N84-31949

NASA

Technical Memorandum 86136

A DEVICE-INDEPENDENT INTERFACE FOR INTERACTIVE IMAGE DISPLAY

D. C. Perkins
M. R. Szczur
J. Owings
R. K. Jamros

MAY 1984

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

NASA's Goddard Space Flight Center (GSFC) has developed a Transportable Applications Executive (TAE) for use in implementing portable applications software that can be shared by different research projects. Since many of the supported disciplines require the interactive display and manipulation of remotely sensed images, a device independent Display Management Subsystem (DMS) is being written as a TAE extension. The DMS attempts to abstract and standardize the device dependent functions that are used in the display and manipulation of image data on image analysis terminals. This paper explores the structure of DMS and the interface routines that are available to the applications programmer for use in developing a set of portable image display utility programs.

1.0 INTRODUCTION

Much of the research carried out at NASA's Goddard Space Flight Center (GSFC) involves computer-aided analysis of remotely sensed image data obtained by meteorological, oceanographic, or earth resources satellite missions. The scientific analysis and interpretation is aided by the ability to display and manipulate the data quickly and easily using raster devices, referred to here as image analysis terminals (IAT).

The rapidly changing market in IATs over the last decade, combined with disparate system development cycles and goals, has led to a proliferation of analysis systems configured around different IATs. The result of such a history of development is a collection of programs, tailored to specific IATs, which cannot be used on different IATs without spending considerable time and effort on conversion. Yet, many of the programs could otherwise be usefully shared among the different systems.

The Display Management Subsystem (DMS) of the Transportable Applications Executive (1) was conceived as a means of standardizing the interface to a variety of IATs in support of new projects. Its purpose was to offer device independence so that programs could be made portable among a variety of IATs. While it is a subsystem to TAE, the concepts and techniques are general and could be readily implemented outside of TAE.

The development of device independent concepts and packages for data display on graphics and imaging devices is an active subfield in computer science research. Much of the work is relevant to DMS development. For example, current standardization work by national and international committees (references 2, 8) is a rich source for ideas on device abstraction. The design of subroutine packages, with respect to internal relationships as well as portability considerations, is an applicable discipline (references 5, 9, 10). Also, a direct relationship can be found in work whose goal is to develop abstract models of raster devices (references 1, 11).

However, DMS also has a goal outside of the realm of defining a conceptual model of an IAT, namely, to give a user easy access to displayed images by cataloging them as named "files", with associated attributes and user descriptions. Some approaches to this goal can be found in various vendor-developed systems, for example, the International Imaging Systems System 575 (reference 7), which keeps a catalog of user-supplied image names, with descriptive information such as the spatial and spectral dimensions of the image. DMS power lies in the combination of such user-oriented features with its device independent concepts.

In this paper we describe the DMS goals, design and current implementation. Because the DMS development cycle was constrained to meet the needs of ongoing projects, the approach taken was to design and develop a *prototype which incorporated the device independent* advantages of the DMS concept, but with functions limited to actual requirements of supported projects. This approach had the advantage of producing an early software implementation which could be evaluated before more comprehensive decisions about a formal DMS model were made. The functional capabilities referred to in this paper are those of the DMS prototype.

In the next section of this paper we explore the DMS concepts which support the goals of naming data and providing generic devices. Following that we describe the DMS data structures and the techniques used to meet the goals. We also discuss actual prototype programs and implementation details. Finally, we present a summary of the work accomplished and discuss future plans.

2.0 DMS CONCEPTS

DMS has two primary objectives: to establish a software environment for an interactive user that allows that user to control, manipulate and do analysis using an IAT without having to understand specific characteristics of the hardware or supporting software; and to allow programs which access IATs to be written independent of any specific IAT type.

