(NASA-TM-102580) RASTER METAFILE AND RASTER METAFILE TRANSLATOR (NASA) 114 p CSCL 09B

N90-18166

Unclas

G3/61 0264792

NASA Technical Memorandum 102588

Raster Metafile and Raster Metafile Translator

Nancy L. Taylor
Eric L. Everton
Donald P. Randall
Raymond L. Gates
Kristi M. Skeens

September 1989

Central Scientific
Computing Complex
Document G-14

ACKNOWLEDGMENTS

The authors would like to acknowledge several individuals from LaRC and CSC who contributed to the content and preparation of this document. First, the authors appreciate the support of the project sponsor, John E. Hogge. Special thanks are awarded to the other LaRC personnel who comprised the initial Raster Metafile specification committee: James B. Hall, Mary-Anne K. Posenau, and W. H. C. von Ofenheim. The authors would also like to thank LaRC personnel Mary S. Adams and Pat Troutman, CSC employees Valerie D. Corker and Scott R. Nolf, and PRC employee Kenneth E. Tatum who provided suggestions and test images which aided in the debugging of the Raster Metafile Translator. Next, the authors appreciate the time and effort of the LaRC Analysis and Computation Division personnel who reviewed the initial draft of the document: Donald L. Lansing, Mary-Anne K. Posenau, and W. H. C. von Ofenheim. Finally, the authors' gratitude is extended to CSC employees Nancy F. Schott, Patricia K. Wright, and Corinne L. Jones who prepared the final document.

TABLE OF CONTENTS

PRECEDING PAGE BLANK NOT FILMED

PAGE ii INTENTIONALLY BLANK

TABLE OF CONTENTS

Section Page

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

The evergrowing requirement for graphic visualization in a scientific computing environment is well documented [1]. In support of this demand for "visualization in scientific computing" (ViSC), an associated requirement for one or more portable device-independent graphics output formats has emerged. Most efforts concerning the generation and processing of such graphics output formats, commonly referred to as graphics metafiles, have been directed towards vector graphics. The most recent of these efforts has resulted in the Computer Graphics Metafile (CGM) [2] which is rapidly gaining acceptance as a metafile standard. Although CGM does support a raster graphics element in the form of a "cell array," it is primarily a vector graphics metafile. In the area of raster graphics, some work with generic image formatting and processing has been done [3] but no standard has emerged. The purpose of this documentation is to present an effort undertaken at NASA's Langley Research Center (LaRC) to design a generic raster image format and to develop software tools for processing images prepared in this format.

The problem of processing raster image data in LaRC's computing environment may be characterized by an increasing number of software package-specific raster data formats that are to be displayed on an increasing number of raster devices which accept only device-specific raster data input. A consequence of this problem is the proliferation of data translators which convert package-specific raster output into device-specific raster input. In order to minimize raster data translation, LaRC has adopted a generic Raster Metafile (RM) format and developed a Raster Metafile Translator (RMT) for processing RM formatted image data.

This document addresses both the RM format and the RMT. The document is intended to serve a varied audience including: users wishing to display and manipulate raster image data, programmers responsible for either interfacing the RM format with other raster formats or for developing new RMT device drivers, and programmers charged with installing the software on a host platform. Section 2 contains general information and relevant terminology peculiar to the RM format. Sections 3 through 6 provide overview information on the use of the RMT and include conventions and pertinent terminology. Individual RMT command features and syntax are detailed in Section 7. The process of device selection and ensuring compatibility between image and device attributes is provided in Section 8. Section 9 illustrates the operation of the RMT through a collection of representative command scenarios. Section 10 addresses the error processing facilities supported within the RMT. Software tools targeted towards interfacing external raster image formats with the RM format are presented in Section 11. The procedures for installing the RMT in a UNIX environment along with guidelines for other environments are detailed in Section 12. Section 13 contains extensive information on existing device drivers and provides guidelines for writing additional drivers. Commonly used RM and RMT terminology is identified and described in the Glossary. A complete specification of the RM format is contained in Appendix A. A skeleton FORTRAN program for generating an RM formatted image is listed in Appendix B. Appendices C and D describe the invocation of the RMT in a UNIX environment. Appendix E lists and categories all RMT error messages. Finally, Appendix F lists sample RMT command file templates.

2.0 RM FORMAT

The Raster Metafile (RM) is an attempt by LaRC to standardize the representation of raster image data. By adopting the RM format, the generation and processing of raster image data may be greatly simplified. The proposed RM format does not attempt to incorporate all popular image formats. Instead, the RM format is compatible with image formats commonly used at LaRC.

An RM formatted image is completely specified through a mandatory image header used to describe the image format and a set of opcodes used to delineate entities within the image structure. The header format and individual opcodes are detailed in Appendix A of this document. In most instances, a potential RM user need not be concerned with the exact structure of an RM formatted image. However, some information about the format is necessary for governing processing.

An understanding of the header parameters is mandatory for RM generation and processing and also for device selection. The header supplies image resolution information in the form of an "X" (horizontal-number of pixels per scanline) and a "Y" (vertical-number of scanlines) resolution. A flag distinguishing color from black-and-white data is also provided in the header. The majority of header parameters are used to categorize images with respect to color look-up table (LUT) or intensity. An LUT image is a single channel raster image where each pixel value is an index into an entry in an LUT. In addition to the LUT/intensity flag, the "size" of the LUT and "range" of each entry in the LUT are specified in the header. An intensity formatted image is a multi-channel image whereby channel data is stored field interleaved. That is, all data is stored before data for the next channel is encountered. For an intensity image, the header contains information on the number of channels and the number of bits used to represent each pixel. Finally, the header provides a background color in the form of an LUT index or an intensity for each channel depending on the value of the LUT/intensity flag. For detailed information on header parameters and allowable parameter ranges consult Appendix A.

An RM formatted file may be comprised of several RM formatted images. Each RM formatted image is composed of a header, an optional LUT, and the image data organized by channel. Figure 1 shows the logical structure of an RM formatted file containing an LUT image followed by an intensity image with three channels. Each RM formatted image must be preceded by an image header. The RM format uses the terms channel and "frame" interchangeably, implying that an LUT image consists of a single frame, and an intensity image consists of one or more frames. The individual sections of an RM image are delineated by the opcodes discussed in Appendix A. An RM opcode is designed to accommodate up to 12 bits of data in a 16-bit word and up to 24 bits of data when used with the continuation opcode. Typically, the user need not be concerned with the details of the RM opcodes except to note that the 12 and 24 data bits limit image parameters such as image resolutions or LUT size and range.

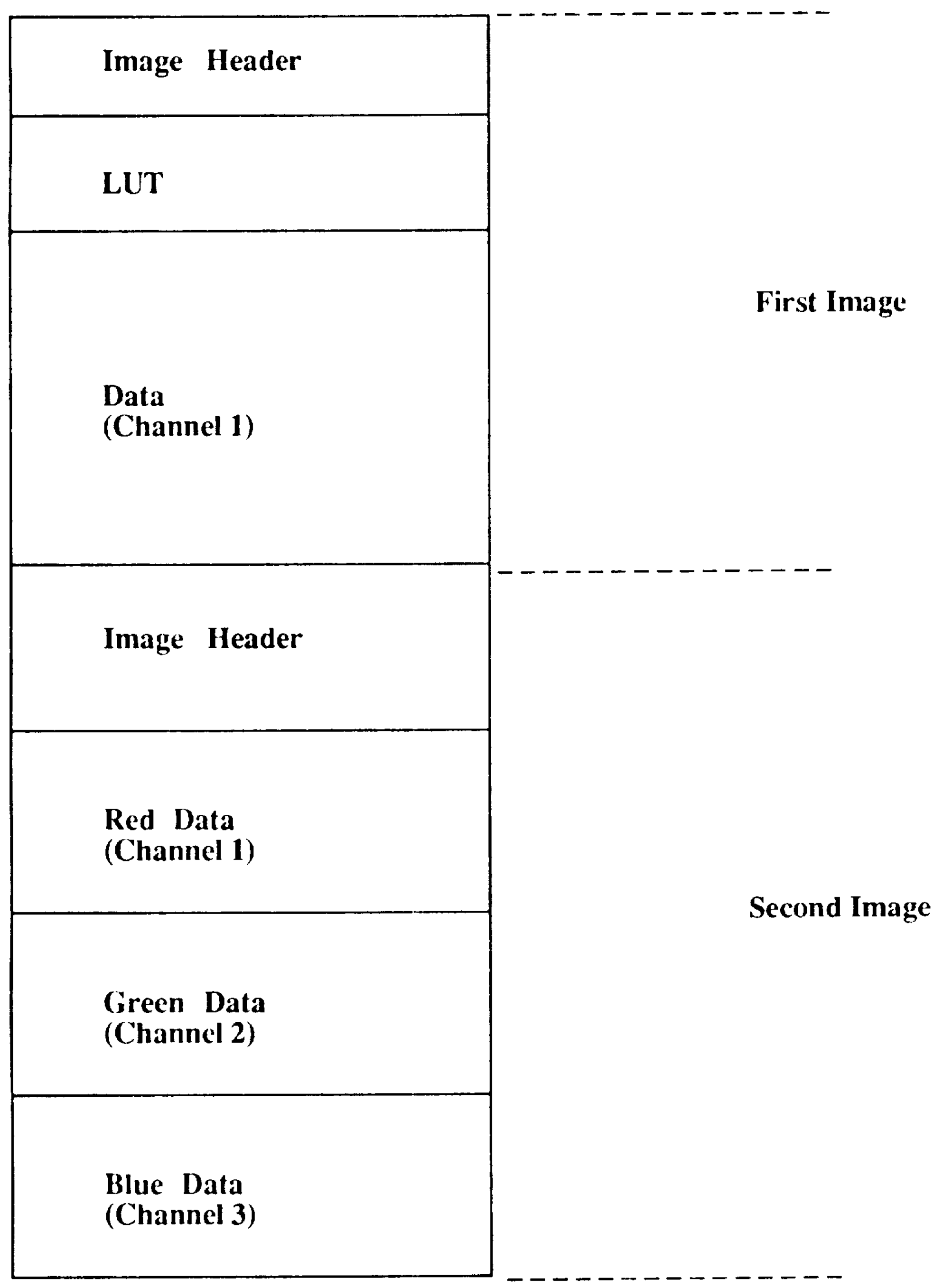


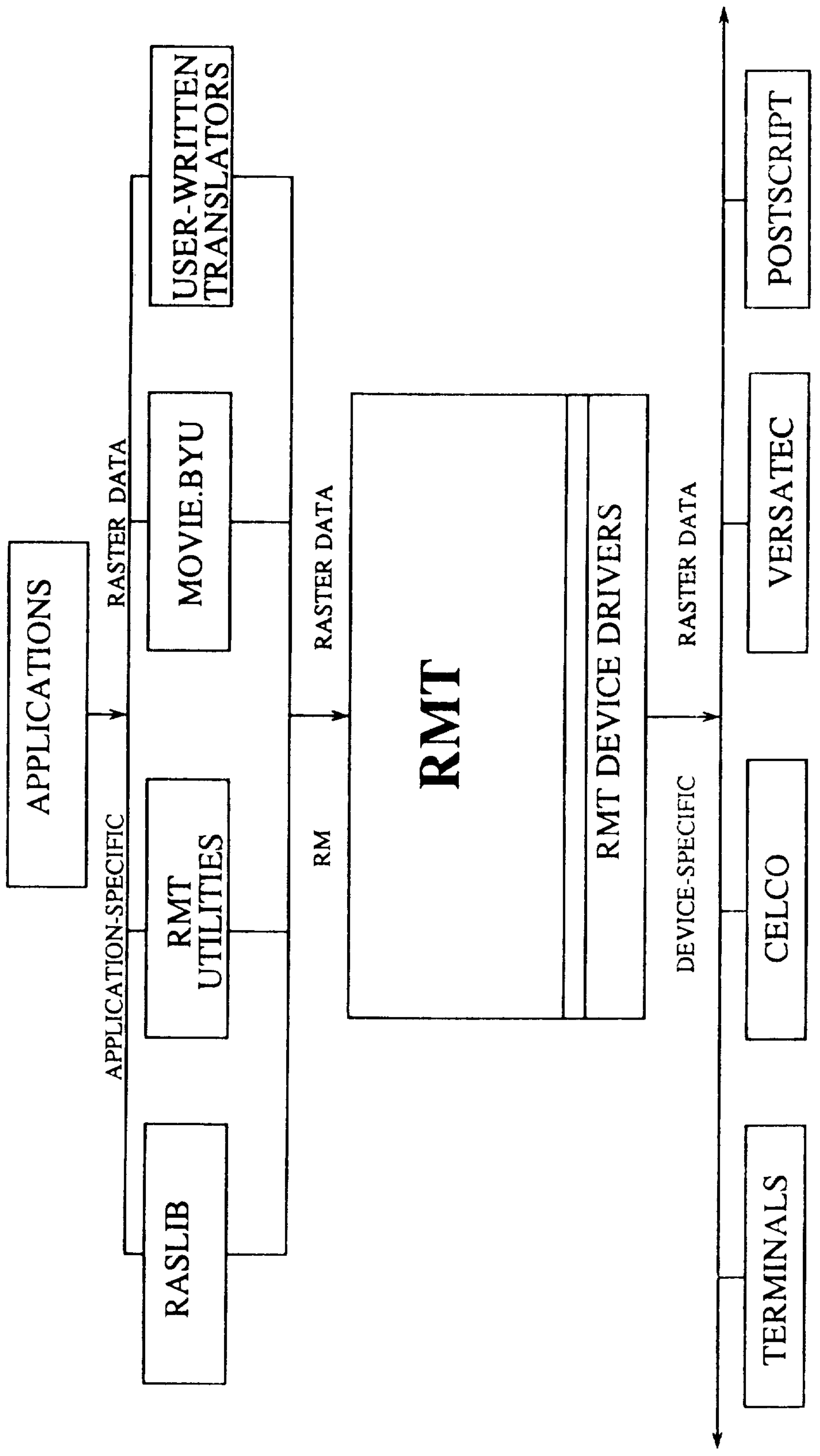
Figure 1.- RM Image Format Logical Structure

3.0 RM PROCESSING

The specification of a generic raster image format does not eliminate the requirement for data format conversion. It remains necessary to convert package-specific data into RM format and to convert RM data into device-specific data. However, software tools are provided to simplify the conversion from package-specific formats to RM format. Additionally, translation from RM to device-specific format is simplified through the invocation of a collection of software device drivers.

As shown in Figure 2, the translation from package-specific formats to RM format may take several forms. Using the RM format specifications, user-written translators may be developed. If a user does not wish to be burdened with the details of the RM format specifications, two alternative paths exist. The translation to RM mechanism may be embedded in the application software, or included as a postprocessing step. The embedded approach has been employed at LaRC for the locally-developed Raster Library (RASLIB) [4] and the locally-supported version of the commercially available MOVIE.BYU software [5] performs postprocessing. In situations where no existing translator is available, an application-independent utility library is provided. This library is the FORTRAN-callable subset of the Raster Metafile Translator (RMT), hereafter referred to as the RMT utility library. A "skeleton" postprocessor for writing an RM formatted image using the utility library is provided in Appendix B.

The process of converting from RM to device-specific formats is handled through software device drivers associated with the RMT. The list of supported raster devices and the specifics of each device interface are discussed in Section 8. The capabilities of the RMT are outlined in Section 7.



Raster Metafile (RM) - LaRC-specified generic raster image format
Raster Metafile Translator (RMT) - Command-driven program to read/write, process, and display RM image data

Figure 2. - RM Processing

4.0 RMT OVERVIEW

The Raster Metafile Translator (RMT) is a command-driven program with the primary function of processing Raster Metafile (RM) formatted raster image data. RMT processing includes reading, writing, manipulating, and displaying RM images.

The principal input to the RMT is one or more RM formatted image files. In the most basic RMT command scenario, an RM file is read as input and displayed on a user-selected raster graphics output device. If the selected device is a supported terminal or workstation, the image is drawn interactively. In the case of a hardcopy device, a device-specific raster file is produced which must be ported to the appropriate device for processing.

Under certain circumstances, an input RM file may require additional processing. For example, the RM formatted image may be incompatible with the selected device. Assume that the RM image is in intensity format, that is, the data is represented by red, green, and blue channel components. Further assume that the selected device is a Tektronix 4125 terminal which requires raster data to be expressed in terms of an index into a color look-up table (LUT). One manipulation function of the RMT is to convert an intensity/color LUT based RM image into a color LUT /intensity based RM image. In this instance, the function of the RMT is not to display an image but to write an output RM file.

Other image manipulation operations such as a minimal compositing operator and a resizing option are available under the RMT command structure. The RMT command set is detailed in Section 7.

5.0 RMT USER INTERFACE

The RMT is an interactive command-driven program but may be exercised in a background environment. Because Precision Visuals, Inc. (PVI) software (including the Metafile Translator [6]) is used extensively at LaRC, the RMT user interface is patterned after that used in PVI's Metafile Translator.

The RMT user interface is diagrammed in Figure 3. Commands may be input one-at-a-time or read from an alternate command file. The RM formatted image file(s) may be output from as well as input to the RMT. RMT output may take the form of an image displayed on a terminal screen or a device-specific raster output file depending on the selected device. As explained later in the document (see Section 8), device selection is performed at program load time. The RMT also optionally outputs a log file that may serve as an audit trail of the RMT session or may be invoked as an alternate command file in a subsequent RMT session.

