A DATABASE SYSTEM TO SUPPORT IMAGE ALGORITHM EVALUATION

FINAL TECHNICAL REPORT

Y. Edmund Lien, Principal Investigagor

The University of Kansas
Department of Computer Science
Lawrence, Kansas 66045

This research was sponsored by the National Aeronautics and Space Administration through Research Grant No. NSG-8046.

FINAL TECHNICAL REPORT

Research Grant Title:	A Database System to Support Image Algorithm Evaluation
Principal Investigator:	Dr. Y. Edmund Lien Associate Professor Department of Computer Science
Grant Period:	November 15, 1976 - December 31, 1977
Grantee Institution:	University of Kansas Lawrence, Kansas 66045
Grant Number:	NASA Research Grant No. NSG-8046

This final report

(1) describes the design and implementation of an image database system IMDB

(2) provided sufficient information for the maintenance and upgrade of the IMDB system

(3) includes a user manual describing use of the IMDB system

(4) provides a list of publications issued during the course of the research.

TABLE OF CONTENTS

—ACKNOWLEDGEMENT—

REPORTS:

I. An Interactive Query Language for an Image Database (by Y. E. Lien and R. Schroff)

II. Implementation of the IMDB System

1. Overview (by Y. E. Lien)
2. Implementation of the Query Module (by Y. E. Lien)
3. Implementation of the Device Module (by S. K. Harris)
4. Implementation of the File Module (by R. Law)
5. Implementation of the Manipulation Module (by R. Law)

APPENDICIES: (by Y. E. Lien)

I. IMDB User Manual
II. Independent Utility Routines
III. Permanent File Header Format
IV. Bibliography of Publications
V. Documentation Package

ACKNOWLEDGEMENT

We would like to thank Dr. Robert R. Jayroe, Jr. of Marshall Space Flight Center for his many helps throughout the period of this research. Without him, this research would have been impossible to carry out to its completion. Mr. Malcomb E. Gillis of Computer Science Corporation, Huntsville has also provided tremendous assistance in the programming phase. He taught us how to use RSX-11D commands, shared with us his experiences in graphics software, wrote the package Z which later became part of the Device Module and many times managed to save our files of thousands of source lines after several severe disk crashes.

Many students at the Department of Computer Science, University of Kansas participated in the early phase of the project. The efforts of Al Poston, Sharon White and Claudia Dale are greatly appreciated.

We also would like to thank Susan Walker, Portia Kibble, and John Ying (of Bell Lab, Murray Hill) for their excellent typing of this report.

REPORT I

AN INTERACTIVE QUERY LANGUAGE FOR AN IMAGE DATABASE

ABSTRACT

Images, such as those created through satellite remote sensing or photography, can be integrated into a central database to permit application-oriented interactive data access and manipulation. Instead of each user building the image processing tools for his own applications, an image database offers to all its users a set of general purpose image data operations. These operations can be used to analyze images, to extract their informational content, to compare images, to overlay images, etc. Unique and essential to image database systems is the capability of conversational system-user interaction through both conventional terminals and graphic display devices. Presented in this paper is the design of an interactive image database system IMDB, which allows the user to create, retrieve, store, display, and manipulate images through the facility of a high-level, interactive image query (IQ) language. The query language IQ permits the user to define false color functions, pixel value transformations, overlay functions, zoom functions, and windows. The user manipulates the images through generic functions. The user can direct images to display devices for visual and qualitative analysis. Image histograms and pixel value distributions can also be computed to obtain a quantitative analysis of images.

REPORT I

AN INTERACTIVE QUERY LANGUAGE FOR AN IMAGE DATABASE

1. INTRODUCTION

This paper addresses itself to an image database system IMDB. The system supports the processing of discrete, geographically-associated images, such as those produced by the LANDSAT satellites. Although primarily intended for use with this type of image, the system design is both general and flexible, permitting its use in other image processing applications. Central to the system design is an interactive user-oriented query language. Here the user is offered a convenient facility by which images may be created, stored, retrieved, manipulated and displayed.

1.1 Database Images

In 1972 and 1975 respectively, satellites LANDSAT 1 and LANDSAT 2 were launched by the National Aeronautics and Space Administration (NASA). The LANDSAT program, formerly Earth Resources Technology Satellite (ERTS), has provided a capability for repeated surveys and assessments of earth conditions and resources.

Each satellite has a multispectral scanner which permits simultaneous imaging in four channels. The digitized images from these channels are sent to the ground where computers are used to reconstruct the images. A four-channel LANDSAT image contains 7,581,000 picture elements (pixels) and for each channel the data value of each pixel is represented as an integer between 0 and 255. This value indicates the intensity of light reflection from an area of 79 x 79 meters.

There is an urgent need to provide a database system to support LANDSAT image processing and application. After years of research and development, the processing of LANDSAT images has not yet provided production facilities capable of extracting and organizing information

contained in the images. Only a small fraction of the images have been fully processed and classified, leaving an enormous fund of data still unavailable for public use. The situation is compounded by the LANDSAT-C project (scheduled for a late 1977 or early 1978 launch) and the LANDSAT-D follow-up project.

An obvious difficulty lies in the extremely large size of the image data. The present LANDSAT will produce a new set of images covering the entire earth every nine days. It is evident, from the sheer magnitude of the data alone, that the processing of LANDSAT images requires a formidable amount of computer resources.

A second type of image is produced from hand-drawn ground-truth maps. These maps represent the data gathered by scientists, for instance, data regarding land use, soil type, slope, mineral resources and energy resources. Quite often satellite images are overlaid or compared with ground-truth maps to obtain meaningful interpretations of the images.

The IMDB design does not address the data magnitude problem of LANDSAT images, nor does it include a direct capability for classifying images and organizing classification data for later retrieval. IMDB is primarily intended for image manipulation and statistical analysis. It can, for example, be used as a tool to assist in the image classification process. It can also be used to evaluate the performance of different image processing algorithms such as image registration and compression. Other applications might include the production of map data, the studying of geographic and geological features, the assessment of crop inventories, etc.

1.2 IMDB System

Within the framework of an image query language (IQ language), the IMDB system incorporates the facilities requisite for the processing of satellite and ground-truth images. Although IMDB is a response to the specific computational requirements associated with LANDSAT images, the system can and should be viewed as a generalized image database system. Let us informally summarize the facilities of such a system. These include

- A database management system capable of maintaining image data.
- Image manipulation facilities which permit images to be edited, colored, transformed, superimposed, compressed, and expanded.
- Display facilities permitting the visual interpretation of image data on one or several output devices.
- Statistical facilities allowing an analysis of an image's composition or spectral distribution.

Although the implementation of an image database poses difficulties not normally encountered in traditional databases such as IMS [6] and IDS [7], the structuring techniques employed by IMDB do not vary significantly from conventional methods. Consequently, we shall not discuss at length the physical characteristics of the IMDB database. Rather we shall focus our attention on the user's view of IMDB as seen through the facilities offered by the IQ language.

The design of the IQ language reflects an adherence to certain general principles:

- The language must combine the display, manipulative, statistical and management facilities within a continuous framework, i.e., a single program.
- The language must view the database from a logical standpoint which is conceptually removed from physical considerations.
- The language must be highly user oriented, employing where feasible, English language prompting.

In presenting the IQ language we have chosen to emphasize conceptual and operational aspects at the expense of syntactic considerations. Certain query statements have been simplified while prompting has been discarded in favor of a less spacious and more direct presentation. A grammar of the IQ language is included in the Appendix.

1.3 Hardware Configuration

IMDB is implemented at the Data Technology Testbed (DTTB) of the Data System Laboratory, NASA Marshall Space Flight Center. The central processing unit is a PDP-11/45 with 128K bytes of memory and 600 million bytes of disk storage. Conventional input-output peripherals such as line printer and magnetic tape are available. The PDP system runs under the standard RSX-11D operating system.

Image display functions of the IMDB system require input-output peripherals which provide graphic and color capabilities. The color graphic device is a Ramtek GX-100B digital TV system which is an on-line device of the PDP-11/45. This particular system has two 19-inch color monitors and one 17-inch black and white monitor, three keyboards and one trackball. The trackball is used for graphic input. Currently there is sufficient refresh memory for two 256 by 256 pixel color monitors and each pixel can be assigned any of the eight colors. Contents of the

refresh memory can also be read, which provides a convenient way to construct composite images interactively. With additional memory each Ramtek display can be expanded to a maximum of 512 by 512 pixels with a maximum of 4096 colors. Graphic capability is also available with two Tektronix 4014-1 storage tube terminals. The screen contains 4096 by 3120 addressable points. Graphic input capability on each Tektronix terminal is provided through a thumb-wheel controlled cursor.

Hardcopy graphic output can be obtained through the hardcopy devices attached to the Ramtek or Tektronix terminals or directed to a Varian 4115 electrostatic printer/plotter. Dicomed D47 image recorder provides the capability for film recording. Each film is formatted as a matrix of 4096 by 4096 points. Each point in the matrix can be assigned an exposure value in the range of 0 through 255.

Although IMDB is implemented on the hardware described above, its logical structure is designed to be flexible as to the particular computer configuration.

1.4 Related Work

Database concepts such as data integration and data independence [1] have not been actively pursued in the field of image and picture processing until only recently. Image and picture processing encompasses a wide range of distinct application areas; the need of database technology in these areas are equally diversified. For example, McKeown and Reddy described a multi-sensor image database system MIDAS designed to perform knowledge acquisition, error analysis, and algorithm evaluation [2]. Chang et al presented a relational database system for managing pictorial and alphanumeric information [4]. Zabrist presented an image-based information system designed especially to manage spatially-referenced

data [3]. Kunni, Weyl, and Tenenbaum proposed a relational database schema for colored picture description [5].

2. BASIC ELEMENTS

The database facilitates the storage and retrieval of images and windows, each of which is addressable by means of a unique name. In addition to these items, the database is also used for the storage of generic functions. These will be discussed in separate sections.

2.1 Images

An image consists of picture elements, or pixels. Each pixel is a one-byte integer associated with a precise geographic area and having a range of values from 0 to 255. The area represented by a pixel is dependent upon the scale of the pixel. In terms of z pixels/km, the pixel represents a square of area $1/z^2$ km^2. The value of a pixel is the image representation of its area; if a pixel results from a photographic measure of grayness, then a pixel value corresponds to a gray-level.

An image is a two-dimensional pixel matrix. Each row and column consists of contiguous non-overlapping pixels which respectively follow the geographic lines of latitude and longitude. The dimensions of an image are a measure of the geographic area represented by its component pixels.

Within the database the unit of storage is the row. This is referred to as a scan line. All images, before being entered into the database, are assumed to have been geometrically corrected and geographically registered.

2.2 Windows

A window is the polygon described by a circuitous ordered set of points, each point located in terms of longitude/latitude. Each pair of

consecutive points determines a vector, and according to an IMDB convention, the vectors are required to form a non-intersecting clockwise closure. Each window is further assumed to be either an inclosure or an exclosure.

The restriction that the window be non-intersecting corresponds to the normal view of a polygon. (It also avoids an unwieldy interpretive problem.) The restriction does not, however, disallow colinear vectors on "richochets". In this context an intersection is interpreted as an actual crossing of two vector chains.

In the construction of the union or intersection of two windows, a determination of the window's direction is essential. Since there exists a relatively simple algorithm for determining direction, the clockwise requirement is seen as a matter of conceptual and computational convenience.

3. FUNCTIONS

Functions provide the user with methods by which images and windows may be combined to form new images and windows. The IQ language supports nine different functions. These can be conveniently classified in two categories, built-in and generic. In this section we shall primarily concentrate on the operational aspects of these functions, leaving to succeeding sections the bulk of the discussion of syntax.

All functions require a specific set of image/window parameters and return, either an image or a window. The masking function, for example, requires an image and a window as parameters. This function returns a masked image.

The query language is constructed such that whenever an image is an appropriate function parameter, an image expression is an equally appropriate parameter. This equivalence holds true for windows as well. In the case of the masking function, since the function returns an image,

it is the expressional equivalent of that image. In the succeeding sections we will use the following notation to describe the relationship between a function, its parameters, and its equivalence:

MASK(<window expression>,<image expression>) → image

3.1 Built-in Functions

There are five built-in functions: JOIN, MASK, CLIP, UNION, and INTERSECTION. These functions are tailored for immediate use much in the same fashion as the built-in functions of PL/I. We shall discuss each of these individually.

A. JOIN(<image expression>,<image expression>) → image.

This function "pastes" two images together to form a new image, according to their geographic coordinates. The dimensions of the new image are those which are minimally sufficient to contain the areas of the originals. The first of the original images is defined to be the dominant image; this image takes precedence when the two images overlap. The following examples show the joining of images A and B (A is dominant). "0" denotes an area that is not contained in either image; all pixels in these areas have zero value.

JOIN is commonly used to create geographically large images by placing component images side by side (or top to bottom).

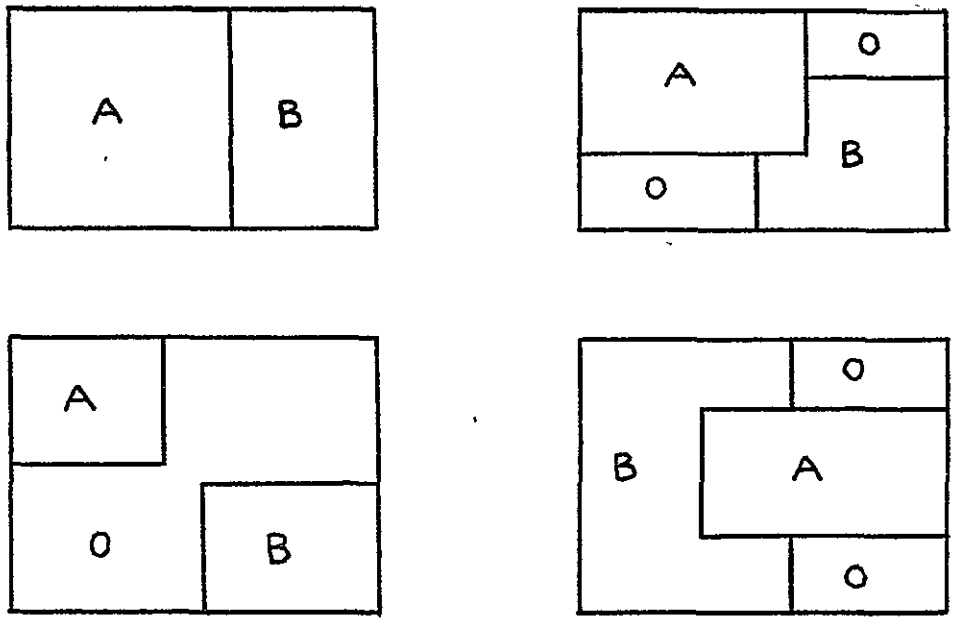


FIGURE 1. Examples of JOIN function.

B. MASK(<window expression>,<image expression>) → image.

This function masks a window onto an image to form a new image. If the window is an inclosure, pixels interior to the window will retain the values of the original image while exterior pixels will be zeroed. Exclosures function in the opposite manner: Exterior pixels are transfered and interior pixels are zeroed. In either case, the resultant image has the same dimension as the original image.

The masking algorithm makes use of the window's topological properties. A pixel is contained within a window if the a ray eminating from the pixel crosses the window boundaries an odd number of times. In terms of a scan line this amounts to an identification of the points where the window intersects the scan line (or its infinite extension), and a determination of interior pixel intervals through a pairing of intersection points.

C. CLIP(<window expression>,<image expression>) → image.

Although clipping is computationally a single operation, it can be conveniently viewed as a two step process. The window is first masked onto the image; thereafter, the image is dimensionally reduced by discarding those outermost rows and columns which do not intersect the window. The resultant image has dimensions which are minimally sufficient to contain the window. The solid lines in Figure 2 show an image and an inclosure window; the dashed lines are the boundaries of the image resulting from CLIP.

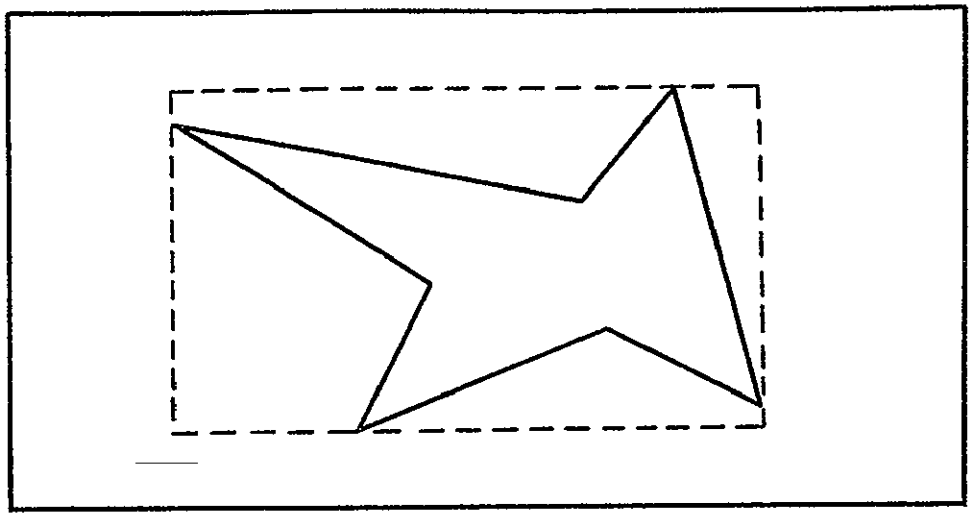


FIGURE 2. Example of CLIP function.

The CLIP function permits the user to reduce storage requirements by discarding the frame that encompasses the relevant data. CLIP is also commonly used to produce a sub-image by masking with a rectangular window.

D. UNION(<window expression>,<window expression>) → window.

INTERSECTION(<window expression>,<window expression>) → window.

These functions permit the user to regard windows as Boolean operands. Included within this framework is the notion of an inclosure and an exclosure: if W1 is an inclosure then ~W1 is an exclosure; likewise, if W1 is an exclosure than ~W1 is an inclosure.

The result of a Boolean operation is always a single window. In some situations this can be accomplished only through the use of artificial colinear vectors. (See Figures 3b and 3c.) Since these vectors do not alter the area defined by a window, they have no effect on the functional usage of the window.

Figure 3a below shows the relative positions of windows A and B, both of which are inclosures. The resultant windows shown in Figures 3b and 3c are inclosures, while the window of Figure 3d is an exclosure. The window in Figure 3b can be described as a sequence of vectors (a,b), (b,c), (c,d), (d,e), (e,f), and (f,a). The arrows in the figure show the clockwise direction of the window. The windows in Figures 3c and 3d can be similarly described.

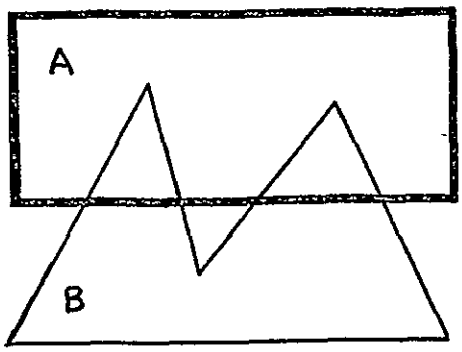


Figure 3a. Windows A and B.

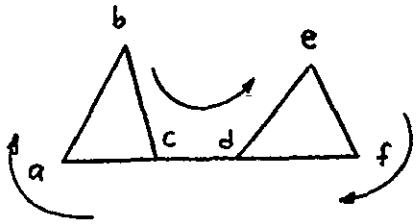


Figure 3b. INTERSECTION (A,B).

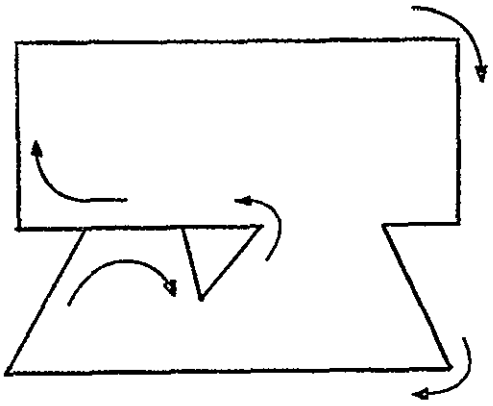


Figure 3c. UNION (A,B).

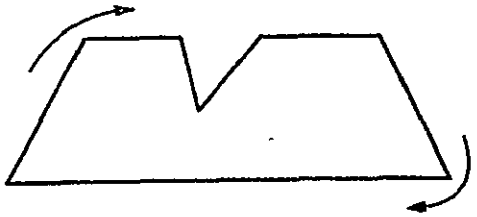


Figure 3d. UNION(A,~B).

3.2 Generic Functions

One of the four generic functions is TRANSFORM. We shall use this function to illustrate the general properties of generic functions. As will be seen in the next subsection, the transformation process creates a new image by a mapping of the pixel values of a given image. For an image I and a mapping M, the new image might be represented as TRANSFORM(I,M). But suppose that M is fixed through the assignment M = a. This defines a new function, say TR, where TR(I) = TRANSFORM(I,M) $\Big|_{M = a}$.

It is evident that an assignment of different values to M determines a family of transformation functions, each function operating under its assigned mapping. These we call the generic functions of TRANSFORM.

There are four generic functions: TRANSFORM, COLOR, OVERLAY, and ZOOM. None of these is useable in the operational sense; instead, these are generalized functions providing a framework within which the user may construct specific operational functions. In defining a function the user must give the function a name. This allows the function to be stored in the database, ready to be invoked by a query statement. In this manner the user is able to build a library of generic functions, each function tailored to a special use.

A. <TRANSFORM-name>(<image expression>) → image.

TRANSFORM allows the user to define a method by which pixel values of a given image can be linearly transformed to create a new image. The transformation consists of a set of subtransformations and a default transformation.

A subtransformation is written (a,b) → (c,d). This specifies that all pixel values in the interval (a,b) are to be linearly mapped to the interval (c,d). For $x_1 \in (a,b)$ and $x_2 \in (c,d)$ this mapping is defined by

$$x_2 = \begin{cases} c + \frac{d - c}{b - a} \ (x_1 - a) & a < b \\ c & a = b \\ \text{ERROR} & a > b \end{cases}$$

Since the general equation $x_2 = mx_1 + k$, $m \geq 0$, is equivalent to the mapping (a,b) → (ma + k, mb + k), it can be seen that this mapping permits all non-inverted linear transformations. An inversion (a negative in the photographic sense) may be achieved by (a,b) → (mb + k, ma + k) for m > 0.

The default transformation is a sort of catch-all. It is used to define the pixel value which results from intervals that are not explicitly defined.

Let us take an example. Suppose that an image has pixel values in the range 0 - 127. The transformation

(0,63) → (0,3)

(96,100) → (4,4)

DEFAULT: 5

will produce the following mapping:

$$\begin{bmatrix} 7 & 60 & 83 \\ 38 & 97 & 18 \\ 120 & 127 & 12 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 3 & 5 \\ 2 & 4 & 1 \\ 5 & 5 & 0 \end{bmatrix}$$

B. <COLOR-name>(<image-expression>) → image.

Color display devices accept streams of integers which are translated into colors. Prior to displaying an image, the user must transform the image to the integer values appropriate to the display device and the desired

color picture. This poses two difficulties: First, the integer/color equivalence varies among devices of different types. Second, the use of a color equivalence is by no means accomodating to the user.

The color function is a response to these difficulties. Here the user is permitted to specify a set of interval-to-color transformations. He might, for example, specify the following:

(0,20) → RED

(50,70) → BLUE

(100,200) → YELLOW

DEFAULT: BLACK

This can be regarded as the first phase of a transformation. The second phase takes place when the user specifies a display device. At this point the user-specified colors can be translated to their device-dependent color equivalents.

C. <OVERLAY-name>(<image expression>,<image expression>) → image. The overlaying of two images produces an image which has the dimensions of the overlapping area. The overlapping area is determined by the geographic coordinates. The relation between the original images and the resultant image is illustrated in Figure 4. Here we see the overlaying of two images, A and B. The shaded area shows the result.

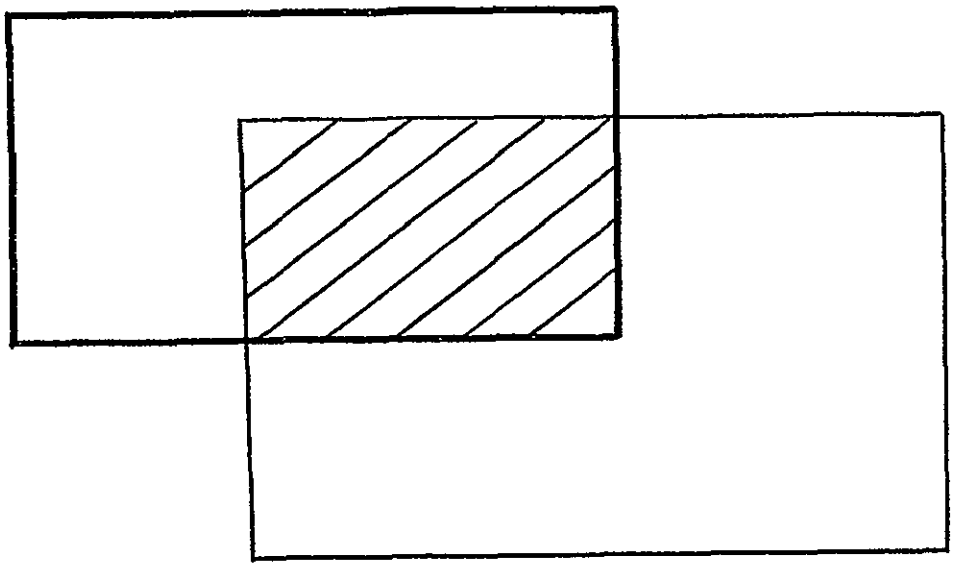


FIGURE 4. Example of OVERLAY function.

Based upon pixel values, the OVERLAY function maps the cartesian product of the two original images onto the resultant image. The mapping is supported by ten binary operators: ADD, SUB, DIV, MULT, MAX, MIN, AVG, AND, OR, and EXOR - and is defined by a transformation set wherein these ten can be freely mixed. The first seven operations refer to the common arithmetic operations between two pixel values: addition, subtraction, integer division, multiplication, maximum, minimum, and averaging. The last three operators refer to the bit-wise logical operations between pixel values: AND, OR, and EXCLUSIVE OR.

An OVERLAY function consists of a set of transformation rules. Each rule is associated with a range of pixel values and an operator. In addition to, or in lieu of transformation rules of this type, the user may specify a default pixel value or a default operator. This

default option defines the transformation that is to take place for otherwise undefined interval products. By using the default pixel value the user specifies that all undefined interval products are to be mapped to that value; the default operator specifies the operation that is to be performed for undefined intervals.

The following example shows a mapping (0,255) x (0,255) → (0,255):

MAX: (0,127) x (0,127)

AVG: (0,127) x (128,255)

DEFAULT: 0

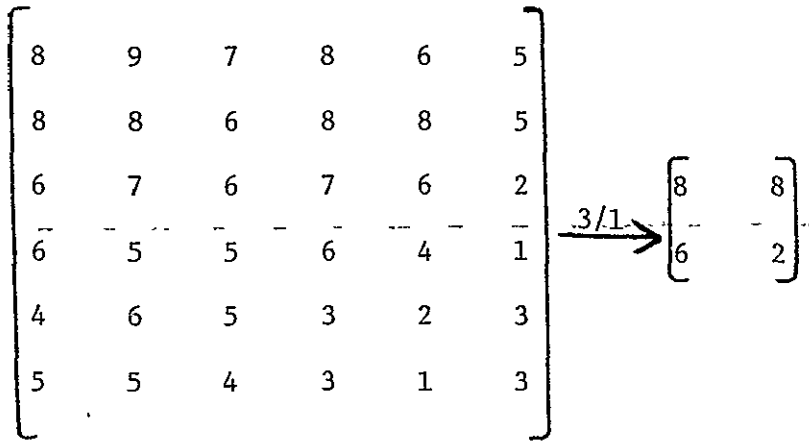
Maximum or averaging will be used for a pixel value pair (x,y) if $0 \leq x \leq 127$ and $0 \leq y \leq 255$. Otherwise, the new pixel value will be zeroed.

D. <ZOOM-name>(<image expression>).

This is a relatively simple function which allows the expansion or compression of an image. A one-to-three zoom expands a 15 x 25 image to 45 x 75, while a five-to-one zoom compresses this same image to 3 x 5. The expansion of an image entails a repetition of pixels as shown in the following one-to-two zoom:

$$\begin{bmatrix} 3 & 0 \\ 4 & 2 \end{bmatrix} \xrightarrow{1/2} \begin{bmatrix} 3 & 3 & 0 & 0 \\ 3 & 3 & 0 & 0 \\ 4 & 4 & 2 & 2 \\ 4 & 4 & 2 & 2 \end{bmatrix}$$

A compression, on the other hand, causes pixels to be selected according to the zoom factor. A 25 x 25 image, for instance, is zoomed five-to-one by selecting pixels 3, 8, 13, 18, and 23 from scan lines 3, 8, 13, 18, and 23. Here is an example:



It should be noted that zooming is not a commutative. That is, for two zoom functions Z1 and Z2, Z1(Z2(A)) is not necessarily equal to Z2(Z1(A)).

The compression scheme adopted is to simplify the implementation of the system and the presentation in this paper, it is understood that other compression schemes such as averaging pixel values can be used as well.

4. THE QUERY LANGUAGE

The IQ language is a command oriented interactive language. The user specifies a command through a terminal keyboard. The execution of the command (normally) begins with question-answer sequence between the terminal and the user. Once the user has supplied the appropriate information, the operation specified by the command is executed.

4.1 Definition Facilities

A. Define Function

The DEFINE verb permits the user to define a function and to attach to this function a unique name. Once a function has been defined, it achieves a status not unlike that of a catalogued object subroutine, and can be immediately invoked by the use of the function name. The following example shows the definition of a transformation function.

DEFINE TRANSFORM TRSO

ENTER TRANSFORMATIONS:

(0,63) → (0,3)

(96,100) → (4,4)

DEFAULT: 5

Underlining denotes that the character string is entered by the user. The first line gives the name of the transform function. The last three lines give the desired mapping.

ZOOM, COLOR, and OVERLAY are defined in a similar manner. ZOOM requires the specification of a zoom-ratio; COLOR requires a set of interval-to-color transformations; OVERLAY requires a set of interval product transformations with overlay operators.

B. Define Window

Windows may be defined by one of two methods. The user may specify points in the window by specifying their absolute coordinate (such as longitude/latitude); or he may select points on an interactive display device, relative to the image on the device. The window is entered into the database with a user-specified name.,

C. LET Statement

This statement allows an expression to be replaced by a simple label. In the following examples, I1 and I7 are images; W1, W2, W3, and W6 are windows; T1 and Z1 are TRANSFORM and ZOOM functions respectively.

```
LET W6 = INTERSECTION(W1,UNION(W2,~W3))
LET I7 = T1(Z1(CLIP(W6,I1)))
```

Note that if these statements were entered consecutively, the second statement would be dependent upon the first (because of W6).

4.2 Maintenance Facilities

WRITE TAPE allows an image, window, or function to be copied from the database to magnetic tape. The data items contained on this tape can at a later time be reentered in the database by READ TAPE. The READ TAPE command is also used to copy into the database images which originate from external sources.

Obsolete or unwanted items can be removed from the database by PURGE. When a data item is originally defined through definition facilities, it is treated as a temporary object, i.e., it will be deleted at the conclusion of the interactive session. The SAVE command permits functions, images, or windows to be stored permanently in the database.

