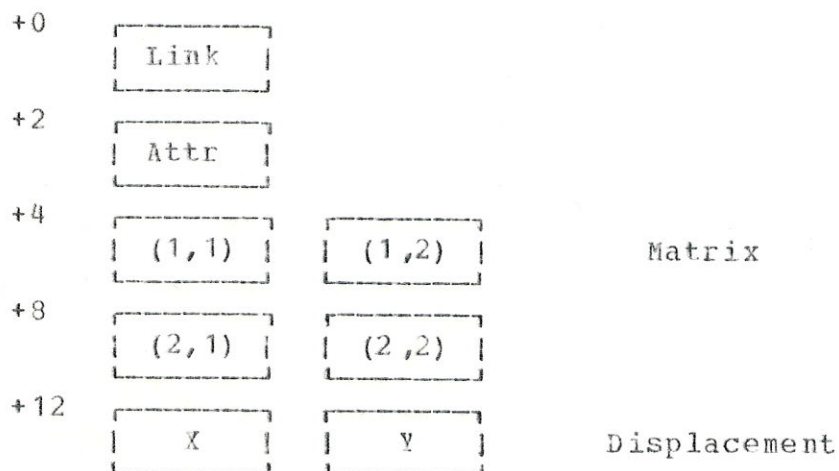
Attr - This is a halfword of attributes which are composed with the present attributes and then applied to the called image.

Matrix - This is an optional transformation matrix which, if specified, is composed with the current transformation matrix and then applied to the called image. If no matrix is specified, the current matrix transformation is applied to the called image.

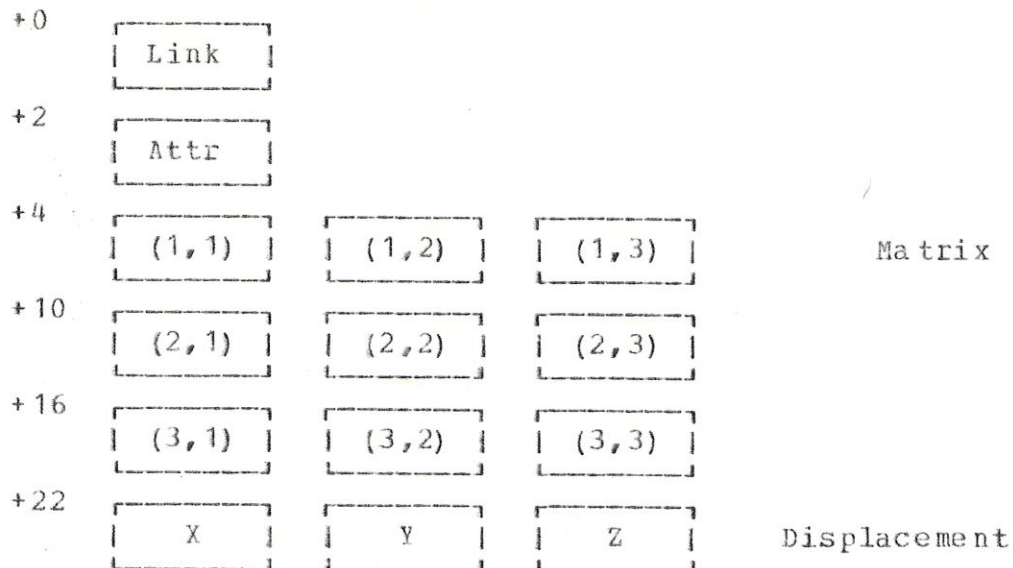
For CALL2D the matrix consists of six halfwords; a 2 x 2 rotation matrix plus an x and y displacement.

For CALL3D the matrix consists of twelve halfwords, a 3 x 3 rotation matrix plus an x, y, and z displacement.

### 3.4.2 2D FORMAT



### 3.4.3 3D FORMAT



### 3.5 RETURN (ENTRY POINTS: RETURN2D, RETURN3D)

This subblock accomplishes a return from a called image. The environment saved on the stack at the time of a sub-image call is restored and processing of the calling image resumes at the next subblock after the CALL which invoked the sub-image. Return subblocks are automatically added to images by PROCUSTES.

### 3.6 EXTENT (ENTRY POINTS: EXTENT2D, EXTENT3D)

The Extent subblock acts like a sub-image return conditional on the results of processing extent information on the current image. Processing of this information can result in a decision that the current image is completely invisible and need not be further processed. It may be possible to tell if the image needs to be clipped or not, or is too small or too big to be displayed.

### 3.6.1 EXTENT SUBBLOCK DATA

Extents - These halfwords define the coordinate limits of the current image. These limits are processed to provide information about all the coordinates in the image.

First the extents are transformed by the current matrix. Next new extents which exactly surround the transformed extents but whose edges are parallel to the co-ordinate axis are found. If the surrounding extents lie completely outside the window, it can be assumed that all lines, characters, and sub-images within the current image will also lie outside the window and a return from the current image can be effected immediately.

If the transformed extents lie completely inside the window, no clipping need be performed, and this fact is passed to all image primitives by resetting the Clip bit of the current image attributes. Otherwise, the extents lie partially within the window, and the clip bit is set.

For EXTENT2D the extents are four halfwords specifying the center x,y position and scale of a rectangular region enclosing the current image.

For EXTENT3D the extents are six halfwords specifying the center x,y,z position and scales of a rectangular solid enclosing the current image.

If the image is totally or partially visible, the size of the image is determined. This is done by first projecting the surrounding extent into dimensionless unit space (see Section 2.1.3). Then one half the square of the distance from the extent center to its upper right hand corner (of the front face in three dimensions) is computed and used for a size criterion.