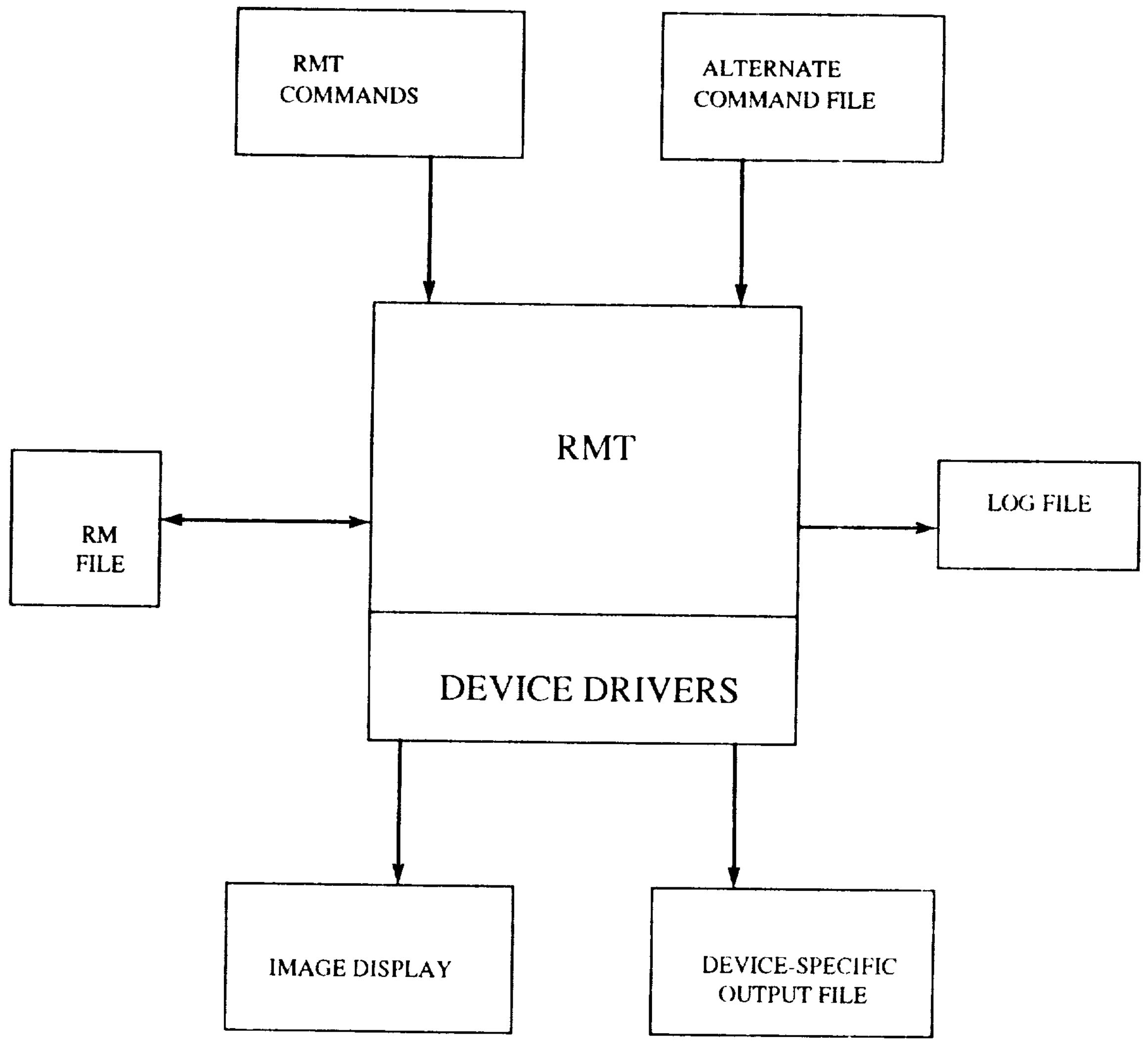


Figure 3. - RMT User Interface

6.0 RMT CONVENTIONS AND TERMINOLOGY

Before presenting individual RMT commands, it is instructive to discuss certain RMT conventions and relevant terminology. In many cases, these conventions refer to the RM format with emphasis on the image header parameters (see Section 2).

The RMT employs an integer coordinate system based on RM image resolution. The RMT coordinates of an RM image with a horizontal resolution of 640 and vertical resolution of 480 range in "X" from 0 to 639 and in "Y" from 0 to 479. The (0,0) RMT coordinate refers to the first pixel encountered on the first scanline, and coordinates are incremented left to right and top to bottom.

The RMT has adopted the graphics constructs of a window and a viewport. An RMT window represents a rectangular subset on an input RM image. An RMT viewport represents a rectangular area of the display surface on which an image is to be drawn. Consequently, an RMT window is mapped into an RMT viewport in the traditional [7] fashion. By default, if the window is larger than the viewport, it is clipped at the right and bottom edges of the viewport. If clipping is inhibited, the window is resized to "fit" the viewport boundaries. This resizing of an image (either "up" or "down") is accomplished using a bilinear interpolation scheme adopted from [8].

An RM directory is a list of the image header parameters presented one image at a time. The RMT groups the metafile number, image (or picture) number, window number, and viewport number into a single entity referred to as a component. Many RMT commands reference this component entity when processing RM formatted files.

Another RMT compound entity is the attribute (or attribute-bundle) which is used to group RMT processing attributes. An example of an attribute in the attribute-bundle is the "clipping/resize" flag. As is the case with components, many RMT commands collectively refer to all attributes using this single entity.

One of the major RMT processing operations is the conversion from/to color LUT to/from intensity format. The conversion from LUT to intensity format is relatively straightforward but the conversion from intensity to LUT is complicated due to the requirement of constructing an appropriate look-up table. Several RMT attributes control the building of the table. The RMT allows for either "uniform" or "color-cube" quantization [9] via an attribute flag. In other cases where a limited number of colors are available, an ordered dithering [10] may be selected to override the quantization flag. For either quantization or dithering, the "size" of the LUT and/or the "range" of LUT entries must be specified by setting the appropriate attributes. There is no firm rule for determining which quantization scheme or dithering scheme should be selected. The process is image dependent and may require experimentation on the part of the user.

By default, the conversion operation preserves whether an image is color or black-and-white. However, by turning the color attribute "off," the result of the conversion operation is a single channel intensity formatted image. The algorithm used for conversion from color to grey scale is:

GREY = 0.3 * RED + 0.59 * GREEN + 0.11 * BLUE

A final RMT operation for tailoring an RM image to a specific device is called scaling. For LUT formatted images, scaling is used to reduce either the "size" of the LUT and/or the "range" of an LUT entry. Only the range of intensity values may be scaled for intensity formatted images. Again, the scaling process is controlled with individual RMT attributes.

The only image composition operator available in the current release of the RMT is overlay. As the name implies, one or more input images are overlaid to form a single output image. The resolution of the output image is determined by the viewport extent of all input images. The images are overlaid in the order they appear on the command line implying that the last input image may obscure all or portions of the other input images. In the default mode, the entire input image is used in the overlay process. However, by activating the "no background color" flag (see Section 7.10.1.2) background pixels are ignored. The overlay operation may be performed only when all input images are in a compatible format. For example, it is not possible to overlay a color LUT and an intensity image. (For further information on overlay restrictions see Section 7.7).

7.0 RMT COMMANDS

The RMT is a command-driven software utility. The commands are organized into a three-level hierarchy. The top level RMT commands are listed in Table 1. The SET and GET commands have a subcommand level which is listed in Table 2. The SET ATTRIBUTE and SET COMPONENT commands have another subcommand level as shown in Table 3. The following subsections detail individual RMT commands including the following information:

name
description
abbreviation
syntax
defaults
limits
example

The command information found in the following subsections may also be obtained by invoking the RMT on-line HELP facility.

Before proceeding to the individual command descriptions, the documentation conventions should be explained. Command names and arguments are case insensitive. The only exception to this rule is commands where one of the arguments is a file name such as SET METAFILE or SOURCE. In these instances, the case is preserved in order that both upper and lower case characters may be entered. Most command names and many command arguments may be abbreviated. The normal abbreviation convention is the first three characters of the name, but the individual command descriptions should be consulted for specifics.

In most instances, RMT commands have an accompanying collection of mandatory and optional parameters with associated defaults. Optional parameters are enclosed in "[ ]." Parameter value descriptions are often delimited by "< >." The "#" character is used to represent a numeric parameter value whose allowable range is specified in the command description.

Defaults are provided for all optional parameters and are command specific. In the case of commands which use components and/or attributes, there are two levels of defaults. The commands use default numbered components and/or attributes, and the parameters comprising each component or attribute are also defaulted. For example, issuing the RMT command abbreviation "D" implies not only to use the default component number "1," but to use the values that comprise component "1" including the metafile, picture, window, and viewport numbers which also default to "1's."

Components and attributes normally occur in pairs. The component identifies a subset of an RM formatted image and the attributes control the manner in which this subset is processed subject to the specific command. Because of this pair relationship, defaults have been established by which one of the pair may be omitted. For example, entering the command "D C 2" assumes an attribute number "2" and is expanded to "D C 2 A 2" by the RMT.

For commands that require both input and output metafiles, the first component refers to input and the second component refers to output. The OVERLAY command, which may involve several input components, assumes the last component refers to output.

Basic Commands

CONVERT
COPY
DIRECTORY
DRAW
GET
HELP
OVERLAY
QUIT
SCALE
SET
SOURCE

Table 1. - RMT Commands

GET Subcommand Name	SET Subcommand Names
ATTRIBUTE	ATTRIBUTE
COMPONENT	COMPONENT
METAFILE	ERROR
OPTION	METAFILE
VIEWPORT	LOG
WINDOW	VIEWPORT
	WINDOW

Table 2. - GET and SET Subcommands

SET ATTRIBUTE Subcommand Names	SET COMPONENT Subcommand Names
BGC (background color)	METAFILE
BGF (background color flag)	PICTURE
BPC (bits per channel)	VIEWPORT
CHANNEL	WINDOW
CLIP	
COLOR	
DITHER	
OPERATOR	
QUANTIFY	
RANGE	
SCALE	
SIZE	

Table 3. - SET ATTRIBUTE and SET COMPONENT Subcommands

7.1 CONVERT

Convert an RM formatted image from intensity (color table) to color table (intensity) format.

```
CONVERT [COMPONENT <input #>] [ATTRIBUTE <input #>] [COMPONENT <output
#>]
CONV [C #] [A #] [C #]
```

CONVERT has no mandatory parameters. The default input component and input attribute numbers are "1" and default output component is "2." (If a component/attribute is designated without an attribute/component, the attribute/component is assigned the component/attribute number.)

When converting from intensity to color table format, the input attribute contains color LUT information. The relevant attributes are LUT "range," "size," and "bits per channel" (see SET ATTRIBUTE for RANGE, SIZE, and BPC subcommands). The use of these LUT attributes is controlled by the "quantization" or "dithering" scheme selected (see SET ATTRIBUTE for QUANTIFY and DITHER subcommands).

When the black-and-white flag is "set" (see SET ATTRIBUTE COLOR), the conversion process produces a single channel intensity image.

```
CONV
CONV C 1 A 1 C 2
CONV C 1 C 2
```

7.2 COPY

Copy designated RM formatted images from an input metafile to the end of an output metafile.

COPY [COMPONENT <input #>] [COMPONENT <output #>]
COPY [C #] [C #]

COPY has no mandatory parameters. The default input component number is "1" and default output component number is "2."
This command does not use attributes.

7.3 DIRECTORY

Display the contents of an RM formatted image file by printing "header" information for the selected image range.

DIRECTORY #
DIR #

DIRECTORY mandatory parameter is the metafile number (from 1-5).
Optional parameters include the image "range."
If the range is omitted, the "headers" for all images on the file are displayed.

The "range" specifcation is available on several commands and takes the form:

FROM <first image> TO <last image>

where the image designators may be the keywords FIRST, CURRENT, or LAST; absolute (integer) image numbers; or relative (integer) offsets designated by "+" or "-."

DIR # [FROM FIRST TO LAST]

```
DIR # [FROM 2 TO 4]
DIR # [FROM -1 TO +2]
```

7.4 DRAW

Draw RM formatted image to selected graphics output device.

```
DRAW [COMPONENT #] [ATTRIBUTE #]
D [C #] [A #]
```

DRAW has no mandatory parameters. The default input component and input attribute numbers are "1."
(If a component/attribute is designated without an attribute/component, the attribute/component is assigned the component/attribute number.)

The DRAW command aborts if the RM image and the selected output device are incompatible.

```
DRAW
DRAW C 1 A 1
DRAW C 1
```

7.5 GET

Get current attribute, component, and option information.

```
GET subcommand [parameters]
```

GET mandatory parameter is the subcommand name.
The GET subcommands include:

ATTRIBUTE (A)
COMPONENT (C)

METAFILE (MF)
OPTION (OPT)
WINDOW (W)
VIEWPORT (V)

The GET subcommands are documented individually.

7.5.1 GET ATTRIBUTE

Inquire about attributes associated with specified attribute number.

```
GET ATTRIBUTE #
GET A #
```

GET ATTRIBUTE mandatory parameter is the attribute number (from 1-10).
The information returned includes:

background color (BGC),
no background color flag (BGF),
bits / LUT channel (BPC), and
number of data channels and data channel ordering (CHA),
clipping / resize flag (CLI),
color / black-and-white flag (COL),
dither flag (DIT),
color-cube / uniform quantization flag (QUA),
composition operator (OPE),
LUT range, (RAN)
number of bits for scaling (SCA), and
LUT size (SIZ).

(See SET ATTRIBUTE [subcommand] for more detailed information on individual attributes.)

7.5.2 GET COMPONENT

Inquire about components associated with specified component number.

GET COMPONENT #
GET C #

GET COMPONENT mandatory parameter is the component number.
The information returned includes:
metafile number,
picture number (or range),
viewport number, and
window number.
(See SET COMPONENT for more detailed information on individual components.)

7.5.3 GET METAFILE

Inquire about metafile name associated with specified metafile number.

GET METAFILE #
GET MF #

GET METAFILE mandatory parameter is the metafile number (from 1-5).

7.5.4 GET OPTION

Inquire about supported RMT options.

```
GET OPTION
GET OPT
```

No parameters.

Reported options include:

"log" file status,

alternate command file name (if any), and

error severity level.

7.5.5 GET VIEWPORT

Inquire about viewport coordinates for specified viewport number.

```
GET VIEWPORT #
GET V #
```

GET VIEWPORT mandatory parameter is the viewport number (from 1-10).

7.5.6 GET WINDOW

Inquire about window coordinates for specifed window number.

```
GET WINDOW #
GET W #
```

GET WINDOW mandatory parameter is the window number (from 1-10).

7.6 HELP

Access the on-line "help" documentation for individual RMT commands.

```
HELP
HELP [command]
HELP [command] [subcommand]
```

The HELP command has no mandatory parameters but optional parameters may be included for specific commands and subcommands.
HELP with no arguments produces a list of all supported commands and subcommands.

7.7 OVERLAY

Overlay one or more RM formatted images to produce a composite image.

```
OVERLAY [COMPONENT <input #>] [ATTRIBUTE <input #>]...[COMPONENT
<output #>]
OVER [C #] [A #] ... [C #]
```

OVERLAY has no mandatory parameters. The default input component and input attribute numbers are "1" and default output component is "2."
Optional parameters include the additional component and attribute pairs that describe the individual component images.
(If a component/attribute is designated without an attribute/component, the attribute/component is assigned the component/attribute number.)

Component images must be compatible. Compatibility is measured by comparing component image header parameters. The header of the first component image is used as the baseline.

The resolution of the composite image is determined by the "extent" of supplied viewport dimensions.

The specification of a picture "range" in a component definition is not permitted.

By setting the "no-background-color" flag (see SET ATTRIBUTE BGF subcommand), the background color pixels of the component images are not written to the output metafile.

```
OVER C 1 A 1 C 2 A 2 C 3
OVER C 1 C 2 C 3
```

7.8 QUIT

Exit the RMT program.

```
QUIT
Q
```

No command parameters.

7.9 SCALE

Scale the number of data bits in an input intensity image or size of LUT entries in an input color table image.

```
SCALE [COMPONENT <input #>] [ATTRIBUTE <input #>] [COMPONENT <output #>]
```

```
SCA [C #] [A #] [C #]
```

SCALE has no mandatory parameters. The default input component and input attribute numbers are "1" and default output component is "2." (If a component/attribute is designated without an attribute/component, the attribute/component is assigned the component/attribute number.)

The number of bits by which to scale an LUT entry or intensity data or LUT size are specified in the SCALE attributes (see SET ATTRIBUTE SCALE).

```
SCA
SCA C 1 A 1 C 2
SCA C 1 C 2
```

7.10 SET

Set components, attributes, and other options.

```
SET subcommand [parameters]
```

The available SET subcommand are:

- ATTRIBUTE (A)
- COMPONENT (C)
- ERROR (ERR)
- LOG (L)
- METAFILE (MF)
- VIEWPORT (V)
- WINDOW (W))

Set subcommands are documented individually.

7.10.1 SET ATTRIBUTE

Set attribute-bundle parameters.

```
SET ATTRIBUTE # [parameters]
SET A # [parameters]
```

SET ATTRIBUTE mandatory parameter is the attribute number.
Optional attribute parameters include:

background color (BGC),
no background color flag (BGF),
bits / LUT channel (BPC),
number of data channels and data channel ordering (CHA),
clipping / resize flag (CLI),
color / black-and-white flag (COL),
dither flag (DIT),
color-cube / uniform quantization flag (QUA),
composition operator (OPE),
LUT range (RAN),
number of bits for scaling (SCA), and
LUT size (SIZ).

(See SET ATTRIBUTE [subcommand] for more detailed information on individual attributes.)

If the SET ATTRIBUTE command extends over a single input line, the "continuation" character "&" may be used or a second SET ATTRIBUTE command (using the same number) may be issued.

7.10.1.1 SET ATTRIBUTE BGC

Override the default image background color.

```
SET ATTRIBUTE # BGC (#1 #2 ... )
SET A # BGC (#1 #2 ... )
```

SET ATTRIBUTE BGC mandatory parameters include the background color table index for color LUT images or the background intensity for each channel in an intensity formatted image.

Attribute is not currently enabled.

7.10.1.2 SET ATTRIBUTE BGF

Set the no-background-color flag used to determine if background color is to be ignored when compositing (see OVERLAY) images.

```
SET ATTRIBUTE # BGF ON
SET A # BGF +
```

The only parameter is the no-background-color flag ON/OFF or +/-. (The default is "OFF" implying use background color.)

7.10.1.3 SET ATTRIBUTE BPC

Specify the number of bits to be used per channel in color LUT.

```
SET ATTRIBUTE # BPC # (#1 #2 #3 ... )
SET A # BPC # (#1 #2 #3 ... )
```

SET ATTRIBUTE BPC mandatory parameters are the number of bits to be used per channel in a color LUT.
(Default values are "0.")

This attribute is applied when "converting" an image from intensity to color LUT format using "uniform quantization."

7.10.1.4 SET ATTRIBUTE CHANNEL

Override the default number of channels and/or the channel selection order for multi-channel image data.

```
SET ATTRIBUTE # CHANNEL # (#1 #2 #3 ... )
SET A # CHA # (#1 #2 #3 ... )
```

SET ATTRIBUTE CHA mandatory parameters include the number of channels and the channel ordering.
By default the RMT assumes 3 channels ordered 1, 2, and 3 corresponding to RED, GREEN, and BLUE channels. The user may alter this by supplying a new number of channels and/or channel ordering.
(Supplying "0" for the number of channels, informs the RMT, to use the number of channels in the input metafile image.)

7.10.1.5 SET ATTRIBUTE CLIP

Set the "clip" flag used to determine whether to "clip" image at viewport boundaries or "resize" image (using a bi-linear interpolation scheme) to fit the viewport.

SET ATTRIBUTE # CLIP ON
SET A # CLI +

SET ATTRIBUTE CLI mandatory parameter is the clip flag ON/OFF or +/-.
(The default is "ON" implying clipping.)