A listing of all items contained in the database can be obtained by LIST DIRECTORY. Condensed information concerning each image, window, and function such as dimensions of an image, definition of a function, etc., can be obtained by the SPOTLIGHT command.

4.3 Display Facilities

A. DISPLAY<image expression>

This command requests that an image be directed to a display unit. The operation is straight-forward if the image is small enough to be displayed on the device. If it is not, the user has several options at his disposal.

1. He can request that the image be displayed starting from the northwest corner and continuing until the display area of the device is full. Here he displays as much of the image as can be fitted on the device.

2. He can request image compression. This option calculates a compression ratio and automatically zooms the image to fit the device.

3. He can enter scrolling mode, permitting him to select and display certain portions of the image. This is most conveniently done by displaying a compressed image on one device (as in 2), and then using crosshairs or a cursor to identify the desired areas to be displayed on another device. Only interactive display devices can be used to perform scrolling.

B. DISPLAY<window expression>

In displaying an image one is very much dependent upon the size and discreteness of the display unit. This is not true of windows since a window can be readily scaled to fit any device. The algorithm used to display windows chooses a scale such that extremities of the window very nearly touch the edges of the display area. The scale and the location of extreme points also appear on the display.

4.4 Statistical Facilities

The purpose of the statistical facilities is to provide an analytical means to study and compare images. For example, pixel values of an image can be examined to obtain a histogram. Pixel values of a pair of images can be correlated to construct a joint histogram. There is no limit to the type of operations that can be included in this category. We present several statistical operations here to illustrate the nature of the statistical operations.

A. EXHIBIT HISTOGRAM<image expression>

Pixel values of the image are tabulated to determine their distribution. A histogram is then displayed to the selected device to show, for each pixel value or each range of pixel values, the number of occurrences in the image. For certain display devices, resolution may not be high enough to indicate a precise pixel value count.

B. EXHIBIT DISTRIBUTION<image expression>

This command produces a list which shows the numerical count of the pixel value occurrences for each pixel value. For example, a distribution can be exhibited on a line printer for studying the composition of an image.

C. EXHIBIT JOINT HISTOGRAM<image expression>,<image expression>

Two images of the same size are retrieved and a frequency count is maintained for each unique pair of values of the corresponding pixels. A diagram is then displayed to show the pixel values of the two images, as the X and Y axes, and the frequency count of each pair of values. For example, on a color display device, different colors will be used to indicate different frequency counts.

D. EXHIBIT JOINT DISTRIBUTION<image expression>,<image expression>

This operation is similar to the joint histogram display except that numerical values of the frequency counts will be exhibited.

5. AN EXAMPLE

In this section we present an example which demonstrates a relatively sophisticated application of the IQ language. So as to emphasize conceptual content rather than syntax, the IQ statements have been pruned to their bare essentials. We begin by presenting the problem and the data which are available for its solution.

5.1 The Problem

We wish to determine the pixel value distribution of a certain agricultural area. The region is identified by its elevation which is between 100 and 149 meters above sea level. So as to make this region readily comparable to other similar regions, the distribution should not contain values from the two major lakes located in the region.

5.2 Available Data

1. Three pairs of images: (A1,B1), (A2,B2), and (A3,B3). Each pair, considered as a unit, contains the region in question. An atmospheric haziness has caused portions of each pair to be somewhat lighter than the expected grayness. Fortunately, this lightness has not affected the same portions of all three pairs. The images have a range 0-255, where 0 is white.

2. A contour image C. Each pixel in the image is linearly related to the elevation of the area represented by the pixel. Elevation is incremented in steps of 25 such that pixel value 0 represents 0-24 meters, 1 represents 25-49 meters, etc. Therefore, elevations between 100 and 149 meters are represented by pixel values 4 and 5. The region to be studied is wholly contained in this image.

5.3 The Solution

We begin by joining the component images of each pair:

```
LET C1 = JOIN(A1,B1)
LET C2 = JOIN(A2,B2)
LET C3 = JOIN(A3,B3)
```

Here we have assumed that the A-images are dominant. This assumption is valid since pixel values in overlapping areas are nearly identical.

The next step is to combine these three images in a way such that effects of atmospheric haziness are eliminated. This is accomplished by comparing the three images and choosing, for each pixel area, the greatest pixel value (i.e., the darkest). This requires an overlay function:

```
DEFINE OVERLAY MAXPIXEL
MAX:  (0,255) x (0,255)
```

We can now define the desired image:

LET D = MAXPIXEL(C1,MAXPIXEL(C2,C3))

The areas which contain the lakes must now be identified. This might be accomplished in terms of absolute coordinates by referring to a hand-drawn map. But for the sake of example, let us try another approach. Assume that past experience has shown bodies of water to have pixel values in the interval 75-90. A color function can be used to accentuate this interval:

DEFINE COLOR BLUELAKES
(0,74) → YELLOW
(75,90) → BLUE
(91,255) → RED

Using the color function we display the image D:

DISPLAY BLUELAKES(D)

We assume that the two lakes are now identifiable as blue areas on the screen. The cursor can now be used to define two windows:

DEFINE WINDOW W1
(The first lake is identified in cursor mode.)
DEFINE WINDOW W2
(The second lake is identified in cursor mode.)

Note that the window containing both lakes is defined by UNION(W1,W2).

Let us now turn our attention to the contour image C. Pixel values 4 and 5 identify the agricultural region of the specified elevation. The following transformation allows the contour image to be converted to a Boolean image.

```
DEFINE TRANSFORM TR1
(4,5) → (1,1)
DEFAULT:  0
```

The Boolean image can now be expressed as TR1(C). By multiplying this image pixel by pixel with the image D, all pixels not at elevation 100-149 will be zeroed. This involves the overlay function which we now define:

```
DEFINE OVERLAY MULTIPLY
MULT:  (0,1) x (0,255)
```

The image defined by MULTIPLY(TR1(C),D) contains zeroes for all areas not lying at the appropriate elevations. Let us simplify by

```
LET F = MULTIPLY(TR1(C),D)
```

The last step is to remove the pixel values contained in the lake areas. This is accomplished by masking the exclosure of UNION(W1,W2) onto the image F. The required pixel value distribution may then be obtained by

```
EXHIBIT DISTRIBUTION(MASK(~UNION(W1,W2),F)
```

6. VIABILITY OF THE IQ LANGUAGE

By its very nature as a language of predefined procedures, the IQ language is not complete, that is, it is not capable of solving every problem. While one user, for example, may find the built-in and generic functions to be more than adequate, another user may find these same functions insufficient and restrictive. This directs our attention to an important feature of any language, to wit, the readiness with which the

language can be adopted to meet new requirements. Within the area of programming languages one needs to search no further than RPG and FORTRAN to find examples of languages which do not have this adaptiveness.

In the last paragraphs of this paper we shall examine several natural and obvious extensions to the IQ language, showing, in each case, that these can be incorporated within the language without a disruption of the language's general structure. In this way we present a persuasive (as opposed to conclusive) argumentation for the language's adaptiveness and viability.

The most obvious extension of the language is the addition of new functions. Since the language is readily compatible with functions which return windows or images, this type of language extension poses no difficulty. Plausible new functions might be the following:

- A zoom function which uses an averaging technique rather than a pixel selection.
- A generic transformation function which is based upon pixel distribution quantiles.
- Join, union, and intersection functions which permit more than two parameters.

Also within the area of functions one might consider a macro function facility which permits the redefinition of composite functions. In this manner Z1(CLIP(W1,OVERLAYA(I1,I2))) might be defined as MACRO1(W1,I1,I2). Another variation of the macro function might permit function names to be used as parameters. In any case, since the macro function returns the same data item as the original composite function, this extension of the language can be readily admitted.

As a final example let us consider the implementation of a new data item, the image plane. By supporting this type of item, the user may request that a collection of images be fitted together to form a plane. The construction of a plane can be viewed as an on-going process where the user may at any time reference the plane as the basis for the definition of an image. This may be done by specifying the area of the plane which corresponds to the desired image. To implement the image plane three commands are required, a command which defines a plane, a command which includes an image within a plane, and a command which defines an image as a portion of a plane.

Although relying on brief explanations, these examples indicate several general approaches to the further development of IMDB. We have shown how these developments might be handled within the IQ language, and on the basis of these examples we submit that the IQ language offers a sound foundation for the continued development of an interactive image query language.

REFERENCES

[1] C.J. Date An Introduction to Database Systems, Addison-Wesley, Reading, Massachusetts 1975.

[2] D.M. McKeown, Jr. and D.R. Reddy A Hierarchical Symbolic Representation for an Image Database, Proceedings of the Workshop on Picture Data Description and Management, April 21-22, 1977, IEEE, pp. 40-44.

[3] A.L. Zobrist Elements of an Image-based Information System, Proceedings of the Workshop on Picture Data Description and Management, April 21-22, 1977, IEEE, pp. 55-60.

[4] S.K. Chang, N. Donato, B.H. McCormick, J. Reuss, and R. Rocchetti A Relational Database System for Pictures, Proceedings of the Workshop on Picture Data Description and Management, April 21-22, 1977, IEEE, pp. 142-149.

[5] T. Kunii, S. Weyl, and J.M. Tenenbaum A Relational Database Schema for Describing Complex Pictures with Color and Texture. Proceedings of the Second International Joint Conference on Pattern Recognition, Lyngby-Copenhagen, Denmark, August 1974.

[6] IBM Manual. Information Management System/360, Version 2, System/Application Design Guide (Program Product). Form SHO-0910, 1975.

[7] Honeywell Information System. Integrated Data Store Reference Manual, Order No. BR69, 1972.

IQ - IMAGE QUERY LANGUAGE

Syntax

The rules below describe the major constructs of the IQ language. Prompting sequences for each query statement are not included here. For example, when a DISPLAY is specified, the user will be asked the question "DEVICE?" as well as other parameters directly related to the device. Also excluded from the description is the semantic rules defining the meaning of a query statement. A statement may be syntactically correct but not executable. For example, T(C(I1)) for an image I1, color function C and transform function T is not meaningful, since the color function maps to symbolic color names. In this case, an error message will be directed to the user.

In the following description, ε is used to denote an empty sentence. Capital letters are system keywords.

<query program> → <program body> STOP

<program body> → <statement><program body> | ε

<statement> → <definition statement> | <let statement>
| <display statement> | <statistics statement>
| <file maintenance statement> | <control statement>

<definition statements → DEFINE WINDOW<window name>
| DEFINE TRANSFORM<transform name>
| DEFINE COLOR<color name>
| DEFINE ZOOM<zoom name>
| DEFINE OVERLAY<overlay name>

<let statement> → LET<image name> = <image expression>
| LET<window name> = <window expression>

<display statement> → DISPLAY<image expression>
| DISPLAY<window expression>

<statistics statement> → EXHIBIT HISTOGRAM<image expression>

| EXHIBIT JOINT HISTOGRAM<image expression>,

<image expression>

| EXHIBIT DISTRIBUTION<image expression>

| EXHIBIT JOINT DISTRIBUTION<image expression>,

<image expression>

<file maintenance statement> → <save command> | <purge command>

| <spotlight command> | <list command>

| <activate command> | <tape command>

<control statement> → <restart command> | <journal command>

<save command> → SAVE<name list>

<purge command> → PURGE<name list>

<spotlight command> → SPOTLIGHT<name list>

<list command> → LIST DIRECTORY

<activate command> → ACTIVATE<name list>

<tape command> → READ TAPE | WRITE TAPE

<restart command> → RESTART

<journal command> → JOURNAL | NO JOURNAL

<image expression> → JOIN (<image expression>,<image expression>)

| MASK (<window expression>,<image expression>)

| CLIP (<window expression>,<image expression>)

| <color name> (<image expression>)

| <zoom name> (<image expression>)

| <overlay name> (<image expression>,<image expression

| <image name>

<window expression> → UNION (<window expression>,<window expression>)

| INTERSECT (<window expression>,<window expression>)

| ∿ (<window expression>)

| <window name>

<name list> → <data item name> | <data item name>,<name list>

<data item name> → <image name> | <window name> | <transform name>

| <color name> | <zoom name> | <overlay name>

<image name> → <ident>

<window name> → <ident>

<color name> → <ident>

<transform name> → <ident>

<zoom name> → <ident>

<overlay name> → <ident>

<ident> → a character string within a fixed length

The IMBD system distinguishes between permanent and temporary data files (images, windows, and functions). When a user initiates a session, no data file is available to him unless he activates the files with the ACTIVATE command. Although not presently implemented, access control and password verification can be incorporated in the ACTIVATE to avoid unauthorized use of files. At any time, the user can access all temporary files he has created and all permanent files he has activated. The system maintains a directory of active permanent files and temporary files. RESTART command can be used to clear the directory and restart the session. All conversation between the user and the system can be recorded if the JOURNAL command is issued. NO JOURNAL clears the journal system.

REPORT II

IMPLEMENTATION OF THE IMDB SYSTEM

This documentation details the IMDB software implemented on a PDP-11/45 computer system running under RSX-11D operating system. Design principles and major concepts of the IMDB system have been presented in a separate paper [1,2]. The details of the query language was also described in [2]. This documentation will provide sufficient information for maintaining and expanding the IMDB system.

The presentation is divided into five parts. We first summarize the main concepts of the system in Part 1. Each of the other four parts explains a module of the IMDB system. The four modules are Query Module, Device Module, File Module and Manipulation Module.

The version of the query language described here varies slightly from the design presented in [2]. This version is described in Part 2, Query Module. This version has been implemented at the Data Systems Laboratory of the NASA Marshall Space Flight Center and has become operational since August 1977.

References:

[1] Y.E. Lien and D.F. Utter, Jr., Design of an Image Database Proceedings of the Workshop on Picture Data Description and Management, IEEE, April 21-22, 1977.

[2] Y.E. Lien and R. Schroff, An Interactive Query Language for an Image Database to appear in the International Journal on Policy Analysis and Information Systems, January, 1978. (see also Report I)

1. OVERVIEW

IMDB is an image database system. The user of the system can access, manipulate and manage imagery data in the database through the facilities provided by the image query language IQ. One objective of the IMDB design is to keep minimal the information the user needs to know in order to manage his data. As a result, the query language has been designed to encourage system - user interactive dialogue. The user needs only enter the 2-character command codes of the 24 commands. All other information can be obtained from the user through prompting. However, a user well versed in the language also has the option to provide parameters directly and thus bypass the prompting sequence.

From the user's view, the IMDB system is what is described to him through the query language. The database consists of five basic elements:

(1) Image: It is assumed that an image is rectangular in shape. If an image is obtained through windowing operation, a background rectangle may have to be created artificially. Each pixel in each scanline of an image is associated with a coordinate. The coordinate of a pixel is always in the set {0,1,2,..., 4095, * } x { 0,1,2,...,4095,* }. The components of each coordinate are referred to as LOQ and LAQ. A component marked as * is a don't care. The coordinate of the left upper corner of an image is either specified directly by the user as the image is entered into the database, or is derived from the parent images when this image is formed as a result of a user specified query.

(2) Window: A window is a polygon described by a circuitous ordered set of points. Each point has a coordinate in the grid structure {0,1,..., 4095} x {0,1,...,4095}. As a convention, no two edges of the polygon can cross each other. Each window also has a closure code to indicate whether an inclosure or an exclosure is of interest.

(3) transform function: Each pixel of an image has an integer value between 0 and 255. A transform function defines a mapping from one set of values to another set of values.

(4) color function: A color function defines a mapping from a set of pixel values to a set of color names. This is the primary tool for the user to perform false coloring onto an image for later display.

(5) zoom function: This type of function adjusts the size of an image. Either expansion or reduction can be specified.

The data base is simply a collection of the five types of basic elements. The user can generate new basic elements through query commands. Windows, transform functions, color functions or zoom functions have a rather straight-forward and direct way to generate. For generating images, a fairly powerful concept called image expressions is introduced. A new image can be defined as an expression of existing images. A uniform representation for concatenating or overlapping two images, for masking a window on an image, for coloring an image, for transforming or zooming an image, or for any repetition or sequence of the above operations is included in the mechanism of an image expression. Thus a user can build new images by entering relatively simple expressions.

From the system point of view, the IMDB software is divided into four modules: Query Module, Device Module, File Module and Manipulation Module.

The Query module interacts directly with the IMDB users. Query commands and other device-independant parameters are received and analyzed by the Query module. Error messages are returned to the user through the Query module. The Query module is designed to be independent of device types and therefore is transportable to other hardware configuration.

The Device module handles all graphic I/O activities. It supports color graphic devices such as Ramtek GX-100B color screen and a DICOMED D47 film recorder, and continuous tone devices such as a Varian 4115

electrostatic printer/plotter. The module may request device dependent parameter directly from the user or send error messages related to the devices to the user. The Device module makes characteristics of graphic devices transparent to the Query module.

The File module provides an interface between other IMDB modules and the RSX-11D Files-11 file system. File retrieval and file storage are performed by the File module upon requests from the Query module, the Device module or the Manipulation module. The File module makes database management essentially transparent to the residue of the IMDB system.

The manipulation module provides an interface between other IMDB modules the data manipulation commands entered by the user. (Other query commands are either performed directly by the Query module or dispatched to the Device module by the Query module.)

The four modules communicate through several common areas. COMMON contains in-core file directories, buffer areas to store file header and two scanlines of images and global flags. LUNIT stores logical unit numbers of the peripheral devices. CURRENT records information about the most recently displayed image, such as image name, size, LOQ and LAQ of the north-west corner, etc.

Because of the restrictions imposed by the RSX-11D operating system, the IMDB program was divided into several overlay segments. The overlay structure is described in the Task Build Files of the Documentation package. (Appendix V).

2. Implementation of the Query Module.

The presentation of the query module is divided into two chapters. The first chapter describes the details of the revised IQ language and is written as a self contained reference manual. The second chapter outlines the implementation of the module.

The reader should refer to Report I: An Interactive Query Language for an Image Database, for insight into and rationale of the IQ design. The present version, as implemented in NASA Marshall Space Flight Center, differs only slightly from the version presented in the said report. The differences will be summarized in Chapter I.

CHAPTER I

IQ Language

This chapter is written as an IQ reference manual. The presentation mimics that of the original report on IQ design [Report I]. We shall skip justifications and explanations of certain design decisions, as they have already been covered in [Report I].

1.1 Basic Elements.

There are five types of basic elements in the IQ language: images, windows, transform functions, color functions and zoom functions. Each basic element is a file in the IMDB system.

1.1.1 Image

An image is a matrix of pixel values along with a header block. Since it is assumed to be a matrix, the image is always rectangular in shape. Pixel values range from 0 to 255. The upper left corner of the image is associated with a coordinate in { *, 0, 1, ..., 4095 } X { *, 0, 1, ..., 4095 }. The first component of the coordinate is referred to as the LOQ, and the second component LAQ. The intention is that when the image is first entered into the database, the user can assign its LOQ and LAQ relative to a

4096X4096 grid structure. The asterisk * is used to denote "don't care". The LOQ and LAQ pair is essential to binary image operations to be presented later.

The header block of an image contains:

(a) type: This field is always filled with ' I' to denote the type image.
(b) LOQ
(c) LAQ
(d) pixels/line: It is the number of pixels in a scan line.
(e) scan lines: It is the number of lines in the image.
(f) description: It is a string of characters entered by the user for annotation. The size is limited to 228 characters.

1.1.2 Window

A window is a sequence of points together with a header block. Each point falls within { 0, 1, ..., 4095 } X { 0, 1, ..., 4095 } grid coordinates. The sequence of points form one not necessarily convex polygon. The header block contains the following information:

(a) type: The field contains ' W'.
(b) maximum LOQ/LAQ: The maximum of LOQs of all points and the maximum of LAQs of all points are encoded into this field.
(c) minimum LOQ/LAQ: This field stores the minimum LOQ and LAQ in a way similar to (b).
(d) closure code: The field denotes whether the window is an enclosure or an exclosure.
(e) number of points: It is the number of points in the window.
(f) description.

1.1.3 Transform

A transform function is a mapping from { 0, 1, ..., 255 } to { 0, 1, ..., 255 }. It usually consists of a

collection of subtransformations. Each subtransformation is in the form of

$$a - b = c \qquad \text{where} \quad a \leq b$$

It means that the pixel values from a to b inclusive are to be transformed into c.

The header block of a transform contains:

(a) type: It is always 'T'.
(b) description.

1.1.4 Color

A color function is a mapping from { 0, 1, ..., 255 } to a set of color symbols. There are two systems of color symbols used in the IC language. The first one uses a 4-bit format and consists of eight different colors: dark (D), blue (B), green (G), red (R), cyanine (C), magenta (M), yellow (Y) and white (W). The user uses the one-character symbols to denote colors. The other system allows sixty-four colors and uses a 6-bit format. The basic components of each color are still blue (B), green (G) and red (R). However each basic color has four shades. For example, 1 part of B, 3 parts of G and 3 parts of R give a yellowish color. The user can use B1G3R3 to denote this formation of color. In general, it is hard for the user to visualize the resulting color from the three components. Hence, the user is provided with a color table which maps each of the sixty-four colors to a number. The user can also use this number

to select a color.

A color function is similar to a transform function; it consists of a collection of subtransformations. Each subtransformation is in the form of

$$a - b = c \quad \text{where} \quad a \leq b$$

where c is a color specification. All subtransformations in a color function are either all in 4-bit format or all in 6-bit format.

The header block of a color function consists of:

(a) type: The content is always " C"
(b) description.

1.1.5 Zoom

A zoom-function contains a mapping from old size to new size and a header block. The zoom ratio is a / b where a is the new size, b is the old size and both a and b are positive integers.

The header block consists of:

(a) type: The content is ' Z'.
(b) new size
(c) old size
(d) description

1.2 System Functions.

There are several built-in functions in the IQ language which can be used to create new images. These functions can

be invoked by name. They consist of JOIN, MASK, CLIP and ten different overlay functions.

1.2.1 Join

This function pastes two images together to form a new image, according to their LOC/LAC coordinates. The dimensions of the new image are those which are minimally sufficient to contain the areas of the originals. The first of the original images is defined to be the dominant image: this image takes precedence when the two images overlap. When the result is padded to become rectangular, the pixel value zero is filled.

The rules used to determine the relative positions of the two images are:

(a). If LOC/LAC of the first and second images do not contain * , then the two pairs of LOC/LAC all refer to well-defined points in the 4096X4096 grid structure. Neighboring pixels along the same scan line differ in LOC by one and neighboring lines differ in LAC by one.

(b). If LOC of the one image is * while the other is not, then the * one is assumed to have the same value as the other one. The * for LACs are treated in a similar way.

(c). If LOCs in both image are * , then both are treated as zero. The * in LACs are treated similarly.

Join function results in a new image and the header block of this new image will be derived from the originals. Description field will be empty.

1.2.2 Mask

This function masks a window onto an image to form a new image. If the window is an inclosure, pixels interior to the window will retain the values while exterior pixels will be zeroed. Exclosure functions in the opposite manner. In either case, the result is an image with the same LOC/LAC and the same dimensions as the original. Again, the description field will be empty.

1.2.3 Clip

This function is similar to MASK except that the result image has dimensions which are minimally sufficient to contain the window. This function discards those outermost rows and columns which do not intersect the window.

1.2.4 Overlay Functions

An overlay function takes two images and produces a new image by performing a binary pixel-to-pixel operation over corresponding pixels. There are ten different overlay functions: ADD, SUB, MULT, DIV, MAX, MIN, AVG, XOR, AND, OR. These functions perform respectively addition, subtraction, multiplication, integer division, maximum, minimum, average, exclusive OR, logical AND, and logical OR. Whenever

overflow occurs (e.g., in multiplication), the result is always truncated by taking the rightmost eight significant bits.

The relative positions of the two images are determined according to the rules specified in Join (Section 1.2.1).

1.3 Image Expression

A salient feature of the IC language is its capability for specifying construction of a new image as a functional expression of existing basic elements and system functions. Such an expression is called an image expression. The rules for constructing image expressions are given below. These rules can be applied recursively.

```
<image expression>:
        <image>
        <transform> ( <image expression> )
        <zoom> ( <image expression> )
        JOIN ( <image expression> , <image expression> )
        MASK ( <image expression> , <window> )
        MASK ( <window> , <image expression> )
        CLIP ( <image expression> , <window> )
        CLIP ( <window> , <image expression> )
        <overlay function> ( <image expression>,
            <image expression>)
```

In the above rules, <image> and <window> refer to an image file and a window file respectively. The symbol <overlay function> refers to one of the ten system overlay functions. Since the rules can be applied recursively, a sophisticated image can often be specified as one single image expression. For example, JOIN(AND(X1(T1(MASK(W1,

11))), X2(I2)), I3) is a legitimate image expression if X1 and X2 are zoom files, T1 is a transform file, I1, I2, and I3 are image files, and W1 is a window file.

Note that color functions are not included in the image expression rules. Strictly speaking, a colored image only contains symbolic color names as its pixel values and hence it is not logical to perform any other operation on it. Nevertheless, the internal representation of a colored image is no different from a regular image and, if the user chooses to do so, a colored image may be used to replace <image> in an image expression without any system error. The interpretation of the result is up to the user.

1.4 Devices

The graphics devices can be and can only be referred to by symbolic names in a query session. The user does not have to know any particular logical or physical device numbers used internally in the IMDB system.

The devices supported by the present version of the IQ language and their corresponding device names are:

(a) Two color Ramtek screens: R1(left) and R2(right), with a trackball attached to R1.

(b) One Tektronix 4014-1 terminal: TK

(c) One user command terminal: UT

(d) One line printer: LP

(e) One Dicomed film recorder: FR

(f) One Varian printer/plotter: PL

(g) Two magnetic tapes: T0 and T1

A future expansion will include keyboards and an additional trackball attached to the Ramtek system. The Tektronix terminal is only used as an alphanumeric CRT, although future expansion can take advantage of its graphic capability.

1.5 Commands

The IQL language is a command oriented query language. Each database command activates one specific operation. A command consists of two parts: command code and parameters. A command code is always a two-character name followed by a separator (blank, comma or carriage return). Parameters may be supplied along with the command code, or deferred until answering system prompted questions. Note that all parameters may be entered through prompting. Therefore, the minimal information needed to be entered by the user to initiate a command will be the 2-character command code.

The commands are grouped into five categories: defini-

tion, display, statistics, file manipulation and control.

1.5.1 Definition Commands (5)

These commands are used to create new basic elements or equivalently new files.

1.5.1.1 Build Image (BI)

The form of a build image command is

BI <new image name> = <image expression>

or BI <new image name> , <image expression>

1.5.1.2 Build Window (BW)

The form of this command is

BW <window name>, <closure>, <mode>, <device>

The <closure> code can be EX or EN for exclosure or enclosure respectively. The default value is EN.

There are two modes in window construction: C for cursor and A for absolute. The default mode is A. In A mode, the user types in LCQ/LAQ pairs of the window vertices from the user command terminal. After the user enters

BW W1,EN,A

The system will repeat the question until all points are entered:

ENTER COORDINATES(ONE POINT PER LINE WITH X AND Y SEPARATED BY ,):

The question can be escaped by a carriage return.

In C mode, the user indicates that a window is to be constructed relative to an image presently displayed on <device>. Since there is only one track ball attached to R1, it is only meaningful to specify R1 as the <device>. The user can move the cursor on R1 and select a point by hitting ENTER key of the track ball. To end the construction of the window, the user hits VISIBLE (to make cursor invisible) and ENTER. In C mode, the LOC/LAC of the selected points are calculated from the LOC/LAC of the displayed image. Whether the image is displayed in its true form or in a compressed form, the calculation will produce actual positions of the points relative to the image.

1.5.1.3 Build Transform (BT)

The form of the BT command is

BT <transform name> , <subtransformations>

Each subtransformation is in one of the two forms:

lower bound - upper bound = new value

old value = new value

The right side of a subtransformation is restricted to be one single value. All unspecified intervals can be assigned to one default value

upon answering

NUMBER FOR UNDEFINED INTERVALS?

1.5.1.4 Build Color (BC)

The form of this command is

BC <color name>,<format>,<color transformations>

The color format can be 4 or 6 for 4-bit or 6-bit formats respectively. Each color transformation is in one of the two forms:

lower bound - upper bound = color symbol

value = color symbol

Here the color symbols referred to the symbolic forms of color representation as described in Section 1.1.4. Again, all unspecified intervals can be assigned to one default color upon answering:

COLOR FOR UNDEFINED INTERVALS?

1.5.1.5 Build Zoom (BZ)

This command has the form

BZ <new zoom name> , <scale ratio>

The <scale ratio> is always NEW/OLD.

1.5.2 Display Commands (3)

1.5.2.1 Erase (ER)

The form of this command is

ER <list of devices>

where <list of devices> are device names separated by commas. The effect of erasure depends on the device specified. For R1 or R2, the screen is erased. For LP, a new page of paper is moved under the print head. For PL, the command also slews the paper. All other devices are nor permitted in ER command.

The future expansion will include a capability to advance the roll film in the device FR.

1.5.2.2 Exhibit Pixel Area (EP)

The form of this command is

EP <input device> , <output device>

This command is used to examine the image pixel values of a rectangular area of no more than 20X20. The image is presently displayed on <input device>. The pixel value array is to be displayed on <output device>.

The user is also required to specify

(a) The dimensions of the rectangular area - number of lines and number of pixels.

(b) The upper-left corner of the area through the track ball.

This command is only meaningful when <input device> is R1. The <output device> is restricted to be UT, LP, R1 or R2.

1.5.2.3 Display (DI)

The form of DI command is

DI <image name>,<device>,<color function name>

or DI <window name> , <device>

The <color function name> is optional.

For window display, the device can only be R1 or R2. The displayed output depends on the existing contents of the selected screen. If the screen is blank, the window will be scaled properly so that it can be displayed entirely on the screen. After the window polygon is drawn, the system will ask

DO YOU WISH THE POINTS LABELED WITH PIXEL/LINE COORDINATES?:

A 'Y' answer will cause the coordinates displayed along with the polygon.

If the selected screen has an image displayed, the win-

dow will be scaled according to the displayed image and the window polygon will be positioned correctly on the image so that the coordinates of the image and the window are consistent. A window may be too large to fit on the image. If so, the command will be aborted and error signaled.

In both cases of window display, the color in which the window is to be displayed will be solicited from the user.

Image display is much more involved than window display. If the user specifies a color transformation, it will be applied to the image to produce a colored image. The colored image can later be saved as a regular image file. The sequence of events can be described as the following procedure:

Step R1. ——

If the device is FR, goto F1.

If the device is PL, goto P1.

If the device is not R1 or R2, then error return.

Step R2.

(The device is R1 or R2.)

Ask the user to select a point on the specified screen. Let the specified screen be X and the other one Y. (The system will attempt to display the image on the

rectangular area VIEW defined by the selected point and the bottom right corner of screen X.)

Step R3.

Can the colored image fit in the area VIEW?

If yes, display the image and goto R9.

Step R4.

(The image does not fit in the area VIEW.)

Ask the user if image compression is desired?

If yes, compress the image sufficiently to fit in VIEW, display it and then goto R5. Otherwise, display the upper left portion of the colored image in VIEW and goto R9.

Step R5.

(The compressed image is on X.)

Ask if the user wants to display legend.

If yes, ask the user to select legend position and to enter legend; then display legend at the position selected.

Step R6.

Ask if the user wants to scroll the compressed image.

If no, exit.

Step R7.

(Scrolling)

Erase screen Y.

Ask the user to select a point on screen Y.