To meet these objectives, the DMS designers established three major requirements:

1. users of DMS must have services, similar to operating system file management services, for managing data displayed on an IAT;

2. programs must be able to perform actions on IATs without addressing a specific vendor's hardware characteristics;

3. an interactive user must be able to exclusively control or selectively (and deliberately) share a particular IAT, and have that device used

automatically by programs he/she runs.

In support of these requirements, the DMS designers formalized definitions of "images" and IAT categories for DMS users.

2.1 Images

In an interactive image display and processing system, the data base that is operated on by the software consists of digital image (2) files and related ancillary information. The images are stored in either disk files, tape files or IAT refresh memories. The last storage medium, refresh memories, differs in that the contents of one or more memories can be dynamically directed to an IAT monitor for viewing by the user, while being enhanced by application of transformation tables or altered by zooming, panning, etc.

In a departure from earlier systems (e.g., references 3, 4), which required that a user first place data into refresh memories, then independently apply intensity transformation (lookup) tables, configure the memories for viewing, and finally, remember the details of this configuration for later viewing, DMS combines these types of sequences into single operations. The result of an operation is named, and the entire configuration can be later recalled by use of that name. So, for example, in one operation a user can load three bands of data, declare them to be a false color image, and name that image WASHINGTON. Later, after displaying other data sets or results, the user can type

```
VIEW WASHINGTON
```

to return to viewing the original false color image.

The following list summarizes the information about a named image which is known to DMS:

- the refresh memories on which the data are stored
- the transformations applied to that data (e.g., intensity or color assignments, shifts, zoom factor)
- the image type (e.g., full color, stereo, black & white)
- the associated protection (locked (cannot be replaced), or unlocked (may be replaced by new image))

- o image age (oldest replaceable images are replaced first)
- o image size (height, width, number of bits per pixel)
- o the image start location in refresh memory
- o the source of the data (disk file(s))
- o header labels (if any) from the source files

2.2 IAT Categories

DMS operations are based on the concept of generic devices. To DMS, any one IAT is identified only as a collection of capabilities for data storage and control. Thus, specific individual IATs within a system are described by their configuration. For example, a simple device may be identified as having three 512 x 512 refresh memories, three lookup tables, one monitor, one cursor, and one graphics overlay plane.

Each of these devices (collections of characteristics) may be given one or more names. The purpose of the naming is twofold. First, IATs can be given meaningful names, which will allow users to allocate them simply. Once a user allocates an IAT by name, then DMS tracks the set of capabilities available to that user for data processing.

Second, programs may also request IATs by name. DMS requires that a program which accesses an IAT declare what capabilities it expects to use. This powerful feature allows programs to be concerned only with what they intend to accomplish, not with specifics of a device. DMS verifies this request against the capabilities of the IAT owned by the user of the program, rejecting the request if the program's requirements cannot be met.

Rather than force a programmer to request a device by listing a complete set of options, DMS defines some standard groupings of characteristics. Thus, a program may state its requirements by naming a particular standard device. It may also request additional features to be added onto its request, such as an extra cursor.

For example, the IAT described above is a standard category, called FULL for full color. A program might declare that it needs FULL, along with an additional overlay plane and hardware zoom. DMS would assure that the user's IAT has the required features.

Figure 1 is an example of a possible name/characteristic association which could be used at a DMS installation. It is expected that while a user will allocate an IAT by its unique name (e.g., TIGER), programs will typically request devices by standard names and options. The program can then be run using any IAT supporting the required characteristics. Only programs which require use of a particular IAT or a particular model of IAT would use the unique names.

A program may also query characteristics of a user's IAT if certain requirements are flexible. E.g., if 10 memories are adequate, but 12 would be better, then the number actually available can be determined at run time.