7.10.1.6 SET ATTRIBUTE COLOR

Set "color" flag used in conversion process when converting from (to) color to (from) black-and-white.

SET ATTRIBUTE # COLOR ON
SET A # COL +

SET ATTRIBUTE COL mandatory parameter is the "color" flag ON/OFF or +/-.
(The default is ON.)
The effect of the "color" flag being OFF is to produce a single channel intensity image (corresponding to a black and white image) during a conversion operation (see CONVERT command).

7.10.1.7 SET ATTRIBUTE DITHER

Set the "dither" flag used in converting from intensity to color table format.

SET ATTRIBUTE # DITHER ON
SET A # DIT +

SET ATTRIBUTE DIT mandatory parameter is the dither flag ON/OFF or +/-.

(The default is "OFF.") If dither is "ON," it overrides the quantization flag (see SET ATTRIBUTE QUANTIFY).

The "ordered dithering" scheme produces a color LUT image with an LUT of "size" "8" and color table entry "range" as declared with the SET ATTRIBUTE RANGE command.

7.10.1.8 SET ATTRIBUTE OPERATOR

Designate image composition operator.

```
SET ATTRIBUTE # OPERATOR <operator name>
SET A # OPE <operator name>
```

Attribute is not currently enabled but is hardwired to the overlay operator (see OVERLAY).

7.10.1.9 SET ATTRIBUTE QUANTIFY

Set the "uniform quantization" flag used in converting from intensity to color table format.

```
SET ATTRIBUTE # QUANTIFY ON
SET A # QUA +
```

SET ATTRIBUTE QUA mandatory parameter is the quantify flag ON/OFF or +/-. "ON" implies the use of uniform quantization and "OFF" implies the use of a color cube quantization scheme. (The default is "ON.")

7.10.1.10 SET ATTRIBUTE RANGE

Specify "range" of color LUT channel entry.

```
SET ATTRIBUTE # RANGE <range>
SET A # RAN <range>
```

SET ATTRIBUTE RANGE mandatory parameter is the "range" (maximum minus minimum) of a color LUT entry expressed as a positive integer.
(Default is "0.")

This attribute is applied when "converting" an image from intensity to color LUT format.

7.10.1.11 SET ATTRIBUTE SCALE

Specify the number of bits to be used when "scaling" an RM image.

```
SET ATTRIBUTE # SCALE #1 #2
SET A # SCA #1 #2
```

SET ATTRIBUTE SCALE mandatory parameters are the two "scaling" parameters. The first parameter controls the "range" of an LUT entry or the number of data bits in an intensity image. The second parameter controls the "size" of the LUT for a color LUT formatted image.
(Default values are "0.")

The allowable range for LUT "range" or intensity scaling is 1 to 12.
The allowable range for LUT "size" scaling is 3 to 12.

CAUTION: The default scaling parameters are not in the allowable range, which signifies that the input image header values associated with the scaling parameter(s) will also be used for the output image. In this manner either type of scaling may be bypassed.

7.10.1.12 SET ATTRIBUTE SIZE

Specify "size" (or length) of color LUT.

```
SET ATTRIBUTE # SIZE <size>
SET A # SIZ <size>
```

SET ATTRIBUTE SIZE mandatory parameter is the "size" (length) of color LUT expressed as a positive number of bits.
(Default is "0.")

This attribute is applied when "converting" an image from intensity to color LUT format.

7.10.2 SET COMPONENT

Set component parameters.

```
SET COMPONENT # [METAFILE #] [PICTURE #] [VIEWPORT #] [WINDOW #]
SET C # [MF #] [P #] [V #] [W #]
```

SET COMPONENT mandatory parameter is the component number.
Optional attribute parameters include:
metafile number,
picture number,

viewport number, and
window number.
The default number for all four component parameters is "1."

Metafile numbers range from 1 to 5.

The picture number may be a "range" (see DIRECTORY command for range specification).

Viewport and window numbers range from 1 to 10.
The viewport and window cooordinates should be specified with the
SET VIEWPORT or SET WINDOW commands.

7.10.3 SET ERROR

Set error termination level.

```
SET ERROR <error termination level>
SET ERR   <error termination level>
```

SET ERROR mandatory parameter is the error termination level.
Possible candidates are: SYNTAX (S), WARNING (W), FATAL (F), NONE (N).
(Default is FATAL).

7.10.4 SET LOG

Toggle the flag to write the RMT log file, RMT.LOG.

```
SET LOG ON
SET L +
```

SET LOG mandatory parameter is the write flag ON/OFF or +/-.
(The default is "OFF.")

The "log" file may be used as RMT input in the form of an
alternate command file (see SOURCE command).

7.10.5 SET METAFILE

Identify Raster Metafile (RM) for reading or writing.

```
SET METAFILE # <file name>  [WRITE]
SET MF # <file name> [W]
```

SET METAFILE mandatory parameters are the metafile number and file name.
The "write" flag is optional; when omitted, the metafile is opened
for reading only.
Metafile numbers may range from 1 to 5.
("Write" flag defaults to "false.")

CAUTION: When a metafile is opened for writing, no other "read"
processing operations are allowed (without first exiting the RMT).
(That is, commands like DIR and DRAW are not allowed for "output"
metafiles.)

CAUTION: The "output" metafile required for several commands (such
as CONVERT or OVERLAY) must be opened for writing.

7.10.6 SET VIEWPORT

Set an RMT viewport (used to position and size an RM image for display or output.

```
SET VIEWPORT # (<x-minimum> <x-maximum> <y-minimum> <y-maximum>)
SET V # (<x-minimum> <x-maximum> <y-minimum> <y-maximum>)
```

SET VIEWPORT mandatory parameters include the viewport number and dimensions in terms of minimum and maximum horizontal and vertical extents. Viewport numbers may range from 1 to 10.
Viewport coordinates may range from 0 to 4095 but should reflect both image and device dimensions. Viewport "1" is defaulted to the resolution of the selected device while the other viewports default to (0 0 0 0).

7.10.7 SET WINDOW

Set an RMT window (used to designate a subset of an RM formatted image).

```
SET WINDOW # (<x-minimum> <x-maximum> <y-minimum> <y-maximum>)
SET W # (<x-minimum> <x-maximum> <y-minimum> <y-maximum>)
```

SET WINDOW mandatory parameters include the window number and dimensions in terms of minimum and maximum horizontal and vertical extents. Window numbers may range from 1 to 10.
Window coordinates may range from 0 to 4095 but should reflect both image and device dimensions. Window "1" is defaulted to (0 4095 0 4095) while the other windows default to (0 0 0 0).

7.11 SOURCE

Read command input from alternate command file.

```
SOURCE <filename>
SOU <filename>
```

SOURCE mandatory parameter is the alternate command file name.

Nesting of alternate command files is not permitted.

8.0 RMT DEVICE INTERFACE

One of the major RMT functions is the display of RM formatted images. As illustrated in Figure 3, RMT output is in one of three formats: RM file, terminal/workstation display, or device-specific file. In all three instances, RMT output is controlled by selecting a device and issuing the appropriate RMT command.

The selection of a graphics output device is performed at load time. Although this process is host-dependent (see Section 9), it is necessary for the user to access the RMT via a command procedure with a parameter designating the desired device. For example, in the UNIX environment, entering

```
rmtran tek4109
```

invokes a shell script which links the required RMT modules and libraries with the Tektronix 4109 device driver to build and invoke the executable image

```
a.out
```

A listing of the *rmtran* script is provided in Appendix C.

Invoking the DRAW command (see Section 7.4) from within the RMT causes the "display" of an RM image. If the selected device is an interactive terminal or workstation, the device driver formats the data for the device and writes it to the display surface. In the case of a graphic hardcopy device, a device-specific output file is written. The porting of this file to the actual device is installation dependent.

A list of supported devices is provided in Table 4. There is no explicit device driver for the CELCO film recorder because the CELCO system software accepts RM formatted input. The DICOMED, PostScript, Sun, and Versatec device drivers produce device-specific output files. This device driver list is dynamic in nature. The RMT has been designed so that new devices may be interfaced with relative ease (See section 13).

Commands such as CONVERT and OVERLAY (see Sections 7.1 and 7.7) produce output in the form of RM files. For these and other commands, no graphics device driver needs to be linked. A "dummy" device driver has been provided for this purpose. The "dummy" device is linked with the RMT in the same manner as other devices.

Device-specific access information is provided in the next subsection. Additional device drivers may be interfaced with the RMT. The approach to constructing new drivers is detailed in section 13.1.

Device Name	Device Type
AED 767	Terminal
CELCO CFR4000	Film Recorder
DICOMED D47	Film Recorder
PostScript (Color/Grey Scale)	Laser Printer
Sun (Color/Monochrome)	Pixrect Image Format
Tektronix 410x	Terminal
Tektronix 41xx	Terminal
Versatec ECP-42	Color Electrostatic Plotter
Versatec C2756	Thermal Plotter

Table 4. - RMT Supported Devices

8.1 Device Selection

A variety of devices are available to the RMT user. These devices run the gamut from high resolution film recorders to laser printers (see Table 4). Each of these devices may be categorized by a collection of characteristics such as: color table or intensity, resolution, number of channels, and size or range of the LUT. Notice that these device characteristics closely parallel the information found in the RM image header (see Section 2). In order to avoid RMT errors and/or incorrect image output, the user must ensure compatibility between the RM image and the selected device.

There are a variety of device versus image compatibility issues which must be resolved such as: intensity versus color table, resolution, and color table or intensity composition. If an incompatibility is encountered when an image is to be displayed, the RMT either aborts the DRAW command with an error message or attempts to perform some corrective action that may produce undesired results. In either case, the user has the option of either selecting another device which is more compatible with the RM image or invoking the CONVERT, SCALE or SET commands modifying the RM image characteristics to look more like those of the device. The various forms of incompatibility together with suggested corrective actions are detailed in the following paragraphs.

A device may be classified as either color table-based or intensity-based. The RMT does not permit the output of color table-based images to intensity-based devices, or vice versa. That is, intensity-based images must be "displayed" on intensity-based devices and color table images must be "displayed" on color table type devices. An alternative display device selection or the CONVERT command (see Section 7.1) may be employed to eliminate the incompatibility.

The second image versus device incompatibility issue concerns image and device resolutions. All of the RMT output devices have a maximum display resolution. If a user attempts to output an image having a resolution larger than the display resolution of the target device then, by default, the output image is clipped (at the right and bottom) to the display device resolution. To alleviate the "clipping" problem, the RMT user has several choices, the simplest of which is to select a higher resolution output device. Another alternative is to use the SET WINDOW command (see Section 7.10.7) to select a region of interest within the image that resides inside the device resolution. Finally, the "SET

ATTRIBUTE # CLIP OFF" command (see Section 7.10.1.5) may be used to "resize" the image to fit within the display boundaries.

The third example of possible image device incompatibility can occur when attempting to output a color table-based image to a color table device whose color table size or range is less than that of the output image. If the image LUT exceeds the size or range of the device LUT, the RMT compensates by using a subset of the LUT with an adjusted entry range, but the quality of the output is unpredictable. In addition to selecting an alternate device, the SCALE command (see Section 7.9) may be used to "scale" the LUT to device characteristics. Another option is to select ordered dithering using the "SET ATTRIBUTE # DIT ON" command (see Section 7.10.1.7) which produces an LUT with eight entries and a user specified range for each entry and no loss in resolution. Because this option is actually intended to be used in converting from color LUT to intensity format, using dithering on a color LUT image is a two step process for the user. The first step is to employ the CONVERT command to create an intermediate intensity RM image, and the second step is to convert the intermediate intensity image back to an LUT image with dithering activated.

The final incompatibility concerns intensity-based devices and intensity based images. The number of data channels in the image must be compatible with the number of channels supported by the device. In general, color devices accept three channels and monochrome devices accept only one channel. For multi-channel images, the "SET ATTRIBUTE # CHA" command (see Section 7.10.1.4) may be used to designate both a desired number of channels and a channel ordering. In the case of monochrome devices, the "SET ATTRIBUTE # COL OFF" command (see Section 7.10.1.6) may be used to convert either LUT or multi-channel intensity images to a single channel intensity image. In addition to matching the number of channels, the number of bits used to specify an intensity value must conform to the device limits. As with LUT images, the RMT automatically truncates intensities to the device maximums, but this may generate undesirable results. The SCALE command may be invoked to "scale" intensity values up or down in order to match the device characteristics.

Now that the variety and severity of image versus device incompatibilities have been explained, the device-specific characteristics of all RMT supported devices must be identified. The characteristics of supported color LUT devices are listed in Table 5, and the characteristics of supported intensity devices are listed in Table 6.

DEVICE	DRIVER NAME	MAXIMUM RESOLUTION	LUT LENGTH	LUT RANGE	OUTPUT
AED767	aed	1024 x 1024	256	[0...255]	
CELCO CFR 4000	celco	4096 x 4096	256	[0...255]	
			1024	[0...1023]	
			2048	[0...2048]	
Dicomed D-47	dicomed	2048 x 2048	256	[0...255]	D47OUT
Sun	sun	1152 x 910	256	[0...255]	sun.img
Tektronix 4109	tek4109	640 x 480	16	[0...100]	
Tektronix 4125	tek4125	1280 x 1024	256	[0...255]	

Table 5. - RMT Color Table Devices

DEVICE	DRIVER NAME	MAXIMUM RESOLUTION	# OF CHANNELS	INTENSITY RANGE	OUTPUT
CELCO CFR4000	celco	4096 x 4096	3	[0...4095]	
PostScript	postscript	4096 x 4096	1	[0...255]	PSOUT
PostScript Color	postscriptcolor	4096 x 4096	3	[0..255]	PSOUT
Sun monochrome	sunmono	1152 x 910	1	[0...1]	Sun.img
Versatec ECP-42	versatec39		3	[0...4095]	VEROUT
DITHER: 8 x 8		1000 x 1000			
4 x 4		2000 x 2000			
2 x 2		4000 x 4000			
1 x 1		8000 x 8000			
Versatec C2756			3	[0...4095]	VEROUT
A size:	versateca				
DITHER: 4 x 4		608 x 608			
2 x 2		1216 x 1216			
1 x 1		2432 x 2432			
B size:	versatecb				
DITHER: 4 x 4		781 x 781			
2 x 2		1562 x 1562			
1 x 1		3124 x 3124			

Table 6. - RMT Intensity Devices

In addition to listing device characteristics, these tables list the driver name to be used by the user when selecting the device, and identify the device-specific output file created by the hardcopy devices.

Several devices listed in Tables 5 and 6 merit further explanation. The CELCO CFR4000 film recorder can function as a color table device or as an intensity device based on RM header parameters. The CELCO film recorder accepts raster metafiles without additional processing and, hence, has no driver name. Additionally, the CELCO film recorder can accept images using 8, 10, or 12-bit color tables or 8, 10, or 12 bits of intensity.

The Sun workstation driver takes two forms: one being the color version, which functions as a color table device, and the other a monochrome version which uses a single bit intensity to determine the on/off status of a pixel. Both drivers write graphical output to a file named sun.img. This file exists in standard Sun Pixrect [11] format and may be displayed on the Sun workstation with Sun-supplied utilities.

The PostScript driver also exists in two forms; the first being the monochrome version, and the second, a color version. The maximum resolution of an image processed with the PostScript drivers is 4096 by 4096, but individual printer memory capacities/configurations may limit this maximum resolution to a lower value.

The Versatec plotters, the ECP-42 color electrostatic plotter and the C2756 color thermal plotter, are devices that can produce graphical output from a maximum of eight colors using different combinations of red, green, and blue. In order to produce more colors, a dithering scheme was introduced, allowing a wider range of colors. The trade-off to this method is that as the number of available colors increases, the usable resolution of the devices is reduced. For example, an image output with a 4 x 4 dither pattern can use approximately 4096 colors. If the same image is produced using an 8 x 8 dither, the number of available colors increases to approximately 262144, but the output resolution of the image is reduced by a factor of two. The user must be aware of this color-resolution trade-off when sending images to the Versatec plotters.

The Versatec drivers differ from other RMT devices in that they prompt the user for device control information. The following partial RMT command session illustrates the Versatec-specific prompts and acceptable values. User responses are typed in bold.

```
d
ENTER DITHER LEVEL:  8,4,2,OR 1
4
ENTER COLOR MAP OPTION
0 = LINEAR MAPPING
1 = LINEAR MAPPING (FIRST 1/2)
2 = LOG MAPPING
0
RMT>
```

The recommended color map option is "0." However, since choosing the most appropriate color map is image dependent, the other two selections have been included for experimentation purposes.

When producing output images for the C2756 thermal plotter (A and B sizes), the user must reverse the channel ordering. The RMT command

```
set a 1 cha 3 (3 2 1 )
```

should be used (see Section 7.10.1.4) prior to issuing the "DRAW" command. (In this manner, the three pass thermal plotter receives color information in the proper order.)

9.0 RMT ACCESS AND OPERATION

Access to the RMT is host and operating system dependent. These dependencies are addressed in Section 12. In this section, access to and operation of the RMT are illustrated for a sample computing environment. The selected computing environment is a CONVEX 210 running CONVEX 7.1 (a version of UNIX BSD 4.3).

In the UNIX environment, the RMT is accessed from a shell script, *rmtran*. A listing of *rmtran* is provided in Appendix C. Detailed access information, including pathname specification and the setting of environment variables, is provided in Appendix D in the format of a UNIX manual page. The *rmtran* script is used to link the desired driver and invoke the RMT. For example, the command line:

rmtran tek4125

links in the Tektronix 4125 device driver and begins RMT execution.

RMT commands were presented in Section 7. In this section, the operation of the RMT is illustrated through three sample command scenarios. The operation of the RMT is host and operating system independent. Once invoked, the RMT prompts the user with

RMT>

at which time the user may begin entering RMT commands. The three sample RMT command sessions include both command input and output. User input is always preceeded by the RMT prompt line, "RMT>," and is further distinguished from command output through the use of bold type. Annotation in the form of comments delimited by "{ }" following the RMT prompt have been added for command clarification.

Assuming the Tektronix 4125 driver was linked as shown above, the first command scenario illustrates the following:

- using the on-line help facility
- identifying an input RM file
- performing a directory of an RM file
- displaying an RM image
- exiting the RMT

RMT command input and output are shown in Figure 4. RMT command input is case-insensitive. However, file names (such as in the SET MF command) are case sensitive. The output from the DIR command illustrates that an RM may contain several images and these images may be of different types. Notice the use of abbreviations and defaults with the use of the DRAW command. Entering simply "D" implies use "COMPONENT 1" or "D C 1." Also, since "COMPONENT 1" was not reset, "C 1" implies using the default metafile, picture, window, and viewport numbers (all of which are "1"). The result of the DRAW command is the image display provided in Figure 5. Finally, the characteristics of the Tektronix 4125 are compatible with the attributes of the first image but incompatible with the second image.

```
                ***** RASTER METAFILE TRANSLATOR *****

RMT>
{Use the on-line help facility for command information}
help set mf
SET METAFILE (SET MF)
Identify Raster Metafile (RM) for reading or writing.
SET METAFILE # [WRITE]
SET MF # [W]
```

Metafile number is a mandatory parameter. The "write" flag is optional; when omitted the metafile is opened for reading only. Metafile numbers may range from 1 to 5. ("Write" flag defaults to "false.")