Ask the user to select the scrolling point, which is a point in the compressed image as presently displayed on screen X.

The selected point on Y and its bottom right corner define a rectangular area called SVIEW. The scrolling point together with SVIEW specifies a rectangular portion of the colored image whose LOC/LAC are those of the scrolling point and whose dimensions are those of SVIEW. Display this rectangular portion in SVIEW.

Step R8.

Ask if the user wants to scroll again?

If yes, goto Step R7.

Step R9.

Ask if the user wants to display legend on the most recently used screen (X if come from R3 or R4 and Y if

from R8).

If yes, ask the user to select legend position and to enter legend; display characters entered at the position selected.

Step R10. Exit.

Step F1.

Ask the user to select a point on the film. The film has 4096 X 4096 positions.

Step F2.

Ask the user to enter Dicomed related parameters such as magnification factor, resolution, intensity, polarity, etc.

Step F3.

Can the colored image fit in the rectangular area defined by the selected point and the bottom right corner of the film?

If yes, display the image and goto F6.

Step F4.

(The image is too large.)

Ask the user if image compression is desired?

If yes, compress the image sufficiently to fit, display the image and goto F6.

Step F5.

(Display the upper left corner.)

Display the upper left portion of the colored image in the selected rectangular area.

Step F6.

Ask if the user wants to display legend.

If yes, ask the user to select legend position on the film and to enter legend; then display legend at the position selected.

Step F7. Exit.

Step P1.

The user can specify either 4x4 dot matrix for one pixel or 5x5 dot matrix.

Step P2.

Calculate the number of strips required to display the entire image. Inform the user.

Step P3.

Ask the user "How many strips do you want printed?:".

Step P4.

Display the strips.

Step P6.

Ask if the user wants to display legend. If yes, ask the user to enter the legend, then display it.

Step P7. Exit.

1.5.3 Statistics Commands (5)

1.5.3.1 Exhibit Histogram (EH)

The form is

EH <image name> , <device>

The device can only be R1 or R2. The user can also specify the color of the histogram upon answering

WHAT COLOR DO YOU WISH THE HISTOGRAM
TO BE DISPLAYED IN?

The output is a two-dimensional colored graph with horizontal coordinate corresponding to pixel values and vertical

coordinate frequencies.

1.5.3.2 Exhibit Distribution (ED)

The form is similar to EH . The device can only be LP. The histogram of the image will be calculated and displayed as "pixel value: frequency" pair.

1.5.3.3 Exhibit Join Histogram (JH)

The form is

JH <image name>,<image name>,<device>

The two images must be of the same dimensions. The frequencies of pixel value pairs will be calculated. The frequency values will be partitioned into at most seven ranges as directed by the user. Each range can be assigned a color by the user. If the user chooses not to define the range or the coloring of the ranges, the frequency values will be equally partitioned into seven ranges and default colors assigned.

When the joint histogram is displayed, the two coordinates correspond to pixel values of the two images. The colors of the displayed points indicate the frequency range of the pixel value pairs.

The user is also given an option to view the magnified

joint histogram. The magnification is by 2 or by 3.

1.5.3.4 Exhibit Joint Distribution (JD)

The form is the same as JH. The device has to be LP. The output is in the form of "pixel value : pixel value - frequency" for each pair of pixel values. The output format can either be sorted by frequency or by pixel value pair.

1.5.3.5 Exhibit Contigency Matrix (CM)

The form of this command is the same as JH. The device can only be LP and the images are restricted to have pixel values between 0 and 7. All higher values are truncated on the left. The purpose of the command is to compare two classified images to find their differences.

1.5.4 File Manipulation Commands (7)

All basic elements in the IMDB system are treated as files. A file can enter into the database in two ways. First, image files can be brought into the database from tape through the use of Read Tape (RT) command, which will be discussed in this section. Secondly, a file may be created through the use of definition commands (Section 1.5.1). We distinguish permanent and temporary files. Files created through definition commands are all temporary in the sense that they will be removed automatically at the end of the query session unless they are explicitly saved. Permanent files are those which last through query sessions.

Specifically, files brought in by RT are considered permanent.

When a user first logs onto a command terminal, a file directory is assigned for his exclusive use. The file directory is separated into two sections: one for temporary files and one for permanent files. Existing database files can not be used in any command until they are 'activated'. Activation of a file is a process of making the file name known to the user's file directory.

1.5.4.1 Activate (AC)

The form of this command is

AC <list of file names>

where <list of file names> is a list of names of existing files separated by commas. File names specified in the command will be entered into the permanent file section of the file directory.

1.5.4.2 Save (SA)

The form is

SA <list of file names>

This command causes existing temporary files to become permanent. File names specified in the command will be moved from the temporary file section of the file directory to the permanent file section.

1.5.4.3 Purge (PU)

The form is

PU <list of file names>

This command causes files in the directory, whether permanent or temporary, to be removed. Removal of a permanent file also purges the file from the database.

This command is not implemented at the present time. The user has to use PIP command of the RSX-11D to remove a file from the file system.

1.5.4.4 Modify (MO)

The form of this command is

MO <file name>

The purpose of this command is to allow the user to change certain information in the header block of the file.

The alterable fields of the header block are listed below according to file types:

(a) image: LOC,LAC,description

(b) window: closure code, description

(c) transform: description

(d) color: description

(e) zoom: new size, old size, description

The above rules apply to only permanent files. Temporary files can also be modified in exactly the same way except that they do not contain the description field.

1.5.4.5 List Directory (LD)

This command has the form

LD <device>

where <device> can be UT or LP. Contents of the file directory will be printed at the specified device. The file directory contains all information stored in the header block about the files activated or created by the user. (In the actual implementation, an activated permanent file has its header information stored both in the physical file as well as in the directory; and a temporary file does not have a header in the physical file, its header information is stored only in the directory.)

1.5.4.6 Spotlight (SP)

The form is

SP <file name>, <device>

The <device> can be LP or UT for image, window, transform or zoom files. It can be LP, UT, R1, R2 or FR for color files.

This command performs a similar function as LD for a single file: it displays header information of the file. However, if the file is a transform or a color file, SP also displays the definition of the mapping in the file. That is, it lists all the subtransformations in the file.

The most interesting use of SP is to spotlight color function onto a graphics device (R1, R2 or FR). It will display, for each subtransformation, the range of pixel values and a small colored square to indicate the actual color of the subtransformation. If SP is used on LP or UT for a color function, symbolic names of the colors will be displayed.

1.5.4.7 Read Tape (RT)

The form of this command is

RT <device>

The device is either T0 or T1, indicating one of the two tape drives. The options available to the users are:

(a) to read any file on the tape;

(b) to read any number of files on the tape;

(c) to edit a tape file by specifying the starting and the ending line numbers and the starting and the ending pixel numbers;

(d) to read multi-channel composite files up to 16 channels: for each channel, the user can indicate whether the image for this channel is wanted or not, and if wanted, a separate file will be created. In general, an n-channel image can be moved into the database and becomes n+1 separate files - one for each channel and one for the original n-channel file.

1.5.5 Control Commands (5)

These commands are special facilities built into the IMDB system to ease the user-system interaction.

1.5.5.1 Stop (ST)

The form is simply

ST

which ends the query session, causes all temporary files to be removed and is the only command for the user to log off the system gracefully.

1.5.5.2 Restart (RE)

The form is

RE

which performs the similar function as ST except that the user is not logged off and is assigned a new file directory with no entry in it. (The old directory is erased.)

1.5.5.3 Help (HE)

The form of this command is

HE <device>

where the <device> can be LP or UT. It lists all IQ commands with explanations at the specified device.

One of the goals of the IQ design is to minimize the information the user has to remember in order to use the IMDB system. In fact the user does not have to remember the forms of the commands. The user can obtain assistance in two ways:

(a) To consult the system for the command format and its function by typing HE, or

(b) To use prompting to enter command parameters. (The absolute minimum needed to initiate system activity is a 2-character command code.)

1.5.5.4 Journal (JC) and No Journal (NJ)

The forms of these commands are

JC <file name>

NJ

The conversation between the user and the system - in general, it is whatever shown on the UT terminal - can be recorded verbatim in a journal file. The <file name> is the name of the journal file. If the file does not exist prior to the JO command, a new one will be created bearing the name given by the user. If the file is an old one, new journal information will be appended at the end. The command NJ is used to turn off the journal activity. With these two commands, the user can specify journal mode or no journal mode at any time during the query session, switch between two modes any number of times, create several journal files and disperse journal information in any way the user desires. The only restriction is that no two journal files can be active at the same time, one has to be closed by NJ before the other can be named in JC.

The contents of the journal files can be printed at the line printer through PIP facility of the RSX-11D.

1.6 Log On

The log-on sequence to start the IMDB system is very

simple. If the user is a legitimate user of the RSX-11D system, that is, the user has a legitimate UID, the following sequence can be followed to start the IMDB system:

Step 1. Turn on a terminal.

Step 2. Type in Control C to get

MCR>

printed on the terminal.

Step 3. Type in

BEL [UID]

so that the operating system can validate whether UID is legal.

Step 4. If UID is legal, the system will come back with

MCR>

then enter "IQL" after MCR>.

At this time the log-on process is completed, the IMDB is activated and a message will be printed:

```
* WELCOME TO THE IMDP SYSTEM
*
```

Any permissible IQ commands can be entered after the second asterisk. In summary, the entire log-on sequence will look like the following if the user has the access right:

```
(Control C)
MCR> HEL[UIC]
MCR> IQL
* WELCOME TO THE IMDB SYSTEM
* (ready to accept IQ command here)
```

A blank file directory has also been created for the user.

1.7 Special Notes

Some conventions and special cases not covered in the previous sections are covered here:

(a) A window is assumed to be a simple polygon. No two edges of the polygon can cross each other, of course, other than meeting end to end for neighboring edges. The system does not check the crossing of edges and the user is responsible for the correctness of polygon formation.

(b) Whenever a question is asked the user, a carriage return is taken as NO, 0(zero), or the default answer, depending on the nature of the question.

(c) The operating system PIP facility can be used to copy files from tape to tape or from disk to tape, to purge files from the data base, and to rename files in the data base. The IMDB system is built on top of the

FILE-11 file system and any file operation available in the operating system can be applied to IMDE files.

(d) A 'Carriage Return' as an answer to the question 'DEVICE?' will cause the list of all permisssible device names to be printed at UT.

(e) The reserved words in the IC language are JOIN, MASK, CLIP, MULT, ADD, SUB, DIV, MAX, AVG, AND, XOR, MIN and OR. These can not be used as a file name of any file.

1.8 Variations from Original IC Design

The differences between the version of IC as implemented and described in this manual and the one in [Report I] can be summarized as follows:

(a) This version uses 2-character command code and the original version does not.

(b) This version does not have window union or intersection capability.

(c) New commands are added in this version: ER, EP, CM, MO, and HE.

(d) Write Tape command is not included in this version.

(e) Overlay functions are not treated as generic functions in this version. That is, only ten system built-in overlay functions are allowed and the user can not

define his own.

(f) Image expressions have to be evaluated and assigned a new name (in BI) before it can be used in DI. In the original version LET and DEFINE are distinguished. In this version they are combined into build commands.

(g) The original version assumes a longitude/latitude coordinate system. This version assumes a 4096 X 4096 grid coordinate system. The images are no longer associated with geographic position.

CHAPTER II

Implementation

The query module is a collection of 31 subroutines. The design of these routines follows strictly two principles: the query module must be independent of the graphics devices and the query module must be independent of the operating system. As such, the only functions performed by the query module are to extract parameters from the user command line, to solicit missing parameters from the user through prompting and to dispatch tasks to device module, manipulation module or file module. The main activity in the query module is to perform lexical analysis.

2.1 Subroutine Structure

The subroutines are organized in such a way that there is one subroutine for each command. These routines perform the following functions:

(a) Check if the parameters needed for the command are missing - if they are missing, ask the user to enter them, otherwise, extract the parameters from the command line.

(b) Check the validity of the parameters - if any error is detected, inform the user and ask for the parameter

again.

(c) Call lower level modules to execute the command.

Sometimes, a parameter can be related to characteristics of a graphics device. For example, resolution of the Dicomed film recorder or dot matrix pattern of the Varian printer. The query module is restricted to handle only device-independent parameter.

The following routines correspond to the basic commands in IQ:

(a) ACTVAT : AC

(b) BUILDC : BC

(c) BUILDI : BI

(d) BUILDT : BT

(e) BUILDW : BW

(f) BUILDZ : BZ

(g) DISPLA : DI

(h) ERASE : ER

(i) EXHIBD : ED, EH

(j) EXHIBP : EP

(k) FINISH : ST

(l) HELP : HE

(m) JOINDI : JD, JH, CM

(n) JOURNA : JC

(o) LISTDI : LD

(p) MODIFY : MO

(q) NOJOU : NJ

(r) READTA : RT

(s) SAVEST : SA

(t) SPOLIT : SP

On top of these routines there is a main program segment which performs the branching upon detection of the command codes listed above, or restarts if the command code is RE.

Commonly used by these routines to perform lexical

analysis are a set of subroutines which parse the input parameters character by character. We shall discuss them in the next section.

2.2 Analysis Subroutines

The routines used to manipulate and analyze the input character stream to obtain parameters are presented in this section.

2.2.1 COMPRE

This routine compresses a character string to get rid of blanks.

2.2.2 COMPUT

This routine converts a string of digits in ASCII code to an integer numeric value. If the input string does not represent an integer, an error flag will be signaled. The routines in Section 2.1 always read the parameters entered by the user as character strings, that is, in FORTRAN A format. If a parameter is expected to be a number, it will be converted by calling COMPUT.

2.2.3 DEVICQ

This routine asks the question 'DEVICE?' and compares the user's answer against the list of permissible device names. A Carriage Return to the question will cause the list of device names to be printed at UT.

2.2.4 ERROUT

This is the routine used to relay an error message to the user. The messages are usually generated as a result of inappropriate parameters detected during lexical analysis.

2.2.5 EXPRES

The lexical tokens are grouped into eleven classes. The classes are:

Class 1: System functions other than MASK or CLIP
Class 2: MASK or CLIP
Class 3: Transform function or zoom function
Class 4: Image
Class 5: Image expression
Class 6: Window
Class 7: Right parenthesis
Class 8: Comma
Class 9: File name error (file not activated or nonexistent)
Class 10: File type error
Class 11: End mark of input string

This subroutine produces a list of class numbers from a string of tokens in an image expression. The class number list is stored in a stack. The tokens representing file

names are stored in a separate stack. Both stacks are then passed to another routine SYNTAX for parsing the syntax of an image expression.

2.2.6 GETTOK

This subroutine actually performs lexical analysis. It takes a string of characters, presumably representing a command line, and produces one token for each activation. Repeated call to this routine can generate a complete list of tokens.

2.2.7 GETYPE

This routine obtains the file type from the file directory for a given file name. The file name is usually passed as a token to GETYPE.

2.2.8 NAMEQ

The routine asks the user to enter a file name. The actual question depends on the type of the file. It may ask file names for image, window, transform, color zoom or unspecified files.

2.2.9 PACK

this routine eliminates the blanks in an input string and returns the number of nonblank characters in the input string.

2.2.10 SYNTAX

This routine takes token class numbers and file names passed from EXPRESS and produces a postfix representation of an image expression. The postfix form can be written into the database as a file (of type ' E') and later be converted to a real image. These two operations are performed by routines WTEXPR and XINTRP of the file module and manipulation module respectively. The activation of WTEXPR and XINTRP takes place in BUILDI.

2.2.11 SYSFUN

This routine checks if the user uses a reserved system function name as a file name. The reserved names are JOIN, MASK, CLIP, MULT, ADD, SUP, DIV, MAX, AVG, AND, XOR, MIN, and OR.

IMPLEMENTATION OF THE DEVICE MODULE

3.1 Chapter I, Introduction

During the past few years there has been an increasing interest in computer graphics. Computer graphics systems design, data structures, languages, user interaction and uses have been discussed and researched to a great extent. Most of this research has been restricted to a few specialized applications involving either vector graphics or image graphics. In vector graphics dots, lines, surfaces or solids are combined to form pictures. Various transformations on pictures such as rotation, translation, clipping and scaling may be performed. [1] Image graphics, on the other hand, deals with images digitized from actual physical data such as aerial or satellite photographs or maps. Complex algorithms for processing such images have been developed, but emphasis of most of the prior work has been in image enhancement, image preprocessing or classification. [2] For instance, images are processed to cut down on blurring, jaggedness, and other undesirable features.

Most hardware used in image graphics operates according to the principle of raster scanning. An image is treated as an array of spots. Each spot is called a picture element (pixel). A horizontal line of pixels is called a scan line. Each pixel has a value which reflects the brightness or color of the spot in the original image. The pixel values may have specific meanings in the user's application. For example, a pixel value in a classification map may indicate the class number of the pixel. The raster scanning concept presents a difficulty when a line or text is to be displayed. Line segments must be converted by computer to raster information, which will then be transmitted to the display hardware. It often requires extensive central processor time to perform the conversion and large memory space to store the scan lines.

This report describes the design principle and implementation of a software package which involves both vector and image graphics. The package is an integral part of a larger software system called the Image Database (IMDB) system.[3] The IMDB system is designed to support image editing, transformation, storage and display. Essential to the operation of the IMDB system is the ability to display images, vectors and text information. The software subsystem

presented in this report, called the device module (DM) of the IMDB system, performs these display functions on several different raster graphic devices. In addition, the device module performs several statistical computations, with the results appearing as listings or in graphical form.

1.1 Overview of IMDB Design

Project Delta, a research project to design and implement an image database system, was begun in January of 1977 at the University of Kansas. The project was motivated by a need to manage imagery data produced by NASA's Landsat program. Landsat images can be utilized in many different areas including geography, geology, cartography, meteorology, oceanography, agriculture, forestry, urban planning, pollution control, energy resources discovery and allocation, and military reconnaissance. But before experts in any of these disciplines can economically and efficiently access the Landsat information, a system must be developed so that a specialist in the area can analyze the data without having extensive knowledge of computer programming or image processing. It is also desirable to have a single system that can be used by specialists in all of these areas.

The goal of project Delta was to design a graphics system which allows the user to view a desired digitized image and perform various operations on the image in order to obtain information helpful in research or problem solving in the user's field. Operations supported by the system include image display, false coloring, clipping and windowing of images and transformation of images, as well as several statistical operations.

The final product of the project is the image database system called IMDB. The user can access the database through an interactive query language called IQ. When a user enters a query statement he specifies one or more files and a database operation to be performed on the files. The database consists of five types of files; image, window, color, transform and zoom. A window file contains coordinates of points defining a window polygon. A window is used for specifying the area of interest in an image. A color file, which is used for

highlighting various properties of an image, specifies for each pixel value the color value to which it should be transformed. Transform files are similar to color files in that a new pixel value is specified for every previous pixel value. A zoom file contains two numbers which form the ratio of new image to old image size for a compressed or enlarged image.

The tasks of the image database system are grouped into four modules; query module, manipulation module, file module and device module. Figure 1.1 shows the relationships between the four modules. Each directed line signifies that routines in one module call routines in the module to which the arrow points.

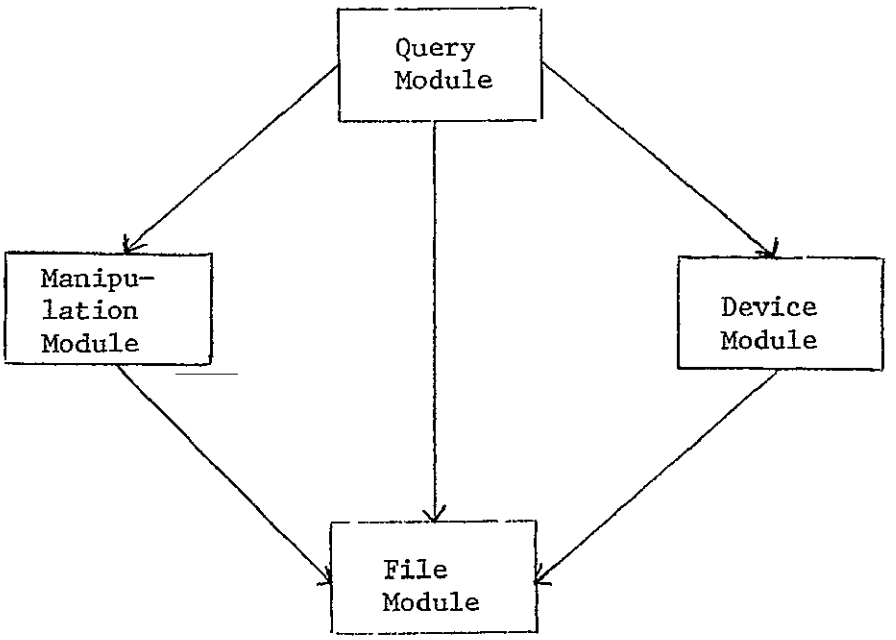


FIGURE 1.1

The query module has four main tasks. First, the query module must identify the database operation to be performed. It must also identify the file or files on which the operation is to be performed and check that the file type and requested operation are compatible. A third task of the query module is to interact with the user and thereby obtain information necessary for carrying out the requested operation. The

fourth task of the query module is to coordinate the other modules.

When using the system, the user inputs a command stating what action or operation he wishes to have performed. The query module then parses this input, asks the user questions as necessary, occasionally performs calculations, and then calls the correct subroutine in one of the other modules. The query module must be responsive to errors occuring at any level, and must instruct the user as to what to do when an error appears.

The manipulation module executes operations such as mask, clip, join, overlay, transform and zoom. When a new image is created by the user, it is expressed in terms of a sequence of these operations on existing files. The manipulation module interprets the operation sequence and constructs the image.

The file module manages storage and retrieval of image scan lines, window vertices, and color or other transform tables. All of the physical input/output operations necessary to perform the file management are handled through the operating system.

Display and listing functions are performed by the device module. These functions include displaying color images, windows, histograms or color tables on raster graphic devices and listing statistical information on character oriented devices such as the line printer and CRT terminals.

Responsibility for the design and implementation of each of these modules was assigned to one person. Managing the project in this way allowed each person to concentrate on a particular aspect of the problem, yet frequent discussions permitted a sharing of ideas and theories and also gave each person an understanding of the overall project. In the initial stages of the project this separation was also important as it permitted project members with varying amounts of experience and time to work at a pace suited to them.

1.2 The Device Module

Design of the device module involves consideration of two major tasks. On one hand, various database operations require graphic display or alphanumeric listing. On the other hand, devices of various types and with varying capabilities must be supported.

1.2.1 Database Operations Supported by the Device Module

Among the database operations supported by the query language IQ [4], ten depend on the device module for input or output. Each of the operations supported by the device module falls into one of four categories.

a. Definition Operations

Build window (BW) is the only definition operation which relies on the device module. The user may specify window vertices through a graphical input device and each vertex point will then be displayed through graphical output.

b. Control and Utility Operations

Spotlight (SP), which describes the contents of a given file, requires alphanumeric output and, in the case of color files, may also use graphic output.

c. Display Operations

The device_module is responsible for all of the display operations which include window display, image display, text display (in DI), erasure (ER) and exhibiting pixel area (EP).

d. Statistical Operations

A variety of statistical operations, all of which require either graphical or alphanumeric output, may be specified by the user. These operations are exhibit histogram (EH), exhibit distribution (ED), exhibit joint histogram (JH), exhibit joint distribution (JD) and exhibit contingency matrices (CM).

1.2.2 Devices supported by the Device Module

The IMDB system has been implemented for the hardware configuration located in the Data System Laboratory of the NASA Marshall Space Flight

Center in Huntsville, Alabama. The major components of the system are shown in Figure 1.2.

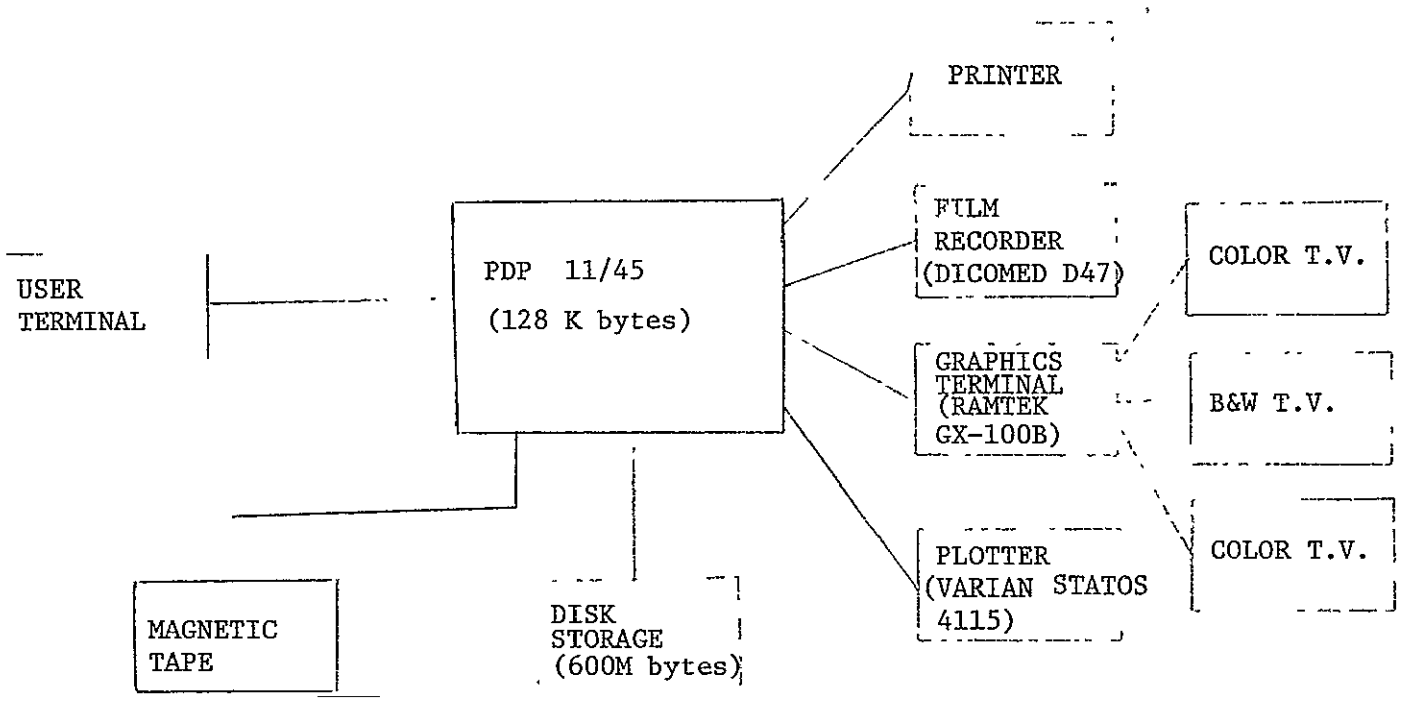


FIGURE 1.2

The Ramtek GX-100B system is a raster graphics device with two 19 inch color monitors, one 17 inch black and white monitor and a trackball. Each screen surface consists of 256 by 256 individually addressable points, each of which may be assigned any of the eight colors red, green, yellow, blue, magenta, cyan, white or dark.

The Dicomed D47 film recorder produces either black and white or color photographs. The film, either standard roll film or polaroid film, is considered to be a matrix of 4096 by 4096 points. Each point in the matrix can be assigned an exposure value in the range 0 through 255 for any one of three color filters or a neutral filter.

The Varian 4115 electrostatic printer/plotter is capable of plotting 1408 points across each line on a strip of paper 14 7/8 inches wide. Each point is either black (printed) or white (blank). Any number of lines may be drawn.

The line printer has the standard format of 132 characters per line. Device module output may also appear on the user terminal, which may be one of several available hard or softcopy terminals.

All of the devices mentioned above are attached to the Input/Output bus of the PDP 11/45 system. I/O commands can be issued to the devices through the operating system RSX-11D. The commands relevant to the device module will be explained in chapter four.

1.2.3 Database Operations and Associated Devices

The two major concerns of the device module, operations and devices, must be integrated to form a system capable of producing whatever output the user desires and on whichever device he wishes. Figure 1.3 shows the relationships between the database operations and the devices on which they may be displayed.

	RAMTEK		DICOMED	VARIAN	LINE PRINTER	USER TERMINAL	
	output	input				output	input
BW	X	X					X
ED,JD					X		X
EH,JH	X						X
CM					X		X
DI (I)	X	X	X	X			X
DI (W)	X						X
DI (L)	X	X	X	X			X
ER	X			X			X
SP	X		X		X	X	X
EP	X	X			X	X	X

FIGURE 1.3

Relationships between database operations supported by DM and devices used by the operations. DI command

may be used to display an image (I), a window (W) or legend information (L). The trackball on the Ramtek system is an input device to be used to select points of a window or to position the image or legend being displayed. All operations require the user to enter parameters through the user terminal.

3.2 Chapter II, Device Module Design Strategy

As Dijkstra pointed out [6], a program must be carefully structured in order for its correctness to be demonstrated in a convincing manner. This job is particularly difficult when the size of the program approaches, say, several thousand lines. It is the programmers obligation to modularize a program, to simplify its control logic and to organize the program into comprehensible form.

For a program of moderate size, it has been shown that the three basic control structures - sequencing, IFTHENELSE and DOWHILE - are adequate for expressing any control logic [7]. This concept can be extended to systematic development of very large programs, as described by Dijkstra [6] and Mills [8,9]. This methodology for large programs is called top down structured programming.

The IMDB system is implemented using top down structured programming. In this chapter we first present an overview of top down structured programming and then describe how the method is used to develop IMDB and in particular, the device module.

2.1 Top Down Structured Programming

Structured programming has long been advocated by Dijkstra[6] as a programming methodology to facilitate correctness proofs and to enhance program readability as well as manageability. Clarity of program structure and simplification of program control logic are achieved through the use of fundamental program constructs such as IFTHENELSE, DOWHILE and blocking, and through the elimination of arbitrary GOTO's. Mills further extended the methodology to top down structured programming, a systematic way to specify, document and code a large program [8].

Top down structured programming is an evolving process, beginning with a reasonably sized functional specification of the entire system, for example, a page or two in size. The functional specification,

called the root segment here, defines the top level control logic and necessarily leaves out many details. Within the root segment, several subspecifications may be defined. Each subspecification may later evolve into a program segment of manageable size. At each step of the process a subspecification is expanded to simpler and simpler functions until all functional specifications or subspecifications are translated into statements of the programming language itself. At the conclusion of the process, the total system is implemented as a tree of program segments.

The main features of the top down programming process can be stated as follow:

1. The entire program is broken into its constituent parts through a series of successive refinements. Each refinement takes one functional specification or subspecification and expands it into a segment of finer functional subspecifications.
2. Program segments are natural units of documentation, specification and coding. Therefore, top down programming allows documentation, specification and coding to proceed concurrently.
3. Each program segment has a single entry and a single exit. Thus, when reading a segment name, one can be assured that control flow will pass through the segment with no side effect on the control logic.
4. Higher level segments may contain program stubs representing sections of code not yet fully developed. In practice, this can be done by inserting dummy routines to hold the place for the code for the next level of expansion.
5. Each segment is written as a structured program. The final system is a hierarchy of segments.
6. Program debugging also becomes a top down process. The process begins with verification of the root segment and proceeds through debugging of each level of refinement.

Top down structured programming requires two software tools. Since each segment is a structured program by itself, a programming language which offers control structures such as IFTHENELSE and

DOWHILE is needed. We decided to use RATFOR (RATional FORTRAN [10]) for two main reasons: RATFOR offers a convenient set of control structures adequate for writing GOTO-free code, and a RATFOR translator produces ANSI Standard FORTRAN code which is accepted by most machines.