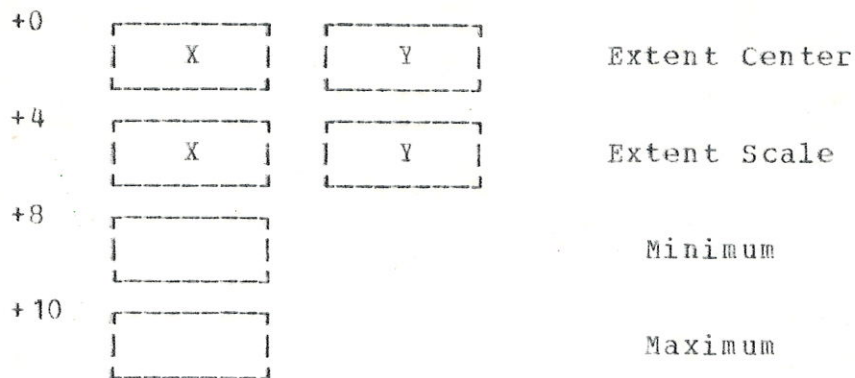
Min size - This halfword is compared with the computed size value. If the size is less than the minimum, a return is forced and the image is not displayed.

Max size - This halfword is compared with the computed size value. If the size is greater than the maximum, a return is forced and the image is not displayed.

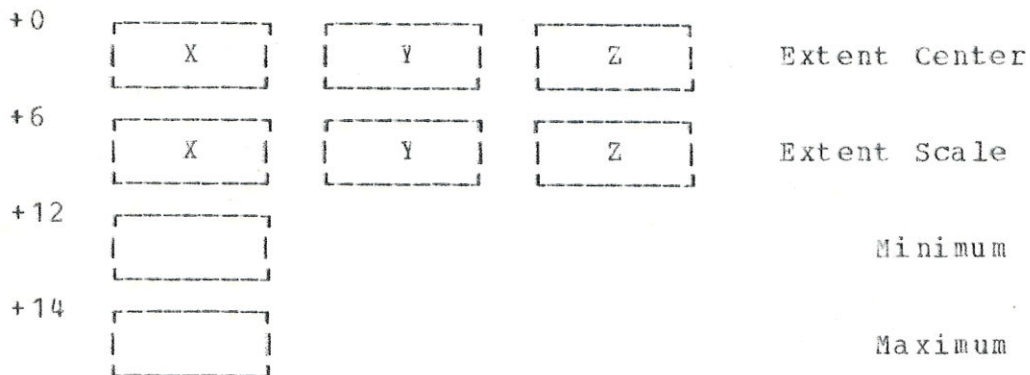
Extents act with the current window specified in a view subblock. Thus varying the window co-ordinates effects the amount of picture detail displayed.



### 3.6.2 2D FORMAT



### 3.6.3 3D FORMAT



### 3.7 NAME (ENTRY POINTS: NAME2D, NAME3D)

The NAME subblock provides a mechanism whereby elements within images may be named. The name specified will be in effect until the next NAME or RETURN subblock.

Name - This halfword, called an element name, is read from the subblock and copied into register 32. This register is automatically stacked and restored by CALL's and RETURN's so at the time of an interrupt it is possible to know the current image element name and call history.

*bad term*

## 4 GRAPHIC PRIMITIVES

Many graphic primitives are easily implemented within the framework of the standard graphic package, and it is expected that many types will be added as users invent them. For now, suffice it to say the standard package will include several types of lines and curves as well as different flows of characters for two and three-dimensional displays. Only the basic line and text subblocks are described below.

### 4.1 CURVE (ENTRY POINTS: CURVE2D, CURVE3D)

This subblock allows continuous line drawing. A starting point is specified, followed by an arbitrary number of points through which line segments are drawn. The line type is determined by the current attributes.

#### 4.1.1 2D FORMAT

|                |   |                |   |                |             |
|----------------|---|----------------|---|----------------|-------------|
| +0             | <table border="1"><tr><td>X<sup>0</sup></td></tr></table> | X <sup>0</sup> | <table border="1"><tr><td>Y<sup>0</sup></td></tr></table> | Y <sup>0</sup> | Start point |
| X <sup>0</sup> |   |                |   |                |             |
| Y <sup>0</sup> |   |                |   |                |             |
| +4             | <table border="1"><tr><td>X<sup>1</sup></td></tr></table> | X <sup>1</sup> | <table border="1"><tr><td>Y<sup>1</sup></td></tr></table> | Y <sup>1</sup> | Next point  |
| X <sup>1</sup> |   |                |   |                |             |
| Y <sup>1</sup> |   |                |   |                |             |
| +12            | <table border="1"><tr><td>X<sup>2</sup></td></tr></table> | X <sup>2</sup> | <table border="1"><tr><td>Y<sup>2</sup></td></tr></table> | Y <sup>2</sup> | ...         |
| X <sup>2</sup> |   |                |   |                |             |
| Y <sup>2</sup> |   |                |   |                |             |
|                | .   |                |   |                |             |
|                | .   |                |   |                |             |
|                | .   |                |   |                |             |

#### 4.1.2 3D FORMAT

|     |       |       |       |             |
|-----|-------|-------|-------|-------------|
| +0  | $X^0$ | $Y^0$ | $Z^0$ | Start point |
| +6  | $X^1$ | $Y^1$ | $Z^1$ | Next point  |
| +12 | $X^2$ | $Y^2$ | $Z^2$ | ...         |
|     | .     |       |       |             |
|     | .     |       |       |             |
|     | .     |       |       |             |

#### 4.2 LINES (ENTRY POINTS: LINES2D, LINES3D)

This subblock allows the drawing of disjoint lines. The endpoints of the lines are specified. The line type is determined by the current attributes.