CAUTION: When a metafile is opened for writing, no other "read" processing operations are allowed (without first exiting the RMT). (That is, commands like DIR and DRAW are not allowed for "output" metafiles.)

CAUTION: The "output" metafile required for several commands (such as CONVERT or OVERLAY) must be opened for writing.

```
RMT>
{Open RM file as metafile "1"}
set mf 1 scenario1.rm
RMT>
{Examine characteristics of all images on metafile "1"}
dir 1
```

Figure 4. - RMT Command Scenario # 1

```
          ***** RMT HEADER INFORMATION *****

RASTER METAFILE NAME = scenario1.rm
RMT METAFILE NUMBER = 1
RMT METAFILE IMAGE NUMBER = 1

MAXIMUM X AND Y RESOLUTIONS = 1280     1024
COLOR TABLE IMAGE DATA
COLOR IMAGE DATA
NUMBER OF BITS PER COLOR TABLE ENTRY = 8
NUMBER OF DATA BITS PER PIXEL = 8
BACKGROUND COLOR OR INTENSITY =      0


          ***** RMT HEADER INFORMATION *****

RASTER METAFILE NAME = scenario1.rm
RMT METAFILE NUMBER = 1
RMT METAFILE IMAGE NUMBER = 2

MAXIMUM X AND Y RESOLUTIONS = 1024    1024
INTENSITY IMAGE DATA
COLOR IMAGE DATA
NUMBER OF DATA CHANNELS = 3
NUMBER OF DATA BITS PER PIXEL = 12
BACKGROUND COLOR OR INTENSITY =     0      0      0
RMT>
{Draw image using current defaults}
d
RMT>
{Exit the RMT}
q
```

Figure 4. - (Continued)

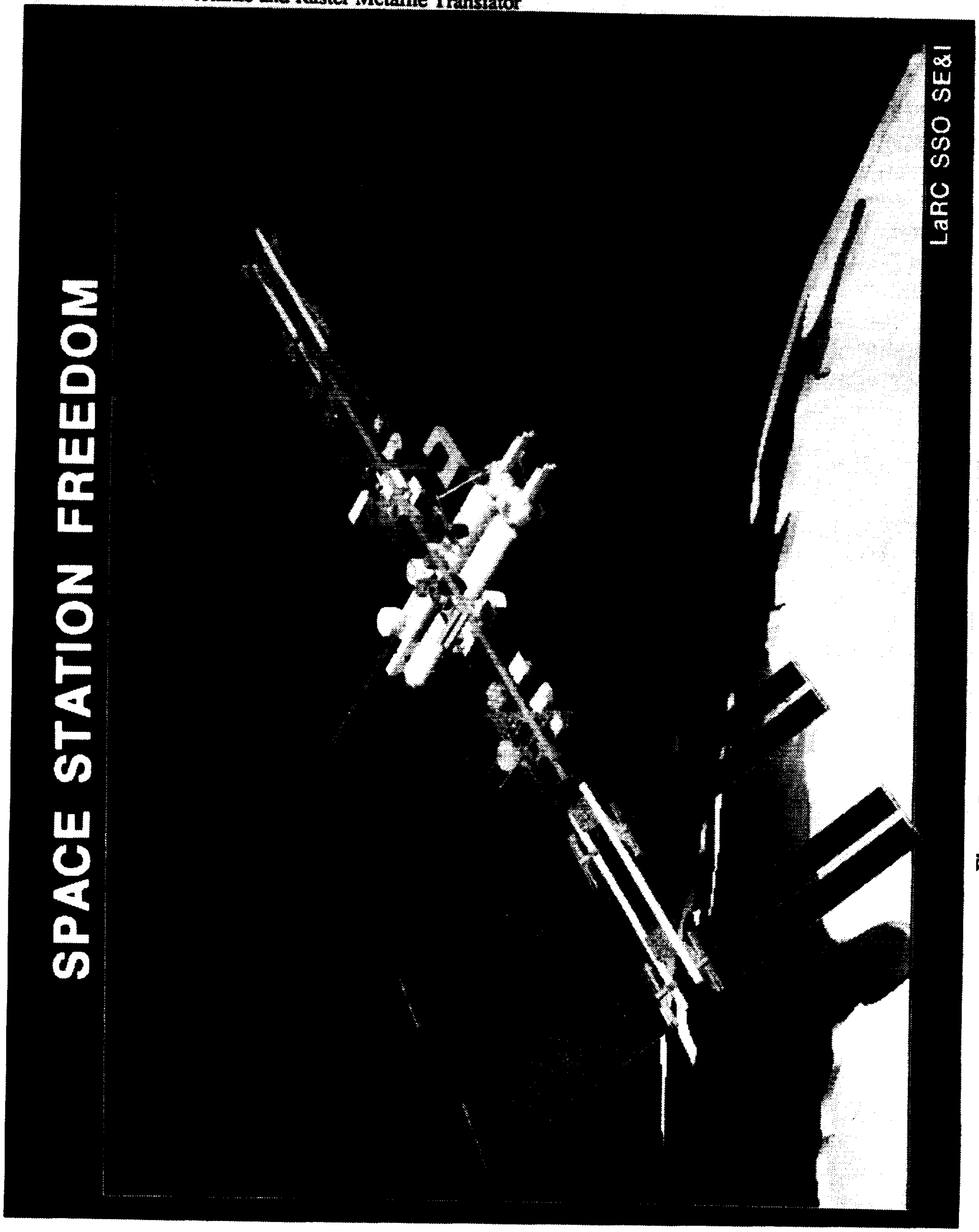


Figure 5.- RMT Command Scenario #1 Output Image

The second command scenario is an application of an image format conversion. In this example, the second image on the metafile used in the first scenario is converted from intensity to color LUT format. No device driver need be linked in this instance, since the primary output is another RM. Consequently, the RMT may be invoked by entering:

rmtran

implying that the "dummy" driver is to be used. The RMT command input and output are provided in Figure 6. Again, some observations concerning the example are important. The "write" flag, "w," is used when designating the output RM. Default component and attribute definitions are not applicable in this case. Consequently, the SET C was used to change the picture number of component "1" from "1" to "2." The GET C or GET A should be used to examine components or attributes. In order to prepare for the conversion operation, the DITHER flag was set and the RANGE of the LUT was set to 100. Notice that several attributes may be set on a single SET A command line. The resultant RM will now be compatible with a Tektronix 4109. The CONVERT command used the component and attribute defaults. CONV implies the command line "CONV C 1 A 1 C 2."

```
***** RASTER METAFILE TRANSLATOR *****

RMT >
{Open the input RM file as metafile "1"}
set mf 1 scenario1.rm
RMT
{Open the output RM file as metafile "2"}
set mf 2 scenario2.rm w
RMT >
{Change the picture number in component "1" to "2"}
set c 1 p 2
RMT >
{Turn on dither option and set LUT entry range to 100}
set a 1 dit + ran 100
RMT >
{Convert second image on metafile "1" to an LUT format}
conv
RMT >
quit
```

Figure 6. - RMT Command Scenario #2

The final RMT command scenario illustrates the image composition operator, OVERLAY. In this example, four component images are combined to form a composite output image. As in the second scenario, no device driver is required since the primary output is an RM. The RMT command input and output are illustrated in Figure 7. All component images must be of compatible format (see Section 6) as shown by the DIR command. Default values were not appropriate for component definitions or the OVERLAY command. The VIEWPORT command is used to position component images within the resultant image. This resultant RM may now be processed as an ordinary RM. The resultant image is displayed in Figure 8.

As part of the standard installation, several templates in the form of RMT alternate command files have been provided to assist users with common RM file operations. These command file templates, named and described in Table 7 and listed in Appendix F, have been annotated with RMT comments for general use.

```
        ***** RASTER METAFILE TRANSLATOR *****
 RMT >
{Open the input metafile }
set mf 1 DITHER.RM
RMT>
{Open  the output metafile}
set mf 2 OVER.RM w
 RMT >
{Ensure component images are compatible}
dir 1

          ***** RMT HEADER INFORMATION *****
 RASTER METAFILE NAME = DITHER.RM
 RMT METAFILE NUMBER = 1
 RMT METAFILE IMAGE NUMBER = 1

 MAXIMUM X AND Y RESOLUTIONS =  512     512
 COLOR TABLE IMAGE DATA
 COLOR IMAGE DATA
 NUMBER OF BITS PER COLOR TABLE ENTRY = 8
 NUMBER OF DATA BITS PER PIXEL = 3
 BACKGROUND COLOR OR INTENSITY =       0

          ***** RMT HEADER INFORMATION *****
 RASTER METAFILE NAME = DITHER.RM
 RMT METAFILE NUMBER = 1
 RMT METAFILE IMAGE NUMBER = 2

 MAXIMUM X AND Y RESOLUTIONS = 512     512
 COLOR TABLE IMAGE DATA
 COLOR IMAGE DATA
 NUMBER OF BITS PER COLOR TABLE ENTRY =  8
 NUMBER OF DATA BITS PER PIXEL =  3
 BACKGROUND COLOR OR INTENSITY =       4
```

Figure 7. - RMT Command Scenario #3

```
    ***** RMT HEADER INFORMATION *****

 RASTER METAFILE NAME = DITHER.RM
 RMT METAFILE NUMBER = 1
 RMT METAFILE IMAGE NUMBER =   3

 MAXIMUM X AND Y RESOLUTIONS =  1024     1024
 COLOR TABLE IMAGE DATA
 COLOR IMAGE DATA
 NUMBER OF BITS PER COLOR TABLE ENTRY = 8
 NUMBER OF DATA BITS PER PIXEL = 3
 BACKGROUND COLOR OR INTENSITY =       0

        ***** RMT HEADER INFORMATION *****

 RASTER METAFILE NAME = DITHER.RM
 RMT METAFILE NUMBER = 1
 RMT METAFILE IMAGE NUMBER = 4

 MAXIMUM X AND Y RESOLUTIONS = 511     484
 COLOR TABLE IMAGE DATA
 COLOR IMAGE DATA
 NUMBER OF BITS PER COLOR TABLE ENTRY =  8
 NUMBER OF DATA BITS PER PIXEL  = 3
 BACKGROUND COLOR OR INTENSITY =       7
 RMT >
{Position four component images on output image using viewports}
set v 1 (0 1023 0 1023)
 RMT >
set v 2 (0 1023 1024 2047)
 RMT >
set v 3 (1024 2047 0 1023)
 RMT >
set v 4 (1279 1791 1279 1791)
 RMT >
{Use components to associate input images with the proper viewport}
set c 2 p 2 v 2
 RMT >
```

Figure 7. - (continued)

```
set c 3 p 3 v 3
 RMT >
set c 4 p 4 v 4
 RMT >
{Use component "5" to identify the output metafile}
set c 5 mf 2
 RMT >
{Overlay the four component images onto the output metafile}
over c 1 c 2 c 3 c 4 c 5
 RMT >
quit
```

Figure 7. - (continued)

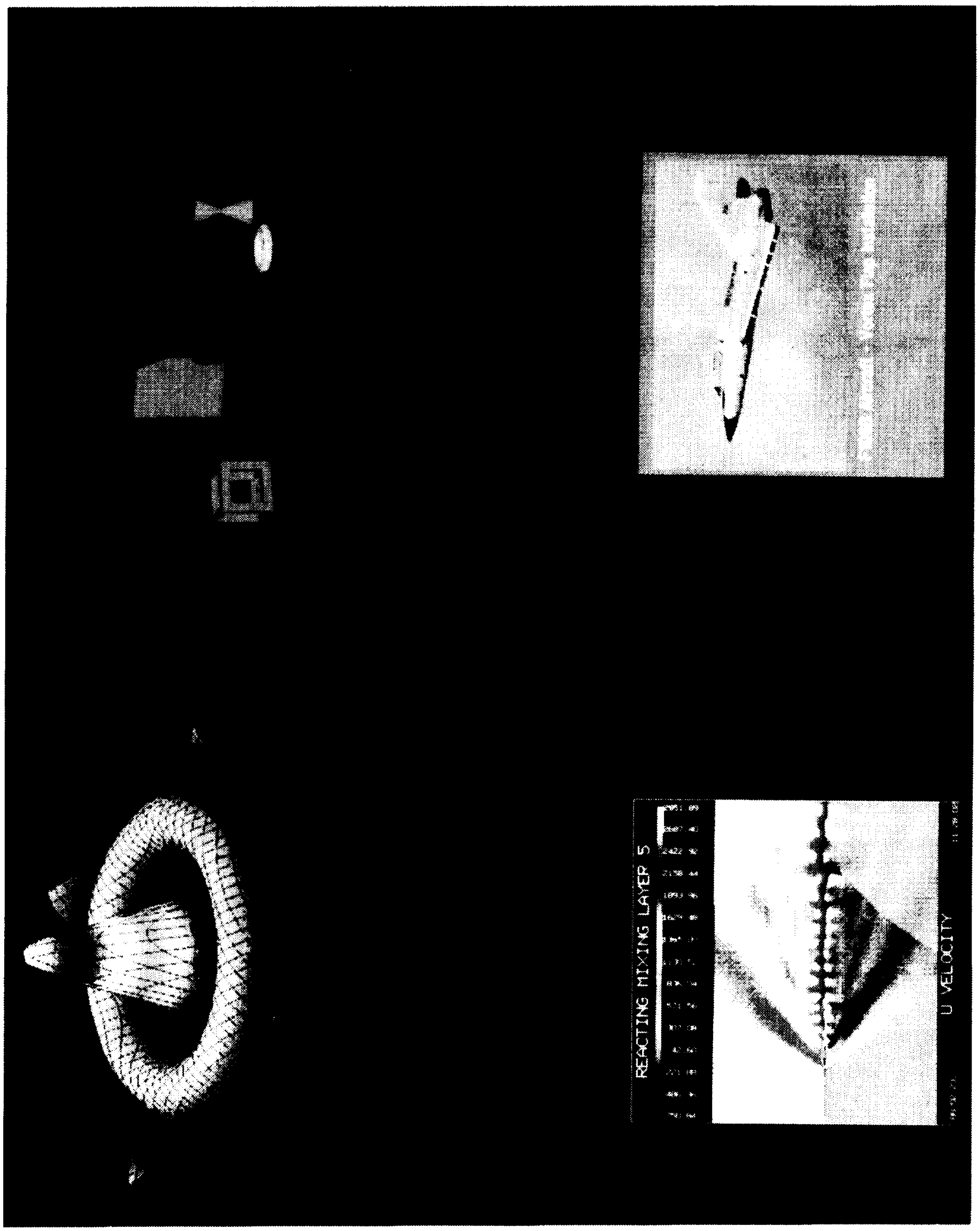


Figure 8. - RMT Image Composite

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

NAME	DESCRIPTION
clipres.acf	Demonstrates clipping and resizing ("up" and "down")
drawver.acf	Demonstrates commands necessary to output to Verastec plotters.
itolut.acf	Demonstrates intensity to LUT conversion using one of three quantization schemes
lut2i.acf	Demonstrates color LUT to intensity conversion
luti2bw.acf	Demonstrates conversion from color LUT or intensity to black and white
ovrlay.acf	Demonstrates "overlaying" images

Table 7. - RMT Alternate Command File Templates

10.0 RMT ERROR PROCESSING

The RMT error processing mechanisms have been designed to prevent premature program aborts. The error processing performed by the RMT is primarily error detection with minimal error correction.

The error detection facility is user controlled via the SET ERROR command (see Section 7.10.3). This command is used to set the error severity level that causes RMT termination. The detectable error conditions are listed in Appendix E. Each error condition has been assigned a severity level of SYNTAX, WARNING, or FATAL. If the error severity level is set to NONE, no error detection is performed. When an error condition is encountered, the error message is printed to the user's terminal. In the case of severity level NONE, a traceback of all offending RMT routines is provided. The user is advised that for other than SYNTAX errors, the normal procedure is to exit the RMT and correct the error. Results occurring when proceeding from an error condition are unpredictable.

As mentioned previously, error correction facilities are minimal. A device driver may truncate a color LUT or color LUT entries if the image and device attributes are incompatible in an effort to provide some form of display. Other error correction mechanisms involve default channel selection for multi-channel images.

11.0 RMT UTILITY LIBRARY

The primary function of the RMT utility library is to provide software tools which permit the conversion to/from RM format from/to an external image format. The RMT utility library is comprised of a collection of FORTRAN callable subroutines which permit the reading and writing of RM image entities such as headers, color LUT's, and scanlines. A skeleton program

skeleton.f

which writes a simple RM image is included with the distribution. The skeleton program (see Appendix B) is heavily commented to facilitate conversion to another application. Every RMT utility library reference is proceeded by a "CRMT" format comment line. In addition, two other RMT utility library sample programs are provided. The program **sun2rm** converts a Sun image file into an RM, and **rm2sun** converts an RM into Sun image format. Again, all RMT utility library calls are preceded by the "CRMT" comment line.

The following two subsections provide information on accessing the RMT utility library and using the RMT utility library routines.

11.1 RMT Utility Library Access

An application program using the RMT utility library must link the library, **rmtran/rmtlib.a**, and the "dummy" device driver, **rmtran/dddummy.o**. In the CONVEX environment assuming an application program **test.f** and a release area directory named **rmtran**, an executable image may be produced by invoking the compile and link command line:

```
fc test.f rmtran/dddummy.o  rmtran/rmtlib.a
```

In addition, the RMT utility library requires access to the RMT errors file (see Appendix E). In the UNIX environment, the location of the file is passed to the RMT through the environment variable RMTERR. For the CONVEX default installation, the error file resides in **rmtran/RMTERR.DAT**. This process is described in the *rmtran* manual page and Section 12.2.5.

11.2 RMT Utility Library Routines

The RMT utility library routines are a subset of all RMT routines. The list of user callable RMT utility library routines follows:

BEGCHA	PRMSL
CLOSMF	RDHEAD

ENDCHA	RDLUT
GRMSL	RMINIT
OPENMF	WRHEAD
POSRM	WRLUT

If the sample conversion programs do not provide sufficient information for using these utility routines, additional documentation is contained in the file **RMTUTIL.DOC** which resides in the release area. This on-line documentation is comprised of the prologues from each of the subroutines and contains a description of each routine and its associated input/output parameters.