Another tool needed for top down programming is an operating system which provides program library facilities. Each segment can be compiled and its object code saved under a symbolic name in an object program library. The linking loader (task builder) can later build an executable task by combining all relevant object segments. RSX-11D provides this capability. In fact, if a segment named by higher level segments is not yet written (not in the library), the task builder will issue a warning flag and proceed to build the rest of the task. During the execution time, if control logic actually reaches the undefined segment, the program will be aborted. Otherwise the program is not aware of the incomplete part. This capability allows the programmer to develop the entire program hierarchy in a depth-first manner when necessary, without having to insert dummy segments in the program library.

2.2 Top Down Structured Implementation of IMDB

The root segment of the IMDB system is a simple branching statement conditioned on IQ query commands. Upon detection of a legal command code, branching takes place and control passes to the next level segments. There is one segment for each command code (each database operation). Depending on the complexity of its corresponding command code, a segment may develop into a large subtree of segments or it may be a leaf segment of the total hierarchy.

As the process evolves, many common functions at the lower levels are identified. For example, a segment to retrieve one scan line of an image may be needed in several places. Many functional primitives for graphics devices are also gradually developed. Examples are: drawing a line between two points, printing a string of characters, erasing a screen, etc. Commonality

of functions gradually leads to partitioning of the entire hierarchy into several independent parts so that each part can be programmed by one person. Once some initial segments are written and the interfaces between segments are defined, several programmers can work concurrently and independently on the coding. Of course, each programmer will have to incorporate his new segments into the main body of the hierarchy.

The IQ language contains 24 commands, hence the total hierarchy can be viewed as a tree with 24 branches at the root level. All database commands requiring graphical or statistical output are allocated to the device module. These commands were listed in Chapter 1. The command BI requires parsing and evaluation of an image expression and is by itself in the manipulation module. Many low level file manipulation operations, mainly those involving direct interface with the operating system, and AC, SA, PU, RT, JO and NJ commands are grouped into the file module. The query module is engaged in device independent, operating system independent interaction with the user, such as collecting command parameters, and also performs the rest of the commands with the help of the file module segments. The query module and the device module are separately implemented, each by one programmer. The file module and the manipulation module are implemented by the third programmer.

Within the device module, each database command corresponds to at least one segment. In case of the DI command, three segments are involved; each of the image, window and character display operations takes one segment. Therefore, the device module is actually a collection of subtrees, with each subtree responsible for one basic database operation. This division of tasks, according to database operations as opposed to devices, allows us to defer the device dependent decisions to the lower levels and hence localize device dependencies to a smaller collection of segments. Some segments found in the lower levels of the hierarchy may be referenced by more than one segment at a higher level. This can be conveniently done through the use of the program library.

At the level of finest refinement of segmentation, each segment

must develop a structure. RATFOR is the programming language used in the implementation of IMDB. RATFOR includes versions of the three basic control structures sufficient for program coding, as well as several other useful structures. Besides basic sequencing, RATFOR has an IFELSE statement and a WHILE statement. Block structuring is available, as one need only enclose several statements in brackets in order to form a block. These blocks may be nested to any desired level.

RATFOR programs can be written so as to be easily read. Statements may start anywhere on a line, so indentation is commonly used to highlight blocks and loops. Flow of control is straightforward, as the available control structures eliminate the need for GOTO's. RATFOR code may be readily documented, as comments are delimited by a '#' occuring anywhere on a line. Anything after the '#' and on the same line is considered to be a comment.

2.3 Advantages of Top Down Structured Programming

Using top down structured programming techniques is advantageous for several reasons. After the process of refinement of subfunctions is completed, each resulting segment has a single function of mapping initial data to final data. By limiting the tasks of each segment, the segments may be more easily coded and more easily read. Structured programs are usually GOTO free, so code can be read sequentially without jumping around mentally to follow the flow of control. This property adds to readability by allowing better mental association of program static text with dynamic execution. Due to the one entrance, one exit property of structured code, each segment flows from top to bottom without any side effects in control logic other than in that particular page or segment. This ease of reading is especially important in a system such as the IMDB system where the original designers and implementors will not be the persons maintaining or using the software.

As well as being designed in a top down fashion, a system may be coded in a top down manner. Segments of code can be successively generated and tested. In order to test each segment, dummy routines can be inserted in place of lower level functions. Progressive

testing of segments isolates problems of syntax and control logic, thus simplifying the task of debugging. Data provided for newly designated segments at the next level can be tested by introducing dummy versions of these new segments.

A major advantage in top down structured programming is the decrease in complexity of a proposed system. Through successive refinement of segments and structuring within these segments a system can be produced that is no more complex than the actual problem [11]. This reduction in complexity subsequently results in significant cost reduction during implementaion because the goals of each segment and its inputs and outputs are clearly stated. The resulting system is also more reliable, easier to maintain and easier to modify than it would be if such a programming scheme were not used.

A segmented, hierarchical system also permits relative ease of expansion. This is important in a graphics system, because new devices or devices with extended capabilities are continually being developed, so it is probable that the user of a graphics software system will wish to extend his system to be used on such devices. The hierarchical structure of a system makes the problem of deciding where new code or new segments need to be inserted less tedious. Changes can be made at the proper levels of the hierarchy without the necessity of redesigning the entire system. More segments can be added as required to extend the device interface capabilities of the system.

In the IMDB system, it is possible that in the future more database operations will be included. This type of expansion of the system is also simplified due to top down structured programming. An entire new branch to the hierarchy can be developed, working top down from the database operation and using the segmentation in the previous branches as a guideline.

Due to the hierarchical structure of the IMDB device module, unit debugging is possible. By including dummy routines at the lower levels, routines near the top of the hierarchy can be checked for correctness in the early stages of coding. Each database

operation can be compiled and taskbuilt without the need for completing the entire IMDB system or even the device module. This step by step testing makes it possible to discover and correct errors in logic or syntax before they cause more errors and become more difficult to find by being embedded in a lengthy piece of code.

Top down structured programming is also a good strategy for designing graphics systems because it aids the localization and minimization of device dependence. All device dependent code can be allocated to segments at a given level or levels. Since the device module is structured according to database operations rather than devices, all of the device dependent routines are placed as low as possible in the hierarchy, thus localizing the device dependent activities.

3.3 Chapter III, Hierarchical Structure of the Device Module

Implementation of the device module employs a top down, hierarchical programming strategy. Routines are grouped according to the database operations for which they are required. Each major database function invokes one single device module routine at the top level of the hierarchy; within this routine, computational or input/output functions are further divided according to the devices. The program control flows from the device-independent routines at the top levels down to the device-dependent routines at the lower levels. At the bottom level of the hierarchy, operating system I/O routines are called to issue commands to specific devices.

In order to facilitate communication between the query module, device module and file module, several common areas have been set aside. The common areas relevant to the device module are Permanent File Core Table (PFCT), Temporary File Core Table (TFCT) and Core Table Entry Buffer (CTEBUF). File name and header information about each permanent or temporary file are stored in the PFCT or TFCT. Information in the file header may include file type, size of an image, number of points in a window or color format. CTEBUF is used to store the header information for the most recently accessed file.

3.1 The Hierarchy

The hierarchial structure of the device module consists of five basic levels. Figure 3.1 illustrates the hierarchical structure and the relationships of the query and file modules to the device module. It also shows the information passed between the levels.

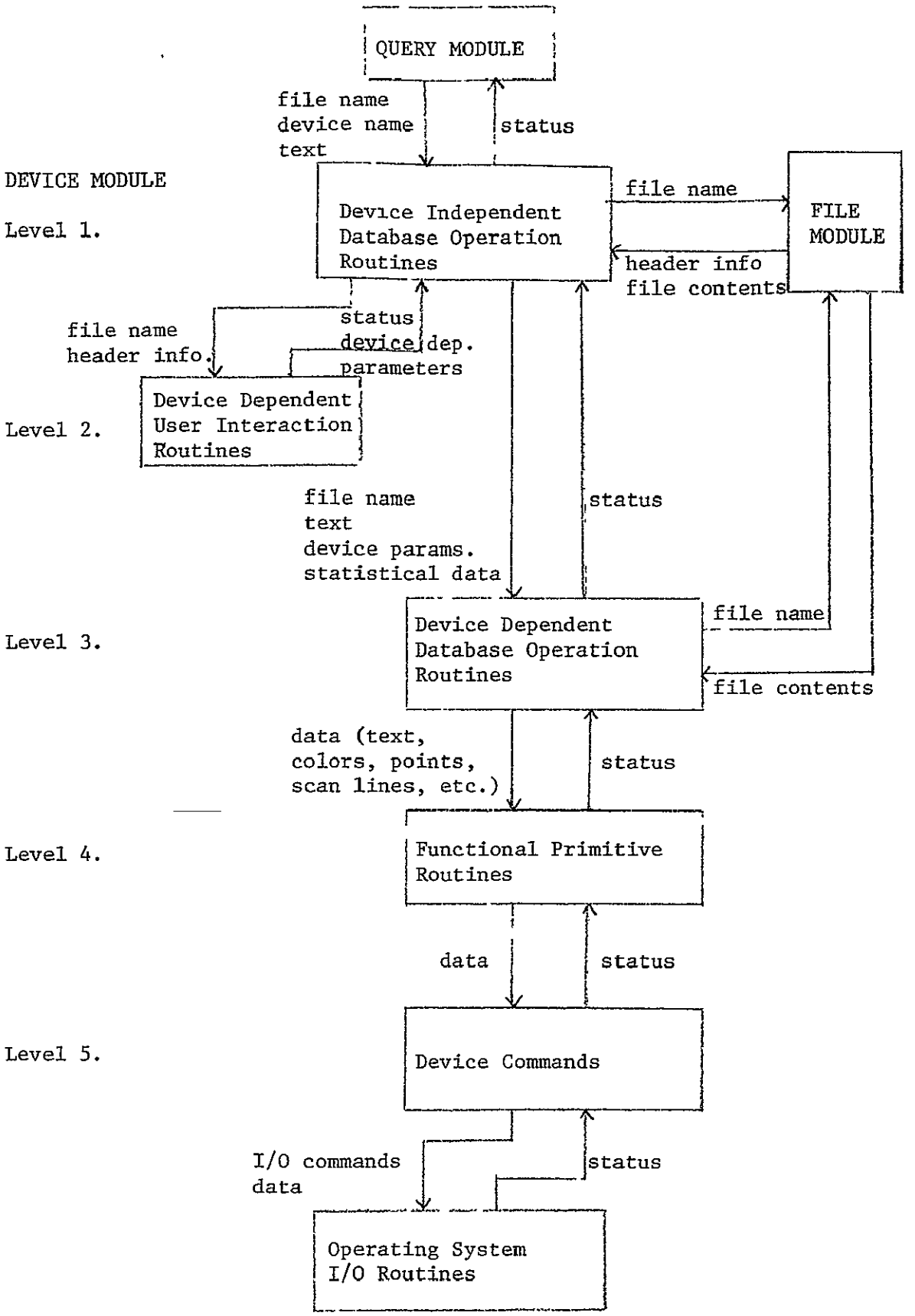


FIGURE 3.1
Device Module Hierarchical Structure

The subroutines in the device module correspond to the five categories of the hierarchy. The top level in the structure provides the interface between the query module and the device module. Before calling device module routines, the query module determines which database operation the user has requested, which files the operation should be performed with, and on which device the output is to appear. Type checking is done by the query module to see if the file and the operation are compatible. The query module then passes information concerning device, file name, color function and text to the top level routines of the device module.

In the top level of the device module as many device independent operations as possible are completed. This usually requires accessing the core table entry for the given file to obtain the header information. When a statistical operation has been requested, the necessary statistical computations can usually be performed at this level.

The top level routines also check compatibility of the database operation and the specified device. If the operation cannot be performed using the designated device, the device module indicates this to the query module by passing back an error status. The query module is then responsible for informing the user of his mistake and advising him of what action to take. A bad status may also be returned to the query module at this time in a few other instances, such as statistical information overflowing the buffer area set aside for it. In such cases, the operation is usually aborted and the user may re-enter a database operation command.

If the specified device is compatible and no other errors have occurred, control passes to a subroutine at the next lower level, level two. Parameters passed from level one to level two generally include file name and core table information. Level two is the point at which device dependent user interaction takes place. Since display devices vary widely in their capabilities and design, each requires different parameters in order to be utilized to its capacity. In order for the query module to remain device independent, these device dependent parameters must be solicited by the device module. These device

dependent parameters are used by the display devices to position the output on the display surface, to select the desired colors, to select the magnification factor and to perform other device specific operations. Various device dependent calculations may also be performed at level two. These calculations include size of display, scaling factors, number of characters to be displayed, and positioning. All of these parameters are then either returned to the top level routine or passed directly to level three. Level two routines occasionally return status codes indicating actions such as compression of images or truncation of character displays. The status codes received from level two usually do not cause the operation to be aborted, but may be interpreted by the query module at a later point in order to indicate possibly unexpected occurrences to the user.

At the third level of the hierarchy are found the device dependent, database operation specific routines. At this level any further calculations involving information acquired in the second level are performed. Data must be read from the database, possibly transformed through calls to the file module, and put in the proper form to be passed to routines at the lower levels in the hierarchy.

The fourth level of the hierarchy contains the routines implementing the functional primitives of the graphics system. It is these basic graphic primitives which are combined in order to execute the database display operations. Functional primitives include beam movement, point drawing, line drawing, rectangle drawing, character display, scan line display and erasure. The functional primitive routines are device dependent, yet each device may theoretically have a corresponding functional primitive at this level.

At the fifth and lowest level are found the routines which send the device commands to the device. These routines take the parameters passed to them by the functional primitives and put them in a form for sending to the I/O subsystem of the operating system. Through this interface with the operating system the desired output device is addressed and command codes provided by the device manufacturer are relayed to the device.

In order for any database display operation to be executed,

control must flow from the query module to level one of the device module and then on through each level to level five, the operating system interface. Within the device module, a separate subroutine could be provided for each of the five levels of the hierarchy. In some cases, however, efficiency is increased and redundancy of code or excessive parameter passing decreased by combining two or more levels in one subroutine. Such is the case for instance when no device dependent questions need to be asked of the user. Occasionally a level may be skipped due to the hardware capabilities of a certain device. For some operations which are very simple or which may only be performed on one device or device type, it is unnecessarily confusing to branch out to numerous subroutines.

3.2 Example of Hierarchical Structure

As an example of the five level device module hierarchy, let's look at what takes place when the user requests that an image be displayed. At the query level, the user is asked for the name of the image file, the name of the device on which he wishes the image to appear, and the color function, if any, that he wishes to be associated with that image. These three parameters are then passed to the top level display subroutine of the device module, DISPL.

The display routine DISPL calls the file module core table entry routine to determine how many pixels and lines there are in the image. DISPL then passes the device, file name, starting scan line and scan element, number of lines, and number of elements to the second level routine. For our example, let's suppose that the user has requested one of the Ramtek screens as the display device. This second level routine, RAMQ, first asks the user to input a point at which he wishes the northwest image corner to appear on the screen. The routine then determines whether or not the entire image will fit on this portion of the screen. If so, the compression ratio is set at one to one, indicating no compression. If not, the user may request compression or may ask that only the northwest corner of the image be displayed. If the first choice is made, a compression ratio will be calculated and the status parameter set to indicate compression to the

query module. If no compression is desired, the ratio is again set at one to one. In all three cases the number of lines and pixels to be displayed on the screen is determined. All of these parameters are then passed back to DISPL.

DISPL now calls the level three routine, RAMD. RAMD attaches the correct Ramtek screen and calls the file module routine to transform the image as required by the color function or the compression factor. Repeated calls to RDSCLN, the file module routine for reading data from a file, are made to obtain the transformed pixel values. RAMD extracts the desired pixel values and sends them to the BLOKZ subroutine. This routine performs the functional primitive operation of image scan line display and is also the fifth level routine.

At the lowest level, BLOKZ is responsible for sending the commands and data to the operating system in order to initiate output on the Ramtek. Through the use of the BLOKZ routine the correct mode, format, color and positioning are selected.

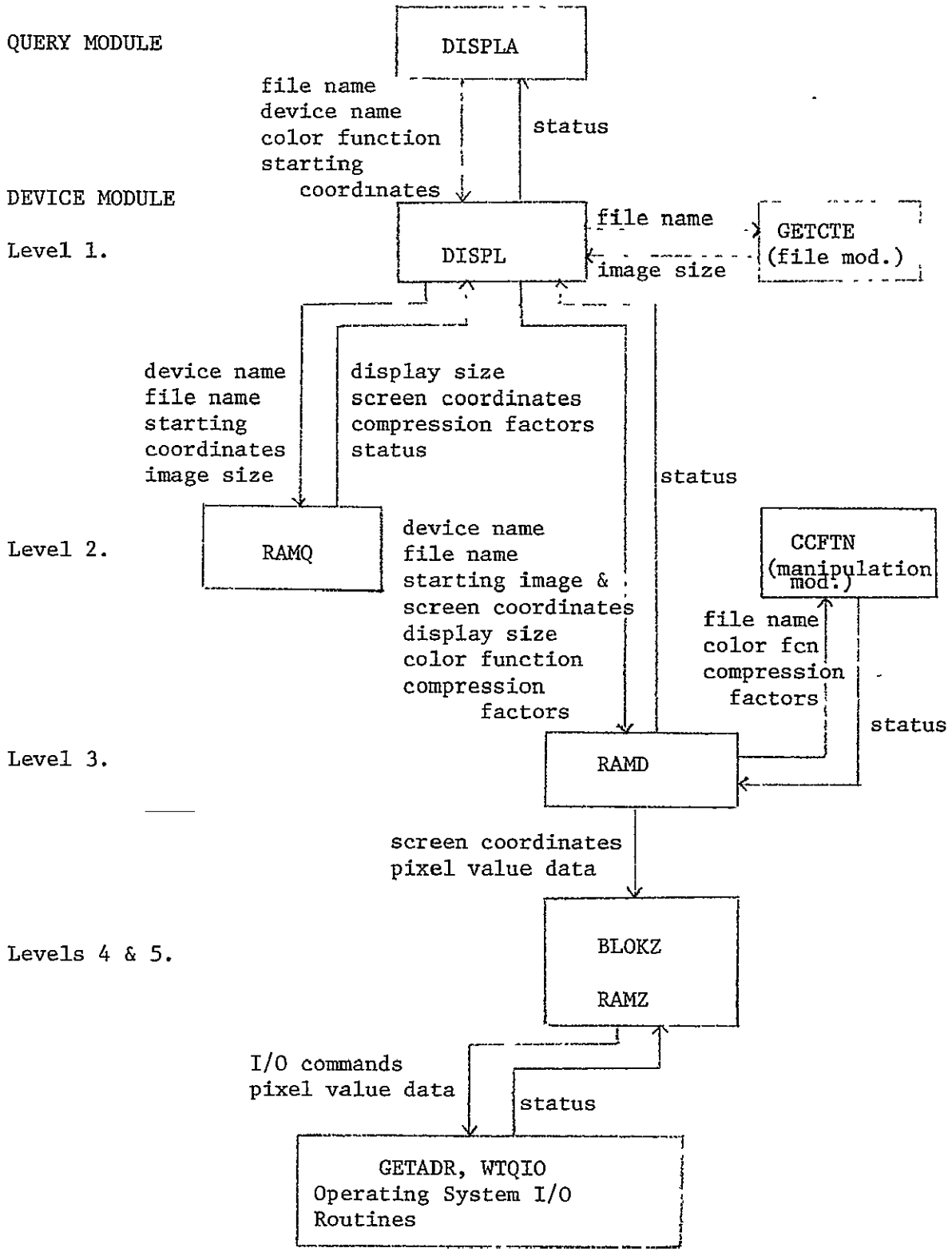


FIGURE 3.2
Device Module Hierarchy for
Image Display Operation

3.4 Chapter IV, Device Commands

The lowest level of the device module hierarchy contains the device dependent routines which are responsible for sending the commands to the device. The commands available depend on the hardware characteristics of the particular device. The routines at this level are used independently of the function being performed. They provide the interface between the programmer and the device through the operating system. Many of these routines fill command buffers and parameter blocks with information about logical unit number of the device, how many commands or bytes of data are to be read, and which instructions are to be performed by the device hardware.

Two system input/output routines which can be called from FORTRAN programs are provided by the operating system: GETADR and WTQIO. GETADR loads the initial address of a specified user buffer into an array for use by the I/O routines. WTQIO issues an input/output command and waits for I/O completion before continuing.

4.1 Ramtek Hardware and Commands

The Ramtek GX-100B display system has the hardware configuration shown in Figure 4.1. This system uses the raster scan technique to produce an image on the cathode-ray screen. [4]

The bi-directional interface interconnects the local CPU and the display generator and device multiplexor. The instruction buffer stores received commands and data into a 256-work FIFO buffer in order to provide asynchronous operation of the CPU and display generator. These commands and data are input, decoded and processed by the instruction processor. Dot-matrix character patterns are generated from ASCII character codes by the character generator, while the vector generator performs the essential parameter manipulation and linear interpolation for automatically drawing lines between arbitrary end points. The scan converter controls the writing of all data into the refresh memory which stores the display image. The video generator

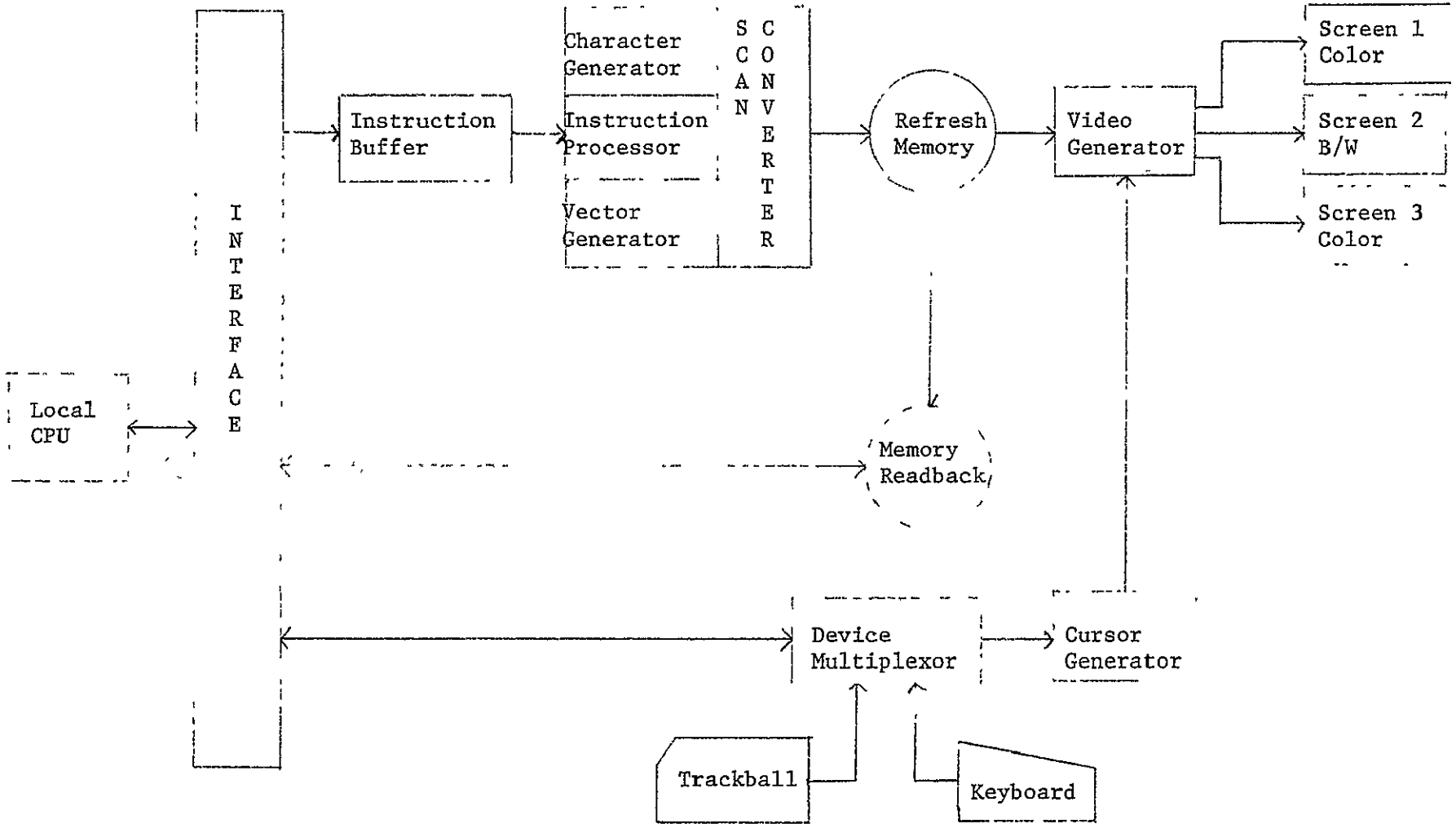


FIGURE 4.1

Ramtek Display Hardware

scans the refresh memory and generates video and synchronization signals which are decoded by the selected television monitor. This television monitor then drives the electron guns which excite the phosphors painted on the picture tube in order to display the image.

The cursor generator interconnects the cursor/trackball registers of the device multiplexor with the video generator for the purpose of displaying a maneuverable cursor which bypasses the refresh memory. The trackball option converts the rotational movements of a mounted sphere into X and Y step codes for serial transmission to the device multiplexor. The keyboard option converts input into ASCII character codes for serial transmission to the device multiplexor. The memory readback option interconnects the CPU and circulating refresh memory in order to read the contents of the refresh memory.

The Ramtek routines at this lowest level in the hierarchy are BLOKZ, ARAMZ, and RAMZ. These routines use the KRAMZ assembly language function to stack the Ramtek opcodes into a buffer for I/O to the Ramtek. These opcodes are the instructions and flags on instructions which are used by the display generator in writing output on the screen. These instructions to the display generator fall into three categories; channel/subchannel partitioning, positional addressing, and control modes and flags. Some special functions are also provided. A summary of the Ramtek instructions is given in Figure 4.2.

Channel and subchannel partitioning allows a particular screen to be addressed and also allows for color selection. The select display channel (SDC) instruction establishes access to the refresh memory of a specific television monitor and thus performs the function of attaching the device. The select subchannels (SSC) instruction establishes access to the subchannels for the various colors. Seven colors, plus dark, capability is achieved using three subchannels, with a fourth subchannel used for white overlay.

There are three modes, absolute, indexed and relative, and eight instructions associated with the positional addressing of the

Instruction Mnemonic	Meaning	Function
SDC	select display channel	attach device
SSC	select subchannels	designates colors
LEX	load element index register	define starting element address
LLX	load line index register	define starting line address
LE2	load element register 2	define terminating element addre
LL2	load line register 2	define terminating line address
LER	load element relative	displace element address
LLR	load line relative	displace line address
LCM	load control mode	specifiy type of graphic output
	AN - alphanumeric	
	TD - transverse data	
	RD - raster data	
	CD - complex data	
	GV - graphic vector	
	GC - graphic cartesian	
	GP - graphic plot	
	GE - graphic element	
LDX	load and execute data	transmit data to display generator one byte at a time.
BLK	block transfer	transfer up to 256 words of data to display generator
ERS	erase	full screen erasure

FIGURE 4.2

Ramtek Instructions

refresh memory. These instructions carry a line or element address to the display generator in order to update the current operating point (COP). In a 256 line by 256 element system, the least significant bit of the effective address is truncated, as this effective address is in the range 0 to 511. The load element index register (LEX) and load line index register (LLX) instructions define the starting element and line address, while the load element register 2 (LE2) and load line register 2 (LL2) instructions define the terminating element and line address. If the indexed addressing mode is selected, these element and line addresses are summed with the element or line index register; otherwise, the origin is assumed to be at element zero, line zero. Relative displacement of the element or line address is accomplished through the load element relative (LER) and load line relative (LLR) instructions.

The control mode, as established by the load control mode (LCM) instruction, stipulates the type of graphic form to be written into refresh memory. Eight control modes are available - four data modes and four graphic modes. In the data modes, alphanumeric, transverse, raster and complex, each programmed instruction sequence accesses and addresses the refresh memory, issuing one or more data sequences to be decoded. After decoding this data, the display generator writes the data into the accessed subchannels beginning at the current operating point, then recalculates the COP. In the graphics modes, vector, plot, cartesian and element, element addresses are issued instead of data sequences.

In alphanumeric mode (AN), each data byte is interpreted as an ASCII character code. In the Ramtek character set, each character consists of a 5 by 7 dot matrix within a 7 by 12 matrix. Double width or double height flags may be set to get larger characters. Characters are written in the same bit pattern as received, rather than being decoded and transformed. Transverse mode results in an eight bit wide vertical column being written, from top to bottom. Raster mode (RD) is identical to the transverse mode, except that the data is written from left to right across a specific raster line instead of in a vertical column. The complex data mode (CD) is most often used

for generating multi-colored displays. Each bit of each pixel description is written into one or more subchannels and the data is written from left to right across a raster line. Either the word format or the byte format flag may be set in complex data mode. Word format specifies that a 16 bit word be decoded for the Z-axis data and byte format selects an 8 bit byte format for decoding.

Two forms for transmitting data to the display generator are available. The first form, using the load and execute data (LXD) instruction, transfers one 8-bit data byte at a time. Although this method is very simple, it is often inefficient. The second form of data transmission is via the block transfer (BLK) instruction. This instruction conditions the display generator to receive a specified number of data words (up to 255) for processing. A reverse packing flag is available with the BLK instruction. This flag may be set for implementations where the right byte is to be processed first, e.g., in the PDP 11.

While operating in any of the four graphics modes, the display generator draws specified graphic forms by decoding and processing positional addressing instruction rather than transferred data. Graphic vector mode (GV) is used for drawing lines between two specified points. These points are defined through the LE2 or LER and LL2 or LLR instruction. For drawing solid rectangles between arbitrary endpoints the graphic cartesian mode (GC) may be used. The graphic plot mode (GP) draws contiguous vertical line segments in order to produce a histogram style plot of consecutive data points. In the graphic element mode (GE), each individual pixel is activated as addressed by line and element. All of these graphic modes permit a third dimension of color or intensity, which is determined by the SDC and SSC instructions.

One useful special function provided in the Ramtek instruction set is erasure. The ERS command performs a full screen erasure. Selective erasure of colors may be performed by using the SSC instruction to select specific subchannels to be erased.

The BLOKZ subroutine can be used for output to the Ramtek while in any mode using block transfers. Parameters required by BLOKZ are

the control mode, starting element and line coordinates, colors to be used and a buffer containing physical device, logical unit number, number of bytes of data and the actual data. BLOKZ places more information concerning the number of commands to be executed in the buffer and then calls ARAMZ. ARAMZ puts all the information for an I/O to the Ramtek in a single array and then calls RAMZ. Inputs to RAMZ, the routine which calls the operating system routines GETADR and WTQIO in order to output the data on the screen, are a buffer containing the Ramtek operations, the logical unit number, the number of bytes of data involved and the desired functions.

4.2 Dicomed Hardware and Commands

The Dicomed model D47 image recorder is based on high-performance cathode-ray tube (CRT) technology [5]. A simplified diagram of the film recorder is shown in Figure 4.3. The image area of a CRT is projected through a lens and a colored filter and focused on the film. The CRT beam intensity level is related to an exposure index which is manually entered for the type of film in use. Each data element received during the recording process consists of a digital value proportional to the desired exposure. This number is entered in the exposure register via the I/O interface. When the beam is positioned to the desired location it is turned on for a length of time proportional to the exposure value stored in the exposure register.