#### 4.2.1 2D FORMAT

|     |       |       |             |
|-----|-------|-------|-------------|
| +0  | $X^0$ | $Y^0$ | Start point |
| +4  | $X^1$ | $Y^1$ | End point   |
| +8  | $X^0$ | $Y^0$ | Start point |
| +12 | $X^1$ | $Y^1$ | End point   |
|     | .     |       |             |
|     | .     |       |             |
|     | .     |       |             |

#### 4.2.2 3D FORMAT

|     |       |       |       |             |
|-----|-------|-------|-------|-------------|
| +0  | $X^0$ | $Y^0$ | $Z^0$ | Start point |
| +6  | $X^1$ | $Y^1$ | $Z^1$ | End point   |
| +12 | $X^0$ | $Y^0$ | $Z^0$ | Start point |
| +18 | $X^1$ | $Y^1$ | $Z^1$ | End point   |
|     | .     |       |       |             |
|     | .     |       |       |             |
|     | .     |       |       |             |

#### 4.3 TEXT (ENTRY POINT: TEXT2D,TEXT3D)

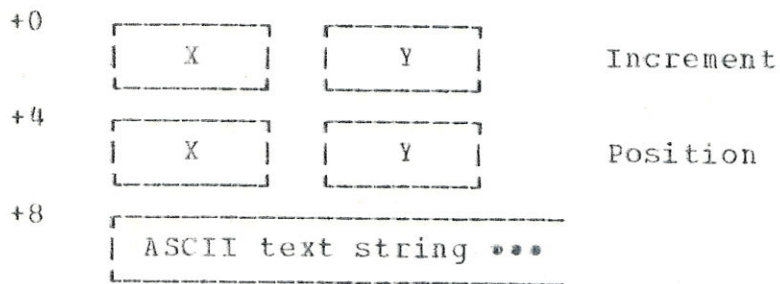
This subblock provides for character display. The position, character spacing, and an ASCII character string are specified. The character spacing is subject to rotation and scaling by the current transformation matrix.

Increment - These halfwords specify inter-character spacing which also determines character size and orientation. If the X increment is greater than the Y increment then the characters are horizontal; if the X increment is less than the Y increment the characters are vertical.

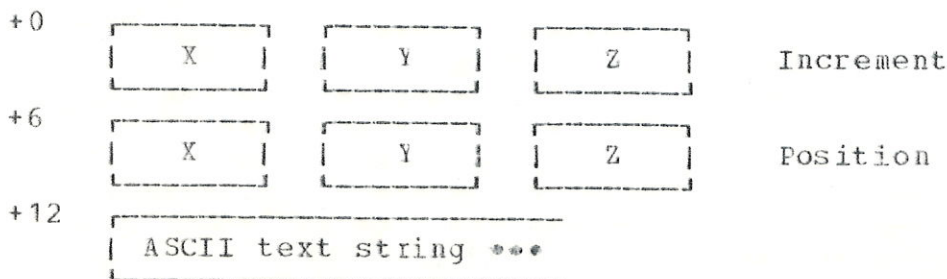
Position - These halfwords specify the position of the center of the first character.

Text - An arbitrary number of ASCII characters.

##### 4.3.1 2D FORMAT



##### 4.3.2 3D FORMAT



## 5 BASIC SOFTWARE

A set of subroutines, PROCUSTES, has been written to allow low level access to the facilities of the SIMALE Standard Graphics Package. It can also be used as a stand-alone graphics access method. PROCUSTES maintains the directory and buffer, starts the display and performs low level interrupt handling and analog device polling. It is intended as a minimal access method to allow flexible access to the graphic data structure for implementing a high level graphics package. All PROCUSTES routines are available from the text library PROCRUST TXTLIB, which resides on the COMMON segment.

### 5.1 THE ROUTINES

The routines that follow are not ALGOL callable. They should be called using the standard M4A call macro (see Meta 4A Assembler User's Guide): R2 points to a parameter list containing the addresses of the parameters and R14 contains the return address. Their parameters are described using ALGOL syntax, but the types refer to standard BUGS types, i.e. INTEGER means a 16 bit integer.

#### 5.1.1 INITIALIZATION AND TERMINATION

##### 5.1.1.1 PROSTART

Function: PROSTART is the routine which initializes PROCUSTES. It must be called before any other routine. PROSTART allocates the buffer, loads the SIMALE, allocates the stacks and starts a Meta 4B program (REFRESHB). REFRESHB executes an INIT subblock, then refreshes the Vector General display, performs first level interrupt handling, and polls analog devices.

Parameters:

- INTEGER VALUE stack\_depth -- this value is the maximum depth of the SIMALE call stack.

- STRING (4) VALUE dimensionality -- an aligned string defining whether the program will use 2D, 3D or both in its display; valid values are "2D", "3D", and "ALL".
- INTEGER VALUE core\_factor -- number in K bytes representing the amount of core PROCRUSTES is allowed to use for images and the directory; a positive number means to use "core\_factor" \* 1024 bytes, a negative number means to use all of core but -"core\_factor" \* 1024 bytes, a zero value means to use all but 4K bytes.
- STRING (8) VALUE simale\_program\_name -- the name of the SIMALE program to be loaded; if X'FF', the standard load, SSGP, will be used. This enables execution of the basic subblocks described in section 3.

#### 5.1.1.2 PROTERM

Function: PROTERM closes out operation of PROCRUSTES and frees up the buffer. PROTERM should always be called before terminating a program.

### 5.1.2 IMAGE MANIPULATION

#### 5.1.2.1 OPENIMAG

Function: OPENIMAG is called to make an image the current image for adding new graphic elements. If the image does not exist in the directory, it will be created. If the image already exists, it will be extended.