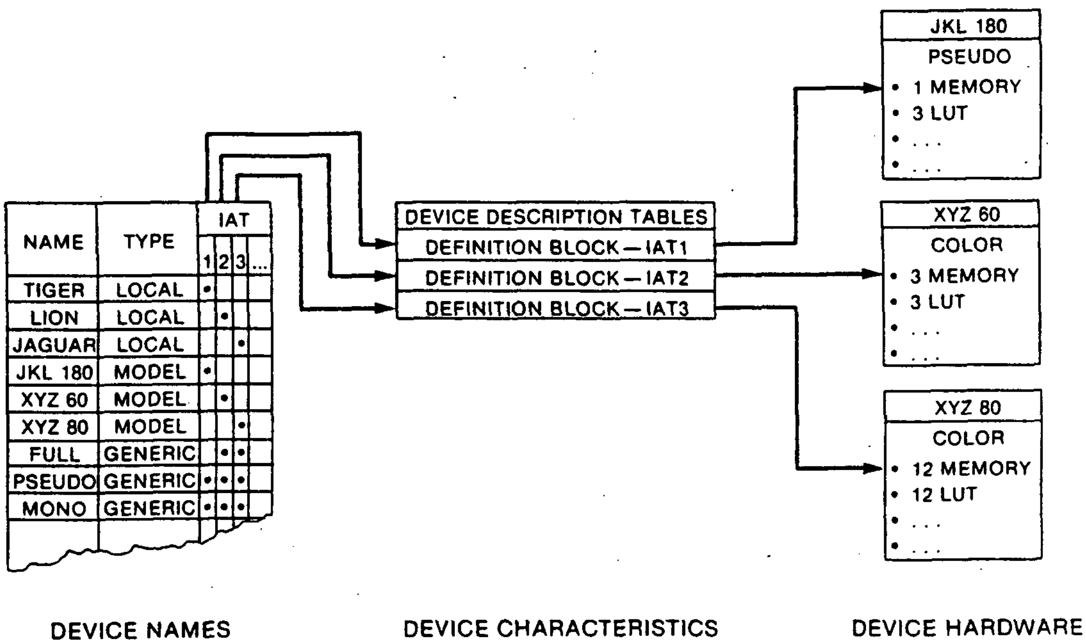

Figure 1: Device Name-Characteristic Association

3.0 DMS STRUCTURE

DMS is founded upon a set of data structures which capture and identify the main DMS concepts. Several software components have been developed to manipulate the data structures and to allow programmer access to the IATs. These include:

- Data structure management
- Generic image manipulation services
- Device dependent services
- Image input/output support
- Image display utility programs

Figure 2 illustrates the relationship of the DMS components to each other and to programs using them.

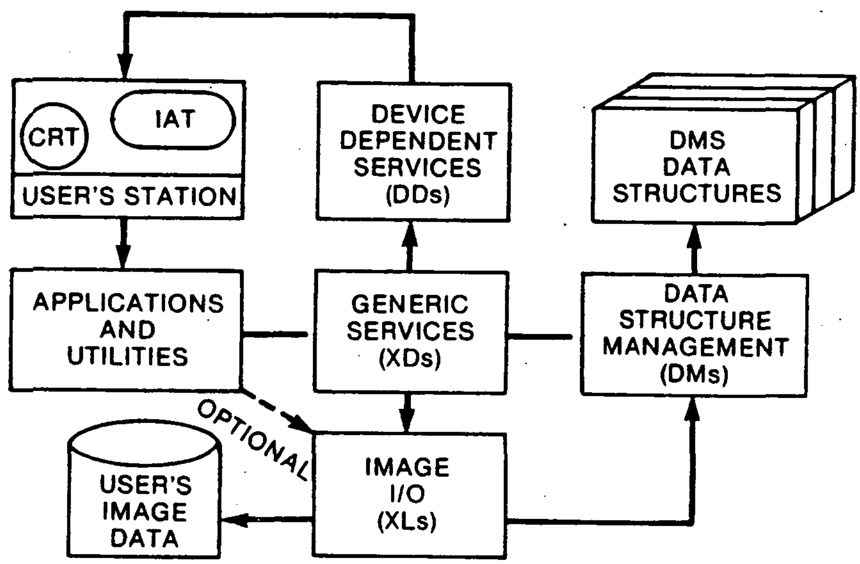

Figure 2: DMS Components

The following sections describe the characteristics and usage of the various components, followed by specific software implementation details.