The recording along a given line (horizontal) is achieved by incrementing the horizontal counter after each point is exposed until an end-of-the line command is received. This command causes the beam to be positioned at the beginning of the next line. The recording of an image is complete when the desired number of lines have been exposed through each filter.

Commands are sent to the Dicomed film recorder through the DICOMD and CINE subroutines. DICOMD accepts a list of commands and sends these commands one at a time to CINE, the routine which builds the calling sequence for the operating system I/O routines GETADR and WTQIO. There are five functions available when using CINE: write commands, read status, attach Dicomed, detach Dicomed and write data.

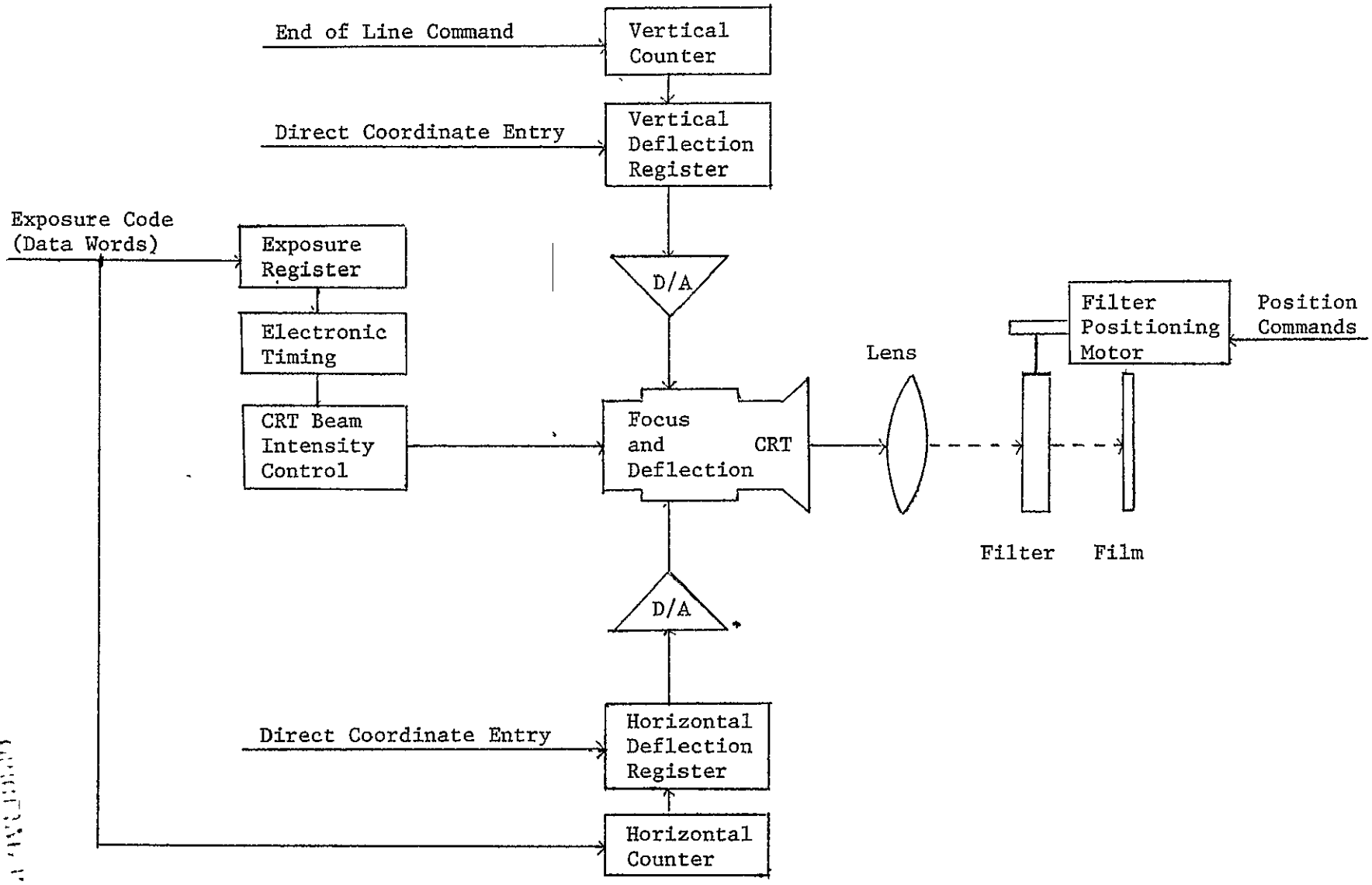


FIGURE 4.3
Dicomed Hardware Diagram

A set of operations along with their functional type is sent to the film recorder through CINE. The Dicomed opcodes and functions are summarized in Figure 4.4.

Before data is output on the Dicomed, the image recorder must be initialized through the initialize command (op code 000). The Dicomed is then ready to receive further commands. Resolution may be set to low (014 - a 4 x 4 point matrix is output for each pixel received), medium (015 - 2 x 2) or high (016 - one point per pixel). This allows an image to be magnified without further software programming. The desired polarity, i.e. whether the image exposure data is to be recorded in uncomplemented or complemented form (201, 202), may also be chosen. Input format of six or eight bits must also be selected (005, 006). In 6 bit input format, the six least significant bits of each exposure data word are interpreted as a value in the range 0 to 63, whereas in 8 bit format, exposure data is interpreted as being of intensity between 0 and 255. A linear or logarithmic transfer function may also be chosen (211, 212). Selection of a logarithmic function means that film transmittance will be linearly proportional to the log of the exposure values while a linear function means that film transmittance will be linearly proportional to the exposure values. A filter of the desired color is positioned in the optical path through command 230 (neutral filter), 231 (red filter), 232 (green) or 233 (blue). The FILTER subroutine is used for filter selection. Starting position on the film may be set through the random horizontal (010) and random vertical (011) commands.

At this point the start-of-input command (001) must be sent to prepare the image recorder for the input of data. After each line of data is sent to the Dicomed, an end-of-line (003) command is sent to reposition the cathode-ray tube beam for the next line of plotting. After all lines have been transmitted, an end-of-input (004) code causes the recorder to recognize an end-transmission operation.

4.3 Varian Commands

The Varian Statos 4115 electrostatic printer/plotter is a black and white hardcopy output device. Only three functions are used with

Command	Code	Function
External Initialize	000	prepare for operations
Start of Input	001	prepare for data input
End of Line	003	position cathode ray tube beam to left margin
End of Input	004	recognize end of transmission operation
6-Bit Select	005	least significant six bits of word interpreted as exposure energy value
8-Bit Select	006	eight bits of data word interpreted as exposure energy value
Random H Position	010	prepare to receive horizontal positioning
Random V Position	011	prepare to receive vertical positioning
Low Resolution Select	014	record at low resolution (sixteen points per pixel)
Medium Resolution Select	015	record at medium resolution (four points per pixel)
High Resolution Select	016	record at high resolution (one point per pixel)
Polarity Normal Select	201	record input exposure data in uncompleted form
Polarity Complement Select	202	Record input exposure data in complemented form
Linear Select	211	select linear steps in transmissivity
Log Select	212	select linear steps in density
Filter Select 1	230	positions neutral filter in optical path
Filter Select 2	231	position red filter in optical path
Filter Select 3	232	position green filter in optical path
Filter Select 4	233	position blue filter in optical path

FIGURE 4.4

Dicomed Instructions

the Varian. These three functions are paper slew, form feed and data write. The subroutine PLOTV forms the interface between the functional primitive level and the operating system. PLOTV prepares a parameter block containing the actual memory address of the data buffer and the number of bytes to be output. The mode (slew, form feed or data write) is set and all this information is passed to the operating system through the WTQIO subroutine.

Slew and form feed are similar operations causing the paper in the plotter to advance. Form feed causes an advance of about four inches and is useful when doing several plotting operations in a sequence. Slew causes the paper to advance about ten inches, allowing the last line which was printed to be seen. Paper advancement can also be done manually. Data write causes a buffer of the specified length to be read as data and printed on the Varian. Each bit is interpreted as a print (1) or skip (0) signal to the printer. 1408 points are plotted across each line.

4.4 Line Printer and User Terminal

Output is directed to the line printer through the FORTRAN WRITE statement and is formatted through the use of a FORMAT statement. Output can also be directed to the user terminal by specifying the user terminal as the output device in a WRITE statement.

4.5 Summary

The routines included at this level are shown in Figure 4.5. These routines make it possible for routines at a higher level to perform display functions without knowing the specific commands associated with the display devices.

Device	Routines	Function
Ramtek	BLOKZ	Set control mode, starting element and line coordinates, colors to be used, physical device, logical unit, number of bytes of data, actual data, commands for Ramtek display generator.
	ARAMZ	
	RAMZ	
Dicomed	DICOMD	Send Dicomed operations and their functional type to the film recorder through operating system I/O routines.
	CINE	
	FILTER	
Varian	PLOTV	Send address of data buffer and number of bytes to be output to printer/plotter.
Line Printer and User Terminal	WRITE statement (FORTRAN I/O)	Direct output to line printer or user terminal.

FIGURE 4.5

Low Level Routines of the Device Module

5. Chapter V, Functional Primitives

At the next level in the device module hierarchy appear the routines which support the functional primitives. These routines represent the capabilities of a graphic output system. Seven functional primitives are considered in this chapter: beam movement, point drawing, line drawing, solid drawing, character display, erasure, and image scan line display. Most devices are capable of performing these functional primitives, but the method of implementing them on the various devices may differ.

5.1 Beam Movement

Beam movement involves initializing or reinitializing the location for display. On the Ramtek, beam movement is accomplished through the DRWABZ routine. By calling DRWABZ with the desired beam coordinates and dark (0) as the color, the beam position is changed from the current operating point (COP) to a new COP. DRWABZ uses RAMZ to set the graphic vector mode and send the LEX, LLX, LE2, and LL2 commands for repositioning. On the Dicomed, the beam is positioned by using the RANDHV routine. Arguments of RANDHV are the desired x and y film coordinates. RANDHV calls CINE to send the positioning commands to the device. The Varian printer/plotter starting x and y coordinates may be changed by instructing the printer to offset each line of points to be plotted. However, this was not implemented, so the appearance of offset may be given by zero filling the desired number of bits at the beginning of each print line buffer area.

5.2 Point Drawing

Point drawing is the basic display operation through which most other graphic entities are derived. Point drawing can also be useful by itself, for example in marking points designated by the user through the Ramtek trackball. Points may be displayed on the Ramtek screen by a sequence of two calls to DRWABZ - one to relocate the COP at the point and one to draw a line segment of length zero from that point to the same point in

the desired color. Point display might also be obtained by using the graphic element mode, but this was not implemented as individual points are seldom drawn. Single point plotting on the Dicomed is not used, but each image recorded on the film is actually just a series of small dots. This is also true of the Varian. Points are plotted on the Varian by setting bits to 1 in a buffer sent to PLOTV along with a write data function specification.

5.3 Line Drawing

Line drawing may be performed on any graphical device, but we did not implement non-vertical, non-horizontal lines on the Dicomed and Varian, as each point on the line must be software defined. Line drawing on the Ramtek is accomplished through a call to RAMLN. RAMLN consists of two calls to DRWABZ, one to move the beam from its current position to the first endpoint of the line and the other to draw the line in the specified color, between the two endpoints. Horizontal or vertical lines may be drawn on the Dicomed and Varian by assigning the proper values to all points in a line of transmitted data (for horizontal line drawing) and by assigning certain values to corresponding points in each line of data transmitted (for vertical lines). Skew lines could be drawn by calculating slope and assigning values according to line number, pixel number and slope.

5.4 Solid Rectangle Drawing

Solid rectangle drawing is a primitive function which is useful when drawing histograms or spotlighting color functions. RECTZ is a subroutine that may be used for drawing rectangles of arbitrary size and color on a Ramtek screen. RECTZ calls RAMZ, setting graphic cartesian mode and using LE1, LL1, LE2, LL2 to define the corners of the rectangle. Raster data mode may be used when rectangles of small size (i.e. 1 X 1) are desired. This mode is used when drawing a joint histogram, which also uses the BLOKZ routine rather than calling RAMZ directly. For drawing rectangles with the film recorder, RECTFR is used. When given the starting coordinates, size, resolution, filter color and intensity for a desired rectangle, RECTFR

calls DICOMD, RANDHV, FILTER and CINE in order to output the rectangle. Solid rectangles may also be drawn on the Varian, but there is no separate routine to handle this operation.

5.5 Character Display

Character display is another functional primitive available on most devices either through hardware or software programming. The Ramtek is equipped with hardware character generation facilities which may be accessed by using the alphanumeric data mode. The ALPHAZ subroutine accepts a string of characters packed two to a word, and through a call to BLOKZ, outputs this character string in the desired color and at the desired position on the Ramtek screen. Currently, software character generation is required on the Dicomed. FRLIST is a subroutine which, given the required parameters of positioning, length of character string, magnification, resolution, character color and background color, writes a character string on the film. FRLIST writes the character string in normal polarity then creates a background by writing the character string in the complement mode in the specified background color. FRLIST calls FRKHAR which is the routine which interfaces with the lower level routine DICOMD thus causing output of the characters. KGEN is the routine used to generate the character dot matrix patterns. Each character is a dot pattern within a 10 X 9 dot matrix. Character size may be enlarged by using medium or low resolution or by using a magnification factor greater than one. Software character generation is also necessary when using the Varian. This character generation is achieved by using the subroutine VARPC. Each character for the Varian consists of an eight by nine dot matrix. VARPC translates the ASCII character codes to the proper dot format, fills a buffer with these dot codes, and calls PLOTV to plot the dots on the paper.

5.6 Erasure

Erasure is another function which applies to all devices. For the Ramtek, erasure is handled through a call to ERASEZ which, through RAMZ, causes the ERS command to be issued. Selective erasure of color is possible, as ERASEZ also uses the SSC command in RAMZ. Erasure of hardcopy devices such as the Varian and Dicomed is interpreted as setting a new

page for output. Since we were concerned primarily with the use of polaroid film on the Dicomed, new film had to be manually inserted. However, if roll film were to be frequently used, instructions for advancing the film through software programming could be included as part of the erase procedure. When using the Varian, after each line is printed the paper is automatically moved in preparation for the next line of printing. In order to have greater spacing between plotted images, the form feed command is sent through PLOTV.

5.7 Image Scan Line Display

Another primitive function available on graphic display devices is display of an image scan line. An image scan line results from reading and, if necessary, transforming all the pixel values across one line of a digitized image. Where software character generation is required, a string of characters may be considered an image so the character drawing is actually a display of several image scan lines.

Image scan line display is achieved on the Ramtek through use of the BLOKZ routine. The complex data control mode is set as is byte format (8 bits per pixel). The Ramtek reads the byte data and displays the pixels according to their color bit format. In order to have all color subchannels available, the color parameter in the BLOKZ command is set at 14 (1110_2).

On the Dicomed, image scan line display requires calls to the CINE subroutine. If more than one color is desired in the final image photograph, each scan line must be written up to three times; once with a red filter, once with green, and once with blue. As each scan line is read in, for each pixel the bits corresponding to the color being displayed must be extracted and transformed into intensities to be sent to the film recorder. CINE is then called with a buffer holding these color pixel values. The number of lines actually recorded on the film for each call to CINE depends on the resolution for which the recorder was set.

Scan line display on the Varian consists of plotting points across one or more lines of the paper. All points are either black or white (blank), with no grey level shading, so various dot patterns must be used to give the final image the appearance of varying grey levels. These patterns are in the form of m x n dot patterns corresponding to the various

pixel values. In our case, 4 X 4 or 5 X 5 dot patterns are available, with 8 different grey levels in the 4 X 4 pattern and 16 in the 5 X 5. Each pixel value must be tranformed to the desired dot pattern before a scan line is sent to PLOTV. M lines must be plotted for each scan line of the image.

5.8 Summary

The functional primitives form an integral part of the database operations included in the device module. They represent the basic display functions required for design and implementation of more complex graphic output. Figure 5.1 summarizes the availability of these seven functional routines and the routines used in implementing them.

Functional Primitive	Devices	Routines	Other Routines Required
Beam movement	Ramtek	DRWABZ	RAMZ
	Dicomed	RANDHV	CINE
Point drawing	Ramtek	DRWABZ	RAMZ
	Varian	PLOTV	
Line drawing	Ramtek	RAMLN	DRWABZ
Solid rectangle drawing	Ramtek	RECTZ	RAMZ
	Dicomed	RECTFR	DICOMD, RANDHV, CINE
Character display	Ramtek	ALPHAZ	BLOKZ
	Dicomed	FRLIST, FRKHAR, KGEN	DICOMD
	Varian	VARPC	PLOTV
Erasure	Ramtek	ERASEZ	RAMZ
	Varian	PLOTV	
Display image scan line	Ramtek	BLOKZ	
	Dicomed	CINE	
	Varian	PLOTV	

FIGURE 5.1

Functional Primitive Implementation

6. Chapter VI, Database Operations

The two levels of the device module hierarchy discussed in chapters four and five are necessary in order to support the main action of the project - performing the requested database operations and showing the results to the user. After the query module determines which operation the user desires, device independent questions are asked, a few preliminary calculations may be performed, and then the query module invokes the device module routine for that operation. Thus the highest level routines in the device module form the interface between the query module and the functional primitives of the device module.

These interface level routines are still device independent. In them, any device independent operations such as statistical calculations and obtaining information about the file contents are completed. At this time, a check is made to see if the operation and the specified device are compatible. If not, control returns to the query module where an error message is given to the user. If the operation is compatible with the specified device, routines dependent on that device are called. As a first step in these routines, any necessary device dependent information is requested of the user. This is desirable in order to more fully utilize the capabilities of each device. Device dependent questions may include inquiries as to color, positioning, magnification, or other factors. Explanatory information for the user may also appear at this time.

The next step is to compile the data into proper form for the requested database operation. This may involve extracting data values, selecting colors and transforming bit patterns. After the output data is prepared, the routines in the two lowest levels (chapters four and five) are called in order to complete the database operation.

Query language commands invoking device module routines fall into four categories. A definition command which uses the device module is build window (BW). Display commands using the device module are display (DI), erase (ER) and exhibit pixel area (EP). Statistical operations supported by the device module are exhibit histogram (EH), exhibit

distribution (ED), exhibit joint histogram (JH), exhibit joint distribution (JD), and exhibit contingency matrices (CM). A utility operation supported by the device module is spotlight (SP). These database operations and their corresponding top level device module routines are listed in figure 6.1.

Database Operation		Device Module Routine
Type	Name	
Definition	BW	GETPT
Display	DI	DISPL, DRWIND, LEGEND
	ER	ERASES
	EP	EXHBPA
Statistical	EH	HISTO
	ED	DISTR
	JH	JHISTO
	JD	JDISTR
	CM	CONMAT
Utility	SP	SPOTLT

FIGURE 6.1

Top Level Device Module Routines

6.1 Definition Operation

Build window is a definitional database operation which allows the user to construct a window through interaction with the query module. If the user chooses to build a window using cursor rather than absolute mode, he selects window vertices by using the trackball. A call to GETPT from the query module is required to obtain each point input through the trackball and also to indicate when no more points are to be entered.

6.2 Display Operations

Five display operations resulting from three query language commands are controlled by the device module. Image display, window display and legend display are all initiated by the query language display command. Erasure is requested through the erase command. Exhibit pixel area is the fifth operation in this display category.

One common area, labelled as the CURRENT common block, is used by the query module and the device module for retaining information concerning the current state of each of the Ramtek screens. If an image is present on a screen, the scale, positioning and name of the image are retained in CURRENT. If no image, or an unmeaningful image is presently showing on the screen, the 'erased' indicator in CURRENT is set. This information may be used when displaying a window or a portion of an image.

6.2.1 Image Display

When an image is to be displayed, the query module must determine the image file name, display device, color function, if any, to be applied to the image, and starting scan line and scan element of the image portion to be displayed. These parameters are then sent to routine DISPL. In DISPL, the file module core table entry routine GETCTE is called to obtain the size of the image in pixels east to west and north to south. [6] Control then flows to the second level device dependent user interaction routines. These routines are RAMQ, DICOQ and VARQ for the Ramtek, Dicomed and Varian, respectively. After obtaining device dependent information, these routines return control to DISPL. DISPL then calls the third

level routine RAMD, DICOD, or VARD.

RAMQ first determines whether the user wishes to use the trackball or the user terminal to input the starting display coordinates. In either case, the user inputs one point. RAMQ then determines, using the image size, screen size and starting screen coordinates, whether the image will fit on the screen. If not, the user may request either that the image be compressed to fit or that only the northwest corner be displayed. Whatever the result, RAMQ then calculates the compression ratio and the size of the image to be displayed. Before returning these parameters to DISPL, RAMQ also must store information concerning this to-be-displayed image in the CURRENT common area for use by the query module.

DISPL then passes the parameters determined in RAMQ to RAMD. RAMD invokes the file module routine which applies the color function and compression factor to the image file and creates a new temporary image file whose name is returned to RAMD. Using the file module, each scan line of the new image is read in and RAMD causes each scan line to be displayed by calling the functional primitive routine BLOKZ until the screen is full or the image is completed, at which time control returns to DISPL and then to the query module.

When the Dicomed film recorder is requested as the display device, DICOQ must obtain several parameters from the user. Besides selecting the film coordinates at which the display is to start, the user must also select how much software image magnification he wants, what hardware resolution he wants, how many different colors may appear, whether the maximum pixel value is 63 or 255, whether the display is to appear in normal or complemented intensities, whether intensity is to be a linear or logarithmic transfer function and which filters are to be used. Before returning to DISPL, DICOQ also calculates the size of the image to be displayed.

DICOD, which receives the parameters obtained in DICOQ from DISPL, calls the file module compression and coloring routine in the same manner as RAMD. For each desired filter, DICOD must then call the functional primitive routines CINE and DICOMD and activate device commands in order to prepare the film recorder for receiving data and outputting it in the

correct format. DICOD then reads each scan line and outputs it through further calls to CINE and DICOMD. This functional primitive operation is repeated until all scan lines have been displayed.

-- Images displayed on the Varian printer/plotter appear as a matrix of black dots and null dots. Grey-level shading is accomplished by having small dot pattern matrices corresponding to various pixel values. Two patterns were implemented - one a 4 X 4 matrix giving eight different grey levels and the other a 5 X 5 matrix giving 16 levels. VARQ asks the user to choose one of these patterns. Then using the image dimensions and pattern size, the amount of paper required for plotting the image is determined. Compression is not necessary on the Varian, as a large image may be plotted in several vertical sections. The user may specify the number of sections to be plotted. Selection of fewer strips than required to plot the entire image is equivalent to requesting display of the western portion of an image. The size of the image to be plotted is then calculated before control returns to DISPL. DISPL then passes the pattern size and image size to VARD.

VARD is divided into two sections, one for displaying eight grey levels using the 4 X 4 dot matrix pattern and one for displaying 16 grey levels using the 5 X 5 pattern. For either pattern choice, the resulting image is framed by a dark border. For the image display, scan lines are read in one at a time, each pixel value is converted to the corresponding dot matrix, and the scan line is displayed through a call to PLOTV.

6.2.2 Legend Display

Display of a legend is an option included in the image display operation at the query level, but is considered a separate database operation at the device module interface level. After an image has been displayed, the user is allowed to enter a legend, which is a string of 80 or fewer characters. The query module counts these characters and then sends them to the device module routine LEGEND. LEGEND sets the character size for each device and calls the device dependent character writing routines RAMPC, DICOPC and VARPC.

When using a Ramtek, the user is allowed to input a point, through the trackball or the terminal, at which to begin the character display. RAMPC checks to see if the characters will fit below and to the right

of this point and prints as many as will fit by calling the functional primitive ALPHAZ.

The user may also select the character display position for the film recorder. DICOPC sets the character size and color as well as determining the size of the area in which the characters are to appear. DICOPC prints a maximum of two lines of characters and returns a status code indicating that not all characters were printed if such is the case. In order to print the characters DICOPC calls FRLIST.

For legend display on the Varian, the user is given no choice of positioning. Each legend is left-justified starting on the next available line on the paper. The VARPC subroutine encompasses three levels of the device module hierarchy. It transforms the ASCII characters to their octal values, determines the dot pattern for these characters and puts these patterns in a buffer, then calls the device command level routine, PLOTV.

6.2.3 Window Display

Another display operation available is drawing a window. Window drawing is only implemented on the Ramtek. The first action in DRWIND is to get the header information about the window file - number of vertices in the window and X and Y minima and maxima. Next the current status of the specified Ramtek screen, stored in the CURRENT common area, is checked to see if there is any image displayed on that screen. If so, the window mode flag is set to one to indicate that the vertices of the window should be drawn to correspond to the coordinates of a currently displayed image. Otherwise, the window mode flag is set to zero. DRWIND then determines the scaling and offsetting factors for the window vertices relative to the screen size and, if applicable, to the image currently displayed on the screen. These factors, along with the number of points in the window and the window mode, are passed to RAMDW, the device dependent window display routine.

In RAMDW, the user is asked which color he wants the window displayed in so if an image is currently on the screen, he can pick a color which will show up on the image. The scaling and positioning factors are then applied to the window vertices to convert them to screen coordinates. These screen

coordinates are then sent two at a time to RAMLN, the line drawing primitive. If the window is not drawn relative to a currently displayed image, the user may request that the window coordinates be displayed so that he has some idea of the actual window size and orientation. These coordinates are encoded to character form and then displayed through ALPHAZ.

6.2.4 Erasure

As previously described, erasure is a functional primitive. It is also a database operation handled by the top level routine ERASES. Allowing the user to suppress erasure permits him to display more than one image on the display surface at the same time.

6.2.5 Exhibit Pixel Area

Exhibit pixel area is an operation used to display the exact pixel values of the pixels in a specified area of a currently displayed image. After displaying an image on a Ramtek screen, the user designates a point on the image and specified how many elements of how many lines south and east of this point he wishes to know the pixel values for. Within the query module, the current image name, size of the area to be exhibited and the image scan element and scan line corresponding to the northwest corner of the area must_be_determined. These parameters are then sent to EXHBPA. This routine reads in the scan lines containing the desired pixels, extracts the designated pixel values, and outputs them on the selected device. For the user terminal and line printer, the WRITE statement is used. For the Ramtek, the values must be encoded to character form and then displayed using ALPHAZ.

6.3 Statistical Operations

Several statistical operations are supported by the IMDB system. These involve computations as well as graphical output. Some of them compare qualities of two images. Because of the type of output required, fewer devices are usually available for displaying the results of each of these statistical operations.

6.3.1 Exhibit Histogram and Exhibit Distribution

Exhibit histogram and exhibit distribution both involve computing the

frequency of occurrence of every pixel value found in an image. The routines in which these calculations take place are HISTO and DISTR. After all pixel values have been tallied, the frequency count array is sent to the device dependent routine. When a histogram on a Ramtek is desired, RAMHI receives the count array from HISTO. RAMHI determines the color in which the histogram is to be output and then passes this color and the count array to DHISTZ. DHISTZ determines the maximum frequency and the largest pixel value in the image in order to scale the histogram to fit the screen. The histogram is displayed as a sequence of rectangles whose heights indicate frequency of occurrence. These rectangles are drawn through calls to the RECTZ functional primitive routine. Included in DHISTZ is code to label the X and Y axes of the histogram through calls to ALPHAZ.

For a distribution, the device is expected to be the line printer and LPDT is the program which is called. LPDT prints out, using the WRITE statement, the number of occurrences of each pixel value within the image.

6.3.2 Exhibit Joint Histogram and Joint Distribution

Exhibit joint histogram and exhibit joint distribution also differ mainly due to the devices on which they can be displayed, as joint histograms may only appear on the Ramtek while joint distributions may only be directed to the line printer. Both JHISTO and JDISTR calculate the frequency of the pixel value pairs obtained by extracting points from the same position on two images. Before the pixel values can be extracted, the northwest corner coordinates and the sizes of the images must be obtained from the file headers. These are used to determine what portion of each image is included in the intersection of the two images.

When first created, images are declared to be in either grid coordinates, with a specified northwest corner, or in absolute coordinates with an arbitrary 'don't care' northwest corner. If two images are both in absolute coordinates, their northwest corners are assumed to have the same coordinates. If one is in absolute and the other in grid coordinates, the northwest corners are also assumed to match up. If both are in grid coordinates, the northwest corner of the overlapping portion and its relative position in each image must be determined.

After the starting and ending scan element and scan line within each image have been determined, one scan line from each image is read. Each pixel value from the first image is concatenated with the corresponding pixel value from the second image and a frequency count of these pixel pair values is tabulated. Scan lines are repeatedly read in and their pixel values extracted until all in the intersecting image portion have been examined.

After this point JHISTO and JDISTR differ, as the device dependent display portion of each routine has been reached. When a joint histogram has been requested, the pixel pair values are sorted in order of ascending frequency. Pixel pair occurrence is shown on a joint histogram by a small rectangle at the point where the X coordinate is the value in one image and the Y coordinate is the value in the other. Frequency is indicated by the color of the rectangle. So that the user may highlight certain frequencies, he is given two options. His first option is choosing the ranges into which he wants the frequencies divided. If he does not use this option, seven equally spaced ranges will be assumed. His second option is selecting a color for each range. Default colors, a different one for each of up to seven ranges, will be assigned if he does not choose this option. After colors and ranges have been assigned, the subroutine JOUT is invoked for each range. JOUT calls BLOKZ using raster data mode to output squares of color on the Ramtek.

When exhibiting a joint distribution, the user may specify whether he wants the output sorted by pixel pair value or frequency, but pixel pair value should only be chosen if all values are less than 128 as larger numbers may cause the pair value to be interpreted as a negative number which would be incorrectly sorted. The pixel pairs and their frequencies are then output to the line printer through repeated use of the WRITE statement.

6.3.3 Exhibit Contingency Matrices

CONMAT is the top level device module routine for computing contingency matrices for two classification maps. The computations take place in CONTIN and the line printer output is handled in CONMXP. Contingency matrices may be used in determining the degree of correspondence between two maps of the same region which have been processed using different algorithms.

Two things are computed when producing contingency matrices - boundary transition type and neighborhood agreement between two maps. Boundary transition type refers to comparison of two pixel values within the same image. The class value of each pixel is compared with its north neighbor and its west neighbor to see which of four boundary transition types it falls into. All three pixels may be in the same class (no boundary), the west neighbor may be the same but the north neighbor different (vertical boundary), the north neighbor may be the same but the west neighbor different (horizontal boundary) or both neighbors may differ from the pixel currently being examined (vertical/horizontal boundary).

To determine the neighborhood agreements between two maps, three comparisons must be made for each pixel. Agreement and disagreement of the corresponding pixels from two classification maps, their north neighbors and their west neighbors is computed. The number of neighborhood agreements for each classification pair with respect to each boundary type is recorded.

6.4 Utility Operation

Of the control and utility database operations available, spotlight is the only one allocated to the device module. Spotlight is a command the user gives when he wishes to know the nature of the data in a certain file. An IMDB file can be an image, a window, a zoom function, a color function, or a transform. At the query level, the user is required to state the name of the file he wishes to spotlight and the device on which he wants the spotlight information to appear. These two parameters are then passed to SPOTLT.

Within SPOTLT, the file type and whether the file is a permanent or temporary file are determined. Depending on the file type, the correct type-dependent routines are called; image - SPIMAGE, window - SPWIND, zoom - SPZOOM, transform - SPTRSF or color - SPCOLR.