#### Parameters:

- INTEGER VALUE image\_name -- the user name of the image to be the open image.
- INTEGER RESULT return\_code  
     return code=0 -- okay  
     return code=2 -- no core for image  
     return code=4 -- no core for expansion of directory

### 5.1.2.2 CALLRDIM

Function: CALLRDIM and its sibling CALLDDIM are intended for use by a routine or routines which manage the CALL/RETURN structure. Since the RETURN function and base pages are different for 2D and 3D objects, PROCUSTES must be advised of the dimensionality of images. This routine returns/sets the dimensionality of the currently open image, and returns its directory offset.

#### Parameters:

- INTEGER VALUE RESULT dim -- dimensionality of image; if 0, then the dimensionality of the image will be returned here; otherwise the image will be set to 2D if "dim" = 2 or 3D if "dim" = 3.
- INTEGER RESULT directory\_offset -- offset from top of directory to directory entry for currently open image.
- INTEGER RESULT return\_code
  - return code=0 -- okay
  - return code=2 -- no open image
  - return code=4 -- dimensionality mismatch (a non-zero value of "dim" was supplied by the caller, but a non-null image of different dimensionality already existed).

### 5.1.2.3 CALLDDIM

Function: This routine returns/sets the dimensionality of the named image and returns its directory offset.

#### Parameters:

- INTEGER VALUE image\_name -- user name of image to be processed
- INTEGER VALUE RESULT dim -- dimensionality of the image; if "dim" = 2, then 2D; if "dim" = 3, then 3D.
- INTEGER RESULT directory\_offset -- offset from top of directory to directory entry for this image.
- INTEGER RESULT return\_code
  - return code=0 -- okay



return code=2 -- image not found  
return code=4 -- dimensionality mismatch  
(same as in Section 5.1.2.2 above)

#### 5.1.2.4 DELIMAGE

Function: DELIMAGE nulls an image. The data representing the image in the buffer is freed, but its directory entry persists. This directory entry will be re-used if the same image name is opened. Directory entries are actually freed by the garbage collection routine (see section 5.1.6 ).

#### Parameters:

- INTEGER VALUE image\_name -- user name of image whose data is to be deleted.

### 5.1.3 SEGMENT MANIPULATION

Segments are a naming and structure feature provided by PROCRUSTES to supplement the image and element naming facilities of the SSGP. A segment is logically a group of primitives within an image, owning a unique name and independently accessible. The segment facility is intended for use in constructing high level data types from groups of primitives. Segments are implemented as blocks within an image.

#### 5.1.3.1 ADDSEGMENT

Function: ADDSEGMENT creates a new segment in the current image.

#### Parameters:

- INTEGER VALUE segment\_name -- the user name of the segment to be created.
- INTEGER RESULT return\_code
  - return code=0 -- okay
  - return code=2 -- no open image
  - return code=4 -- segment already exists
  - return code=6 -- no core for segment

### 5.1.3.2 DELSEGMT

Function: DELSEGMT deletes a segment from the display.

Parameters:

- INTEGER VALUE image\_name -- the user name of the image in which the segment to be deleted resides
- INTEGER VALUE segment\_name -- the user name of the segment to be deleted.
- INTEGER RESULT return\_code
  - return code=0 -- okay, segment deleted.
  - return code=2 -- image not found
  - return code=4 -- segment not found

## 5.1.4 SUBBLOCK MANIPULATION

### 5.1.4.1 ADDSUBLK

Function: ADDSUBLK adds a subblock to the currently open image.

Parameters:

- STRING (10) VALUE type -- name of SIMALE entry point to process the data in this subblock; if "VG" then the subblock contains Vector General orders.
- STRING (\*) VALUE data -- the actual data to put in the subblock.
- INTEGER VALUE data\_length -- length of data in bytes.
- INTEGER RESULT return\_code
  - return code=0 -- okay
  - return code=2 -- no currently open image
  - return code=4 -- no more buffer space
  - return code=6 -- invalid type

#### 5.1.4.2 BLKTYPE

Function: BLKTYPE returns the type of a subblock.

Parameters:

- STRING (\*) VALUE sub\_block -- the first two (or more) halfwords of the subblock whose type is to be returned.
- STRING (10) RESULT type -- same as in section 5.1.4.1 above; if the type is undefined, BLKTYPE will return a blank string.

#### 5.1.4.3 SIMHTRAN

Function: SIMHTRAN is used to determine the initialization halfword for a subblock. Passed the name of the SIMALE entry point to process the subblock, it searches the SIMALE symbol table for the initialization halfword.

Parameters:

At entry R2 points to STRING (10) VALUE type.

At exit R2 contains the initialization halfword or zero if the name was not found.

#### 5.1.5 DYNAMIC SUBBLOCK MANIPULATION

Data within the graphic data structure can be referenced dynamically using symbolic names or dynamos. The following routines provide this facility.

##### 5.1.5.1 ADDDYNAM

Function: ADDDYNAM associates a symbolic name to data within a subblock of an image. The image name, element name, subblock offset, and byte offset can be obtained from the interrupt scanout area.

Parameters:

- INTEGER VALUE dynamo\_name -- name of the dynamo.
- INTEGER VALUE image\_name -- name of the image.
- INTEGER VALUE element\_name -- A NAME subblock containing the element\_name is added to the image before the subblock specified by the dynamo in order to maintain compatibility of names. This is an optional parameter, if the address of this parameter is 0, it is ignored.
- INTEGER VALUE subblock\_offset -- offset within the image to the subblock.
- INTEGER VALUE byte\_offset -- offset within the subblock to the data.
- INTEGER RESULT return\_code
  - return code=0 -- okay
  - return code=2 -- no core for dynamo

5.1.5.2 ADDMARK

Function: ADDMARK associates a symbolic name with the next subblock added to to the currently open image.