3.1 DMS Data Structure Management

Four data structures support all environmental and control information maintained by DMS:

<u>Device ID Block.</u> This block identifies the IAT currently owned by a user. It is accessed by the DMS initialization routine which every program calls to attach itself to a user's IAT.

Display Memory Table. The DMT associates images defined by a user with a device's memories and maintains characteristics of all images, e.g., image type, name, physical location(s), age, lock switch, etc. This table is used extensively by DMS routines to translate between a user's image and device-specific addresses and registers. The table is accessible through a subroutine package, known as the DM package (3), which queries and updates table entries. The following is a sample of DM routines:

- DMCHAR - verify characteristics of a user's IAT
- DMCTYP - get image type
- DMDEFI - define image
- DMDID - validate IAT name
- DMGETM - get refresh memory IDs for an image

Display Device Table. The DDT contains a physical description of IAT characteristics (e.g., number of refresh memories, number of cursors, number of buttons, other hardware features) plus user information (e.g., name of IAT, current owner). IAT hardware characteristics are initially identified to DMS through an interactive program which is run as part of DMS installation. Whenever a facility adds a new IAT or wishes to modify an existing IAT description, the extension is a simple "edit" of the table. Thus, an IAT can be upgraded without changing existing application software.

Device Name Table. The DNT contains all names by which any IAT on a given system can be known. IATs may have more than one name, and names may be generic (e.g., monocolor device), specific (e.g., XYZ 60) or facility-inspired (e.g., TIGER). The table is established as part of the DMS installation procedure through an interactive program. This table may also be edited.

3.2 Generic Image Manipulation Services

Application programs interact with IATs through a set of generic DMS image manipulation functions, collectively known as the XD package. Use of this package for all device interaction makes the program usable on any IAT within a system.

The XD package serves the purpose of hiding all DMS table manipulation and physical device access from the application programmer. A program makes an initial call to XDGETD to declare what IAT category and optional characteristics it requires to function. Thereafter, the program merely issues instructions to perform the required action on the IAT.

For example, a program initializes itself by calling XDGETD to declare its device requirements. It then calls XDZOOM to zoom an image. XDZOOM performs the following sequence of events:

- o calls table management routine DMGETM to get the memory IDs associated with an image;
- o calls XDCOTR to translate from image to screen coordinates;
- o calls DDZOOM (device dependent zoom routine) to perform the actual image zoom. Whether the zoom action is done in hardware or software, and the specifics of the actual protocols for accomplishing the task, are transparent to the programmer.

DMS also handles the windowing of IAT-resident images at the XD level. Incoming data may be written into the refresh memory at an offset from the base coordinates of that memory. The data is mapped into the window by element - no spatial transformations are performed. Any subsequent references to points in the resulting image are made relative to the beginning of the image. That is, a user works in the coordinates of the image; DMS translates between that system and refresh memory coordinates.

The following is a sample of the available XD routines.

Initiation and Termination Routines

- o XDGETD - connect an application to an IAT
- o XDILUT - initialize lookup tables
- o XDCLR - initialize refresh memories
- o XDMNDF - define button menu
- o XDEXIT - clean up at end of process

Image transfer and setup

- o XDDROP - move image from disk to IAT
- o XDSAVE - move image from IAT to disk
- o XDDEFI - define a new image
- o XDIMRD - read image subarea
- o XDIMWR - write image subarea