SPIMAGE prints out, on either the user terminal or the line printer, the image file name, the northwest corner coordinates of the image, the size of the image in pixels east to west and north to south, and, if the file is a permanent file, the file description. SPWIND prints out the window file name, the maximum and minimum X and Y coordinates, whether the window is an enclosure or an exclosure, and how many vertices the window has. If the file

is a permanent file, the file description is also printed. SPZOOM prints the two numbers forming the new size to old size ratio as well as the file name and description.

SPTRSF prints the file name and description as well as the entire transform table. This indicates to the user which pixel values are transformed into which new values when that transform is used. SPTRSF also only appears on the user terminal or line printer. SPCOLR is more complicated because a color function spotlight may be displayed on the Ramtek or the Dicomed as well as on the user terminal or line printer. SPCOLR calls the file module routine which puts the color table in a designated buffer before passing control to the device dependent routines. For the Ramtek, RAMSC determines which colors are used in the color function and in which pixel value ranges they occur. RAMSC then calls the functional primitive routines RECTZ and ALPHAZ to output a square of each color and the ranges of pixel values that will acquire this coloring when the color function is applied. The Dicomed routine, DISOSC, also determines the colors and ranges and then invokes the functional primitive rectangle and character routines RECTFR and DICOPC. If the line printer or user terminal is specified, UTSC is used to output either the color name or number and the ranges to which it is to be mapped.

REFERENCES

[1] Newman, W.M., and Sproull, R.F., Principles of Interactive Computer Graphics, McGraw Hill, 1973.

[2] Haralick, R.M. and Currier, P., "Image Discrimination Enhancement Combination System (IDECS)," Computer Graphics and Image Processing, Vol. 6, 1977, pp. 371-381.

[3] Lien, Y.E. and Schroff, R., "An Interactive Query Language for an Image Database," University of Kansas Dept. of Computer Science, 1977.

[4] Ramtek GX-100B Programming Manual, 1975.

[5] Operation and Programming Manual, Dicomed Image Recorders, Dicomed Corporation, 1974.

[6] Dijkstra, E.W., "A Constructive Approach to the Problem of Program Correctness," BIT, Vol. 8, No. 3, 1968, pp. 174-186.

[7] Bohm, C. and Jacopini,G., "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules," Comm. ACM, Vol. 9, 1966, pp. 366-371.

[8] Mills, H.D., "Top Down Programming in Large Systems," Debugging Techniques in Large Systems, Courant Computer Science Symposium, NYU, 1971, pp. 41-45.

[9] Mills, H.D., "On the Development of Large Reliable Programs," IEEE Symposium on Computer Software Reliability, 1973, pp. 155-159.

[10] Kerighan, B.W. and Plaugher, P.J., Software Tools, Addison Wesley, 1976.

[11] Fish, R.C., "Structured Design Ensures High Ouality Systems." Computer World.

APPENDIX

Description of Subroutines Included in the Device Module of the IMDB System

DISPL (DEVICE, FILENM, SCANEL, SCANLI, COLFTN, STATUS)

DEVICE = device on which image is to be displayed - Ramtek, Dicomed, Varian.
FILENM = name of image file.
SCANEL, SCANLI = northwest corner to display.
COLFTN = name of desired color function.
STATUS = 0: okay-image displayed.
= 1: illegal device for display.
= 2: image was compressed.
> 4: error from file module routine.

Routines called:
GETCTE - file module
RAMQ
RAMD
DILOQ
DILOD
VARQ
VARD

DISPL call device dependent routines for displaying an image.

RAMQ (DEVICE, FILENM, SCANEL, SCANLI, EW, NS, NELEM, NLINE, JX, JY, OLDCF, NEWCF, STATUS)

DEVICE = specific Ramtek screen on which to display image.
FILENM = name of image file.
SCANEL, SCANLI = starting northwest corner to display.
EW, NS = pixels, lines in image.
NELEM, NLINE = number of elements and lines to display.
JX, JY = screen coordinates for northwest corner of image.
OLDCF, NEWCF = compression factors: old size and new size.
STATUS = 0: okay.
= 2: image was compressed.

Routines called:
MSGOUT - utility
GETPT

RAMQ determines image display size, positioning and compression.

RAMD (DEVICE FILENM, SCANEL, SCANLI, NELEM, NLINE, COLFTN, JX, JY, OLDCF, NEWCF, STATUS)

DEVICE = specific Ramtek screen
FILENM = name of image file
SCANEL, SCANLI = northwest corner of image
NELEM, NLINE = number of elements and lines to be displayed.
COLFTN = color function (if any) to be applied to image.
JX, JY = northwest screen coordinates for display.
OLDCF, NEWCF = compression factors.
STATUS = 0: okay
> 0: otherwise - error in file or manipulation module routine.

Routines called:
CCFTN - manipulation module
RDSCLN - file module
MOVE - manipulation module
DVICEZ
BLOKZ

RAMD displayed an image on a Ramtek screen.

DICOQ (FILENM, MAGFR, RES, NKOLOR, NBIT, POLAR, LOGLIN, KOLRCD, SCANEL, SCANLI, EW, NS, NELEM, NLINE, JX, JY, OLDCF, NEWCF, STATUS)

FILENM = name of image file.
MAGFR = magnification factor.
RES = resolution (0,1,2 for low, med, high).
NKOLOR = number of colors (8 or 64).
NBIT = 6-bit or 8-bit intensity values.
POLAR = polarity: normal or complemented exposure.
LOGLIN = logarithmic or linear transfer function.
KOLRLD = color code: specifies which filters to use.
SCANEL, SCANLI = northwest corner to be displayed.
EW, NS = size of image in elements and lines.
NELEM, NLINE = number of elements and lines to display.
JX, JY = film coordinates for northwest corner.
OLOCF, NEWCF = compression factors.
STATUS = 0: okay.

Routines called:
MSGOUT - utility

DICOQ determines parameters to be used by Dicomed film recorder in displaying an image.

DICOD (FILENM, SCANEL, SCANLI, JX, JY, OLDCF, NEWCF, MAGFR, RES, NKOLOR, NBIT, POLAR, LOGLIN, KOLRCD, COLFTN, NELEM, NLINE, STATUS)

FILENM = name of image file.
SCANEL, SCANLI = northwest corner of image to be displayed.
JX, JY = film coordinates for northwest corner.
OLDCF, NEWCF = compression factors.
MAGFR = magnification factor.
RES = resolution (o,1,2 for low, med, high)
NKOLOR = number of colors (8 or 64).
NBIT = number of bits for each intensity = 6 or 8.
POLAR = polarity = normal or complemented.
LOGLIN = linear or logarithmic transfer function.
KOLRCD = color code: designates filters to be used.
COLFTN = color function to be applied to image.
NELEM, NLINE = number of elements and lines in image to be displayed.
STATUS = 0: okay.
> 4: error from file or manipulation module routine.

Routines called:
CCFTN - manipulation module
RDSCLN - file module
MOVE - manipulation module
CINE
DICOMD
FILTER
RANDHV

DICOD displays an image on film using the Dicomed film recorder.

VARQ (FILENM, SCANEL, SCANLI, EW, NS, NELEM, NLINE, NSTRIP, PCODE)

FILENM = name of image file to be displayed.
SCANEL, SCANLI = northwest corner of image.
EW, NS = size of image in pixels east to west and north to south.
NELEM, NLINE = number of elements and lines to be displayed.
NSTRIP = number of strips (image sections) to be printed.
PCODE = pattern code: 4 for 4 X 4 pattern or 5 for 5 X 5 pattern.

Routine called:
MSGOUT - utility

VARQ determines the number of sections of the image to be plotted on the Varian.

VARD (FILENM, SCANEL, SCANLI, EW, NS, NELEM, NLINE, NSTRIP, PCODE, STATUS)

FILENM = name of image file to be displayed.
SCANEL, SCANLI = northwest corner of image
EW, NS = size of image in pixels east to west and north to south.
NELEM, NLINE = number of elements and lines to be displayed.
NSTRIP = number of sections to be printed.
PCODE = pattern code: 4 for 4 X 4 pattern, 5 for 5 X 5 pattern.
STATUS = 0: okay.
= 3: improper pattern code.
< 0 and > -256: bad device status in PLOTV.
> -1000: bad directive status in PLOTV.
> 4: error from file module routine.

Routines called:
RDSCLN - file module
MOVE - manipulation module
PLOTV
PACK85.

VARD displays a grey level image on the Varian printer/plotter.

PACK85. (BUF, BUF1)
BUF - input buffer, 2 bytes per word.
BUF1 - result buffer, byte array.

Routines called: none

PACK85 packs the right most 5 bits of the 8 words of BUF into 5 bytes of BUF1. The packing results in the following structure: 11111222 22333334 44445555 56666677 7778888.

DRWIND (DEVICE, FILENM, STATUS)

DEVICE = device on which to display window - Ramtek.
FILENM = name of window file.
STATUS = 0: okay
= 1: improper device.
= 3: window won't fit on image on screen.
≥ 4: error from file or manipulation module routine.

Routines called:
GETCTE - file module
MOVE - manipulation module
MSGOUT - utility
RAMDW

DRWIND determines coordinates for drawing a window polygon on the Ramtek screen.

RAMDW (DEVICE, FILENM, DSCALE, DMOVEX, DMOVEY, NPTS, WMODE, STATUS)

DEVICE = specific Ramtek screen on which to display window.
FILENM = name of window file.
DSCALE = scale factor.
DMOVEX = X-offset.
DMOVEY = Y offset.
NPTS = number of points in window polygon.
WMODE = 0 - no current image on screen; 1-image currently on screen.
STATUS = 0: okay.
> 4: error from file module routine.

Routines called:
RDWNDW - file module
MSGOUT - utility
RAMLN
ALPHAZ

RAMDW draws a window polygon on a Ramtek screen.

LEGEND (DEVICE, BUFF, NCHAR, STATUS)

DEVICE = device on which legend is to appear - Ramtek, Dicomed, Varian.
BUFF = buffer containing characters to be displayed.
NCHAR = number of characters to be displayed.

STATUS = 0: output okay.
= 1: illegal device.
= 3: character string was truncated because characters didn't all fit in display area.

Routines called:
RAMPC
DICOPC
VARPC

LEGEND displays a character string. Character color is white on Ramtek and Dicomed, dark on Varian.

RAMPC (DEVICE, BUFF, NCHR, CHRSZ, STATUS)

DEVICE = Ramtek screen to be used for character display.
BUFF = buffer containing characters to be displayed.
NCHR = number of characters in BUFF.
STATUS = 0: okay.
3: not all characters will fit.

Routines called:
MSGOUT - utility
GETPT
ALPHAZ

RAMPC displays characters in white at point specified by user on Ramtek screen.

DICOPC (BUFF, NCHR, CHRSZ, STATUS)

BUFF = buffer containing characters to be displayed.
NCHR = number of characters in BUFF.
CHRSZ = character size: 1,2,3 or 4 - 1 is largest, 4 smallest.
STATUS = 0: okay
3: characters didn't all fit so character string was truncated.

Routines called:
MSGOUT - utility
CINE
FRLIST

DICOPC prints characters on Dicomed film starting at point specified by user.

VARPC (BUFF, NCHR, STATUS)

BUFF = buffer containing characters to be output
NCHR = number of characters in BUFF.
STATUS = 0: okay
< 0 and >-256: bad device status.
< -1000: bad directive status

Routine called:
PLOTV

VARPC prints one line of characters on the Varian printer/plotter. Characters are in an 8 X 9 dot matrix.

ERASES (DEVICE)

DEVICE = device to be erased or set to a new page - Ramtek, Varian, line printer.

Routines called:
DVICEX
ERASEZ
PLOTV

ERASES erases a Ramtek screen, advances the paper on the Varian or goes to the top of a new page on the line printer.

EXHBPA (DEVICE, FILENM, SCANEL, SCANLI, NELEM, NLINE, STATUS)

DEVICE = output device for display of pixel values in a given area - user terminal, line printer or Ramtek.
FILENM = name of image file from which pixel values are to be extracted.
SCANEL, SCANLI = starting image coordinates of area to be exhibited.
NELEM, NLINE = number of elements and likes to be exhibited - must be no greater than 20.
STATUS = 0: okay
= 1: improper device.
> 1: error from file module routine.

Routines called:
RDSCLN - file moudle
MOVE - manipulation module
MSGOUT - utility
DVICEZ
ALPHAZ

EXHBPA displays pixel values of an NELEM by NLINE area starting at SCANEL, SCANLI in image FILENM.

HISTO (DEVICE, FILENM, STATUS)

DEVICE = device on which to display histogram - Ramtek.
FILENM = name of image file for which histogram is to be displayed.
STATUS = 0: okay
= 1: improper device.
> 1: error from file module routine.

Routines called:
GETCTE - file module
RDSCLN - file module
MOVE - manipulation module
MSGOUT - utility
RAMHI

HISTO computes frequency counts used for drawing a histogram of an image.

RAMHI (PARRAY, DEVICE)

PARRAY = 256 word array containing frequency counts for pixel values.
DEVICE = specifies which Ramtek screen.

Routines called:
MSGOUT - utility
DVICEZ
DHISTZ

RAMHI initiates display of a histogram on the Ramtek screen.

DHISTZ (KOUNT, KOLOR)

KOUNT = 256 word array containing frequency count of pixel values.
KOLOR = color in which histogram is to be displayed.

Routines called:
RECTZ
ALPHAZ
DRWABZ

DHISTZ displays a histogram on the Ramtek screen.

DISTR (DEVICE, FILENM, STATUS)

DEVICE = device on which to display distribution of pixel values - line printer.
FILENM = name of image file for which distribution is to be displayed.
STATUS = 0: okay
= 1: improper device.
> 1: error from file module routine.

Routines called:
GETCTE - file module
RDSCLN - file module
MOVE - manipulation module
MSGOUT - utility
LPDT

DISTR computes the frequency of each pixel value in a given image.

LPDT (FILENM, PARRAY)

FILENM = name of image file for which distribution is to be printed.
PARRAY = 256 word array containing frequency counts of pixel values.

Routines called:
MSGOUT - utility

LPDT prints the distribution of pixel values on the line printer.

JHISTO (DEVICE, IMAGE1, IMAGE2, STATUS)

DEVICE = display device for joint histogram - Ramtek
IMAGE1, IMAGE2 = names of image files for which joint histogram is to be displayed.

STATUS = 0: okay.
= 1: improper device.
= 2: too many different pixel pairs - joint histogram not possible.
= 4: images don't intersect.

Routines called:
GETCTE - file module
RDSCLN - file module
MOVE - manipulation moudle
MSGOUT - utility
JSORT
JOUT
ALPHAZ
ERASEZ

JHISTO computes pixel pair values and frequencies in preparation for display of a joint histogram. User may select ranges and colors for display of pixel values.

JSORT (KBUFF, NPAIRS, KEY)

KBUFF = 2 by NPAIRS array of pixel value pairs and frequencies.
NPAIRS = number of different pairs to be sorted.
KEY = indicates whether sorting is to be first or second word.

Routines called: none

JSORT sorts a 2 X N array in ascending order.

JOUT (KOUNT, NAPIRS, KOLAR, MA6)

KOUNT = (2,N) array of pixel value pairs and frequency counts.
NPAIRS = number of data pairs in KOUNT.
KOLOR = color histogram is to be displayed in.
MAG = magnification factor.

Routines called:
BLOKZ
BITSET

JOUT displays a line of the joint histogram on the Ramtek.

BITSET (IARRAY, NBIT)

IARRAY = target array.
NBIT = bit to be set.

Routines called: none

BITSET sets a bit in an array.

JDISTR (DEVICE, IMAGE1, IMAGE2, STATUS)

DEVICE = output device - line printer.
IMAGE1, IMAGE2 = names of image files for which joint distribution is to be displayed.
STATUS = 0: okay.
= 1: improper device
= 2: too many different pairs.
= 4: images don't intersect.

Routines called:
GETCTE - file module
RDSCLN - file module
MOVE - manipulation module
MSGOUT - utility
JSORT

JDISTR computes pixel pair values in preparation for display of a joint distribution.

CONMAT (DEVICE, MAP1, MAP2, STATUS)

DEVICE = output device for display of contingency matrix - line printer.
MAP1, MAP2, = names of image files for which contingency matrix is to be computed.
STATUS = 0: okay
= 1: images not of same size.
= 2: pixel number exceeds limit.
= 3: improper device.
> 3: error from file module routine.

Routines called:
GETCTE - file module
MSGOUT - utility
CONTIN
CONMXP

CONMAT computes contingency matrices of two classification maps by calling CONMAT and prints out the matrices by calling CONMXP. Each map has eight or fewer classes.

CONTIN (MAP1, MAP2, NPIXEL, NLINE, NCLAS1, NCLAS2, IX, IY, MATRIX, STATUS)

MAP1, MAP2 = names of image files (classification maps) for which contingency matrices are to be completed.
NPIXEL, NLINE = number of elements and lines in the ranges.
NCLAS1, NCLAS2 = number of classes in MAP1 and MAP2.
IX, IY = buffers for scan lines.
MATRIX = array for contingency matrix.
STATUS = 0: okay.
≠ 0: error in file module routine.

Routines called:
RDSCLN - file module
MOVE - manipulation module
AGREE

CONTIN computes contingency matrices for two classification maps. Boundary transition types and neighborhood agreements are computed.

CONMXP (DEVICE, MAP1, MAP2, TITLE, IH, IH2, INV1, INV2, NPIXEL, NLINE, M, N)

DEVICE = output device - line printer
MAP1, MAP2 = names of image files for which contingency matrices are to be computed.
TITLE = 80 character (max) title to be printed on top of each page of output.
IH, IH2 = contingency matrices computed by CONTIN.
INV1, INV2 = inventory matrices.
NPIXEL, NLINE = number of lines and pixels in images.
M, N = number of classes in MAP1 and MAP2 respectively.

Routine called:
MSGOUT - utility

CONMXP prints matrices showing number of agreements and disagreements of transitions for each pair of classes.

AGREE (IX, IY, NPIXEL, I1, I2, MATRIX)

IX = array holding two scan lines of one map.
IY = array holding two scan lines of other map.
NPIXEL = number of pixels in each scanline.
I1, I2 = either 1 or 2, tells which part of IX or IY has the present line and the previous line.
MATRIX = contingency matrix.

Routine called:
INTLOG

AGREE compares one pair of scan lines for agreement for the two maps.

INTLOG(L) (function)

L = true or false,
INTLOG returns 1 if true, 0 if false.

SPOTLT (DEVICE, FILENM, STATUS)

DEVICE = device on which to display spotlight output,
FILENM = name of file to be spotlighted.
STATUS = 0: okay
= 1: device incompatible with file type
= 2: file type not recognized
= 4: improper color format for color spotlight on Ramtek.

Routines called:
GETCTE - file module
SPIMAGE
SPWIND
SPZOOM
SPCOLR
SPTRSF

SPOTLT displays core table and file content information.

SPIMAGE (DEVICE, IMGNM, STATUS)

DEVICE = device on which to display information about an image-line printer or user terminal.
IMGNM = name of image file for which information is desired.
STATUS = 0: okay.
= 1: improper device.

Routines called:
RDDSCR - file module
MSGOUT - utility

SPIMAGE prints out coordinates, size and description of an image.

SPWIND (DEVICE, FILENM, STATUS)

DEVICE = device on which information about window file is to be displayed-line printer or user terminal.
FILENM = name of window file.
STATUS = 0: okay.
= 1: improper output device.

Routines called:
MOVE - manipulation module
MSGOUT - utility
RDDSCR - file module

SPWIND prints out closure maximum and minimum coordinates, number of vertices and description of a window polygon.

SPZOOM (DEVICE, FILENM, STATUS)

DEVICE = device on which to display information about zoom function-line printer or user terminal.
FILENM = name of zoom file.
STATUS = 0: okay.
= 1: improper output device.

Routines called:
MSGOUT - utility
RDDSCR - file module

SPZOOM prints name, new to old image size ratio, and description of a zoom file.

SPTRSF (DEVICE, FILENM, STATUS)

DEVICE = device on which to display information about transform function-line printer or user terminal.
FILENM = name of transform file.
STATUS = 0: okay.
= 1: improper output device.

Routines called:
MSGOUT - utility
RDDSCR - file module
RDTNSF - file module
MOVE - manipulation module

SPTRSF prints each pixel value and what value it is transformed to.

SPCOLR (DEVICE, COLFCN, STATUS)

DEVICE = device on which to display color function information.
COLFLN = name of color file.
STATUS = 0: okay.
= 1: improper device.
= 4: color format not compatible with display device.

Routines called:
RDCOLR - file module
RAMSC
DICOSC
UTSC

SPCOLR outputs a square of each color and the ranges in which it occurs on the Ramtek or Dicomed and prints the number or name of each color and its ranges on the line printer or user terminal.

RAMSC (DEVICE, STATUS)

DEVICE = Ramtek screen on which color function is to be displayed.
STATUS = 0: okay.

Routines called:
MOVE - manipulation module
DVICEZ
RECTZ
ALPHAZ

RAMSC draws rectangles on Ramtek to illustrate colors of ranges for a color function.

DICOSC (COLFRM, STATUS)

COLFRM = color format: 4-bit for eight colors, 6-bit for 64 colors.
STATUS = 0: okay.

Routines called:
MOVE - manipulation module
MSGOUT - utility
CINE
RECTFR
FRLIST

DICOSC displays a rectangle of each color used in the color function and the ranges in which it is used on Dicomed film.

UTSC (DEVICE, FILENM, COLFRM, STATUS)

DEVICE = display device - user terminal or line printer.
FILENM = name of color file.
COLFRM = color format: 4-bit for 8 colors, 6-bit for 64 colors.
STATUS = 0: okay

Routines called:
MSGOUT - utility
MOVE - manipulation module
RDDSCR - file module

UTSC displays color name, if 4-bit format, or color number, if 6-bit format, and ranges.

GETPT (DEVICE, X, Y, STATUS)

DEVICE = Ramtek screen from which to obtain point.
X = X-coordinate of point.
Y = Y-coordinate of point.
STATUS = 0: cursor visible-point returned.
= 1: cursor not visible - point not returned.
= 2: improper device.

Routines called:
DVICEZ
KURSRZ

GETPT returns the screen coordinates of a point entered through the trackball by the user.

KURSRZ (JX, JY) (function)

JX, JY = trackball coordinates.

Routine called:
RAMZ

KURSRZ is used to input a point through the Ramtek trackball. KURSRZ returns a value giving the condition of the trackball switeches.

DRPT (DEVICE, X, Y)

DEVICE = Ramtek screen on which to display a point.
X = x-coordinate of point.
Y = y-coordinate of point.

Routines called;
DVICEZ
DRWABZ

DRPT displays a point in white on the specified Ramtek screen.

DRWABZ (JX, JY, KOLOR)

JX, JY = ending coordinates of line.
KOLOR = desired color: 1 - white overlay
2 - red
4 - green
8 - blue

Routines called:
COORDZ
RAMZ
KRAMZ

DRWABZ draws a vector from the point last drawn by this routine to the point specified in the call. The current point is remembered until the next call.

RAMLN (DEVICE, X1, Y1, X2, Y2, COLOR)

DEVICE = which Ramtek screen to draw line on.
X1, Y1 = starting coordinates for line.
X2, Y2 = ending coordinates for line.
COLOR = color in which to draw line.

Routines called:
DVICEZ
DRWABZ

RAMLN draws a line on a Ramtek screen.

RECTZ (KX1, KY1, KX2, KY2, KOLOR)

KX1, KY1 = starting coordinates for rectangle.
KX2, KY2 = ending coordinates for rectangle.
KOLOR = color in which to draw rectangle.

Routines called:
RAMZ
KRAMZ
NVERSZ

RECTZ draws rectangles on the Ramtek using graphic cartesian mode.

BORDER (JX, JY, NPIX, NLINE, JXB, JYB, IRES, KOLOR, INTENS, IBUF, JBUF)

JX, JY = upper left coordinates.
NPIX = picture width.
NLINE = picture height.
JXB = vertical border width.
JYB = horizontal border width.
IRES = resolution (0,1,2 for low, med, high).
KOLOR = COLOR (0,1,2,3 for whiter, red, green, blue).
INTENS = intensity level (0-255).
IBUF = integer array to give border temporary storage.
JBUF = annotation for bottom border.

Routines called
RECTFR

BORDER draws a border around pictures or text on the Dicomed film recorder.

RECTFR (JX, JY, NX, NY, IRES, KOLOR, INTENS, IBUF)

JX, JY = upper left corner of border (scaled).
NX = number of data points across.
NY = number of data points down.
IRES = resolution (0,1,2 for low, med, high).
KOLOR = color (0,1,2,3 for white, red, green, blue).
ITENS = intensity value (0-255).
IBUF = temporary array for use by RECTFR (of size NX+1 words).

Routines called:
DICOMD
RANDHV
FILTER
CINE

RECTFR draws filled-in rectangles on the Dicomed film recorder.

RANDHV (IPOSH, IPOSV)

IPOSH = horizontal position.
IPOSV = vertical position.

Routine called:
CINE

RANDHV positions output through the Dicomed film recorder, permitting plotting to begin anywhere on the film.

ALPHAZ (JX, JY, KOLOR, IARRAY, NCHAR)

JX, JY = screen coordinates for character display.
KOLOR = color.
IARRAY = array containing the characters to be displayed.
NCHAR = number of characters in IARRAY.

Routines called: none

ALPHAZ outputs characters on the Ramtek display screen selected by the most recent call to DVICEZ.

FRLIST (JX1, JY1, JX2, JY2, ISTR, ISIZE, MAG, KX, KY, IRES, KOLOR, KBACK, ITEMP, NCSIZE)

JX1, JY1, JX2, JY2 = coordinates of rectangle to be filled in with backround color (scaled per resolution).
ISTR = buffer containing characters to be printed.
ISIZE = number of characters in string.
MAG = magnification.

KX, KY = coordinates (scaled) of upper left corner of first character. If either is zero, characters will be centered in the rectangle.
IRES = resolution (0,1,2 for low, med, high).
KOLOR = 4 word array containing intensity of characters for the four filter colors.
KBACK = 4 word array giving intensity of backround for the four filter colors.
ITEMP = temporary buffer for output.
NCSIZE = size of field characters are to be spread over. If zero, no extra spaces are added.

Routines called:
FRKHAR
RECTFR

FRLIST places a line of characters and backround on Dicomed film.

FRKHAR (KBUF, NCHAR, MAG, JX, JY, IRES, KOLOR, INTENS, KOMP, ITEMP, NCSIZE)

KBUF = array of characters to be output.
NCHAR = number of characters in KBUF.
MAG = magnification.
JX, JY = film coordinates.
IRES = resolution (0,1,2,3 for low, med, high)
KOLOR = color (0,1,2,3 for white, red, green, blue).
INTENS = intensity (0-255)
KOMP = polarity (0,1 for normal, complemented).
ITEMP = temporary array for output.
NCSIZE = number of character spaces string is to be spread over.

Routines called:
DICOMD
FILTER
RANDHV
KGEN
CINE

FRKHAR places a string of characters on Dicomed film in the color, position and intensity specified.

KGEN (KBUF, NCHAR, MAG, MAGV, IROW, INTENS, ITEMP, NCSIZE)

KBUF = buffer containing characters to be output.
NCHAR = number of characters in KBUF.
MAG = magnification.
MAGV = which row is to be formatted.
INTENS = intensity value for the 'on' bits.
NCSIZE = number of character spaces string is to be spread over.

Routines called:
KFILL
TFILL
SPCL

KGEN fills output buffer with one row of nine row character string pattern.

TFILL (KDATA, MAG, MAGV, INTENS, KPTR, ITEMP)

KDATA = 7 X 3 array with pattern and corner fill
MAG = magnification
MAGV = which row from 1 to MAG is being done
INTENS = value to which ITEMP is set if pattern is 'on'
KPTR = pointer used to fill ITEMP
ITEMP = output buffer.

Routines called: none

TFILL fills the output buffer with one row of one 7 X 9 character pattern..

KFILL (IBUF)

IBUF = 7 X 3 array across a character with a 1 for a character element and a 0 for a blank element.

Routines called: none

KFILL tests for corner fill in character generation.

SPCL (IROW, KHAR, KTEMP)

IROW = row being filled (one of 9 rows).
KHAR = code of character being generated.
KTEMP = 7 X 3 array with bit pattern and corner indicators.

Routines called: none

SPCL sets corner indicators for special cases not handled by KFILL.

ERASEZ (KOLOR)

KOLOR = colors to be erased.
1023 erases all.

Routines called:
RAMZ
KRAMZ

ERASEZ erases a color or colors from the Ramtek screen. Use of a negative color sets the reverse backround bit and causes the color to be written instead of erased.

BLOKZ (MODE, JX, JY, KOLOR, IBUF)

MODE = any Ramtek mode that uses block transfers.
JX, JY = starting screen coordinates.
KOLOR = subchannels for color.
IBUF = buffer containing data and header information.
IBUF(1) = physical device or zero.
IBUF(2) = logical unit or zero.
IBUF(3) = number of bytes of data
IBUF(4-11) = header filled in by BLOKZ
IBUF(12-) = data

Routines called:
KRAMZ
ARAMZ

BLOKZ outputs data in a buffer to the Ramtek using the BLK command and the specified mode.

PLOTV (BUFFER, BUFSEZ, IOFCN, STATUS)

BUFFER = buffer containing data.
BUFSIZ = number of bytes of data in buffer.
IOFCN = function: 0-slew.
1-form feed.
2-data write.

STATUS = 0: output okay.
< 0 and ≥ -256 - bad device status.
< -1000: bad directive status.

Routines called:
GETADR - operating system I/O routine.
WTQIO - operating system I/O routine.

PLOTV performs I/O functions on the Varian plotter.

DVICEZ (IPD, LVN)

IPD = physical device.
LUN = logical unit number.

Routine called:
RAMZ

DVICEZ sets default values for the Ramtek device and unit.

ARAMZ (IBUF)

IBUF(1) = physical device
IBUF(2) = logical unit number.
IBUF(3) = number of bytes of Ramtek commands.
IBUF(4) = function: 1=write
2-read
3-attach
4-detach
IBUF(5-) = Ramtek commands

Routine called:
RAMZ

ARAMZ sends data to the Ramtek from a single array.

RAMZ (IBUF, LUN, NBYTES, IFUNC)

IBUF = Ramtek commands.
LUN = logical unit number.
NBYTES = number of bytes.
IFUNC = desired function: 1-write
2-read
3-attach
4-detach

Routines called:
GETADR
WTQIO
ERRCKZ

RAMZ builds the calling sequence for WTQIO, calls WTQIO to I/O data and commands to the Ramtek, and then does error checking.

KRAMZ (args) (function)

args = Ramtek op codes, modes, flags, colors

KRAMZ is used to stack op codes for Ramtek commands into a buffer for I/O to the Ramtek.

ERRCKZ (ISW, ISTAT, IPRM, MUNIT)

ISW = system status word.
ISTAT = handler status word.
IPRM = parameter list.
MUNIT = unit for error messages.

Routines called: none

ERRCKZ checks system I/O status, handler I/O status and number of bytes input or output.

COORDZ (JX1, JY1, JX2, JY2)

JX1, JY1 = previous coordinates.
JX2, JY2 = new coordinates.

Routines called: none

COORDZ accepts a set of coordinates and returns the set sent over in the previous call.

NVERSZ (KOLOR) (function)

KOLOR = color - if color is negative, reverse backround mode is selected.

Routines called: none

NVERSZ returns the code for reverse backround if the color is negative.