Parameters:

- INTEGER VALUE dynamo\_name -- name of the dynamo.
- INTEGER VALUE element\_name -- A NAME subblock containing the element\_name is added to the image. This is an optional parameter, if the address of this parameter is 0, it is ignored.
- INTEGER VALUE byte\_offset -- offset within the subblock to the data.
- INTEGER RESULT return code
  - return code=0 -- okay
  - return code=2 -- no more core
  - return code=4 -- no open image

5.1.5.3 QUERYDYN

Function: Before data can be referenced in the graphic data structure, QUERYDYN must be called to provide the absolute address from the symbolic name. This address

becomes invalid if any change is made to the graphic data structure, so QUERYDYN must be called each time the dynamo data is used.

Parameters:

- INTEGER VALUE dynamo\_name -- dynamo name.
- ADDRESS RESULT pointer -- address of the data.
- INTEGER RESULT length -- length of the data.
- STRING (10) RESULT type -- type of subblock.
- INTEGER RESULT return\_code
  - return code=0 -- okay
  - return code=2 -- dynamo not found
  - return code=4 -- specified dynamo was invalid (not associated with an image, or associated with a null image, or subblock offset > length of image, or byte offset > length of subblock) and has been deleted.

5.1.5.4 DELDYNAM

Function: DELDYNAM deletes a dynamo.

Parameters:

- INTEGER VALUE dynamo\_name -- dynamo name.
- INTEGER RESULT return\_code
  - return code=0 -- okay
  - return code=2 -- dynamo does not exist

5.1.6 GARBAGE COLLECTION -- MRCLEAN

Function: MRCLEAN eliminates all images which are not connected to the graphic data structure. It has two modes of operation, controlled by the parameter passed. The normal mode deletes all images which are not being displayed. In override mode, MRCLEAN will override extents and only eliminate images which are not connected to the root image.

Parameters:

- STRING (10) VALUE mode
  - mode="NOOVERRIDE" -- normal mode

mode="OVERRIDE" -- override mode

## 5.2 INTERRUPTS AND ANALOG DEVICES

The control and status information for the I/O devices attached to the Vector General are defined by the macro CRUSTY, which resides in PROCUST MACLIB on the COMMON segment. Appendix II contains an expansion of CRUSTY. The instruction sequence for establishing addressability to CRUSTY is as follows:

```
LI    Rx,A(PROCRUST)      GET ADDR. OF CRUSTY
USING CRUSTY,Rx           TELL ASSEMBLER
```

### 5.2.1 FUNCTION KEYS

The function key control area consists of a mask with a bit for each function key (PROFKEY1 and PROFKEY2), a one byte data field (PROFKEY), and a one byte mode field (PROFMODE). When function keys are enabled in the masks, the corresponding function key lights are lit. Function key 32 is always enabled. The modes are as follows:

PROFMODE -- current mode

- 0 (PROFIDLE) -- no interrupts will be accepted
- 1 (PROFWAIT) -- function keys enabled
- 2 (PROFKHIT) -- interrupt has been received, data byte (PROFKEY) contains function key number.

### 5.2.2 ALPHANUMERIC KEYBOARD

PROCRUSTES maintains and displays one line of characters for prompt messages and input from the alphanumeric keyboard. The alphanumeric keyboard control area consists of a one byte mode field (PROAMODE) and a one byte length (PROANKSZ) that is the total length of the combined prompt/input line, i.e. the length of the prompt line plus the maximum length of the input buffer.

PROAMODE -- current mode

- 1 (PROASTBT) -- start input mode

#### 4.1.2 3D FORMAT

|     |       |       |       |             |
|-----|-------|-------|-------|-------------|
| +0  | $X^0$ | $Y^0$ | $Z^0$ | Start point |
| +6  | $X^1$ | $Y^1$ | $Z^1$ | Next point  |
| +12 | $X^2$ | $Y^2$ | $Z^2$ | ...         |
|     | .     |       |       |             |
|     | .     |       |       |             |
|     | .     |       |       |             |

#### 4.2 LINES (ENTRY POINTS: LINES2D, LINES3D)

This subblock allows the drawing of disjoint lines. The endpoints of the lines are specified. The line type is determined by the current attributes.

#### 4.2.1 2D FORMAT

|     |       |       |             |
|-----|-------|-------|-------------|
| +0  | $X^0$ | $Y^0$ | Start point |
| +4  | $X^1$ | $Y^1$ | End point   |
| +8  | $X^0$ | $Y^0$ | Start point |
| +12 | $X^1$ | $Y^1$ | End point   |
|     | .     |       |             |
|     | .     |       |             |
|     | .     |       |             |

- 2 (PROTTAB) -- track tablet
- 3 (PROTHERE) -- display cursor at current position; in this mode, the user can maintain direct control over the cursor position by specifying the cursor position in PROCURXY.

PROTMODE -- tracking mode

- 0 (PROTNONE) -- just display cursor
- 1 (DRAWING) -- this mode may be entered explicitly by the user or implicitly by setting "DRAWWAIT". If the processing is successful, it will enter mode 4 ("PICKED"), the interrupt scanout area will contain correct information about the viewport which encloses the cursor and the user co-ordinates of the cursor within that viewport will be placed in PROCUSER; it will enter mode 5 ("DRAWWAIT").
- 2 (PICKWAIT) -- automatic picking mode; whenever the tracking trigger is pressed, an attempt will be made to pick something.
- 3 (PICKING) -- this mode may be entered explicitly by the user or implicitly by setting "PICKWAIT"; PROCURST will attempt a correlation hit. If successful it will enter mode 4 ("PICKED"), else it will enter mode 2 ("PICKWAIT").
- 4 (PICKED) -- a successful correlation or inversion has been made; the interrupt scanout area contains the pertinent information.
- 5 (DRAWWAIT) -- automatic drawing mode; whenever the tracking trigger is activated, an attempt will be made to return the current cursor position in user coordinates.