Image viewing and alteration of viewing

- XDSDIW - set (define) image window
- XDVIEW - display an image on the screen
- XDLUTR - read the lookup table(s) associated with an image
- XDLUTW - write lookup table(s) for an image
- XDLUTI - write a linear lookup table
- XDALIN - register images interactively (e.g., by trackball)
- XDENGR - logically "OR" graphics overlay with an image
- XDFADE - fade between images
- XDFLIC - loop through a sequence of images
- XDSHFT - shift image (horizontally or vertically)
- XDZOOM - zoom an image (by pixel replication)

Cursor and interrupt routines

- XDCRDF - define shape(s) of cursor(s)
- XDCRON - turn on cursor
- XDCROF - turn off cursor
- XDCRRD - read the cursor position
- XDCRWR - move the cursor to a given position
- XDWTIR - wait on interrupt (from cursor or function key)

Miscellaneous routines

- XDCOTR - translate between image and screen coordinates
- XDFILI - retrieve source file name of IAT image
- XDWAIT - pause for a given amount of time
- XDXCOL - translate color names into red-green-blue values

3.3 Device Dependent Services

The DMS software layer which directly addresses an IAT is known as the DD package. Where the XD package is portable across IATs, the DD package must be reimplemented for each different IAT architecture, and thus serves as a "device driver". The DD routines are called by the XD routines. It is possible for an application program to call DD routines (or, for that matter, vendor supplied routines) directly. However, it will no longer be transparently portable to other IATs.

Unlike the XD routines, which deal in "images" and windows, DD routines work with refresh memories and other hardware elements and access an IAT directly through vendor supplied or other device specific software.

3.4 Image I/O Support

As part of the DMS development effort, an ancillary package to perform I/O on disk-resident image data was implemented. DMS was targeted for use on several systems, each of which had its own disk-based data structures for images. Rather than try to accommodate system-specific formats, the DMS team developed a set of protocols for accessing image data which could be layered over locally used I/O, and which were independent of any physical data structure. (4)

This package, known as XL, allows an application program to read, write and update image files and their labels. In its initial implementation, the XL package requires that an image file be a single band image stored as a disk file. Subsetting of a disk-based image for input is supported. (Note: XL can also handle a single refresh memory as an input or output image.)

The XL routines maintain their independence of data structure by using keyword parameters to describe the data. Thus, when a program needs a particular kind of information, such as pixel size, it calls an XL routine with that keyword. The XL routine queries the data structure and returns the required information. The implementation of the XL routines and the tables they use are system dependent. The calls are generic. Use of these routines will facilitate the porting of programs to other systems which also have an XL implementation. The following lists some XL routines.

- XLOPEN - Open an image file
- XLREAD - Read from an image file
- XLWRIT - Write to an image file
- XLCLOS - Close an image file
- XLFTCH - Retrieve information about a file
- XLADD - Update file description information
- XLUNIT - Get a logical unit number for a file name
- XLGET - Retrieve fields from image file header
- XLPUT - Put fields into image file header

It should be noted that this package was developed as a convenience for application programmers. It is used within DMS only in those few XD routines which do disk-based I/O.

3.5 Utility Programs

A set of image display utility programs are provided along with the DMS. These programs serve two purposes. They do many simple image display operations, making it unnecessary for each installation to code them. They also serve as a model of the application interface to DMS for the discipline specific programs that are developed by each site. The following lists many of the DMS-supplied programs.

- ALLOC - allocate an IAT to a user session
- DEALLOC - free a previously allocated IAT
- IATINIT - set the IAT and DMS tables to a known initial state
- IATSTAT - list capabilities, status of system IATs
- TOTV - create an image on an IAT from disk image file(s)
- FROMTV - create disk image file(s) from an IAT image
- IMGLST - list current images with their attributes
- IMGUTIL - update image list
- VIEW - display an image on a monitor screen
- LOOP - show a sequence of images
- ALIGN - interactively register images by shifting
- FADE - interactively fade between two images
- SPLIT - display portions of images simultaneously
- ZOOM - enlarge an image area by pixel replication
- PAINT - dynamically make assignments to grey levels
- STRETCH - make LUT assignments for contrast alteration
- HIST - compute the intensity histogram of an image
- PROFILE - list the intensity values along a line
- LOADLUT - load lookup tables to the IAT
- SAVELUT - save lookup tables to disk

3.6 Implementation

A prototype DMS has been implemented on a VAX 11/780 under VMS. The primary implementation language for table manipulation and image I/O (DM and XL packages) is C, while FORTRAN 77 is used for other subroutine packages. All routines for application programmers are FORTRAN-callable.