FILTER (KOLOR)

KOLOR = color of filter to position on Dicomed:
0-neutral.
1-red.
2-green.
3-blue.

Routines called:
DICOMD
CINE

FILTER positions a filter in the optical path of the Dicomed film recorder.

DICOMD (IARRAY, ISIZE)

IARRAY = array of commands to be sent to the Dicomed.
ISIZE = number of commands in the array.

Routine called:
CINE

DICOMD sends commands to the Dicomed film recorder.

CINE (IBUF, LUN, NBYTES, IFUNC)
IBUF = buffer containing Dicomed operations.
LUN = logical unit number.
NBYTES = number of bytes of data

IFUNC = desired function: 1-write commands.
2-read status.
3-attach
4-detach
5-write data.

Routines called:
GETADR - operating system I/O routine.
WTQIO - operating system I/O routine.

CINE sends Dicomed I/O operations to WTQIO.

Description of routines invoked by the device module but not included in the device module.

GETCTE: searches the core tables for a given file name and returns the core table information in a common buffer. Core table information includes file type, coordinates and size.

MOVE: moves an arbitrary number of bytes from a specified source address to a specified destination.

CCFTN: compresses and performs a color transform on a given image file and returns the name of a file containing the resultant image.

RDCOLR: reads a color transformation into a common buffer.

RDDSCR: reads the description field of a file header into a buffer.

RDSCLN: reads a scan line from a file into a buffer.

RDTNSF: reads a transformation into a common buffer.

RDWNDW: reads window vertices into a buffer.

MSGOUT: outputs a message on the specified logical unit.

4. IMPLEMENTATION OF THE FILE MODULE

The IMDB File Module provides an interface between the IMDB program structure and the RSX-11D FILES-11 file system, making data management essentially transparent to the other modules of IMDB.

4.1 IMDB DATA STRUCTURES

4.1.1 FILES

The primary IMDB data structure is the file. This is organized as a FILES-11 direct access file having a 256 byte header record followed by one or more 256 byte data records.

The header record of a permanent file consists of six 4-byte fields containing the information corresponding to columns 5 through 10 of the core table entry (see 4.1.2), a 4-byte file size field recording the number of records in the file, and a 228-byte file description field.

A distinciton is made between temporary files and permanent files. For a temporary file, the file header record is not maintained and the file is deleted at the end of the IMDB session. Files may be converted from temporary to permanent by invoking the IMDB SAVE command.

4.1.2 DIRECTORIES

The file directory structures maintained by IMDB are in-core file tables TFCT (Temporary File Core Table) and PFCT (Permanent File Core Table). Only files entered into the core tables are accessible to the user. Directory entries are created by the user ACTIVATE, BUILD, and SAVE commands.

Structurally, the PFCT and TFCT are INTEGER*4 (50,10) arrays. Each of the 50 rows may contain an entry. The first four columns of an entry contain the 16 character file name (left justified, blank filled). Unused entries are completely blank filled- Column 5 contains a file type indicator and columns 6 through 10 contain file specific header information as indicated in the table on the following page.

	CORE TABLE COLUMN					
File Type	5	6	7	8	9	10
IMAGE	'I '	LOQ	LAQ	*	Pixels/ scan line	Scan Lines
WINDOW	'W '	Maximum X & Y coordinates of window**	Minimum X & Y coordianates of window**	Closure code	Number of points in window	*
EXPRESSION	'E '	Number of 16 byte terms	*	*	*	*
TRANSFORM	'T '	*	*	*	*	*
COLOR FUNCTION	'C '	*	*	*	*	*
ZOOM	'Z '	Relative size of resultant image	Relative size of base image	*	*	*
LINE (not Implemented	'L '	X_1	Y_1	X_2	Y_2	*

* indicates field is unused

** MAXX, MAXY and MINX, MINY are packed with the X portion in the low address word and the Y portion in the high address word

4.2 IMDB FILE MANAGEMENT PRIMITIVES

These routines handle all file create, open, delete, read and write operations. Also included in this group are the routines which access and enter core table information.

4.2.1 MISCELLANEOUS PRIMITIVE ROUTINES

MOVE - A MACRO-11 routine which copies (left to right) an arbitrary number of bytes from any memory address to any other address

SEARCH - Attempts to find the core table and row number for a given file name

FDSCRP - Converts a 16 byte IMDB system file name to a null-character terminated FILES-11 file descriptor. If the IMDB system file name does not specify device, directory, type, and version information, the following defaults are assumed:
Device - SY0
Directory - directory under which IMDB is currently running
Type - .DLT
Version - ;1 (in order to avoid proliferation of versions)

DSDSCR - Creates the dataset descriptor information required by the CHKFIL routine

CHKFIL - (module FILAUX.MAC) - Checks whether a file exists by attempting to open it (written in the MACRO-11 language)

4.2.2 THE FILPRM FILE PRIMITIVE ROUTINE

FILPRM performs three different functions - create, open and delete - on the IMDB standard random files. The function is specified by the FCN parameter.

Create calls CHKFIL to determine whether the file exists, then uses the FORTRAN OPEN statement to open the file with TYPE = 'NEW'. The file is then closed before returning. Open uses the FORTRAN OPEN statement to open the file with TYPE = 'OLD'. Delete opens the file, then uses the FORTRAN CLOSE statement to DISPOSE = 'DELETE'.

4.2.3 FILE READ AND WRITE PRIMITIVES

For purposes of reading and writing, IMDB files are viewed as continuous byte strings. Data is transferred to or from a file by calculating the byte offset from the beginning of the file (byte 0) and calling the READP or WRITEP routine.

GETSIZ - Calls SEARCH (see 4.2.1) to determine the row number and core table of the given file, uses this information to find the number of records in the file from the file size tables (PFST and TFST) in the FILCOM area.

PUTSIZ - Calls SEARCH, then updates information in the PFST or TFST. If the file is permanent, the file size field of the file header (bytes 24-27 of the file) are also updated.

READP - Reads the specified bytes of the named file, one random record at a time, unblocking the data into the buffer area designated by the caller.

WRITEP - Writes data form caller-designated buffer area to a specified byte offset within a file and performs blocking as necessary. GETSIZ is called when a preliminary readback is indicated in order to avoid attempted reads beyond the end of the file. If the write operation resulted in extension of the file, PUTSIZ is called to update the file size information.

4.2.4 CORE TABLE MANAGEMENT PRIMITIVES

These routines provide an interface between all other IMDB routines and the information stored in the system core tables TFCT and PFCT. Data transfers rely on CTEBUF in the /GENCOM/ common area which is a six entry INTEGER*4 array corresponding to core table columns 5 through 10.

CTENAM - Returns the name portion of the core table entry for the given table and row number.

GETCTE - Calls SEARCH to determine the table and row number of the specified file. The requested core table entry is returned to CTEBUF and the STATUS parameter is set to indicate in which table the entry was found.

UPDCTE - Calls SEARCH to determine the core table and row number

for the specified file, then copies the contents to CTEBUF to the core table entry. If the specified file is permanent, its header record is updated from CTEBUF.

DELCTE - Calls SEARCH to find the core table entry for the specified file, then blanks out that entry.

PUTCTE - Finds a blank row in the specified core table, then enters the filename in columns 1-4 of the core table entry and copies the contents of CTEBUF to columns 5-10. If the entry is made in PFCT, CTEBUF is also written to the file header.

4.3 FILE MODULE UPPER LEVEL ROUTINES

4.3.1 FILE MANAGEMENT ROUTINES

MAKFIL - Create a permanent file.

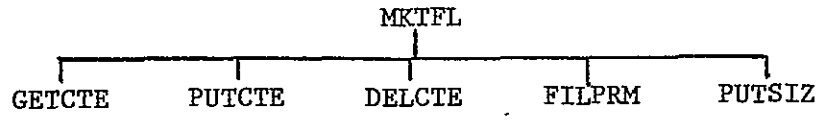
Call GETCTE to verify that no file by the given name is already active.

Call PUTCTE to make a PFCT entry for the new permanent file using information passed in CTEBUF.

Call FILPRM to create the file and call PUTSIZ to initilize the PFST entry.

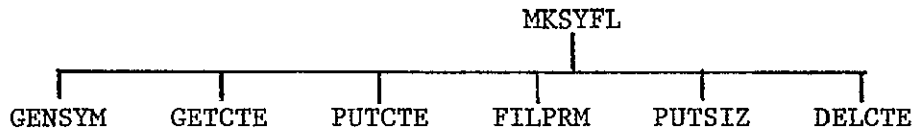
Finally, call WRITEP to copy header information to the file.

MKTFL - Create a temporary file.



The procedure is the same as for MAKFIL except that the core table entry is made in the TFCT instead of the PFCT and no header is written.

MKSYFL - Create a temporary file having a system generated name.



GENSYM is called to obtain a file name, then the procedure followed parallels that of MKTFL.

GENSYM - Generates a name for a system temproray file.

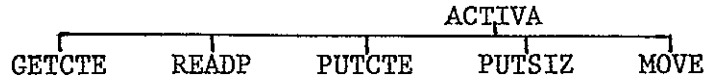
MKZOOM - Because the BZ (build zoom) command always results in a zoom having a temporary status and because all zoom information can be kept in the core table entry, no file is actually created. The create function of FILPRM is called to ascertain that there is not already a file having the specified name, but the file is immediately deleted by a second call to FILPRM. PUTCTE is called to make a core table entry using the information passed in CTEBUF.

DLTFLS - Clears the TFCT, deleting all temporary files, TFCT is searched for non-blank entries, and for each file found, DELCTE is called to delete the entry and FILPRM is called to delete the file.

DELETE - Calls DELCTE and FILPRM to delete a file from the system.

DELBLK - This routine is passed a 126-word linear array, each successive 8 words of which either specify a file name or are zero. For each non-zero eight word field, DELBLK calls DELCTE and FILPRM to delete the named file from the system.

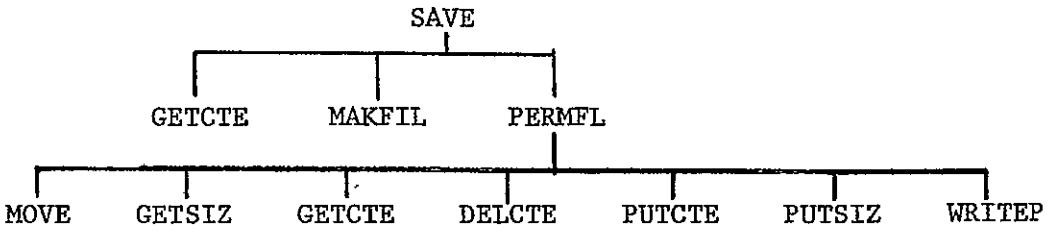
ACTIVA - Activate (make a core table entry for) a permanent file.



GETCTE is called to make sure the file name is not currently active. The CREATE function of FILPRM is called to determine whether the file exists. READP is called to read the core table information from the file header into CTEBUF. PUTCTE is called to make the

core table entry, and finally PUTSIZ is called to initialize the PFST entry.

SAVE - Make a temporary file permanent.



GETCTE is called to verify that the file is indeed temporary. If the named file is a zoom function, MAKFIL must be called. Otherwise, PERMFL is called to move the core table entry from TFCT of PFCT.

PERMFL - GETSIZ is called to obtain the number of records in the file. GETCTE is called to obtain the core table information. DELCTE deletes the TFCT entry and PUTCTE makes an entry in the PFCT. PUTSIZ makes the entry in PFST and WRTIEP is called to write the file header.

INICOM - Opens the system message file and initializes the /GENCOM/ common area which includes the core tables. (Note that the constants TYPEI, TYPEW, BLANKS, etc, must be initialized in this way. The FORTRAN DATA statement does not handle initializing INTEGER*4 variables with 4-byte Hollerith constants correctly)

JSTART - Begin journalizing. If the JMODE flag in /common/ is set, the current journal file is closed. FDSCRP and DSDSCR are called to create a file descriptor and dataset descriptor for the file CHKFIL is called to determine whether the named file already exists and the FORTRAN OPEN statement is used to open the file with TYPE = 'NEW' or else with TYPE = 'OLD' and ACCESS = 'APPEND' depending on the status returned by CHKFIL.

MSGOUT - Writes message lines from the system message file to the specified device, and, if appropriate, to the current journal file.

4.3.2 DATA ACCESSING ROUTINES

This group of routines provides caller transparent access to data contained in IMDB system files. The general procedure for each of these routines if to call GETCTE to verify that the file is active and of the appropriate type. The location of the specified information in the file is calculated and READP or WRITEP is called to effect the requested data transfer.

RESCLN - Return the specified scanline in the designated buffer.
WTSCLN - Copy the specified scanline from buffer to file.
RDWNDW - Read a window into the designated buffer.
WTWNDW - Copy window data from buffer to file.
RDDSCR - Read description field of file header into specified buffer.
WTDSCR - Copy from buffer to description field of file header.
RDEXPR - Read expression from buffer to expression file.
WTEXPR - Copy expression from buffer to expression file.
RDCOLR - Read color table from file to TCBUF in /GENCOM/.
WTCOLR - Copy color table from TCBUF to file.
RDTNSF - Read transform table from file to TCBUF.
WTTNSF - Copy transform table from TCBUF to file.

5. IMPLEMENTATION OF THE MANIPULATION MODULE

The manipulation module of IMBD provides data handling routines which realize the data manipulation commands entered by the user at the terminal. The three sections of the manipulation module are

1) the expression interpreter which realizes the image expressions entered through the BI (build image) command,
2) the display support routine which is responsible for coloration and reduction to screen size of images to be displayed, and
3) the data entry routine which provides for the reading, editing and channel separation of ANSI standard tape image files so that new image data may be introduced into the IMDB system.

5.1 MANIPULATION MODULE DATA STRUCTURES

The manipulation moudle creates, combines, and modifies IMDB system standard files, relying on the file module facilities discussed in Part 4. Of special interest is the expression file. The data portion of an expression file contains a sequence of 16-byte IMDB system filenames or operation symbols representing the postfix form of an image expression as entered through the BI command. The only addtional data structure of significance is the operand stack managed by the expression interpretation section of the manipulation module. This is the 160 word INTEGER array STACK in the /EXPCOM/ common area which allows for the stacking of up to twenty operand file names.

5.2 MANIPULATION MODULE PRIMITIVE ROUTINES

These routines provide caller transparent stack management and dynamic allocation and deletion of intermediate temporary files for the expression interpreter routines.

PUSH - After checking for stack overflow, PUSH copies the name of an operand file to the first empty entry in the stack, sets the deletion flag for the new stack entry as directed by the DFLAG parameter, and adjusts the stack pointers.

POP - After checking for stack underflow, POP adjusts the stack pointers, effectively removing an entry from the top of the stack. Also, POP checks for the deletion flag (i.e., entry of the DELMK array in /EXPCOM/) corresponding to the deleted stack entry and, if indicated, calls DELCTE and the delete function of FILPRM to delete the file from the system.

NXTERM - GETCTE is called to determine the number of terms in the expression being interpreted. This is compared with the term pointer TNUM in /EXPCOM/ to determine whether the last term of the expression has already been returned by a previous call. If so, an end of expression status is returned and the end of expression flag FINISH in /EXPCOM/ is set. Otherwise, READP is called to obtain the next term of the expression. If this term is a filename (as opposed to a system-defined operation), GETCTE is called so that the core table information is returned to the caller in CTEBUF.

GTOUFL - This routine returns to the user the name of the file which is to receive the result of the next operation performed by the expression interpreter.

GETCTE is called to obtain the length of the expression being interpreted, and the expression length is composed with TNUM and the FINISH flag tested. If the operation to be performed is not the final operation of the expression, MKSYFL is called to generate a temporary intermediate file and DFLAG = 1 is returned to indicate that the file is to be deleted after being used as an operand and popped off the stack. If the operation to be performed is the final operation of the expression, the expression file itself is designated to receive the result, FINISH is set to 1, and DFLAG = 0 is returned along with the name of the expression file.

Note that the information passed in CTEBUF is copied to the core table entry of the file whose name is returned. This is accomplished by MKSYFL when a temporary file is generated and by UPDCTE when the expression file itself is returned for use as the output file for the next operation.

5.3 LOWER LEVEL MANIPULATION ROUTINES

These routines implement most of the actual data manipulation functions of IMDB. For clarity of presentation, a distinciton is made between image manipulation routines which operate only on images, and window manipulation routines which take an image and a window as operands.

5.3.1 IMAGE MANIPULATION ROUTINES

OLINE - The overly function code parameter is used to effect a case branch (via a COMPUTED GOTO) to perform the specified function on a byte from SCLBF1 and the corresponding byte from SCLBF2 (both in the /GENCOM/ common area). The correspondence between bytes of SCLBF1 and SCLBF2 is determined by the BYTE1 and BYTE2 parameters. This byte by byte approach is adopted to avoid overflow which could result from some of the overlay functions. Each overlaid byte is reinserted in SCLBF1.

ZPREP - GETCTE is called once to determine the zoom ratio and a second time to find the size in lines and pixels of the image to be zoomed. The size of the resulting image is computed and UPDCTE is called to enter this information in the core table entry for the file which will receive the zoomed image.

ZLINE - This routine handles the actual magnification or reduction of an image on a scanline by scanline basis. When called upon to produce a specified line of the zoomed image, ZLINE calculates which line of the original image is to be used as the basis for the requested line of the resultant image. RDSCLN is called to obtain the base line. Reduction is achieved by averaging groups of adjacent pixels. Each group consists of approximately one plus the reciprocal of the zoom ratio pixels. Magnification is achieved by duplicating pixels. The zoomed scanline is returned left justified in the specified buffer (this buffer must be at least 4096 bytes long).

5.3.2 WINDOW MANIPULATION ROUTINES.

WOPRND - Call GETCTE for the top two operands on the stack and determine which is the window and which is the image.

BLDLST - Build an intersection list for the given window and scanline. An intersection list is defined as a list, sorted in ascending order, of the x-coordinates of all the points between 0 and 4095 inclusive where the window crosses the line and the points 0 and 4095. Such a list should always contain an even number of points.

RDWNDW is called to read the window into SCLBF1 and GETCTE is called to determine the number of points in the window. X-coordinate 0 is always the first point of intersection. A point of the window is selected which does not lie on the given scanline. The window is then traversed point by point until the starting point is reached. After each advance to a new point, a test is made to determine whether the advance has resulted in an intersection. An intersection is considered to have occurred if and only if:

1) the previous point and the new point lie on opposite sides of the scanline, or
2) the previous point lies off the scanline and the new point lies on the scanline, or
3) the previous point lies on the scanline and the new point lies off the scanline and the last point visited

which does not lie on the scanline lies on the same side of the scanline as the new point. Some examples are given in Figure 5.1.

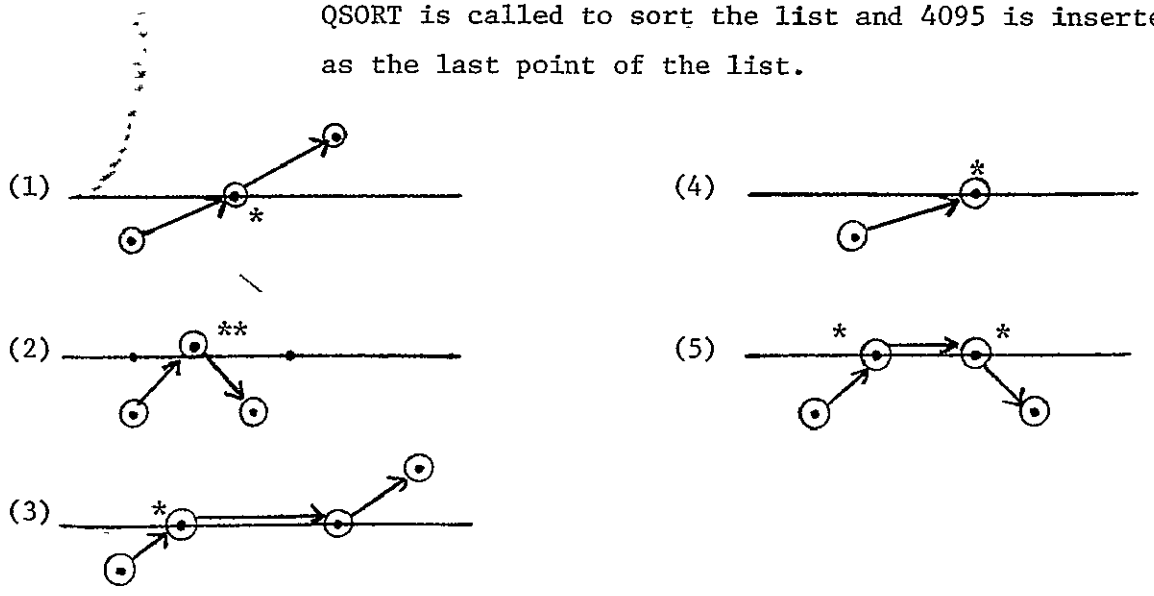
Each time an intersection occurs, its x-coordinate is entered in the intersection list being built in SCLBF2.

In case 1) above, the point of intersection must be calculated. The formula

$$x = x_1 + y - y_1 \left(\frac{x_2 - x_1}{y_2 - y_1}\right)$$

is used to (note the use of INTEGER*4 variable DISP to avoid overflow and truncation) where (x_1, y_1) are the coordinates of the previous point, (x_2, y_2) are the coordinates of the new point, and y is the scan line number.

QSORT is called to sort the list and 4095 is inserted as the last point of the list.



* indicates one point of intersection

** indicates two points of intersection

Figure 5.1 Example of intersections

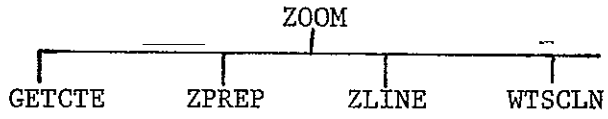
QSORT - An adaption of the version of C.A.R. Hoare's quicksort algorithm presented in Kernighan and Plauger, Software Tools, p. 115.

WINDOP - The intersection list in SCLBF2 is consulted in order to determine which sections of the scanline in SCLBF1 are to be zeroed. If enclosure is specified, then the areas of SCLBF1 between the points specified by each odd-even pair of list elements (SCLBF2 (1)-SCLBF2 (2), SCLBF2 (3)-SCLBF2 (4), etc.) inclusive is zeroed. If exclosure is specified, the area between the points of SCLBF1 specified by even-odd pairs (SCLBF2 (2)-SCLBF2 (3), etc.) exclusive are zeroed.

5.4 HIGH LEVEL DATA MANIPULATION ROUTINES

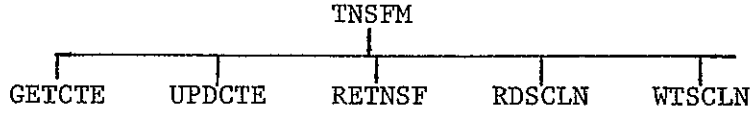
These routines are characterized by the fact that they operate on entire images through calls to file module and low level manipulation module routines.

ZOOM -



ZPREP is called to set output file core table information. GETCTE is called to determine the number of lines in the zoomed image. For each line in the zoomed image, ZLINE is called and the zoom line is written on the output file via WTSCLN.

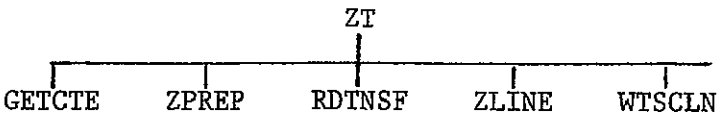
TNSFM -



GETCTE obtains core table information for the input file. Since this information will be exactly the same for the result image, UPDCTE is then called to set the core table entry for the output file. RDTNSF reads the transform table into TCBUF. Then each

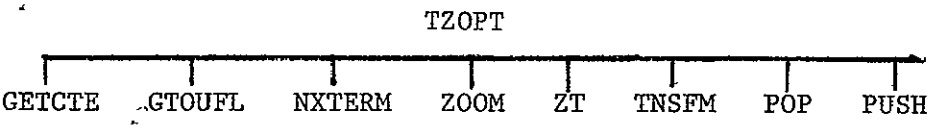
line is read via RDSCLN, transformed byte by byte - each byte used as an index into the transform table in order to pick up the value to which it maps - and written to the output file via WTSCLN.

ZT -



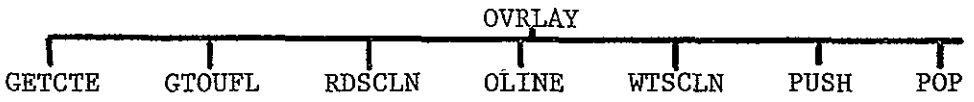
A zoom and a transform are combined. ZPREP sets the core table entry for the output file. GETCTE obtains the number of lines in the zoomed, transformed image. RDTNSF reads the transform table into TCBUF. Then, for each line of the result image, ZLINE is called, the line is transformed as in TNSFM, and the resulting line written by WTSCLN to the output file.

TZOPT -



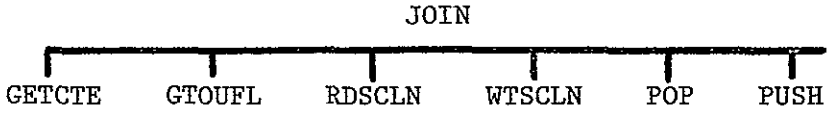
Called whenever a zoom or transform operation is to be done, TZOPT acts as a traffic controller for ZOOM, TNSFM, and ZT. GETCTE determines the type of the operation. NXTERM is called to obtain the type of the next term of the expression. TZOPT optimizes a zoom followed by a transform or a transform followed by a zoom into a single call to ZT. If no optimization is indicated by the call to NXTERM, the TNUM term pointer is backed up by subtracting one. GTOUFL is called to obtain an output file and the appropriate routine - ZOOM, TNSFM, or ZT - is called. A call to POP then removes the operand from the stack and the result is stacked through a call to PUSH.

OVRLAY -



GETCTE is called twice to obtain the core table entries for the images to be overlaid. If translation of the images before overlaying is indicated, the translation factors are calculated. The resultant image will be the overlay of those portions of the two images which, after translation, overlap. GTOUFL obtains the output file and sets its core table entry. For each line of the result, RDSCLN is called twice to read lines from the operand files. OLINE is called to overlay the scanlines and WTSCLN copies the result line to the output file. POP is called twice to remove the operands from the stack, and PUSH stacks the result.

JOIN -



Two calls to GETCTE obtain core table information for the operands. Translation factors, if any, are calculated along with the dimensions of the result image. GTOUFL returns an output file and sets it core table entry. The concatenation process is done on a line by line basis in three steps. First, SCLBF1 is zeroed. Then a line from image two to read by RDSCLN, if indicated by the calculated translation. Finally, a line from image one is read by RDSCLN, if indicated, on top of the line from image two. Thus, in areas of overlap, image one will have precedence. The scanline is then written by WTSCLN to the output file. When the entire result image has been built, two calls to POP unstack the operands and a call to PUSH stacks the result.

CLMASK

WOPRND is called to get the operand pointers. GETCTE is called twice for the operand core table information and the core table entry for the result is calculated. If an operation is specified, the result image will have the same size and location as the operand image. If CLIP is specified, the image will be the smallest rectangle which is bounded on each of its four sides by either the image boundary or, if the window does not extend beyond the image boundary, by a line paralled to the image boundary which passes through the point of the window nearest the image boundary. GTOUFL obtains the output file and sets its core table entry.

For each line of the image,

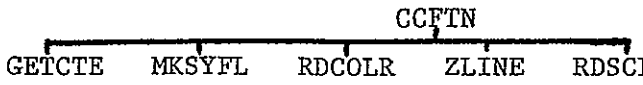
1) BLDLST is called, and the intersection list translated to reflect the origin of the scanlines comprising the operand image
2) RDSCLN reads a line from the operand image
3) WINDOP is called
4) WTSCLN writes the masked or clipped line to the output file.

Two calls to POP and a call to PUSH update the stack

5.5 MANIPULATION MODULE TOP LEVEL COMPONENTS

5.5.1 ROUTINE TO PERFORM COLORING AND COMPRESSION - CCFTN

CCFTN - Display support routine



GETCTE is called to obtain the core table information for the image to be displayed.

If the image is to be compressed, the size of the new image is calculated.

If neither compression nor coloring is requested, the name of the image to be displayed is returned as the name of the final display file. Otherwise, MKSYFL is called to obtain a display file.

If the image is to be colored, RDCOLR reads the color transformation table into TCBUF.

For each line of the final display image,

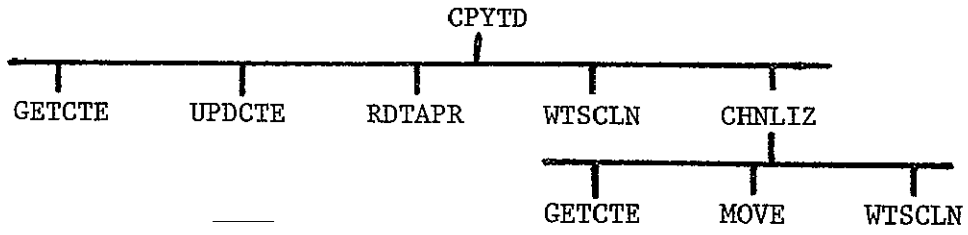
if the image must be compressed, ZLINE is called.
Otherwise RDSCLN is called.

if the image must be colored, the line is transformed byte by byte according to the transform table.

The line of the final display image is written by WTSCLN.

5.5.2 Routine to copy tape to files to disk - CPYTD

CPYTD - Tape to disk interface routine.



This routine implements the tape reads requested by the user RT command.

5.5.2.1 CPYTD SPECIAL DATA STRUCTURES

NAMBLK - A 136 word array which may contain up to 17 file names - one for a multichannel image file and up to 16 for single channel file. Words 1 - 8 are reserved for the file name for the entire image, words 9 - 16 for the file name for channel 1, etc. If the first word of the 8 word file name field is 0, no file has been requested for the corresponding channel.

EDBLK - A 5 word array indicating which portion of the image the user wants to enter into the system image file. If word 1 is 0, the entire file is to be copied and words 2-5 are ignored.

Otherwise, words 2 and 3 specify the beginning and ending pixels within a scanline which are to be retained in the system file(s). Words 4 and 5 specify the beginning and ending scanline (tape record) numbers. Pixels and scanlines are assumed to be numbered from 0.

5.5.2.2 CPYTD SPECIAL ROUTINES

RDTAPR - Perform tape read by calling WTQIO and check for error or end of file status.

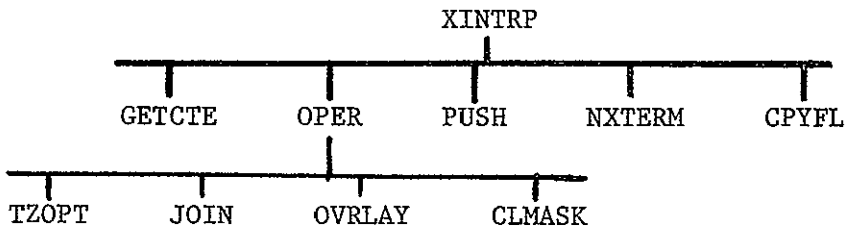
CHNLIZ - Separate the channels of a scanline and copy them to individual files.

For each file in NAMBLK,
call GETCTE for line length
assemble the single channel scanline (except for NAMBLK(1))
call WTSCLN to copy the line to the file.