12.0 RMT INSTALLATION

The RMT was developed on a CDC host running NOS/VE. Since this initial development effort, the RMT has been ported to a DEC VAX 11/750 running VMS 4, a DEC VAX 11/750 running Ultrix, a CONVEX C210 running CONVEX 7.1, a Sun 3 system running SUNOS 3.5, and a CRAY-2 running UNICOS 4.0. The intent of this section is to describe a sample installation and to identify all host dependent code.

12.1 RMT UNIX Installation

The installation procedures for any software package are host dependent with respect to file management, compilation, linking, and execution. Because the current version of the RMT is evolving in a UNIX environment, this environment was selected to discuss installation procedures. Although installation on a non-UNIX host may prove more difficult, the RMT is written in ANSI FORTRAN 77 with some C utilities for performing input/output.

The following subsections assume a CONVEX host running CONVEX 7.1. UNIX commands and shell script names are highlighted in italics, routine and file names embedded within the text are highlighted in bold print. The reader should be familiar with UNIX utilities such as *tar*, *make*, *ar*, *ranlib*, etc., as well as the FORTRAN compiler and loader on the target host.

12.1.1 Reading the Distribution

The installer should first create a directory where the RMT software is to reside. Environment variables may be employed to further automate the installation procedure, but in this instance make a subdirectory under the current directory:

mkdir rmtran

The RMT software is distributed in *tar* format either via magnetic tape or file transfer across a network. In this case, assume the RMT software has been ported across a network and resides in the current directory under the name:

rmt.tar

The next step is to extract the RMT files from *tar* files as follows:

cd rmtran

tar xvf ../rmt.tar

This last command should create a directory structure as shown in Figure 9.

12.1.2 Installing the RMT

Installing the RMT consists of constructing the RMT utility library and the RMT command

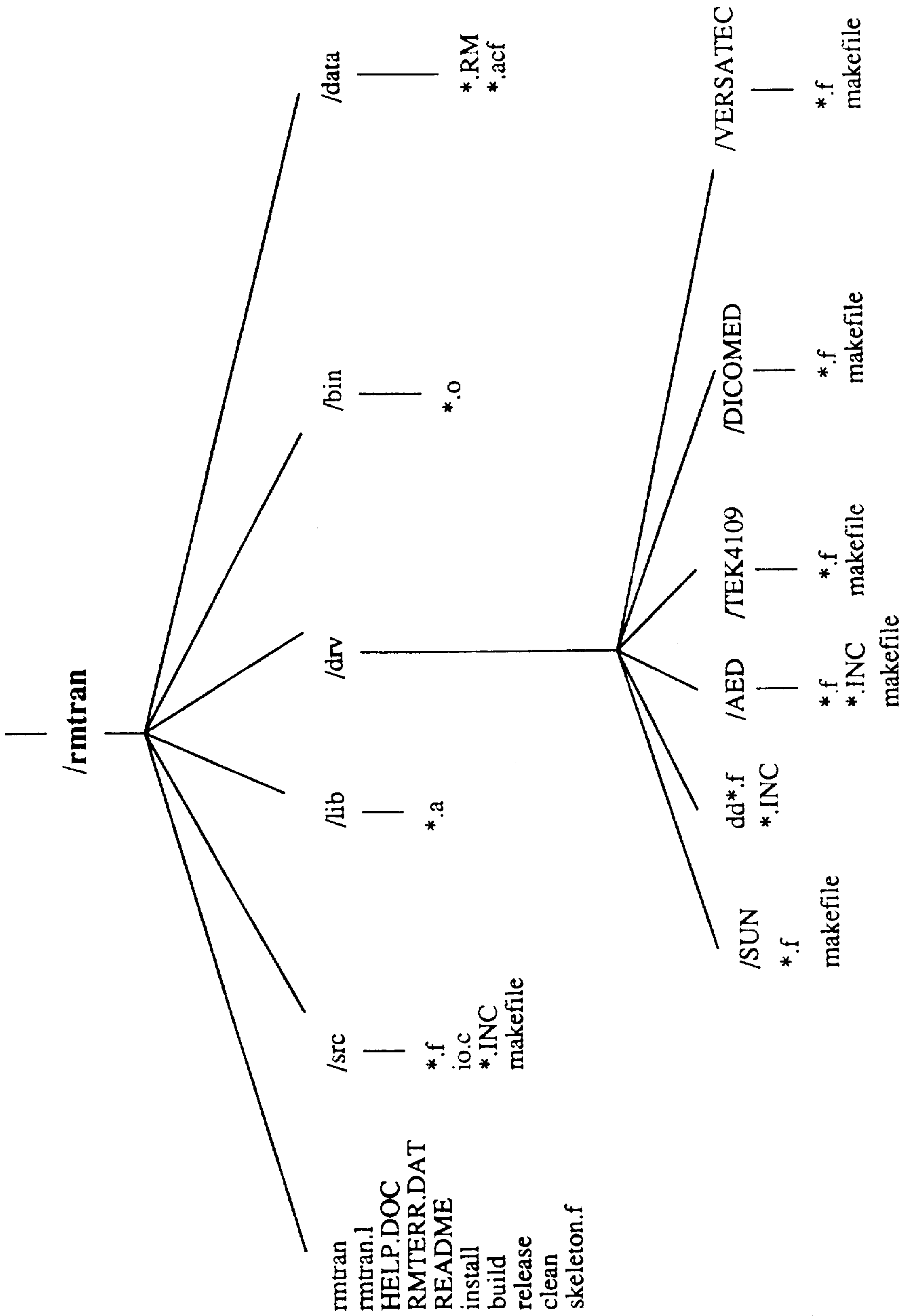


Figure 9. - RMT Directory Structure

interpreter library. This procedure is automated through invoking the *install* shell script:

```
install >& install.log &
```

An examination of the **install.log** file will indicate any errors in the installation process. It may be necessary to tailor either *install* or **src/makefile** to the host environment. Particular attention should be given to compiler name and options whose values are set in the shell variables F77 and F77OPT. As a result of the installation process, the two user callable libraries

rmtlib.a

drvlib.a

should reside in the **rmtran/lib** subdirectory.

12.1.3 Building RMT Device Drivers

Once the RMT libraries have been constructed, the next step in the installation process is to build the desired device drivers. This process is automated by invoking the shell script *build* with the desired device name

```
build [device-name] > &build.log
```

The result of running *build* is that the RMT device driver will be placed in the **rmtran/bin** area and the driver utility library (if required) will be placed in the **rmtran/lib** subdirectory. The file **build.log** may be examined to determine the status of the driver build.

For example, to construct the Tektronix 4109 device driver issue the command

```
build tek >&build.log
```

which places **ddtek.o** in **rmtran/bin** and **tek4100.a** in **rmtran/lib**.

The installer may be required to edit the *build* script and/or the **makefile**, located in the device specific subdirectory subordinate to **rmtran/drv**, particularly with respect to compiler names and options. As with *install*, pay particular attention to the shell variables F77 and F77OPT.

12.1.4 Releasing the RMT

If users are to access the RMT from an area other than the installation area, the *release* script may be used to move the necessary files. This script assumes the release area is identified in the shell environment variable RMT_ROOT. The *release* script is invoked by entering

```
release
```

from the installation area.

If limited disk space is a problem, object and library files may be removed from the installation area by invoking the *clean* shell script. The *clean* script should be run only after the *release* script has successfully completed.

12.1.5 RMT Execution

The final step in the installation procedure is to link the RMT command interpreter, utility library, and device driver to form an executable image. The shell script *rmtran* is provided to perform this function. The *rmtran* is invoked as shown below

rmtran [device-name]

If the device name is omitted, the "dummy" driver is loaded. A detailed discussion of *rmtran* is provided in Appendix D. Note the use of the shell environment variables RMTERR and RMTHELP to locate the RMT errors file (see Appendix E) and the on-line help file (see Section 7.6). If *rmtran* is released to prospective users, it will be necessary to either hardwire a pathname for the local variable

RMTROOT

or employ an environment variable. The installer should set the *rmtran* shell variables LINK and LINKOPT to the desired loader.

12.1.6 RMT Validation

The first step in validating an implementation of the RMT should be to repeat the three sample RMT command sessions provided in Section 9 and verify the results. In addition, the subdirectory

rmtran/data

contains sample RM files (using file extensions ".RM") and sample command files (using file extension ".acf"). The supplied alternate file names together with a brief description are provided in Table 8. All sample command files are based on the RM files included with the distribution. However, since some command sequences are steps in solving a more complex imaging problem, the input to one command file may be dependent on the output from another command file. These contingencies are identified in comments preceding each command file. When attempting to display an RM image, insure that the image format and device characteristics are compatible.

12.2 RMT Host Dependencies

As listed in Section 12.0, the RMT has been ported to several host environments. The purpose of this section is to identify and describe the host dependent modifications necessary to make these ports. The host dependencies have been categorized into distinct groups each of which is discussed in a subsequent subsection.

Most dependencies associated with the installation and creation of device drivers are deferred until Section 13.

NAME	DESCRIPTION
bw2hc.acf	Demonstrates writing a black-and-white image to a PostScript file
c2bw.acf	Demonstrates the conversion of an intensity image to a black-and-white image
ccquant.acf	Demonstrates the conversion of an intensity image to a color table image using color cube quantization
conv2cd.acf	Demonstrates the conversion of an intensity image to a color table dithered image
conv2i.acf	Demonstrates the conversion of a color table image to an intensity image
cpy2rm.acf	Demonstrates the COPY command
dirndrw.acf	Demonstrates the DIRECTORY, DRAW, GET, and SET commands
ovrlay.acf	Demonstrates the OVERLAY command
quant.acf	Demonstrates viewing (separately) the 3 different quantization schemes (dithering, color cube, uniform)
resize.acf	Demonstrates the RESIZE option (performed by turning "clipping" off)
scale.acf	Demonstrates the SCALE command
uquant.acf	Demonstrates the conversion of an intensity image to a color table image using uniform quantization
vuover.acf	Views an overlayed image
vuscal.acf	Views an image (unscaled and scaled)
vutwo.acf	Demonstrates viewing a multi-image metafile and the use of keywords in a command string

Table 8. - RMT Installation Alternate Command Files

The FORTRAN COMMENT statement is utilized to depict host dependent code within the RMT. As listed in the following table, the form of the comment identifies the host.

FORTRAN COMMENT	HOST	SYSTEM
CVE	CDC	NOS/VE
CVMS	DEC VAX	VMS 4
cltrx	DEC VAX	Ultrix
ccnvx	CONVEX	CONVEX UNIX 7.1
ccray	CRAY-2	UNICOS 4.0
csun	SUN 3	SUNOS 3.5
cunix	-	UNIX

The "cunix" comment is sometimes used to refer to a generic UNIX system modification. By uncommenting and/or recommenting these source lines, the RMT may be ported to one of these hosts.

12.2.1 RMT Include Files

Although the RMT is essentially written in ANSI FORTRAN 77, the INCLUDE file construct was employed to designate frequently used labeled COMMON blocks. The format of an INCLUDE statement may vary slightly from host to host, but the form

INCLUDE 'file-name'

appears to be the de facto standard. Within the RMT, all "include" file names are in upper case with a ".INC" extension. It may be necessary to modify either or both of the include files RMTPAR.INC and RMTHOST.INC. In RMTPAR.INC, the maximum size (in 16-bit words) of an RM record, MAXREC, may require modification. The number 3840 (or 7680 bytes) was selected at LaRC for compatibility with all supported hosts. In the case of RMTHOST.INC, the host dependent parameter which must be set correctly is the number of bits in a host computer word, NUMBIT. If the host, does not have a C compiler, it may be necessary to adjust the LENREC parameter which is used as the value of REC on the FORTRAN OPEN statement for an RM. This later condition is expanded upon in subsection 12.2.4.

12.2.2 RMT Intrinsics

In performing basic bit manipulation operations, the RMT employs the standard "and," "or," and "shift" operations. Unfortunately, host compilers differ with respect to the name and syntax of these intrinsic functions. In order to localize the host dependencies for these bit operators, the RMT uses the three routines:

iand.f

ior.f

ishift.f

The FORTRAN commenting scheme described in Section 12.2 is used to isolate host dependent syntax in each of three routines. An exception to the scheme is the CONVEX installation where neither "iand" nor "ior" are required; and therefore, should not be included as part of the RMT utility library.

12.2.3 RMT Network Portability

For reasons of efficiency and minimizing file size, a byte addressable binary format was selected for the RM. However, because RM portability across supported local area networks (LANs) was also a concern, some consideration was afforded to host byte ordering.

The initial target device was a CELCO color film recorder driven by a VAX 11/750 running VMS 4, implying that the DEC ordering scheme became the norm. Consequently, to insure the portability of RM files across an LAN, byte and/or (16-bit) word swapping is performed within the RMT on all non-DEC hosts.

Unless the RMT is to be installed on a new host, only a single modification is required. An alternative to the aforementioned "commenting" scheme was used in this case. The host is named in a DATA statement within the FORTRAN BLOCK DATA routine **rmtbd.f.** The variable to be named is **hname** and the allowable names are: VE, VMS, ultrix, convex, cray, and sun. The value of **hname** is used to control byte and/or word swapping within the RMT routines

rrmrec.f

wrmrec.f

grmbuf.f

prmbuf.f

If a new host is added, the IF-THEN-ELSE statements within these four routines will require modification.

12.2.4 RMT I/O Routines

The routines to perform input/output on RM files within the RMT are written in C using standard UNIX system calls such as read and write. These routines all reside in the single module **io.c.** Use of these C routines within the RMT are bracketed with the "cunix" comment.

Because the interfacing of FORTRAN and C may be host dependent with respect to entry point names and/or parameter passing conventions, some modifications to these routines may be required. However, the C routines ran without modification under Ultrix, CONVEX and SUNOS and only a name change (identified within the C source) under UNICOS was required.

If a C compiler is unavailable, the C calls may be replaced with FORTRAN input/output statements. This approach was utilized for both NOS/VE and VAX/VMS installations. The major routines for performing input/output operations on RM files are **openmf.f, closmf.f, rrmrec.f, and wrmrec.f.** Other RMT routines using the C utilities perform file positioning operations (**crewnd**) and are identified with the "cunix" comment.

12.2.5 Environment Variables

When executing the RMT, it is necessary to have access to the errors file (see Appendix E) and the on-line help file (see Section 7.6). In a UNIX installation of the RMT, the shell environment variables RMTERR and RMTHELP are used to locate these files. The pathnames of the error and on-line help files are placed in these variables and these variables are accessed by the RMT routines **rminit.f** and **exthlp.f** using GETENV, a FORTRAN interface to the *getenv* UNIX system call. The GETENV related code is marked in each routine using the comment conventions described previously. The FORTRAN GETENV call may be host-dependent.

On a non-UNIX host the simplest alternative to environment variables is to hardwire the error and help file pathname in the FORTRAN OPEN statement within **rminit.f** and **exthlp.f.** On a DEC machine, it may be convenient to employ logical symbols to hold the file pathnames.

13.0 RMT DEVICE DRIVERS

The device interface to the RMT was presented in Section 8 from a user's perspective. The intent of this section is to present RMT device drivers from the viewpoint of an installer or a programmer wishing to write new device drivers. The first subsection provides information on the general structure of an RMT device driver and the second subsection provides device specific information.

13.1 RMT Device Driver Structure

The primary function of an RMT device driver is to output RM image data in device-specific format. Consequently, an RMT device driver must be capable of scanline data output with support functions for device initialization, setting color LUT's, cursor positioning, etc. In many instances, one or more user callable subroutine libraries which performs the device specific graphics function may already exist. The philosophy behind the development of an RMT device driver dictates the use of such libraries if they exist.

An RMT device driver is comprised of one or two components. The first component is the "dd" layer which provides the routines which interface directly to the RMT. Every device has an associated "dd" module which is bound to the RMT by routine names and argument calling sequences. The "dd" modules for currently supported drivers reside in the **rmtran/drv** subdirectory. A heavily commented "skeleton" RMT device driver

ddskeleton.f

is provided as a template to assist in the development of new drivers.

The second layer in the RMT device driver structure is optional and consists of a user callable device-specific graphics library as described above. The routines in this library are referenced from the "dd" layer. As an example, the Tektronix 4109 RMT device driver consists of the "dd" component, **ddtek**, which calls a locally supported Tektronix 4100 graphics device library, **tek4100**. If a device driver utility library is unavailable or unnecessary, all device specific code may be placed in the "dd" layer. The PostScript drivers, **ddpost**, **ddcpostlut**, and **ddcpostrgb** are examples of RMT device drivers without an associated utility library.

Because the device-specific libraries are locally supported, RMT device drivers may not be portable. The sample RMT distribution does provide compatible utility libraries for all devices listed in Section 8.

13.2 Supported RMT Device Drivers

The unique characteristics of supported RMT drivers are presented in this section from the viewpoint of the installer and the user. A subsection is devoted to each supported device with the exception of the CELCO film recorder since it accepts RM formatted input.

13.2.1 AED 767

The AED 767 is a color raster terminal with 8-bit entries (an LUT of length 256 with 8-bit entries for red, green, and blue) and a viewable resolution of 767 by 535, and a computational resolution of 1024 by 1024 accessible via a hardware pan capability. The RMT AED driver consists of the "dd" layer **ddaed** and the utility library **aedlib** which invokes the terminal I/O utilities in **ddio**. The source for **ddaed** and **ddio** reside in the **drv** subdirectory and the source for the utility library resides in the **drv/AED** subdirectory. The AED driver "pauses" between frames and execution continues by entering any character followed by a carriage return.

13.2.2 DICOMED D-47

The DICOMED D-47 is a color film recorder which supports an LUT of length 256 with 8-bit entries in red, green, and blue and a maximum resolution of 2048 by 2048. The "dd" layer of the DICOMED driver is **dddicomed**. The driver employs a subset of the LaRC supported RASLIB [4] raster utility library named **raslibrmt**. The RMT driver source code resides in the **rmtran/drv** and the utility library source resides in the subdirectory drv/DICOMED.

The output from the RMT DICOMED driver is the binary file **D47OUT**. The output file is opened, written, and closed with standard FORTRAN I/O statements. The file name and/or file attributes as designated in the OPEN statement may be modified by editing the DDINIT routines located in the **dddicomed.f** module.

13.2.3 PostScript

The RMT PostScript device driver refers to any PostScript compatible laser printer. The driver treats a PostScript printer as either a single channel intensity device (black and white) with a maximum resolution of 4096 by 4096 with an 8-bit grey scale value, or as a color device with either color table or intensity capabilities. The three channel intensity device (**postscriptrgb**) has 8-bit entries for red, green, and blue, while the color table device (**postscriptlut**) supports an LUT of length 4096 with 8-bit entries in red, green, and blue. The "dd" layer for the black and white device is **ddpost**, while the color devices are

ddcpostlut and **ddcpostrgb**, where the "lut" and "rgb" suffix denotes color table and intensity respectively. All source code for these drivers resides in the **rmtran/drv** subdirectory. No device utility library is required.