Device independence is attained through layering the software into link and run time libraries. Application programs link to the XD package. The selection of the particular DD package to be used (i.e., the mapping to a particular IAT) is done at run time, based on a user's allocated IAT. Under VMS, DMS uses sharable libraries to provide the run time library linking.

The DMS control tables are stored in a global section.

Detailed DMS documentation (Functional Specification, Applications Programmer's Guide, System Programmer's Guide) is available from the authors.

4.0 SUMMARY

Advancing technology and increasing maintenance costs on older equipment will continue to make system upgrades necessary. In particular, more powerful and less expensive image analysis terminals are being marketed by a variety of vendors. At the same time, research laboratories like GSFC have invested tens of man years in program development for applications that are not commercially available. A great deal of effort has been expended in making the user interface to these programs a "tool that fits the hand" of the scientist. To preserve the unique algorithms and familiar user interfaces, difficult and expensive (and boring) conversions are required to add new IATs to an existing facility.

The DMS is an attempt at making these changes easier than in the past. The requirement to make the software more portable among disparate image devices is basic to the design of the DMS. In particular, the DMS distances both the end user and the applications programmer from the IAT hardware. We are persuaded that the systems we build based on the DMS will be more portable, easily maintained, and usable than without it.

The DMS prototype was completed in the Spring of 1984. Its functional capabilities are those discussed in this paper. Some areas of refinement for development of a mature DMS are being explored. To date, these include expanding control for the display of data (e.g., managing viewports); generalizing the concept of image type, for example, to support n-band images associated with arbitrary transformation tables; supporting the naming of groups of images, for example, for loop sequences, mosaics, or images combined through boolean operations; supporting new functions such as creation of perspective images, image rotation, and Fourier transforms; and formalizing the relationship between DMS and graphics packages.

5.0 ACKNOWLEDGEMENTS

DMS is being developed by the Image and Information Analysis Center of the NASA/Goddard Space Flight Center. The work is sponsored by the NASA Information Systems Office, which is part of the Office of Space Science and Applications.

The first use of DMS is in the upgrade to the Atmospheric and Oceanographic Information Processing System being developed for the Goddard Laboratory for Atmospheric Sciences. The EROS Data Center of the U.S. Geological Survey and the upgraded GSFC Land Analysis System (LAS) will also be DMS users.

The authors would like to acknowledge the significant contributions of Shernaz Contractor, William Kilmartin, and Arnold Voketaitis, all of Goddard, and of Stephen Ryan, of the EROS Data Center, to both the DMS software development and refinement of DMS concepts. We would also like to express our appreciation to Michael Girard of JPL for his collaborative work on the XL package design, and to Dr. Milton Halem and Lawrence Novak, both of GSFC, for their helpful review of this paper.

6.0 FOOTNOTES

1. The Transportable Applications Executive was developed by the NASA/Goddard Space Flight Center's Image and Information Analysis Center to provide a standard and portable interface for users of scientific research and analysis systems (reference 6). It is being used within the GSFC on several systems, and at several other research facilities within and outside the USA.

2. An image is defined here to be a set of intensities at grid points on an MxN grid. More formally, an image I is the set of i=f(x,y) where x,y are integers such that x lies between 1 and m and y lies between 1 and n. The i may be single points or vectors. When i = (i1,i2,...,ik), I is said to be a K-band image.