#### 5.2.4 ANALOG INPUT

Data from analog input devices is contained in the analog input scanout area. The first word is a filtering value for all devices. The filtering formula is:

$$A(t) = A(t-1) + A(t)/2**f - A(t-1)/2**f$$

where A(t) and A(t-1) are the analog values at times t and t-1 and f is the filtering value (default is 3). The next three words are the tablet x,y,z. The tablet z indicates if the tablet stylus is out of range, in range, or touching. The next three words are the joystick x,y,z. The last ten words are the dial values.






## 6 APPENDIX I -- GRAPHIC DATA STRUCTURE FORMATS

The graphic data structure described in the following sections is particular to the operation of PROCRUSTES and is in no way intended to be the definitive structure.

### 6.1 THE DIRECTORY

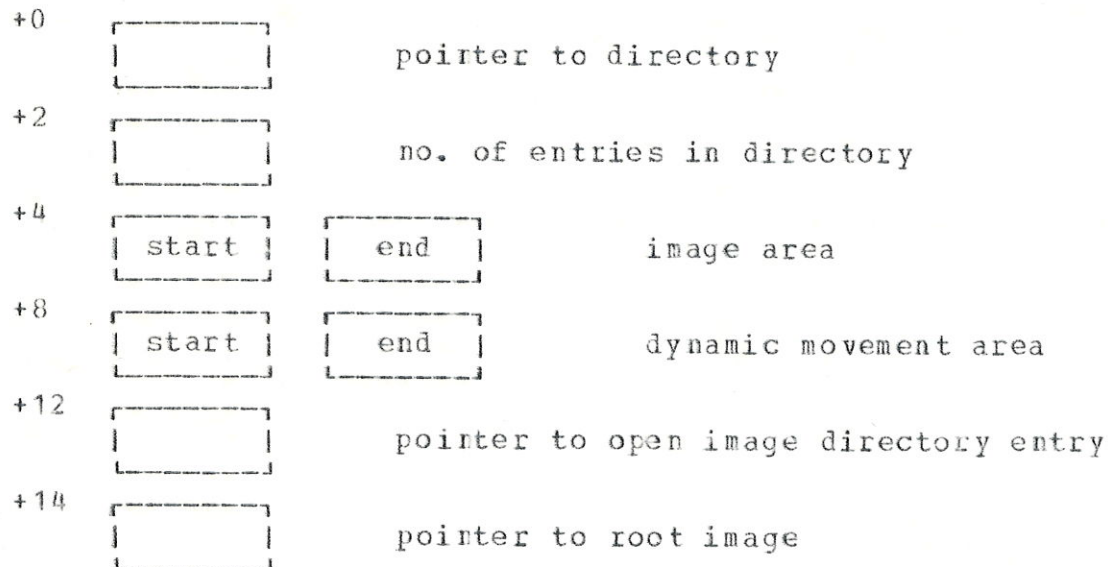
The directory maintained by PROCRUSTES consists of six byte entries. The directory is expanded as needed in increments of 8 entries (48 bytes). Two names are reserved for unused entries and for the root image, X'8000' and X'8001' respectively.

The format of each entry is as follows:

|    |   |                     |
|----|---|---------------------|
| +0 |  | pointer to image    |
| +2 |  | user name for image |
| +4 |  | flags               |

## 6.2 THE BUFFER

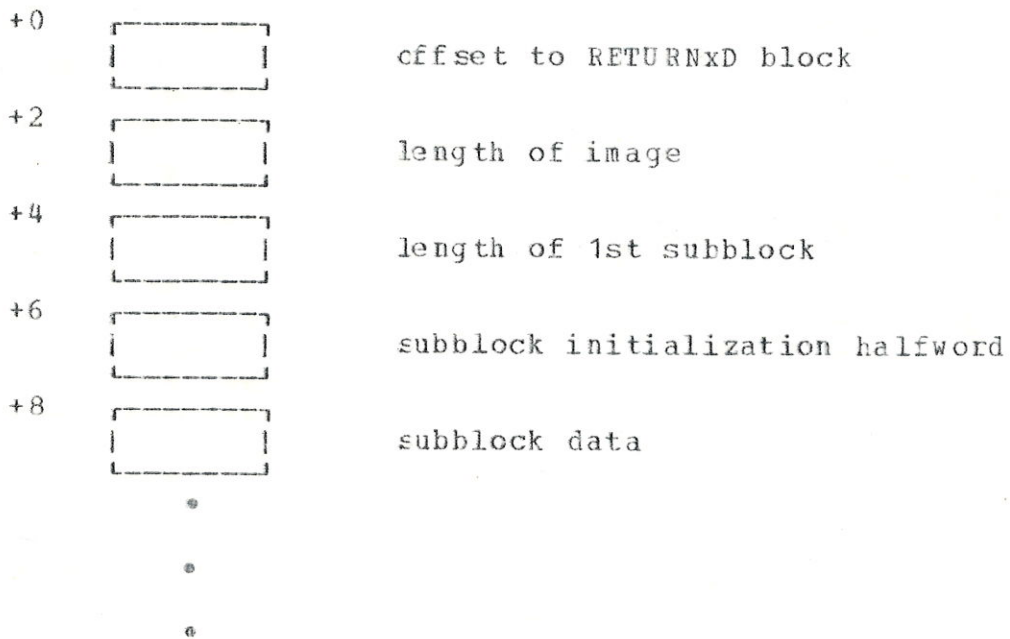
The buffer area obtained by PROSTART consists of the buffer header, the SIMALE call stack, the interrupt scanout area, the directory, the image area, and the dynamic movement (in that order). All free space resides between the image area and the dynamic movement area. The currently open image is always in the last position, next to the free space in the buffer. The format of the buffer header is as follows:



| 6.3 IMAGE FORMAT

| Each image consists of an ETC ring. If an image is  
| unsegmented, the ring is one block chained to a return  
| sub-block in the common area. If the image is segmented, it  
| will consist of multiple contiguous blocks; the first block  
| will contain the length in its second halfword, and subsequent  
| blocks will correspond to segments, with the segment name in  
| the second halfword.