The RMT PostScript driver produces a formatted PostScript compatible file named **PSOUT**. All file I/O is performed with standard FORTRAN. In order to change the output file name or attributes, the DDINIT routine of the PostScript modules must be edited.

13.2.4 Sun

The RMT assumes the Sun is a color raster device with an 8-bit LUT and 8-bit entries for red, green, and blue and a resolution of 1152 by 900. The Sun monochrome driver assumes the Sun is a single channel intensity device with single bit entries for on or off. The monochrome driver incorporates an ordered dithering scheme [10] to simulate grey scales. The RMT Sun drivers consist of **ddsun** or **ddsunbw**. Source code for these drivers resides in the **rmtran/drv** subdirectory. The device utility library resides in the subdirectory **drv/sun**.

The distributed version of the RMT Sun device driver generates a file in "standard" Sun Pixrect format [11] named **sun.img**. The file I/O operations are performed using the C utilities described in Section 12.2.4. The DDINIT routine of the **ddsun.f** module may be edited to rename the output file.

If the RMT is to be Sun-resident, the driver could be modified to output the Pixrect image to the display screen instead of a file.

13.2.5 Tektronix 4109 and 4125

The Tektronix 4109 is a color raster device with an LUT of size 16 and LUT entry range of 0 to 100. The Tektronix 4125 is assumed to have 8-bit planes with an LUT entry range of 0 to 255. The "dd" layers of these RMT device drivers are **ddtek** and **ddtek4125**, respectively. Both drivers invoke the utility library **tek4100**. The source code for the "dd" layers of both devices reside in the subdirectory **drv/TEK4100**. Because the RMT writes to the "dialogue" area and not the graphics area, no user controlled "pause" capability was incorporated into this driver.

13.2.6 Versatec ECP-42 and C2756

The Versatec ECP-42 is a color electrostatic plotter and the C2756 is a thermal printer. The RMT treats both devices as three channel intensity devices with a maximum resolution of 8000 by 8000 for the ECP-42, 2432 by 2432 for the A-size C2756, and 3124 by 3124 for the B-size C2756. The "dd" layer **ddvtek39** is used for the ECP-42, while **ddvteka** and **ddvtekb** are used for the C2756 A and B sizes, respectively. The device specific utility library **versalib** is referenced by the Versatec drivers. The source for the drivers resides in **rmtran/drv** and the utility library resides in the **drv/VERSATEC** subdirectory.

The Versatec drivers differ from other RMT devices in that they prompt the user for device control information. A partial RMT command session in section 8.1 illustrates the Versatec-specific prompts and acceptable values.

A user may have to trade-off "dither level" with input image or output file size. Maximum resolutions presented previously are dependent on selected dither level (see Table 6). More specifically, a selected dither level of 4 reduces the maximum image resolution by a factor of 4. Some experimentation with the "color map" option may be required. It is also important to note that the Versatec-compatible plot file produced is named **VEROUT**. Additionally, it is important, when producing a color output image for the C2756 (A and B sizes), that channel ordering be reversed. That is, the RMT command

```
set a 1  cha 3 (3  2  1)
```

should be used (see section 7.10.1.4).

GLOSSARY

attribute (or **attribute-bundle**)

An RMT attribute is a compound entity whose parameters control the processing of RM images. Defined with the SET ATTRIBUTE command, individual attributes include: color table controls, channel designation, conversion options, and scale parameters.

channel

Multi-spectral image data is often specified in terms of one or more components referred to as channels. Red, green, and blue image data may be represented using three channels. In RM terminology, multi-channel image data is said to be in intensity format.

clipping

Clipping is a graphics term associated with a window to viewport mapping. When clipping is enabled, data which maps to outside the viewport dimensions is not displayed. With clipping turned "on", the RMT adheres to the graphics analogy. However, with clipping "off", the RMT "re-sizes" the image data to fit the viewport dimensions.

color (LUT)

Image data may be specified by indexing into a table referred to as a color lookup table (LUT). Using this approach each LUT entry provides a red, green, and blue component while the image data provides only an index into the LUT. The RM supports image data in color table format.

component

A component is a compound entity specified with the SET COMPONENT command to designate a metafile, picture (or image) within the metafile, window, and a viewport.

compositing

Compositing is an image operation whereby one or more component images are combined to form a resultant image. The OVERLAY command is the only image composition operator provided by the current version of the RMT.

conversion

Conversion is an RMT operation (CONVERT Command) which converts images from one format to another normally for the purpose of ensuring compatibility between the image and the output display device. The conversion options are color table to intensity to color table, and color table or intensity to black and white (or greyscale).

dithering (ordered)

Ordered dithering is an RMT option used when converting from an intensity to a color LUT format. Ordered dithering differs from typical dithering (often referred to as halftoning for monochrome devices) in that no resolution is lost during the dithering process. The technique employs an ordered dither matrix as a threshold in determining whether to "intensify" a pixel at a given horizontal and vertical position. Invoking ordered dithering from the RMT produces an RM image with an LUT of size 3 and a user designated range.

extent

The extent of an RMT image is determined by it minimum and maximum dimensions in both the horizontal and vertical directions. In the case of a composite image, the extent is measured by collectively considering the minimums and maximums of the component images.

frame

In an RM formatted image, each channel of data is delineated by a START FRAME and END FRAME opcode.

header

An RM header is a 512-byte logical record preceding every image providing information which completely categorizes the image with respect to resolution, color LUT or intensity, ... etc.

intensity

Image data in which pixel information is stored in separate channels is said to be in intensity format. Most commonly, the intensity at a specific pixel is determined from its red, green, and blue component, although more than three channels are permissible. Multi-channel data and intensity data are synonymous terms.

interleaving

The method by which intensity (or multi-channel) image data is stored is referred to as interleaving. The three interleaving schemes are: pixel interleaved where the red, green, and blue component for each pixel are stored contiguously; scanline interleaved where the red component for every pixel in a scanline is stored followed by the green and blue components; and field interleaved where all information for the red channel is encountered prior to the green and blue channel components. The RM employs a field interleaved scheme.

opcode

An RM formatted image is comprised of entities known as opcodes. These opcodes, normally consisting of an identifier and a data value, completely specify all image components.

overlay

Overlay is an image composition operator used by the RMT to combine one or more component images into a single resultant image. Images are "overlayed" onto each other in the order they appear on the command line. That is, if A and B are component images, the resultant values of the pixels in the area at which A and B overlap are the values from image B.

pixel

A pixel (often referred to as a "picture element") is the smallest unit in a raster image. A pixel is identified by a color LUT index or an intensity value and a horizontal and vertical position.

quantization

Quantization is a technique used in converting an RM image from intensity to color LUT format. The quantization scheme selected determines the method by which (the normally large number of) intensity values are mapped into (a normally small number of) LUT entries. The RMT permits either uniform or color-cube quantization. In the case of uniform quantization, the size of the LUT and/or the number of bits to be maintained from each of the red, green and blue

LUT entries. The RMT permits either uniform or color-cube quantization. In the case of uniform quantization, the size of the LUT and/or the number of bits to be maintained from each of the red, green and blue channels along with the range of an LUT entry must be user supplied. The designated number of bits per channel associated with the most significant bits of a pixel intensity for each channel are used to construct the LUT while the least significant bits are discarded. For color-cube quantization, the size of the LUT and range of an LUT entry must be supplied by the user. The quantization scheme applies this information when transversing the standard color-cube to determine an appropriate set of LUT entries.

range (of LUT entry)

A color LUT entry is specified in terms of a red, green, and blue intensity. The RMT uses the term range to refer to the maximum intensity permissible per component. In an RM image header, the number of bits per color table entry is comparable to range. Within the RMT, range is normally used as an integer value.

resolution

The resolution of an RM image is given in terms of number of scanlines (vertical resolution) and number of pixels per scanline (horizontal resolution).

RM (Raster Metafile)

The Raster Metafile (RM) format is a generic raster image format adopted at LaRC for portability concerns.

RMT (Raster Metafile Translator)

The Raster Metafile Translator (RMT) is a program used to process RM formatted images.

resizing

In RMT terminology, resizing refers to the application of a bilinear interpolation scheme to "fit" image data from a specified window into a specified viewport. Depending on the relative dimensions of the window and viewport, the image subset may be resized "up" or "down." Resizing is invoked when "clipping" is disabled.

scaling

Scaling is an RMT operation which is used to modify the "size" or "range" of an LUT, or the number of data bits in an intensity image. The operation is typically used to tailor an image to the characteristics of an output device.

scanline

A horizontal run of pixels is referred to as a scanline. The number of scanlines in an image is the vertical resolution of the image.

size (of LUT)

The size of an LUT is the length or number of entries comprising the LUT. In an RM image header, the size of the LUT is the number of data bits/pixel. In both the RM format and the RMT, the size of an LUT is always specified in terms of the number of bits.

viewport

An RMT viewport is a rectangular region of the display surface on which a subset of an RM image is to be drawn.

window

An RMT window is a rectangular subset of an RM image.

REFERENCES

1. McCormick, B. H.; Defanti, T. A.; Brown, M. D. "Visualization in Scientific Computing," ACM SIGGRAPH, Volume 21, Number 6, November 1987.

2. "Computer Graphics-Metafile for the Storage and Transfer of Picture Description Information," American National Standards Institute, ANSI X3.122-1986, 1986.

3. Thomas, Spencer W., "Design of the Utah RLE Format," Technical Report 86-15, Alpha_1 Project, CS Department, University of Utah, November, 1986.

4. "RASDOC," NASA LaRC Central Scientific Computing Complex Document GD-8, 1986.

5. Christianson, H; and Stephenson, M. MOVIE.BYU Training Manual, Brigham Young University, Department of Civil Engineering, 1982.

6. "Metafile Translator User's Guide," Precision Visuals, Inc., 1987.

7. Foley, J. D., Van Damm, A. Fundamentals of Computer Graphics, Addison-Wesley, 1982, pp. 40-42.

8. "SADIE 2.4 Image Processing Software User's Manual," University of Arizona, July 1981.

9. Heckbert, P. S., "Color Image Quantization for Frame Buffer Display," Massachusetts Institute of Technology, May 1980.

10. Jarvis, J. F.; C. N. Judice, and W. H. Ninke, "A Survey of Techniques for the Image Display of Continuous Tone Pictures on Bilevel Displays," Computer Graphics and Image Processing, 5(1), March 1976, pp. 13-40.

11. "Pixrect Reference Manual," Sun Microsystems, 1986.

APPENDIX A
Raster Metafile Format

This appendix describes a generic raster image format adopted at LaRC. The format, hereafter referred to as the Raster Metafile (RM) format, parallels the concept of a vector metafile which is promoted by the graphics industry as a means of supporting device independent graphics. The proposed RM format is to provide a common interface format between application- or package-specific raster output formats and device-specific raster input formats. The RM format does not eliminate the requirement for raster image translation programs, but it should reduce both the number and complexity of these translators.

The proposed RM format does not attempt to incorporate all popular image formats. Instead, the RM format is compatible with image formats commonly used at LaRC. More specifically, the format design emphasizes critical graphics output devices such as the CELCO color film recorder. In general terms, an RM formatted file may be characterized as a multi-image file where individual images may be:

- variable resolution
- color or black-and-white
- specified by a user supplied color table or as intensity values (in a field interleaved format)
- straight pixel data and/or compressed in a run-length encoded structure.

An RM formatted image is completely specified through a mandatory image header used to describe the image format and a set of opcodes used to delineate entities within the image structure. A separate image header must precede each image within an RM formatted file. The header format and individual entries are detailed in Section A.1 of this document. The use and format of the RM opcodes are described in Section A.2. The RM format is designed to accommodate up to 12-bits of data in a 16-bit word and up to 24-bits using the continuation opcode.

The recommended physical record size for an RM formatted image record is 7680 (15 x 512) bytes. The motivation for selecting this binary record size was based on ensuring proper word boundary alignment for all LaRC supported computer systems (including CDC NOS with its 60-bit word size).

APPENDIX A
Raster Metafile - Header Format

SECTION A.1

The image header is used to describe the data format of each image in the Raster Metafile (RM) formatted file. The header is designated by its own opcode (Section A.2 - Raster Metafile Opcodes) and consists of a 512-byte block of data. Each header entry occupies 32 bits of storage. The header format and individual entries are described in the following two tables.

RM Header Format

Word Number	Description
1	Header Opcode
2	Maximum X Resolution
3	Maximum Y Resolution
4	Color Table/Intensity Format Flag
5	Black and White/Color Flag
6	Number of Bits per Color Table Entry
7	Number of Channels ($\leq$ 16)
8	Number of Data Bits/Pixel
9	Background Color Number or Intensity of Channel 1
10	Intensity of Channel 2
11	Intensity of Channel 3
⋮	⋮
24	.
25 - 128	Unused (Reserved)

APPENDIX A
RM Header Entries

Word #	
1	Header opcode
2	X Resolution Defines maximum X resolution for following image (32-bit maximum)
3	Y Resolution Defines maximum Y resolution for following image (32-bit maximum)
4	Color Table/Intensity format flag Defines type of image data 1 = Color table data 0 = Intensity data
5	Black and White/Color Flag Defines color type of data 1 = Color 0 = Black-and-White
6	Number of bits per color table entry Defines number of bits for each entry (red, green, or blue component) in the color table
7	Number of Data Channels Defines the number of data channels which comprise the image (for an R, G, B - intensity image this number would be three). Limited to 16 channels.
8	Number of Data Bits/Pixel Defines the number of bits per pixel for intensity data. Defines the length (number of entries) of the color table for color table data.
9-24	Background color number or intensity. Defines color of background for image number of values dependent upon number of channels (word 7 in header) available.
25-128	Reserved (Unused)

APPENDIX A
Raster Metafile Opcodes

SECTION A.2

The raster metafile is made up of 16-bit opcodes. The upper four bits or "tag bits" define the opcode. The lower 12 bits provide the "data" for the opcode. Below is a table of the raster metafile opcodes. The following pages provide greater detail about each opcode.

	H															L
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RL	0	0	0	0	Run Length											
PD	0	0	0	1	Pixel Value											
I	0	0	1	0	Intensity											
COL	0	0	1	1	Color Number											
SF	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LDY	0	1	0	1	Number of Lines											
EL	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
EF	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	Reserved											
	1	0	0	1	Reserved											
AD	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
MAP	1	0	1	1	Map Value											
CON	1	1	0	0	High Order Data Bits											
LDX	1	1	0	1	Number of Pixels											
HDR	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
NOP	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

APPENDIX A

RL - Run Length Opcode

The Run Length Opcode is always followed by an Intensity Opcode or a Color Number Opcode. The lower 12 bits define the number of times to repeat the following intensity or color number. The valid range is 1-4095. A value of zero is illegal.

PD - Pixel Data Opcode

The Pixel Data Opcode is dependent on the color table/intensity format flag. If the flag specifies color table, then the Pixel Data Opcode defines a single color number. If the flag specifies intensity, then the Pixel Data Opcode defines a single intensity. The valid range is 0-4095.

I - Intensity Opcode

The Intensity Opcode is always preceded by a RL Opcode. The lower 12 bits define the intensity of pixels to be run length encoded. The valid range is 0-4095. Intensity Opcodes are only valid for intensity images.

COL - Color Number Opcode

The Color Number Opcode is always preceded by a RL Opcode. The lower 12 bits define the color table location to use when looking up the intensity of pixels to be run length encoded. The valid range is 0-4095. Color Number Opcodes are only valid for color table images.

SF - Start Frame Opcode

The Start Frame Opcode defines the beginning of a channel in the image. It must be the first opcode of a 512-byte/256-word record. The lower 12 bits must be zero. Color table images only have one channel. Intensity images may have up to 16 channels.

LDY - Load Y Opcode

The Load Y Opcode plots lines with length equal to the maximum X resolution and using the background color. The lower 12 bits define the number of lines to plot. The valid range is 1-4095. A value of zero is illegal.

EL - End Line Opcode

The End Line Opcode defines the end of the current line. The lower 12 bits must be zero.

APPENDIX A

EF - End Frame Opcode

The End Frame Opcode defines the end of the current channel. The opcodes following this opcode to the end of the 512-byte/256-word block are ignored.

AD - Application Data Opcode

The Application Data Opcode defines the beginning of a user record. It must be the first opcode of a 512-byte/256-word record. The entire record is ignored.

MAP- Map Opcode

The Map Opcode defines a map value for color table images. The lower 12 bits define the map value. These values are used by the PD and COL Opcodes in color table images. The map structures are defined in Appendix C.

CON - Continuation Opcode

The Continuation Opcode allows the use of up to 24 bits of data. When a continuation opcode is encountered, the next 16-bit quantity is examined to determine the appropriate opcode. The lower 12 bits of the Continuation Opcode become the most significant portion of the data and the lower 12 bits of the next opcode form the least significant part of the data.

LDX - Load X Opcode

The Load X Opcode is translated into a run length of the background color. The lower 12 bits define the length of the run length. The valid range is 1-4095. A value of zero is illegal.