CPYTD - Call WTQIO to rewind the tape if requested.
Call WTQIO to skip files if requested.
Call RDTAPR to read the first record and obtain the record length.
Calculate the core table size entry for each NAMBLK file based on NCHANS and EDBLK information and call UPDCTE to enter the appropriate pixels-per-line value in the core table entry.
If the line is not to be edited out.
call WTSCLN if no single channel files are required or else call CHNLIZ to copy tape data to single channel files.
Read the next record from tape and if not end of file, request the output process described above.
When the end of the tape file has been reached, call UPDCTE to enter the number of lines for each NAMBLK file.

5.5.3 Routine to interpret image expression - XINTRP

XINTRP - Expression interpreter.



5.5.3.1 EXPRESSION COMPONENTS

An expression is composed of operands and operators (each 16 bytes long) arranged in postfix sequence. There are currently only two valid operand types - the name of an image file and the name of a window file. A third possibility, the name of a line file, has not yet been implemented. Operators may be transform or zoom file names (these two classes are the only file names which are valid operators) or system defined operations. A system defined operation is stored as the ASCII characters '**' in the first word followed by a code number between 1 and 13 inclusive in the second word. Words 3-8 are not used. The code numbers correspond to the IMDB system reserved words.

CODE	OPERATION
1	JOIN
2	MASK
3	CLIP
4	MULT
5	ADD
6	SUB
7	DIV
8	MAX
9	AVG
10	AND
11	XOR
12	MIN
13	OR

5.5.3.2 EXPRESSION INTERPRETER ROUTINES

CPYFL - Invoked when an expression consists of only one term. The operand file is copied to the expression file. GETCTE is called to determine the file type, UPDCTE makes the core table entry for the new file (formerly the expression file), and a case branch is effected via a COMPUTED GOTO to copy the file using the data accessing routines described in section 5.3.2.

OPER - Called whenever an operation is to be performed by the expression interpreter.

If the operator is not system defined, GETCTE is called to verify that the operation is legal (i.e., file type T or Z) and TZOPT is called. Otherwise, a COMPUTED GOTO case branch based on the operation code number (see section 5.5.3.1) accomplishes a call to CLMASK or OVRLAY or JOIN.

XINTRP - GETCTE is called to obtain the expression length. If the expression length is exactly 1, NXTERM obtains the term and CPYFL is called. Otherwise, the following steps are repeated.

NXTERM is called.

If no terms remain in the expression XINTRP returns.

Otherwise, if the term is an operand, it is stacked by a call to PUSH.

If the term is an operator, OPER is called.

APPENDIX I

IMDB USER'S MANUAL

IQ LANGUAGE

This chapter is written as an IQ reference manual. The presentation mimics that of the original report on IQ design [Report I]. We shall skip justifications and explanations of certain design decisions, as they have already been covered in [Report I].

1.1 Basic Elements.

There are five types of basic elements in the IQ language: images, windows, transform functions, color functions and zoom functions. Each basic element is a file in the IMDB system.

1.1.1 Image

An image is a matrix of pixel values along with a header block. Since it is assumed to be a matrix, the image is always rectangular in shape. Pixel values range from 0 to 255. The upper left corner of the image is associated with a coordinate in { *, 0, 1, ..., 4095 } X { *, 0, 1, ..., 4095 }. The first component of the coordinate is referred to as the LCQ, and the second component LAQ. The intention is that when the image is first entered into the database, the user can assign its LCQ and LAQ relative to a 4096X4096 grid structure. The asterisk * is used to denote

"don't care". The LCC and LAC pair is essential to binary image operations to be presented later.

The header block of an image contains:

(a) type: This field is always filled with ' I' to denote the type image.
(b) LCC
(c) LAC
(d) pixels/line: It is the number of pixels in a scan line.
(e) scan lines: It is the number of lines in the image.
(f) description: It is a string of characters entered by the user for annotation. The size is limited to 228 characters.

1.1.2 Window

A window is a sequence of points together with a header block. Each point falls within { 0, 1, ..., 4095 } X { 0, 1, ..., 4095 } grid coordinates. The sequence of points form one not necessarily convex polygon. The header block contains the following information:

(a) type: The field contains ' W'.
(b) maximum LCC/LAC: The maximum of LCCs of all points and the maximum of LACs of all points are encoded into this field.
(c) minimum LCC/LAC: This field stores the minimum LCC and LAC in a way similar to (b).
(d) closure code: The field denotes whether the window is an enclosure or an exclosure.
(e) number of points: It is the number of points in the window.
(f) description.

1.1.3 Transform

A transform function is a mapping from { 0, 1, ..., 255 } to { 0, 1, ..., 255 }. It usually consists of a collection of subtransformations. Each subtransformation is in

the form of

$$a - b = c \quad \text{where} \quad a \leq b$$

It means that the pixel values from a to b inclusive are to be transformed into c .

The header block of a transform contains:

(a) type: It is always ' T'.
(b) description.

1.1.4 Color

A color function is a mapping from { 0, 1, ..., 255 } to a set of color symbols. There are two systems of color symbols used in the IC language. The first one uses a 4-bit format and consists of eight different colors: dark (D), blue (B), green (G), red (R), cyanine (C), magenta (M), yellow (Y) and white (W). The user uses the one-character symbols to denote colors. The other system allows sixty-four colors and uses a 6-bit format. The basic components of each color are still blue (B), green (G) and red (R). However each basic color has four shades. For example, 1 part of B, 3 parts of G and 3 parts of R give a yellowish color. The user can use B1G3R3 to denote this formation of color. In general, it is hard for the user to visualize the resulting color from the three components. Hence, the user is provided with a color table which maps each of the sixty-four colors to a number. The user can also use this number to select a color.

A color function is similar to a transform function; it consists of a collection of subtransformations. Each subtransformation is in the form of

$$a - b = c \quad \text{where} \quad a \leq b$$

where c is a color specification. All subtransformations in a color function are either all in 4-bit format or all in 6-bit format.

The header block of a color function consists of:

(a) type: The content is always " C"
(b) description.

1.1.5 Zoom

A zoom function contains a mapping from old size to new size and a header block. The zoom ratio is a / b where a is the new size, b is the old size and both a and b are positive integers.

The header block consists of:

(a) type: The content is ' Z'.
(b) new size
(c) old size
(d) description

1.2 System Functions.

There are several built-in functions in the IQ language which can be used to create new images. These functions can be invoked by name. They consist of JOIN, MASK, CLIP and

ten different overlay functions.

1.2.1 Join

This function pastes two images together to form a new image, according to their LOC/LAC coordinates. The dimensions of the new image are those which are minimally sufficient to contain the areas of the originals. The first of the original images is defined to be the dominant image: this image takes precedence when the two images overlap. When the result is padded to become rectangular, the pixel value zero is filled.

The rules used to determine the relative positions of the two images are:

(a). If LOC/LAC of the first and second images do not contain * , then the two pairs of LOC/LAC all refer to well-defined points in the 4096X4096 grid structure. Neighboring pixels along the same scan line differ in LOC by one and neighboring lines differ in LAC by one.

(b). If LOC of the one image is * while the other is not, then the * one is assumed to have the same value as the other one. The * for LACs are treated in a similar way.

(c). If LOCs in both image are * , then both are treated as zero. The * in LACs are treated similarly.

Join function results in a new image and the header block of this new image will be derived from the originals. Description field will be empty.

1.2.2 Mask

This function masks a window onto an image to form a new image. If the window is an inclosure, pixels interior to the window will retain the values while exterior pixels will be zeroed. Exclosure functions in the opposite manner. In either case, the result is an image with the same LOC/LAC and the same dimensions as the original. Again, the description field will be empty.

1.2.3 Clip

This function is similar to MASK except that the result image has dimensions which are minimally sufficient to contain the window. This function discards those outermost rows and columns which do not intersect the window.

1.2.4 Overlay Functions

An overlay function takes two images and produces a new image by performing a binary pixel-to-pixel operation over corresponding pixels. There are ten different overlay functions: ADD, SUB, MULT, DIV, MAX, MIN, AVG, XOR, AND, OR. These functions perform respectively addition, subtraction, multiplication, integer division, maximum, minimum, average, exclusive OR, logical AND, and logical OR. Whenever

overflow occurs (e.g., in multiplication), the result is always truncated by taking the rightmost eight significant bits.

The relative positions of the two images are determined according to the rules specified in Join (Section 1.2.1).

1.3 Image Expression

A salient feature of the IQ language is its capability for specifying construction of a new image as a functional expression of existing basic elements and system functions. Such an expression is called an image expression. The rules for constructing image expressions are given below. These rules can be applied recursively.


```
<image expression>:
      —— <image>
         <transform> ( <image expression> )
         <zoom> ( <image expression> )
         JOIN ( <image expression> , <image expression> )
         MASK ( <image expression> , <window> )
         MASK ( <window> , <image expression> )
         CLIP ( <image expression> , <window> )
         CLIP ( <window> , <image expression> )
         <overlay function> ( <image expression>,
             <image expression>)
```

In the above rules, <image> and <window> refer to an image file and a window file respectively. The symbol <overlay function> refers to one of the ten system overlay functions. Since the rules can be applied recursively, a sophisticated image can often be specified as one single image expression. For example, JOIN(AND(X1(T1(MASK(W1,

I1))), X2(I2)), I3) is a legitimate image expression if X1 and X2 are zoom files, T1 is a transform file, I1, I2, and I3 are image files, and W1 is a window file.

Note that color functions are not included in the image expression rules. Strictly speaking, a colored image only contains symbolic color names as its pixel values and hence it is not logical to perform any other operation on it. Nevertheless, the internal representation of a colored image is no different from a regular image and, if the user chooses to do so, a colored image may be used to replace <image> in an image expression without any system error. The interpretation of the result is up to the user.

1.4 Devices

The graphics devices can be and can only be referred to by symbolic names in a query session. The user does not have to know any particular logical or physical device numbers used internally in the IMDB system.

The devices supported by the present version of the IQ language and their corresponding device names are:

(a) Two color Ramtek screens: R1(left) and R2(right), with a trackball attached to R1.

(b) One Tektronix 4014-1 terminal: TK

(c) One user command terminal: UT

(d) One line printer: LP

(e) One Dicomed film recorder: FR

(f) One Varian printer/plotter: PL

(g) Two magnetic tapes: T0 and T1

A future expansion will include keyboards and an additional trackball attached to the Ramtek system. The Tektronix terminal is only used as an alphanumeric CRT, although future expansion can take advantage of its graphic capability.

1.5 Commands

The IQ language is a command oriented query language. Each database command activates one specific operation. A command consists of two parts: command code and parameters. A command code is always a two-character name followed by a separator (blank, comma or carriage return). Parameters may be supplied along with the command code, or deferred until answering system prompted questions. Note that all parameters may be entered through prompting. Therefore, the minimal information needed to be entered by the user to initiate a command will be the 2-character command code.

The commands are grouped into five categories: defini-

tion, display, statistics, file manipulation and control.

1.5.1 Definition Commands (5)

These commands are used to create new basic elements or equivalently new files.

1.5.1.1 Build Image (BI)

The form of a build image command is

BI <new image name> = <image expression>

or BI <new image name> , <image expression>

1.5.1.2 Build Window (BW)

The form of this command is

BW <window name>, <closure>, <mode>, <device>

The <closure> code can be EX or EN for exclosure or enclosure respectively. The default value is EN.

There are two modes in window construction: C for cursor and A for absolute. The default mode is A. In A mode, the user types in LOC/LAC pairs of the window vertices from the user command terminal. After the user enters

BW W1,EN,A

The system will repeat the question until all points are entered:

ENTER COORDINATES(ONE POINT PER LINE WITH X AND Y SEPARATED BY ,):

The question can be escaped by a carriage return.

In C mode, the user indicates that a window is to be constructed relative to an image presently displayed on <device>. Since there is only one track ball attached to R1, it is only meaningful to specify R1 as the <device>. The user can move the cursor on R1 and select a point by hitting ENTER key of the track ball. To end the construction of the window, the user hits VISIBLE (to make cursor invisible) and ENTER. In C mode, the LOC/LAC of the selected points are calculated from the LOC/LAC of the displayed image. Whether the image is displayed in its true form or in a compressed form, the calculation will produce actual positions of the points relative to the image.

1.5.1.3 Build Transform (BT)

The form of the BT command is

BT <transform name> , <subtransformations>

Each subtransformation is in one of the two forms:

lower bound - upper bound = new value

old value = new value

The right side of a subtransformation is restricted to be one single value. All unspecified intervals can be assigned to one default value

upon answering

NUMBER FOR UNDEFINED INTERVALS?

1.5.1.4 Build Color (BC)

The form of this command is

BC <color name>,<format>,<color transformations>

The color format can be 4 or 6 for 4-bit or 6-bit formats respectively. Each color transformation is in one of the two forms:

lower bound - upper bound = color symbol

value = color symbol

Here the color symbols referred to the symbolic forms of color representation as described in Section 1.1.4. Again, all unspecified intervals can be assigned to one default color upon answering:

COLOR FOR UNDEFINED INTERVALS?

1.5.1.5 Build Zoom (BZ)

This command has the form

BZ <new zoom name> , <scale ratio>

The <scale ratio> is always NEW/OLD.

1.5.2 Display Commands (3)

1.5.2.1 Erase (ER)

The form of this command is

ER <list of devices>

where <list of devices> are device names separated by commas. The effect of erasure depends on the device specified. For R1 or R2, the screen is erased. For LP, a new page of paper is moved under the print head. For PL, the command also slews the paper. All other devices are nor permitted in ER command.

The future expansion will include a capability to advance the roll film in the device FR.

1.5.2.2 Exhibit Pixel Area (EP)

The form of this command is

EP <input device> , <output device>

This command is used to examine the image pixel values of a rectangular area of no more than 20X20. The image is presently displayed on <input device>. The pixel value array is to be displayed on <output device>.

The user is also required to specify

(a) The dimensions of the rectangular area - number of lines and number of pixels.

(b) The upper-left corner of the area through the track ball.

This command is only meaningful when <input device> is R1. The <output device> is restricted to be UT, LP, R1 or R2.

1.5.2.3 Display (DI)

The form of DI command is

DI <image name>,<device>,<color function name>

or DI <window name> , <device>

The <color function name> is optional.

For window display, the device can only be R1 or R2. The displayed output depends on the existing contents of the selected screen. If the screen is blank, the window will be scaled properly so that it can be displayed entirely on the screen. After the window polygon is drawn, the system will ask

DO YOU WISH THE POINTS LABELLED WITH PIXEL/LINE COORDINATES?:

A 'Y' answer will cause the coordinates displayed along with the polygon.

If the selected screen has an image displayed, the win-

dow will be scaled according to the displayed image and the window polygon will be positioned correctly on the image so that the coordinates of the image and the window are consistent. A window may be too large to fit on the image. If so, the command will be aborted and error signaled.

In both cases of window display, the color in which the window is to be displayed will be solicited from the user.

Image display is much more involved than window display. If the user specifies a color transformation, it will be applied to the image to produce a colored image. The colored image can later be saved as a regular image file. The sequence of events can be described as the following procedure:

Step R1.

If the device is FR, goto F1.

If the device is PL, goto P1.

If the device is not R1 or R2, then error return.

Step R2.

(The device is R1 or R2.)

Ask the user to select a point on the specified screen. Let the specified screen be X and the other one Y. (The system will attempt to display the image on the

rectangular area VIEW defined by the selected point and the bottom right corner of screen X.)

Step R3.

Can the colored image fit in the area VIEW?

If yes, display the image and goto R9.

Step R4.

(The image does not fit in the area VIEW.)

Ask the user if image compression is desired?

If yes, compress the image sufficiently to fit in VIEW, display it and then goto R5. Otherwise, display the upper left portion of the colored image in VIEW and goto R9.

Step R5.

(The compressed image is on X.)

Ask if the user wants to display legend.

If yes, ask the user to select legend position and to enter legend; then display legend at the position selected.

Step R6.

Ask if the user wants to scroll the compressed image.

If no, exit.

Step R7.

(Scrolling)

Erase screen Y.

Ask the user to select a point on screen Y.

Ask the user to select the scrolling point, which is a point in the compressed image as presently displayed on screen X.

The selected point on Y and its bottom right corner define a rectangular area called SVIEW. The scrolling point together with SVIEW specifies a rectangular portion of the colored image whose LCC/LAC are those of the scrolling point and whose dimensions are those of SVIEW. Display this rectangular portion in SVIEW.

Step R8.

Ask if the user wants to scroll again?

If yes, goto Step R7.

Step R9.

Ask if the user wants to display legend on the most recently used screen (X if come from R3 or R4 and Y if

from R8).

If yes, ask the user to select legend position and to enter legend; display characters entered at the position selected.

Step R10. Exit.

Step F1.

Ask the user to select a point on the film. The film has 4096 X 4096 positions.

Step F2.

Ask the user to enter Dicomed related parameters such as magnification factor, resolution, intensity, polarity, etc.

Step F3.

Can the colored image fit in the rectangular area defined by the selected point and the bottom right corner of the film?

If yes, display the image and goto F6.

Step F4.

(The image is too large.)

Ask the user if image compression is desired?

If yes, compress the image sufficiently to fit, display the image and goto F6.

Step F5.

(Display the upper left corner.)

Display the upper left portion of the colored image in the selected rectangular area.

Step F6.

Ask if the user wants to display legend.

If yes, ask the user to select legend position on the film and to enter legend; then display legend at the position selected.

Step F7. —Exit.

Step P1.

The user can specify either 4x4 dot matrix for one pixel or 5x5 dot matrix.

Step P2.

Calculate the number of strips required to display the entire image. Inform the user.

Step P3.

Ask the user "How many strips do you want printed?:".

Step P4.

Display the strips.

Step P6.

Ask if the user wants to display legend. If yes, ask the user to enter the legend, then display it.

Step P7. Exit.

1.5.3 Statistics Commands (5)

1.5.3.1 Exhibit Histogram (EH)

The form is

EH <image name> , <device>

The device can only be R1 or R2. The user can also specify the color of the histogram upon answering

WHAT COLOR DO YOU WISH THE HISTOGRAM
TO BE DISPLAYED IN?

The output is a two-dimensional colored graph with horizontal coordinate corresponding to pixel values and vertical

coordinate frequencies.

1.5.3.2 Exhibit Distribution (ED)

The form is similar to EH . The device can only be LP. The histogram of the image will be calculated and displayed as "pixel value: frequency" pair.

1.5.3.3 Exhibit Join Histogram (JH)

The form is

JH <image name>,<image name>,<device>

The two images must be of the same dimensions. The frequencies of pixel value pairs will be calculated. The frequency values will be partitioned into at most seven ranges as directed by the user. Each range can be assigned a color by the user. If the user chooses not to define the range or the coloring of the ranges, the frequency values will be equally partitioned into seven ranges and default colors assigned.

When the joint histogram is displayed, the two coordinates correspond to pixel values of the two images. The colors of the displayed points indicate the frequency range of the pixel value pairs.

The user is also given an option to view the magnified

joint histogram. The magnification is by 2 or by 3.

1.5.3.4 Exhibit Joint Distribution (JD)

The form is the same as JH. The device has to be LP. The output is in the form of "pixel value : pixel value - frequency" for each pair of pixel values. The output format can either be sorted by frequency or by pixel value pair.

1.5.3.5 Exhibit Contigency Matrix (CM)

The form of this command is the same as JH. The device can only be LP and the images are restricted to have pixel values between 0 and 7. All higher values are truncated on the left. The purpose of the command is to compare two classified images to find their differences.

1.5.4 File Manipulation Commands (7)

All basic elements in the IMDB system are treated as files. A file can enter into the database in two ways. First, image files can be brought into the database from tape through the use of Read Tape (RT) command, which will be discussed in this section. Secondly, a file may be created through the use of definition commands (Section 1.5.1). We distinguish permanent and temporary files. Files created through definition commands are all temporary in the sense that they will be removed automatically at the end of the query session unless they are explicitly saved. Permanent files are those which last through query sessions.

Specifically, files brought in by RI are considered permanent.

When a user first logs onto a command terminal, a file directory is assigned for his exclusive use. The file directory is separated into two sections: one for temporary files and one for permanent files. Existing database files can not be used in any command until they are 'activated'. Activation of a file is a process of making the file name known to the user's file directory.

1.5.4.1 Activate (AC)

The form of this command is

AC <list of file names>

where <list of file names> is a list of names of existing files separated by commas. File names specified in the command will be entered into the permanent file section of the file directory.

1.5.4.2 Save (SA)

The form is

SA <list of file names>

This command causes existing temporary files to become permanent. File names specified in the command will be moved from the temporary file section of the file directory to the permanent file section.

1.5.4.3 Purge (PU)

The form is

PU <list of file names>

This command causes files in the directory, whether permanent or temporary, to be removed. Removal of a permanent file also purges the file from the database.

This command is not implemented at the present time. The user has to use PIP command of the RSX-11D to remove a file from the file system.

1.5.4.4 Modify (MO)

The form of this command is

MO <file name>

The purpose of this command is to allow the user to change certain information in the header block of the file.

The alterable fields of the header block are listed below according to file types:

(a) image: LCC,LAC,description

(b) window: closure code, description

(c) transform: description

(d) color: description

(e) zoom: new size, old size, description

The above rules apply to only permanent files. Temporary files can also be modified in exactly the same way except that they do not contain the description field.

1.5.4.5 List Directory (LD)

This command has the form

LD <device>

where <device> can be UT or LP. Contents of the file directory will be printed at the specified device. The file directory contains all information stored in the header block about the files activated or created by the user. (In the actual implementation, an activated permanent file has its header information stored both in the physical file as well as in the directory; and a temporary file does not have a header in the physical file, its header information is stored only in the directory.)

1.5.4.6 Spotlight (SP)

The form is

SP <file name>, <device>

The <device> can be LP or UT for image, window, transform or zoom files. It can be LP, UT, R1, R2 or FR for color files.

This command performs a similar function as LD for a single file: it displays header information of the file. However, if the file is a transform or a color file, SP also displays the definition of the mapping in the file. That is, it lists all the subtransformations in the file.

The most interesting use of SP is to spotlight color function onto a graphics device (R1, R2 or FR). It will display, for each subtransformation, the range of pixel values and a small colored square to indicate the actual color of the subtransformation. If SP is used on LP or UT for a color function, symbolic names of the colors will be displayed.

1.5.4.7 Read Tape (RT)

The form of this command is

RT <device>

The device is either T0 or T1, indicating one of the two tape drives. The options available to the users are:

(a) to read any file on the tape;

(b) to read any number of files on the tape;

(c) to edit a tape file by specifying the starting and the ending line numbers and the starting and the ending pixel numbers;

(d) to read multi-channel composite files up to 16 channels: for each channel, the user can indicate whether the image for this channel is wanted or not, and if wanted, a separate file will be created. In general, an n-channel image can be moved into the database and becomes n+1 separate files - one for each channel and one for the original n-channel file.

1.5.5 Control Commands (5)

These commands are special facilities built into the IMDB system to ease the user-system interaction.

1.5.5.1 Stop (ST)

The form is simply

ST

which ends the query session, causes all temporary files to be removed and is the only command for the user to log off the system gracefully.

1.5.5.2 Restart (RE)

The form is

RE

which performs the similar function as ST except that the user is not logged off and is assigned a new file directory with no entry in it. (The old directory is erased.)

1.5.5.3 Help (HE)

The form of this command is

HE <device>

where the <device> can be LP or UT. It lists all IQ commands with explanations at the specified device.

One of the goals of the IQ design is to minimize the information the user has to remember in order to use the IMDB system. In fact the user does not have to remember the forms of the commands. The user can obtain assistance in two ways:

(a) To consult the system for the command format and its function by typing HE, or

(b) To use prompting to enter command parameters. (The absolute minimum needed to initiate system activity is a 2-character command code.)

1.5.5.4 Journal (JC) and No Journal (NJ)

The forms of these commands are

JO <file name>

NJ

The conversation between the user and the system - in general, it is whatever shown on the UT terminal - can be recorded verbatim in a journal file. The <file name> is the name of the journal file. If the file does not exist prior to the JC command, a new one will be created bearing the name given by the user. If the file is an old one, new journal information will be appended at the end. The command NJ is used to turn off the journal activity. With these two commands, the user can specify journal mode or no journal mode at any time during the query session, switch between two modes any number of times, create several journal files and disperse journal information in any way the user desires. The only restriction is that no two journal files can be active at the same time, one has to be closed by NJ before the other can be named in JO.

The contents of the journal files can be printed at the line printer through PIP facility of the RSX-11D.

1.6 Log On

The log-on secuence to start the IMDE system is very

simple. If the user is a legitimate user of the RSX-11D system, that is, the user has a legitimate UID, the following sequence can be followed to start the IMDE system:

Step 1. Turn on a terminal.

Step 2. Type in Control C to get

MCR>

printed on the terminal.

Step 3. Type in

HEL [UID]

so that the operating system can validate whether UID is legal.

Step 4. If UID is legal, the system will come back with

MCR>

then enter "IQL" after MCR>.

At this time the log-on process is completed, the IMDB is activated and a message will be printed:

```
* WELCOME TO THE IMDE SYSTEM
*
```

Any permissible IQ commands can be entered after the second asterisk. In summary, the entire log-on sequence will look like the following if the user has the access right:

```
(Control C)
MCR> HEL[UID]
MCR> IQL
* WELCOME TO THE IMDB SYSTEM
* (ready to accept IQ command here)
```

A blank file directory has also been created for the user.

1.7 Special Notes

Some conventions and special cases not covered in the previous sections are covered here:

(a) A window is assumed to be a simple polygon. No two edges of the polygon can cross each other, of course, other than meeting end to end for neighboring edges. The system does not check the crossing of edges and the user is responsible for the correctness of polygon formation.

(b) Whenever a question is asked the user, a carriage return is taken as NO, 0(zero), or the default answer, depending on the nature of the question.

(c) The operating system PIP facility can be used to copy files from tape to tape or from disk to tape, to purge files from the data base, and to rename files in the data base. The IMDB system is built on top of the

FILE-11 file system and any file operation available in the operating system can be applied to IMDB files.

(d) A 'Carriage Return' as an answer to the question 'DEVICE?' will cause the list of all permisssible device names to be printed at UT.

(e) The reserved words in the IQ language are JOIN, MASK, CLIP, MULT, ADD, SUB, DIV, MAX, AVG, AND, XOR, MIN and CR. These can not be used as a file name of any file.

1.8 Variations from Original IQ Design

The differences between the version of IQ as implemented and described in this manual and the one in [Report I] can be summarized as follows:

(a) This version uses 2-character command code and the original version does not.

(b) This version does not have window union or intersection capability.

(c) New commands are added in this version: ER, EP, CM, MC, and HE.

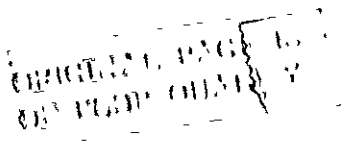
(d) Write Tape command is not included in this version.

(e) Overlay functions are not treated as generic functions in this version. That is, only ten system built-in overlay functions are allowed and the user can not

define his own.

(f) Image expressions have to be evaluated and assigned a new name (in DI) before it can be used in DI. In the original version LET and DEFINE are distinguished. In this version they are combined into build commands.

(g) The original version assumes a longitude/latitude coordinate system. This version assumes a 4096 X 4096 grid coordinate system. The images are no longer associated with geographic position.



APPENDIX II

INDEPENDENT UTILITY ROUTINES

FIXCTE - useful in emergency situations when the file data is known to be valid and the file parameters are known by the file header information has been corrputed or non-existent. Interactive. FIXCTE asks for a filename, calls ACTIVA to activate the file as a permanent file, and prints the current file header information corresponding to core table columns 5-10. FIXCTE then asks for new information to be inserted and prints this information back to the terminal for verification and asks the user if he wishes to go ahead with the update. Any response except 'Y' terminates the program. The Y response causes the new information to be written to the file header, and the program terminates.

(FIXCTE needs to be rewritten to allow alteration of the number of records in file field (header bytes 24-27) and to allow more than one file to be fixed without having to rerun the program.)

MESSFILE - interactive message file updata and examination routine. CURRENTLY, MESSFILE MUST BE RUN UNDER THE UIC IN WHOSE DIRECTORY THE SYSTEM MESSAGE FILE 'IMDB.MSG' RESIDES.
MESSFILE asks the user is he wishes to examine the message file or to update it. The user responds with a 'U' or an 'E' -- any other response terminates the program. If examine is requested, MESSFILE asks for a message number and reads the message from the file and prints it on the user terminal. If update is specified, a message number and message contents are requested, and the message is written to the file. Specifying a message number less than or equal to zero (or a carriage return) returns the user to the question 'examine or update?'.

(The message file is organized as a random access file of 72-character records. Each record contains exactly one message, with the message numbers corresponding to the record numbers.

Note that since MESSFILE opens the message file with TYPE= 'UNKNOWN', MESSFILE can also be used to create a new message file.)

APPENDIX III

PERMANENT FILE HEADER FORMAT

FILE TYPE	BYTES							
	0-3	4-7	8-11	12-15	16-19	20-23	24-27	28-255
IMAGE	TYPE	LOQ	LAQ	UNUSED	PIXELS /LINE	SCAN LINES	RECORDS IN FILE	DESCRIPTION
WINDOW	TYPE	MAXIMUM LOQ & LAQ	MINIMUN LOQ & LAQ	CLOSURE CODE	NUMBER OF POINTS	UNUSED	FILE SIZE IN RECORDS	DESCRIPTION
EXPRESSION	TYPE	NUMBER OF TERMS	UNUSED	UNUSED	UNUSED	UNUSED	FILE SIZE IN RECORDS	DESCRIPTION
TRANSFORM FUNCTION	TYPE	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	FILE SIZE IN RECORDS	DESCRIPTION
COLOR FUNCTION	TYPE	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	FILE SIZE IN RECORDS	DESCRIPTION
ZOOM FUNCTION	TYPE	RELATIVE SIZE OF RESULT	RELATIVE SIZE OF BASE	UNUSED	UNUSED	UNUSED	FILE SIZE IN RECORDS	DESCRIPTION
LINE (NOT IMPLEMENTED	TYPE	LOQ POINT 1	LAQ POINT 1	LOQ POINT 2	LAQ POINT 2	UNUSED	FILE SIZE IN RECORDS	DESCRIPTION

APPENDIX IV

BIBLIOGRAPH OF PUBLICATIONS

1 Y.E. Lien and D.F. Utter, Jr. "Design of An Image Database", Proceedings of the Workshop on Picture Data Description and Management, IEEE, April 21-22, 1977.

2 Y.E. Lien and R. Schroff "An Interactive Query Language for an Image Database", to appear in the International Journal on Policy Analysis and Information Systems, January, 1978.

3 R. Schroff, "Boolean Operations on Polygons", M.S. Thesis, Department of Computer Science, University of Kansas, Lawrence, Kansas, December, 1977.

4 S. Harris, M.S. Thesis, Department of Computer Science, University of Kansas, Lawrence, Kansas, December, 1977.

5 R. Law, "Design of an Interactive Digital Image Analysis System", M.S. Thesis, Department of Computer Science, University of Kansas, Lawrence, Kansas, December, 1977.

6 C.J. Chen M.S. Report, Department of Computer Science, University of Kansas, Lawrence, Kansas, December, 1977.

APPENDIX V

DOCUMENTATION PACKAGE

(This package provides a complete RATFOR and Assembly Language listing of the IMDB system. It also includes a listing of the utility routines, the message file and the task build file. One complete package and a magnetic tape containing the entire program have been provided to Dr. Robert R. Jayroe , Data Systems Laboratory, Marshall Space Flight Center. Since the package consists of over 300 pages of program listing, it is not duplicated here.)