| 6.3.1 UNSEGMENTED FORMAT

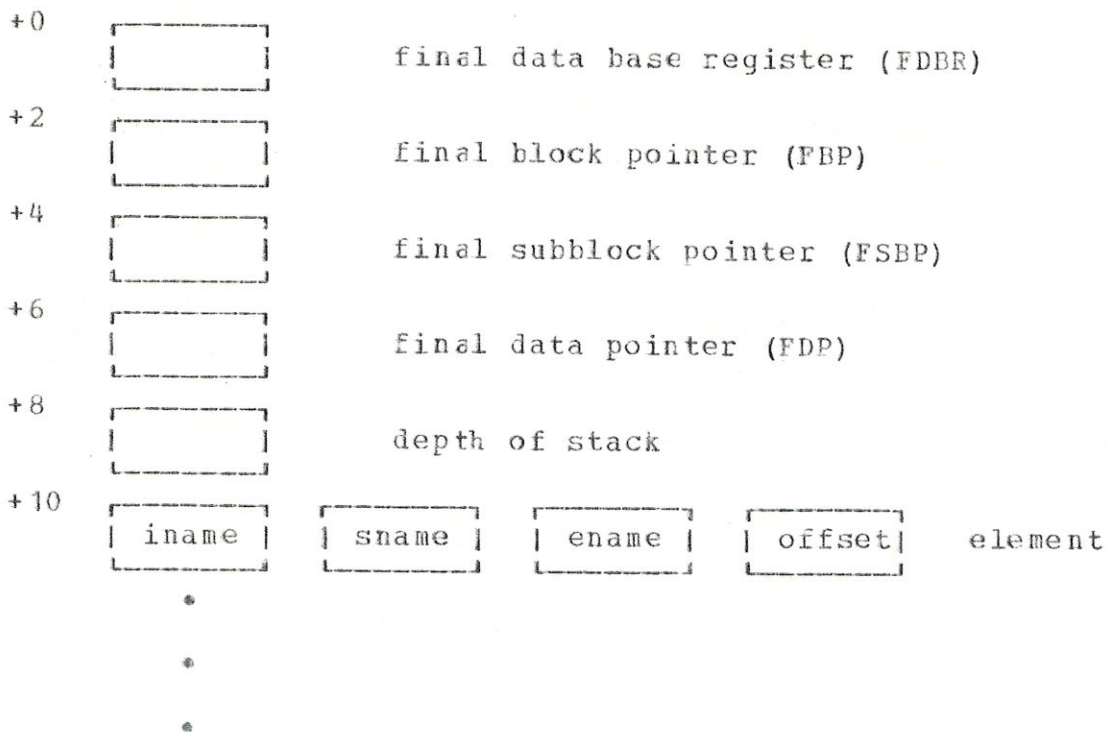


### 6.3.2 SEGMENTED FORMAT

|        |                                |   |
|--------|--------------------------------|---|
| +0     | <input type="text"/>           | offset to first segment                     |
| +2     | <input type="text"/>           | length of image                             |
| +4     | <input type="text" value="0"/> |   |
| +6     | <input type="text"/>           | offset to next segment or RETURNxD<br>block |
| +8     | <input type="text"/>           | segment name                                |
| +10    | <input type="text"/>           | length of 1st subblock                      |
| +12    | <input type="text"/>           | subblock initialization halfword            |
| +14    | <input type="text"/>           | subblock data                               |
|        | •                              |   |
|        | •                              |   |
|        | •                              |   |
| + n    | <input type="text"/>           | offset to next segment or RETURNxD<br>block |
| + (n+2 | <input type="text"/>           | segment name                                |
| + n+4  | <input type="text"/>           | length of 1st subblock                      |
| + n+6  | <input type="text"/>           | subblock initialization halfword            |
|        | •                              |   |
|        | •                              |   |
|        | •                              |   |

#### 6.4 INTERRUPT SCANOUT AREA

The interrupt scanout area maintained by PROCRUSTES is filled in whenever an inversion or correlation completes successfully. Its format is as follows:



The stack elements are ordered with the lowest level first. Each element contains the image name (iname), segment name (sname), element name (ename) and subblock offset (offset).

## 6.5 DYNAMO FORMAT

Each dynamo consists of the following information:

|    |                      |                 |
|----|----------------------|-----------------|
| +0 | <input type="text"/> | dynamo name     |
| +2 | <input type="text"/> | image name      |
| +4 | <input type="text"/> | element name    |
| +6 | <input type="text"/> | subblock offset |
| +8 | <input type="text"/> | byte offset     |