HDR - Header Opcode

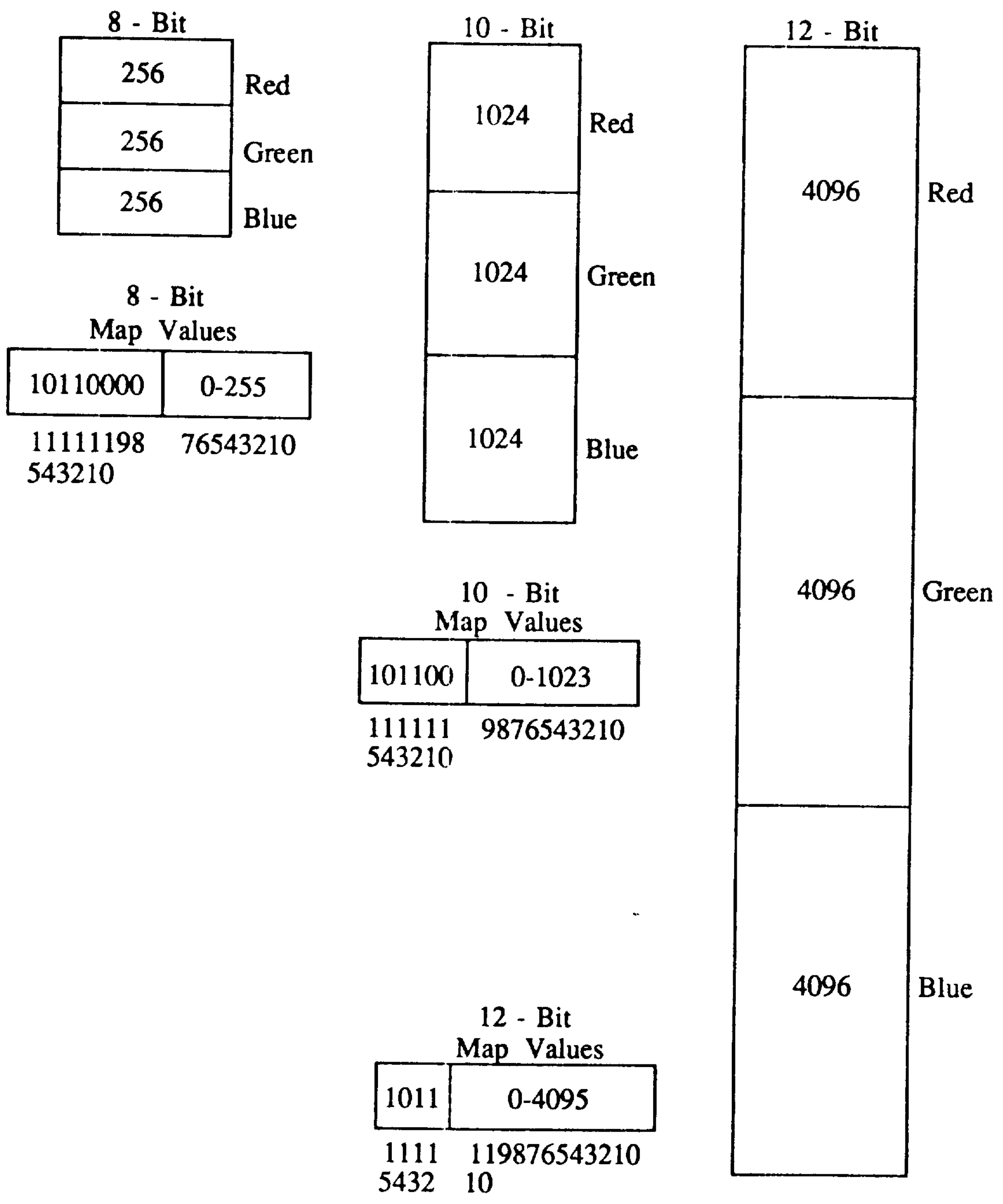
The Header Opcode defines the beginning of a new image. It must be the first opcode of a 512-byte/256-word record. The Header Opcode is a 32-bit opcode.

NOP - No Operation Opcode

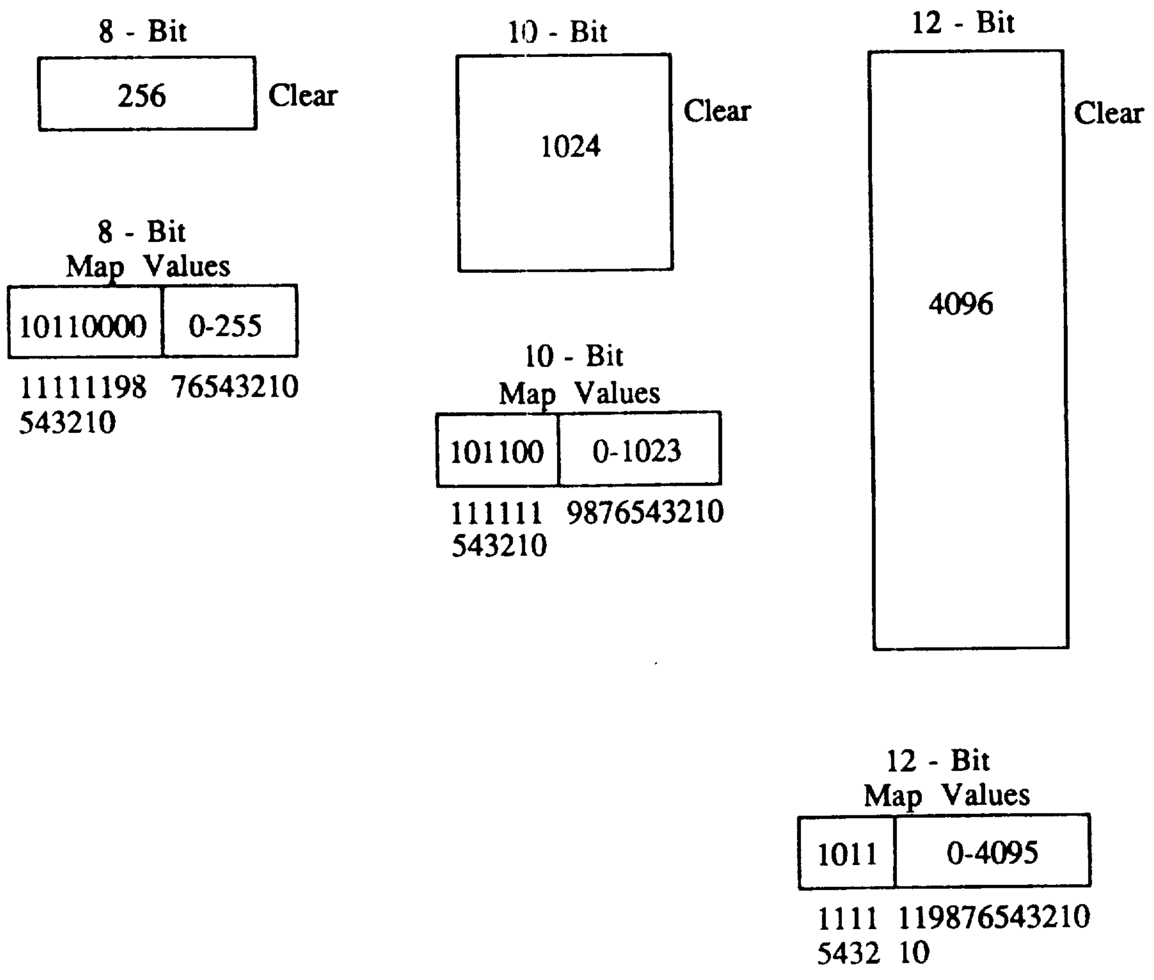
The No Operation Opcode is ignored. It may be used for filler in the Header Record and in records containing EF Opcode

APPENDIX A
Color Table Maps
(Color)

SECTION A.3



APPENDIX A
Color Table Maps
(Black and White)



APPENDIX B

RMT Utility Library Skeleton Program

The RMT utility library is a FORTRAN callable subset of the RMT used to convert from/to an arbitrary raster format to/from RM format. This appendix contains a simplified "skeleton" FORTRAN program that internally constructs an RM formatted image using the utility library. The skeleton code is commented so that the major conversion steps and associated RMT utility library routine calls are highlighted. The basic ordering of these processing steps should be adhered to for any RM conversion effort.

```
      PROGRAM CRERM
C
C***********************************************************************
C*
C*     "SKELETON" PROGRAM TO CREATE AN "RM" FORMATTED IMAGE  (JUNE 1988)
C*
C*     THIS PROGRAM IS INTENDED TO SERVE AS A GUIDE TO ANYONE USING THE
C*     RMT UTILITY LIBRARY ROUTINES TO CREATE A RM IMAGE FILE.
C*
C*     THE CODE ASSUMES A COLOR TABLE IMAGE IS TO BE GENERATED (BUT THE
C*     CHANGES REQUIRED FOR PRODUCING AN INTENSITY IMAGE (WITH 3
C*     CHANNELS) ARE COMMENTED WITH "CII").  THE IMAGE RESOLUTION AND
C*     COLOR LUT SIZES WERE ARBITRARILY SELECTED.
C*
C*     CODE TO EITHER CONSTRUCT THE IMAGE OR EXTRACT THE REQUIRED DATA
C*     FROM ANOTHER IMAGE FORMAT IS NOT INCLUDED.
C*
C*     THE UTILITY LIBRARY OUTPUTS SOME ERROR INFORMATION BUT THE USER
C*     MUST CONTROL THE FLOW.  IN THIS SKELETON PROGRAM,  ALL ERRORS
C*     RESULT IN AN ABORT.
C*
C*     THIS SKELETON MAY BE EASILY MODIFIED TO PRODUCE A PROGRAM THAT
C*     READS RM IMAGE DATA.
C*
C***********************************************************************
C
C      DIMENSION AN ARRAY TO HOLD SCANLINE DATA
       PARAMETER (MAXSL=512)
       INTEGER SCANLN(MAXSL)
C
C      DIMENSION AN ARRAY TO HOLD THE COLOR LUT
       PARAMETER (MAXLUT=8)
       INTEGER LUT(8,3)
```

APPENDIX B

```
C
C     DIMENSION AN ARRAY TO STORE THE RM HEADER
      PARAMETER (MAXHED=9)
      INTEGER HEADR(MAXHED)
C
C     DIMENSION CHARACTER ARRAY FOR OUTPUT RM FILE NAME
      CHARACTER*80 FILNAM
C
C     TYPE LOGICALS FOR RMT METAFILE WRITE FLAG AND UTILITY LIBRARY
C     ERROR RETURN FLAG
      LOGICAL LFWRIT,LFERR
C
C     NORMALLY THE COLOR TABLE WOULD BE CONSTRUCTED OR INPUT FROM
C     AN EXTERNAL SOURCE BUT HERE A DATA STATEMENT SUFFICES
      DATA LUT /   0, 255,   0, 255,   0, 255,   0, 255,
     1             0,   0, 255, 255,   0,   0, 255, 255,
     2             0,   0,   0,   0, 255, 255, 255, 255/
C
C     HARDWIRE FILE NAME
      FILNAM = 'RMIMAG'
C
C     HARDWIRE RMT METAFILE WRITE FLAG
      LFWRIT = .TRUE.
C
C     INITIALIZE THE RMT UTILITY LIBRARY
CRMT  RMT UTILITY LIBRARY REFERENCE
      CALL RMINIT
C
C     OPEN THE RM FILE FOR WRITING USING METAFILE NUMBER 1
CRMT  RMT UTILITY LIBRARY REFERENCE
      CALL OPENMF (FILNAM,1,LFWRIT,LFERR)
      IF (LFERR)  CALL EXIT
C
C     CONSTRUCT RMT HEADER VARIABLES
C     ASSUME 16-BIT DATA
      HEADR(1) = 0
C     ASSUME X AND Y RESOLUTIONS OF 512
      HEADR(2) = 512
      HEADR(3) = 512
C     ASSUME COLOR TABLE IMAGE
      HEADR(4) = 1
CII   HEADR(4) = 0
C     ASSUME COLOR (VERSUS BLACK-AND-WHITE)
      HEADR(5) = 1
C     ASSUME 8 BITS FOR COLOR TABLE ENTRY
      HEADR(6) = 8
CII   HEADR(6) = 0
```

APPENDIX B

```
C     ASSUME ONE CHANNEL FOR LUT FORMAT
      HEADR(7) = 1
CII   HEADR(7) = 3
C     ASSUME LENGTH OF COLOR TABLE IS 3-BITS (OR 8 ENTRIES)
      HEADR(8) = 3
CII   FOR INTENSITY DATA THIS IS NUMBER OF DATA BITS PER PIXEL
CII   HEADR(8) = 8
C     ASSUME 1ST COLOR LUT ENTRY USED FOR BACKGROUND
CII   SINCE LOOP FOR NUMBER OF CHANNELS,  THIS CODE WOULD ALSO ASSIGN
CII   A (0,0,0) INTENSITY FOR BACKGROUND
      DO 10  I=1,HEADR(7)
         HEADR(8+I) = 0
   10 CONTINUE
C
C     WRITE THE HEADER TO THE RM
CRMT  RMT UTILITY LIBRARY REFERENCE
      CALL WRHEAD (MAXHED,HEADR,LFERR)
      IF (LFERR)  CALL EXIT
C
C     WRITE THE COLOR LUT
CII   NOTICE THIS CODE IS SKIPPED FOR INTENSITY IMAGES
      IF (HEADR(4) .EQ. 1)  THEN
         LUTLEN = 2**HEADR(8)
         CALL WRLUT (MAXLUT,LUTLEN,3,LUT,LFERR)
CRMT  RMT UTILITY LIBRARY REFERENCE
         IF (LFERR)  CALL EXIT
      END IF
C
C*****      LOOP THRU THE NUMBER OF CHANNELS     *****
C
      DO 1000  I=1,HEADR(7)
C
C        WRITE RM FRAME/CHANNEL INFORMATION
CRMT  RMT UTILITY LIBRARY REFERENCE
         CALL BEGCHA (LFERR)
         IF (LFERR)  CALL EXIT
C
C        EXTRACT BACKGROUND COLOR FROM HEADER FOR USE IN LATER CALLS
         IBGC = HEADR(8+I)
C
C        LOOP THRU THE NUMBER OF SCANLINES
         DO 500  J=1,HEADR(3)
C
C           NORMALLY CODE SHOULD BE INSERTED TO CONSTRUCT OR READ SCANLINE
C           DATA BUT IN THIS CASE, EACH PIXEL IS SET TO AN ARBITRARY COLOR
C           LOOP THRU THE NUMBER OF PIXELS PER SCANLINE
            ICOLOR = MOD (J,MAXLUT)
```

APPENDIX B

```
              DO 100  K=1,HEADR(2)
               SCANLN(K) = ICOLOR
  100        CONTINUE
C
C            WRITE THE SCANLINE TO THE RM
CRMT   RMT UTILITY LIBRARY REFERENCE
             CALL PRMSL (.TRUE.,MAXSL,SCANLN,IBGC,LFERR)
             IF (LFERR)  CALL EXIT
C
  500     CONTINUE
C         END OF SCANLINE LOOP
C
C         WRITE RM FRAME/CHANNEL TERMINATION INFORMATION
CRMT   RMT UTILITY LIBRARY REFERENCE
          CALL ENDCHA (LFERR)
          IF (LFERR)  CALL EXIT
C
 1000 CONTINUE
C*****      END OF CHANNEL LOOP     *****
C
C      CLOSE THE RM FILE
CRMT   RMT UTILITY LIBRARY REFERENCE
       CALL CLOSMF (1,LFERR)
C
       CALL EXIT
       END
```

APPENDIX C
INVOKING THE RMT IN A UNIX ENVIRONMENT

In a UNIX environment the shell script *rmtran* may be used to link in the specified graphics device driver and invoke the RMT. A listing of the *rmtran* script is provided in this appendix. The shell variable RMTROOT contains the directory path where all relevant RMT files reside and, hence, must be edited by the installer.

```
#!/bin/csh
#
# script to build RMT executables
#
set RMTROOT = ~dpr/rmtran
set rmtbin = $RMTROOT/bin
set rmtlib = $RMTROOT/lib
set loadmodules = ${rmtbin}/dddummy.o
set libraries
#
#use shell variables for compiler name and options
set LINKOPT =
#for CONVEX use
set LINK = fc
#for CRAY use
#set LINK = segldr
#set LINKOPT = "-D DUPENTRY=NOTE"
#for Sun use
#set LINK = f77
#
if  ($#argv != 0) then
  switch ($1)
  case tek4109:
     set loadmodules = "${rmtbin}/ddio.o ${rmtbin}/ddtek.o"
     set libraries = ${rmtlib}/tek4100.a
     breaksw
  case  aed767:
     set loadmodules = "${rmtbin}/ddio.o ${rmtbin}/ddaed.o"
     set libraries = ${rmtlib}/aedlib.a
     breaksw
  case tek4125:
     set loadmodules = "${rmtbin}/ddio.o ${rmtbin}/ddtek4125.o"
     set libraries = ${rmtlib}/tek4100.a
     breaksw
  case postscript:
```

APPENDIX C

```
      set loadmodules = ${rmtbin}/ddpost.o
      breaksw
    case postscriptrgb:
      set loadmodules = ${rmtbin}/ddcpostrgb.o
      breaksw
    case postscriptlut:
      set loadmodules = ${rmtbin}/ddcpostlut.o
      breaksw
    case dicomed:
      set loadmodules = ${rmtbin}/dddicomed.o
      set libraries = ${rmtlib}/raslibrmt.a
      breaksw
    case versatec39:
      set loadmodules = ${rmtbin}/ddvtek39.o
      set libraries = ${rmtlib}/versalib.a
      breaksw
    case versateca:
      set loadmodules = ${rmtbin}/ddvteka.o
      set libraries = ${rmtlib}/versalib.a
      breaksw
    case versatecb:
      set loadmodules = ${rmtbin}/ddvtekb.o
      set libraries = ${rmtlib}/versalib.a
      breaksw
    case sun:
      set loadmodules = ${rmtbin}/ddsun.o
      set libraries = ${rmtlib}/sunrmtlib.a
      breaksw
    case sunmono:
      set loadmodules = ${rmtbin}/ddsunbw.o
      set libraries = ${rmtlib}/sunrmtlib.a
      breaksw
    case ibm:
      set loadmodules = ${rmtbin}/ddibm.o
      set libraries = ${rmtlib}/ibmlib.a
      breaksw
    default:
      echo " RMT executable not linked to device"
      breaksw
  endsw
endif
#echo $loadmodules $libraries
```

APPENDIX C

```
$LINK $LINKOPT  ${rmtbin}/rmtran.o  $loadmodules \
   ${rmtlib}/drvlib.a ${rmtlib}/rmtlib.a $libraries
unset loadmodules
unset libraries
setenv RMTERR ${RMTROOT}/RMTERR.DAT
setenv RMTHELP ${RMTROOT}/HELP.DOC
a.out
```

APPENDIX D
RMT UNIX MANUAL PAGE

A copy of the UNIX manual page for the RMT is provided in this appendix. The manual page in *nroff* format is provided as part of the standard distribution under the file name **rmtran.l**. Note that the location of the required RMT files is hardwired into the manual page. Consequently, the manual page must be edited by the installer.

NAME

rmtran – link RMT with device driver and execute

SYNOPSIS

rmtran [*device–name*]

DESCRIPTION

The *Raster Metafile (RM)* format is a generic raster image format adopted by LaRC. The *Raster Metafile Translator (RMT)* is an *RM* interpreter capable of reading, writing, displaying, and manipulating *RM* formatted images.

The *rmtran* script links the *RMT* with the user-specified device driver and executes.

If the *device-name* is omitted or an illegal name is specifed, the *RMT* is linked with the *dummy* device. This form of the *RMT* is useful in performing image conversion or image composition operations where no graphics output is generated.