3. The convention for naming subroutines within the different components of the DMS is to use the two-letter package mnemonic followed by up to four descriptive letters, e.g., DMDEFI (define image), XDZOOM (zoom an image).

4. Early design work on the XL package was done with the Jet Propulsion Laboratory under a collaborative software

development agreement between the JPL Multimission Image Processing Laboratory and the TAE project.

7.0 REFERENCES

1. Acquah, J., J. Foley, J. Sibert and P. Wenner, "A Conceptual Model of Raster Graphics Systems", ACM Computer Graphics, Volume 16 No. 3, pp. 321 - 328

2. Bono, Peter R., Jose L. Encarnacao, F. Robert A. Hopgood, and Paul J. W. ten Hagen, "GKS - The First Graphics Standard," IEEE Computer Graphics, Volume 2, No. 5, pp. 9 - 23

3. Bracken, P.A., J.T. Dalton, J.B. Billingsley, J.J. Quann, Atmospheric and Oceanographic Information Processing System (AOIPS) System Description", NASA X-933-77-148, March, 1977

4. Dalton, J.T., R.K. Jamros, D.P. Helfer, D.R. Howell, "The Visible and Infrared Spin Scanning Radiometer (VISSR) Atmospheric Sounder (VAS) Ground Data Systems", Society of Photo-Optical Instrumentation Engineers Technical Symposium, April, 1981

5. Hamlet, R.G. and R.M. Haralick, "Transportable Package Software", Software - Practice and Experience, Vol. 10, 1980, pp. 1009 - 1027

6. Helfer, Dorothy P., David R. Howell, Jan Owings, Martha R. Szczur and John T. Dalton, "A Transportable Executive for Interactive Applications", Proceedings of the Harvard Computer Graphics Conference, July, 1981

7. International Imaging Systems, "System 575 Software, Digital Image Processing System Version 3.0, User's Manual", June, 1983

8. International Standards Organization, "Graphical Kernel System (GKS) - Functional Description," Draft International Standard ISO/DIS 7942, November, 1982

9. Krusemark, Scott, and R.M. Haralick, "An Operating System Interface for Transportable Image Processing Software," from Computer Vision, Graphics and Image Processing Vol. 23, 1983, pp. 42-66

10. Meyer, Bertrand, "Principles of Package Design", CACM, Volume 25, No. 7, pp. 419 - 428

11. Warnock, John and Douglas K. Wyatt, "A Device Independent Graphics Imaging Model for Use with Raster Devices," ACM Computer Graphics, Volume 16 No. 3, July, 1982, pp. 313 - 319

BIBLIOGRAPHIC DATA SHEET

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle A Device - Independent Interface for Interactive Image Display		5. Report Date May, 1984
		6. Performing Organization Code
7. Author(s) Dorothy C. Perkins, Martha R. Szczur, Jan Owings, Rita K. Jamros		8. Performing Organization Report No.
9. Performing Organization Name and Address NASA/Goddard Space Flight Center Image & Information Analysis Center/630.3 Greenbelt, Maryland 20771		10. Work Unit No.
		11. Contract or Grant No.
		13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Goddard Space Flight Center Greenbelt, Maryland 20771		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

NASA's Goddard Space Flight Center (GSFC) has developed a Transportable Applications Executive (TAE) for use in implementing portable applications software that can be shared by different research projects. Since many of the supported disciplines require the interactive display and manipulation of remotely sensed images, a device independent Display Management Subsystem (DMS) is being written as a TAE extension. The DMS attempts to abstract and standardize the device dependent functions that are used in the display and manipulation of image data on image analysis terminals. This paper explores the structure of DMS and the interface routines that are available to the applications programmer for use in developing a set of portable image display utility programs.

17. Key Words (Selected by Author(s)) Raster devices, Image display Device-Independence, Transportable Applications Executive, Portability		18. Distribution Statement	
19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*

*For sale by the National Technical Information Service, Springfield, Virginia 22151.

GSFC 25-44 (10/77)