## 7 APPENDIX II -- CRUSTY DUMMY SECTION

### 7.1 PREFIX

```
CRUSTY      DSECT ,           DUMMY SECTION
*****
*****      FIRST WE HAVE THE FLAGS
*****
READY       DS 0XL128 .       IF ON, BUFFER IS READY
TERMINAT    DS 0XL64 .       IF ON, TIME TO TERMINATE
GRABAGE     DS 0XL32 .       GARBAGE COLLECTION IN PROGRESS
GRABAGEC    DS 0XL16 .       CONTINUE GARBAGE COLLECTION
GRABAGEO    DS 0XL8 .        OVERRIDE EXTENTS
            DC X'00'

*****
*****      NOW THE POINTER TO THE CURRENT BUFFER
*****
PROBHPTTR   DC A(*-*) .      --> CURRENT BUFFER
*****
*****      NOW SOME COMMONLY USED BLOCKS
*****
PRONIT      DC A(0,0,4) .     INITIALIZATION DATA AREA
PRONITH     DC H'0'
PROBRET2    DC A(0,0,4) .     RETURN BLOCK (2-D)
PROBRIH2    DC H'0' .        INITIALIZATION HALFWORD
PROBRET3    DC A(0,0,4) .     RETURN BLOCK (3-D)
PROBRIH3    DC A(*-*) .     INITIALIZATION HALFWORD
PROCURSE    DC A(0,0,4) .     CURSOR BLOCK
PROCURIH    DC H'0' .        INITIALIZATION HALFWORD
            DC H'0' .        AND LENGTH OF NEXT

*****
*****      SOME IMPORTANT VALUES TO REMEMBER
*****
PRODYNAM    DC 2A(*-*) .     ADDR. AND LEN. OF DYNAMIC AREA
PROSTAKB    DC A(*-*) .     ETC STACK BASE POINTER
PROSTAKE    DC H'0' .       LENGTH OF STACK ENTRY
PROSTAKM    DC H'0' .       MAXIMUM STACK OFFSET
```

## 7.2 FUNCTION KEY CONTROL

|          |          |                                 |
|----------|----------|---------------------------------|
| PROKEY1  | DC 2H'0' | UPPER FUNCTION KEY MASK         |
| PROKEY2  | DC 2H'0' | LOWER FUNCTION KEY MASK         |
| PROFKEY  | DC X'00' | FUNCTION KEY NO. HIT            |
| PROFMODE | DC X'00' | FUNCTION KEY MODE               |
| PROFIDLE | EQU 0    | -- NO FUNCTION KEY HITS ALLOWED |
| PROFWAIT | EQU 1    | -- WAITING FOR FUNCTION KEY HIT |
| PROFKHIT | EQU 2    | -- FUNCTION KEY HAS BEEN HIT    |

## 7.3 CURSOR CONTROL

|          |            |   |
|----------|------------|---|
| PROCURXY | DC 2H'0'   | X,Y POSITION OF CURSOR                            |
| PROCSAL  | DC 2H'256' | CORRELATE SCALE FACTOR X AND Y                    |
| PROCUSER | DC 2H'0'   | POSITION OF CURSOR IN VIEWPORT                    |
| PROTDEV  | DC X'0'    | TRACKING DEVICE                                   |
| PROTNONE | EQU 0      | -- TRACK NOTHING                                  |
| PROTJOY  | EQU 1      | -- TRACK JOY STICK                                |
| PROTTAB  | EQU 2      | -- TRACK TABLET                                   |
| PROTHERE | EQU 3      | -- TRACK THIS LOCATION                            |
| PROTMODE | DC X'00'   | TRACKING MODE                                     |
| DRAWING  | EQU 1      | -- DRAWING MODE                                   |
| PICKWAIT | EQU 2      | -- PICKING WAIT                                   |
| PICKING  | EQU 3      | -- PICKING  |
| PICKED   | EQU 4      | -- SOMETHING HAS BEEN PICKED                      |
| DRAWWAIT | EQU 5      | WAITING FOR DRAW MODE0-                           |
| PROISCAN | DC A(*-*)  | --> INT. SCANOUT AREA (ISCAN)                     |
| PROTRIGR | DC X'FF'   | NO. OF FUNCTION KEY TO ACT AS<br>TRACKING TRIGGER |

## 7.4 ANALOG DEVICES SCANOUT AREA

|          |           |                         |
|----------|-----------|-------------------------|
| PROFILTR | DC H'3'   | FILTERING VALUE         |
| PROTABX  | DC H'0'   | TABLET X POSITION       |
| PROTABY  | DC H'0'   | TABLET Y POSITION       |
| PROTABZ  | DC H'0'   | TABLET Z POSITION       |
| *****    |           | -- 0 MEANS OUT OF RANGE |
| *****    |           | -- 1 MEANS IN RANGE     |
| *****    |           | -- 2 MEANS TOUCHING     |
| PROJOYX  | DC H'0'   | JOYSTICK X POSITION     |
| PROJOYY  | DC H'0'   | JOYSTICK Y POSITION     |
| PROJOYZ  | DC H'0'   | JOYSTICK Z POSITION     |
| PRODIALS | DC 10H'0' | AND THE DIAL VALUES     |



7.5 ALPHA-NUMERIC KEYBOARD CONTROL

```
PROAMODE DC X'00' .          CURRENT MODE
PROASTRT EQU 1 .            START INPUT MODE
PROAINP EQU 2 .             INPUT MODE
PROADONE EQU 3 .            INPUT AVAILABLE
PROANKSZ DC AL1(80) .       NUMBER OF CHARACTERS TO INPUT
PROAVIEW DC X'000000007FFF7FFF' . VIEWPORT FOR CHARS.
PROANKX DC X'8000' .        X POSITION FOR BUFFER
PROANKY DC X'0000' .        Y POSITION FOR BUFFER
PROANBUF DC A(*-*) .        --> INPUT BUFFER (ASCII)
PROANLEN DC H'0' .          LENGTH OF INPUT
PROAPBUF DC A(*-*) .        --> PROMPT MESSAGE (EBCDIC)
PROAPLEN DC H'0' .          THE LENGTH OF SAME
*****
***** TRANSLATE TABLES
*****
TRANATOE DS 256X .          TRANSLATE TABLE ASCII
TRANETOA DS 256X .          TRANSLATE TABLE EBCDIC
```