The *CELCO* film recorder accepts *RM* formatted images as a standard input format. Other supported devices are listed below.

aed767	- AED 767
dicomed	- Dicomed D47 color film recorder
postscript	- PostScript laser printer
postscriptrgb	- Color PostScript laser printer (for intensity images)
postscriptlut	- Color PostScript laser printer (for color table images)
sun	- Sun (color)
sunmono	- Sun (monochrome)
tek4109	- Tektronix 4109
tek4125	- Tektronix 4125
versatec39	- Versatec color electrostatics plotter
versateca	- Versatec color thermal plotter (A size)
versatecb	- Versatec color thermal plotter (B size)

For non-terminal devices the *RMT* produces a *device-specific* graphics output file that must be "ported" to the designated device. (See postprocessor documentation for device-specific details.) The *device-name* and the resulting permanent file name are listed below.

dicomed	- D47OUT
postscript	- PSOUT
sun	- sun.img
versatec	- VEROUT

The **rmtran** script is located in

```
/usr/local/graphics/rmtran/rmtran
```

Upon exit from the *RMT* the executable *a.out* remains in the user's current directory. If the *RMT* is executed from outside of the **rmtran** script, it is necessary to set two environment variables as follows:

```
setenv RMTERR /usr/local/graphics/rmtran/RMTERR.DAT
setenv RMTHELP /usr/local/graphics/rmtran/HELP.DOC
```

For further information on the functionality of the *RMT* use the on-line "help" facility or consult the file

```
/usr/local/graphics/rmtran/HELP.DOC
```

Templates in the form of *RMT* alternate command files are provided to assist users with some commonly performed image operations. These templates are located and named as listed below:

/usr/local/graphics/rmtran/*.acf

Users wishing to read or write *RM* formatted images without knowing specifics of the *RM* format may invoke the *RMT* utility library through a series of FORTRAN subroutine calls. On-line documentation (*RMTUTIL.DOC*) describes the individual user-callable utility routines. In addition, a "skeleton" FORTRAN program (*skeleton.f*) is provided to serve as a template for utilizing the utility library. The location of the *RMT* utility library, on-line documentation and "skeleton" program are listed below:

/usr/local/graphics/rmtran/rmtlib.a
/usr/local/graphics/rmtran/RMTUTIL.DOC
/usr/local/graphics/rmtran/skeleton.f

Before using the *RMT* utility library, the user should define the environment variable *RMTERR* as defined above and use the following compile/load sequence

```
fc skeleton.f /usr/local/graphics/rmtran/dddummy.o \
          /usr/local/graphics/rmtran/rmtlib.a
```

Certain supported graphics software packages such as *RASLIB* or *MOVIEBYU* are capable of producing *RM* formatted images without the aid of an external translator. (See the appropriate package documentation for details.)

FILES

/usr/local/graphics/rmtran/rmtran	- script for executing rmt
/usr/local/graphics/rmtran/rmtlib.a	- rmt utility library
/usr/local/graphics/rmtran/HELP.DOC	- rmt on-line help information
/usr/local/graphics/rmtran/RMTERR.DAT	- rmt error messages
/usr/local/graphics/rmtran/drvlib.a	- rmt main driver library
/usr/local/graphics/rmtran/dddummy.o	- rmt "dummy" device driver
/usr/local/graphics/rmtran/ddio.o	- rmt device IO routines
/usr/local/graphics/rmtran/ddtek.o	- rmt Tektronix 4109 device driver
/usr/local/graphics/rmtran/ddtek4125.o	- rmt Tektronix 4125 device driver
/usr/local/graphics/rmtran/ddaed.o	- rmt AED 767 device driver
/usr/local/graphics/rmtran/ddpost.o	- rmt PostScript device driver
/usr/local/graphics/rmtran/ddcpostrgb.o	- rmt color PostScript device driver
/usr/local/graphics/rmtran/ddcpostlut.o	- rmt color PostScript device driver
/usr/local/graphics/rmtran/ddvtek39.o	- rmt Versatec electostatic driver
/usr/local/graphics/rmtran/ddvteka.o	- rmt Versatec thermal driver
/usr/local/graphics/rmtran/ddvtekb.o	- rmt Versatec thermal driver
/usr/local/graphics/rmtran/dddicomed.o	- rmt Dicomed D47 device driver
/usr/local/graphics/rmtran/ddsun.o	- rmt Sun color device driver
/usr/local/graphics/rmtran/ddsunbw.o	- rmt Sun monochrome device driver
/usr/local/graphics/rmtran/sunrmtlib.a	- rmt Sun utility library
/usr/local/graphics/rmtran/tek4100.a	- Tektronix 4100 utility library
/usr/local/graphics/rmtran/aedlib.a	- AED 767 utility library
/usr/local/graphics/rmtran/versalib.a	- Versatec raster utility library
/usr/local/graphics/rmtran/raslibrmt.a	- rmt Dicomed D47 utility library
/usr/local/graphics/rmtran/*.acf	- rmt command file templates
/usr/local/graphics/rmtran/RMTUTIL.DOC	- rmt utility library documentation

SEE ALSO

moviebyu, raslib, rmdev

APPENDIX E
RMT ERROR MESSAGES

RMT error messages reside in the standard distribution file, RMTERR.DAT. This appendix provides a list of this file and, thus, all RMT error messages. The three fields in each error entry are the error number, error message, and the (abbreviated) error severity level.

NO.	MESSAGE	SEVERITY
1005	COMMAND INPUT TRUNCATED	S
1010	ILLEGAL COMMAND INPUT SUBCOMMAND INPUT	S
1025	ILLEGAL WINDOW SELECTION	S
1030	ILLEGAL WINDOW COORDINATE SPECIFICATION	W
1035	OVERLAY ERROR - INCOMPATIBLE FORMAT ON COMPONENT IMAGES	W
1040	ILLEGAL RASTER METAFILE NUMBER	W
1045	ILLEGAL PICTURE NUMBER OR ILLEGAL RANGE SPECIFICATION	S
1050	ILLEGAL KEYWORD OR ARGUMENT IN RANGE SPECIFICATION	S
1055	DIRECTORY COMMAND ERROR ENCOUNTERED DURING READ	F
1060	SET METAFILE COMMAND ERROR	F
1065	ILLEGAL RASTER METAFILE NAME	W
1070	DRAW COMMAND SYNTAX ERROR	S
1075	DRAW COMMAND ERROR	F
1080	SET WINDOW COMMAND ERROR	W
1085	SET VIEWPORT COMMAND ERROR	W
1090	CONVERSION COMMAND SYNTAX ERROR	S
1095	OVERLAY COMMAND SYNTAX ERROR	S
1100	OVERLAY COMMAND ERROR	F
1105	SET LOG COMMAND ERROR	W
1110	LOG FILE OPEN ERROR	W
1115	LOG FILE CLOSE ERROR	W
1120	ALTERNATE COMMAND FILE READ ERROR	F
1125	ALTERNATE COMMAND FILE OPEN ERROR	F
1130	ALTERNATE COMMAND FILE CLOSE ERROR	F
1135	SOURCE COMMAND ERROR	W
1140	NESTING OF ALTERNATE COMMAND FILES NOT ALLOWED	F
1145	ALTERNATE COMMAND FILE CANNOT BE FOUND	F
1150	MAXIMUM RMT COMMAND ARGUMENT LIMIT EXCEEDED	S
1155	SET QUANTIFY COMMAND ERROR	S
1160	SET DITHER COMMAND ERROR	S
1165	INCOMPATIBLE QUANTIZATION COLOR TABLE PARAMETERS	F
1170	SET DEFAULT COMMAND SYNTAX ERROR	S
1175	COPY PICTURE COMMAND SYNTAX ERROR	S
1180	GET METAFILE COMMAND ERROR	W
1185	GET WINDOW COMMAND ERROR	W
1190	GET VIEWPORT COMMAND ERROR	W

APPENDIX E

NO.	MESSAGE	SEVERITY
1195	SET CLIP COMMAND SYNTAX ERROR	S
1200	SET CHANNEL COMMAND SYNTAX ERROR	S
1205	ILLEGAL DATA CHANNEL DESIGNATION	W
1210	SET COLOR COMMAND SYNTAX ERROR	S
1215	SCALE COMMAND SYNTAX ERROR	S
1220	SCALE FACTOR (NUMBER OF BITS) OUT OF RANGE	W
1225	ILLEGAL HELP COMMAND INPUT	S
1230	ILLEGAL HELP SUBCOMMAND INPUT	S
1235	END OF FILE ENCOUNTERED ON HELP FILE	S
1240	NO ERROR TERMINATION LEVEL SPECIFIED, ASSUMED FATAL	S
1245	ILLEGAL ERROR TERMINATION LEVEL	S
1250	ILLEGAL GET SUBCOMMAND INPUT	S
1255	ILLEGAL/MISSING COMPONENT NUMBER	S
1260	ILLEGAL COMPONENT SPECIFICATION	S
1265	ILLEGAL/MISSING ATTRIBUTE NUMBER	S
1270	ILLEGAL ATTRIBUTE SPECIFICATION	S
1275	MISSING NO-BACKGROUND-COLOR-FLAG	S
1280	ILLEGAL/MISSING COMPOSITION OPERATOR	S
1285	ILLEGAL/MISSING BACKGROUND COLOR SPECIFICATION	S
1290	ILLEGAL/MISSING LUT RANGE SPECIFICATION	S
1295	ILLEGAL/MISSING LUT SIZE SPECIFICATION	S
1300	ILLEGAL/MISSING LUT BITS/CHANNEL SPECIFICATION	S
2005	METAFILE DOES NOT EXIST	W
2010	METAFILE CANNOT BE OPENED	F
2015	METAFILE READ ERROR	F
2025	METAFILE WRITE ERROR	F
2030	METAFILE HEADER READ ERROR	F
2035	METAFILE HEADER WRITE ERROR	F
2040	METAFILE COLOR TABLE READ ERROR	F
2045	METAFILE COLOR TABLE WRITE ERROR	F
2050	METAFILE NOT OPEN / CANNOT READ	F
2055	METAFILE NOT OPEN / CANNOT WRITE	F
2060	END-OF-FILE ENCOUNTERED ON METAFILE	F
2065	METAFILE POSITIONING ERROR	F
2070	METAFILE CANNOT BE CLOSED	F
2075	METAFILE ALREADY OPEN	W
2080	METAFILE UNIT IN USE	W
3005	METAFILE ILLEGAL OPCODE ERROR	F
3010	UNEXPECTED METAFILE OPCODE ENCOUNTERED	F
3015	ILLEGAL OPCODE AFTER RUN LENGTH OPCODE	F
3020	HEADER - OPCODE MISMATCH	F
3025	COLOR TABLE NOT PERMITTED FOR INTENSITY DATA	F

APPENDIX E

NO.	MESSAGE	SEVERITY
3030	METAFILE HEADER PARAMETER ERROR	F
3035	ILLEGAL RUN LENGTH ENCOUNTERED	F
3040	ILLEGAL X OFFSET EMBEDDED IN SCANLINE	F
3045	SCANLINE "EXPANSION" ERROR	F
3050	SCANLINE COMPACTION ERROR	F
3055	SCANLINE BUFFER SIZE EXCEEDED	F
3060	ILLEGAL DATA CHANNEL NUMBER	F
3065	MAXIMUM Y RESOLUTION EXCEEDED	F
3070	METAFILE IMAGE IN COLOR TABLE FORMAT - CONVERSION IMPOSSIBLE	W
3075	ILLEGAL QUANTIZATION SCHEME SPECIFIED	W
3080	COLOR LUT SIZE DISCREPENCY - CONVERSION IMPOSSIBLE	W
3085	CONVERSION ERROR - COLOR TABLE TO INTENSITY FORMAT	F
3090	CONVERSION ERROR - INTENSITY TO COLOR TABLE FORMAT	F
3095	COPY PICTURE COMMAND ERROR	F
3100	PICTURE "RE-SIZING" ERROR	F
3105	USER CHANNEL DESIGNATION EXCEEDS AVAILABLE IMAGE CHANNELS	W
3110	USER CHANNEL DESIGNATION INCOMPATIBLE WITH CONVERSION	W
3115	IMAGE COLOR TABLE SIZE EXCEEDS RMT LIMITS	F
3120	ERROR IN "SCALING" IMAGE	F
4005	INTERNAL (RMT) FILE CANNOT BE OPENED	F
4010	INTERNAL (RMT) FILE IS NOT OPEN	F
4015	INTERNAL (RMT) FILE READ ERROR	F
4020	INTERNAL (RMT) FILE WRITE ERROR	F
4025	INTERNAL (RMT) UNIT LIST EXHAUSTED	F
4030	INTERNAL (RMT) FILE CLOSE ERROR	F
4035	INTERNAL (RMT) FILE BUFFER PROCESSING ERROR	F
5005	GRAPHICS OUTPUT FILE OPEN ERROR	F
5010	GRAPHICS OUTPUT VIEWPORT SELECTION ERROR	W
5015	GRAPHICS OUTPUT VIEWPORT COORDINATE ERROR	W
5020	GRAPHICS DEVICE AND IMAGE FORMAT INCOMPATIBLE	W
5025	GRAPHICS OUTPUT DEVICE INCOMPATIBLE WITH CHANNEL DESIGNATION	W
5030	ILLEGAL DEVICE QUERY	W
6005	FRAME-BUFFER (RMT) FILE CANNOT BE OPENED	F
6010	FRAME-BUFFER (RMT) FILE IS NOT OPEN	F
6015	FRAME-BUFFER (RMT) FILE READ ERROR	F
6020	FRAME-BUFFER (RMT) FILE WRITE ERROR	F
6025	FRAME-BUFFER (RMT) UNIT LIST EXHAUSTED	F
6030	FRAME-BUFFER (RMT) FILE CLOSE ERROR	F

APPENDIX E

NO.	MESSAGE	SEVERITY
6035	FRAME-BUFFER (RMT) NOT INITIALIZED	F
6040	FRAME-BUFFER (RMT) RECORD NUMBER LIMITS EXCEEDED	F

APPENDIX F
RMT COMMAND FILE TEMPLATES

This appendix presents RMT command file templates for common RM processing operations. These command file templates are annotated to enhance understanding. All of the command file templates listed below are supplied on-line as part of the standard RMT release.

clipres.acf:

```
*****************************************************
*  DESCRIPTION - This template demonstrates clip/resize
*            (resizing "up" and "down").
*****************************************************
* Load input image
set mf 1 input_metafilename
* Window set to input image resolution
set w 1 ( 0 511 0 511)
* Viewport set to upper left quadrant of display surface
set v 1 ( 0 255 0 255)
* Display the clipped image (clipping is the default)
draw
* Turn re-sizing "on"
set a 1 clip off
* Resize image "down"
draw c 1 a 1
* Viewport set larger than image resolution
set v 2 (0 1024 0 1024)
* Associate "large" viewport with component "2"
set c 2 v 2
* Resize image "up"
draw c 2 a 1
quit
```

APPENDIX F

drawver.acf:

```
*******************************************************
*  DESCRIPTION - This template demonstrates command
*              sequence necessary to output to
*              Versatec plotters.
*******************************************************
* Load input image
set mf 1 input_metafilename
* If thermal Versatec (11 in.), uncomment next line
*set a 1 channel 3 (3 2 1)
* Draw picture
* The next two input lines are in response to the Versatec driver
* prompts.  (This is the only driver that prompts for informa-
* tion.)
* These two lines correspond to a 4x4 dither and linear mapping.
draw
4
0
quit
```

APPENDIX F

i2lut.acf:

```
*****************************************************
*  DESCRIPTION - This template demonstrates intensity
*              to color LUT conversion using the
*              three available quantization schemes.
*****************************************************
*
* This templates assumes an LUT with 256 entries (size) where each entry has a red, green,
*  and blue range from 0 to 255.
* Load input image
set mf 1 input_metafilename
* Uncomment the following 3 lines to perform color cube quantiza-
* tion
*set a 1 quantify off
*set a 1 range 255
*set a 1 size 8
* The above three lines could be combined as follows:
*set a 1 quantify off range 255 size 8
* Uncomment the following 2 lines to perform uniform quantization
* (The next line assumes 3 bits for red and green and 2 bits for
* blue.)
*set a 1 size 8
*set a 1 bpc 3 3 2
* Uncomment the following 2 lines to perform dithering
*set a 1 dither on
*set a 1 range 255
* Define output metafile name
set mf 2 output_metafilename w
* Associate output metafile with component
set c 2 mf 2
* Convert to intensity
convert
quit
```

APPENDIX F

luti2bw.acf:

```
*****************************************************
*  DESCRIPTION - This template demonstrates color LUT
*              or intensity conversion to black and
*              white.
*****************************************************
*
* Load input image
set mf 1 input_metafilename
* Turn color off
set a 1 color off
* Define output metafile name
set mf 2 output_metafilename w
*Associate output metafile with component
set c 2 mf 2
* Convert to black and white
convert
quit
```

APPENDIX F

lut2i.acf:

```
*****************************************************
*  DESCRIPTION - This template demonstrates color LUT
*              to intensity conversion.
*****************************************************
*
* Load input image
set mf 1 input_metafilename
* Define output metafile name
set mf 2 output_metafilename w
*Associate output metafile with component
set c 2 mf 2
* Convert to intensity
convert
quit
```

APPENDIX F

ovrlay.acf:

```
*****************************************************************
* DESCRIPTION - This command file demonstrates the OVERLAY command.
*****************************************************************
* This template assumes a multi-image RM file with four component * images.
* The first and third images have a resolution of 512x512 and the * second and fourth images have a resolution of 1024x1024. The
* four component images are to be placed in the four quadrants of * a 2048x2048 output image.
* Load input image
set mf 1 input_metafilename
* set viewports to four quadrants of output image
set v 1 (0 1023 0 1023)
set v 2 (1280 1791 256 767)
set v 3 (0 1023 1024 2047)
set v 4 (1280 1791 1280 1791)
* load metafile for writing
set mf 2 out_metafilename
* associate images and viewports with components
set c 2 p 2 v 2
set c 3 p 3 v 3
set c 4 p 4 v 4
* associate output metafile with component
set c 5 mf 2
*perform the overlay
overlay c 1 c 2 c 3 c 4 c 5
* exit the rmt
quit
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-102588	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Raster Metafile and Raster Metafile Translator		5. Report Date September 1989
		6. Performing Organization Code
7. Author(s) Nancy L. Taylor, Eric L. Everton, Donald P. Randall, Raymond L. Gates, Kristi M. Skeens		8. Performing Organization Report No.
		10. Work Unit No. 505-60-01-01
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665-5225		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001		14. Sponsoring Agency Code

15. Supplementary Notes

Nancy L. Taylor and Eric L. Everton: Langley Research Center, Hampton, Virginia.
Donald P. Randall, Raymond L. Gates, and Kristi M. Skeens: Computer Sciences Corporation, Hampton, Virginia.

16. Abstract

The intent of this report is to present an effort undertaken at NASA Langley Research Center to design a generic raster image format and to develop tools for processing images prepared in this format. This document addresses both the Raster Metafile (RM) format and the Raster Metafile Translator (RMT). This document is intended to serve a varied audience including: users wishing to display and manipulate raster image data, programmers responsible for either interfacing the RM format with other raster formats or for developing new RMT device drivers, and programmers charged with installing the software on a host platform.

17. Key Words (Suggested by Author(s)) graphics software		18. Distribution Statement Unclassified - Unlimited Subject Category - 61	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 115	22. Price A06

NASA FORM 1626 OCT 86