Fifth Annual Users' Conference

Proceedings of a Conference held at
Goddard Space Flight Center
Greenbelt, Maryland
June 4-6, 1985
The Transportable Applications Executive (TAE) was conceived in 1979. It was proposed to be a general purpose software executive that could be applied in various systems.

The success of this concept and of TAE was demonstrated at the Fifth Annual TAE Users' Conference. More than 120 people attended, representing six universities, five government agencies, 10 private industries, Goddard Space Flight Center, and NASA Headquarters. The attendees included both TAE users and potential users. Presentation topics ranged from "TAE and the Space Station," "Unidata -- The Next Generation Meteorological Data System in Universities," and "System 9000: A TAE-based Interactive Digital Image Processing Workstation," to "Porting TAE to the Apollo Micro Computer".

We look forward to the Sixth Annual TAE Users' Conference and meeting again with the TAE user community.

TAE Project
NASA/Goddard
TABLE OF CONTENTS

TAE Current Status
Martha Szczur
NASA/GSFC

TAE Development

TAE Version 1.3
Lora Albanese,
Century Computing, Inc.

An Introduction to the Display Management System
Shernaz Contractor
NASA/GSFC

Remote Communications Job Manager
Dahritri Misra and Phil Miller
Century Computing, Inc.

CATMAN: The Catalog Manager
Scott Ross
Science Applications Research, Inc.

TAE Window Management
Dorothy Perkins
NASA/GSFC

TAE Applications

Storage, Retrieval, Processing and Display of Teal Ruby Imagery
Daniel Rice and Thomas Parris
Environmental Research Institute of Michigan

TAE and BISHOP in a Teaching Environment
Lesley Grove
Imperial College of Science and Technology

Workstation Activities of the Pilot Land Data System
William Likens
NASA/Ames Research Center

TAE and the Space Station
Karen Moe and Dorothy Perkins
NASA/GSFC

Unidata - the Next Generation Meteorological Data System in Universities
George Huffman
University of Maryland
Dynamically Constructing a TAE System of Menus and Procs

Michael Gough and Tamara Weaver
Science Applications Research, Inc.

TAE as the Foundation of a Transportable Executive for Helicopter Analysis

Richard Gemoets
Computer Sciences Corp./NASA/Ames Research Center

Use of TAE to Facilitate Color Film Generation

Ken Gacke
EROS Data Center

HELPME: Providing Expert Help from TAE

Barbara Lowrey and Phil Pease
NASA/GSFC

New Enhancements to the LAS Image Processing System

Jon Robinson
Science Applications Research, Inc.

The Communications Network for NASA's Pilot Land Data System

Harry Jones
NASA/Ames Research Center

Interactive Format Conversion System

Steven Kempler
NASA/GSFC

TAE and Interactive Research Imaging System - IRIS

Neil Allen and Grant Burton
Cooperative Institute for Research in the Atmosphere, Colorado State University

UNIX Emphasis

Transporting TAE to UNIX Environment

Philip Miller
Century Computing, Inc.

Implementation of TAE on an Apollo Network

James Cooper
University of Maryland

Conversion of the Land Analysis System to UNIX

Tony Butzer
EROS Data Center

System 9000: A TAE-based Interactive Digital Image Processing Workstation

Stephen Borders and Michael Guberek
Global Imaging, Inc.

A Transportable Display Management Subsystem

Kenneth Johnson
EROS Data Center
Workshop Summaries

The Land Analysis System
Future Directions

Lyn Oleson
NASA/GSFC

Martha Szczur
NASA/GSFC

Lists of conference Attendees

Alphabetical
By Location/Affiliation
TAE CURRENT STATUS
TAE CURRENT STATUS

MARTHA SZCZUR
IMAGE ANALYSIS FACILITY

NASA/GODDARD SPACE FLIGHT CENTER

JUNE 4, 1985
<table>
<thead>
<tr>
<th>SYSTEMS SOFTWARE ACCOMPLISHMENTS/PROGRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>• RELEASING A NEW ENHANCED VERSION OF TAE (1.3)</td>
</tr>
<tr>
<td>• TAE PORTED TO UNIX OPERATING SYSTEM</td>
</tr>
<tr>
<td>- INITIAL CONVERSION TO VAX/UNIX: 6 WEEKS</td>
</tr>
<tr>
<td>- SUN WORKSTATION: 1 WEEK</td>
</tr>
<tr>
<td>- SEL COMPUTER: 2 WEEKS</td>
</tr>
<tr>
<td>- APOLLO WORKSTATION WITH HYBRID UNIX: 1 WEEK</td>
</tr>
<tr>
<td>• DEVELOPED PROTOTYPE REMOTE COMMUNICATIONS JOB MANAGER (RCJM)</td>
</tr>
<tr>
<td>• COMPLETED DISPLAY MANAGEMENT SYSTEM (DMS) PROTOTYPE</td>
</tr>
<tr>
<td>• ENHANCED AND UPGRADED CATALOG MANAGER</td>
</tr>
<tr>
<td>• CONTINUED COOPERATIVE DEVELOPMENT WITH EROS DATA CENTER (CATALOG MANAGER, DMS UPGRADE TO UNIX, RCJM/EGNET NETWORKING)</td>
</tr>
<tr>
<td>• TAE DEVELOPMENT TEAM RECEIVED NASA GROUP ACHIEVEMENT AWARD 1984</td>
</tr>
</tbody>
</table>
**TAE USER SUPPORT ACCOMPLISHMENTS/PROGRESS**

- CONTINUED EXCELLENT SUPPORT TO TAE SCIENTIST USERS AND PROGRAMMERS
- PRODUCED TAE BROCHURE
- PUBLISHED TRI-ANNUAL NEWSLETTER: DISTRIBUTION INCREASED TO 500
- MOVED CLOSER TO MOU WITH IMPERIAL COLLEGE
- INCREASED TAE USER SITES TO 28 LOCATIONS
- 5TH TAE USERS CONFERENCE, JUNE 1985
- DEMONSTRATED TAE TO UNIDATA, WHO SELECTED IT AS THE SHELL SYSTEM FOR UCAR COMMUNITY
- TAE CONTAINED AS WORK STATION EXECUTIVE FOR GLOBAL IMAGING
- TUTORIALS GIVEN TO NEW USERS AND PROGRAMMERS
<table>
<thead>
<tr>
<th>TYPES OF APPLICATIONS USING TAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>● METEOROLOGY</td>
</tr>
<tr>
<td>● IMAGE PROCESSING</td>
</tr>
<tr>
<td>● DATA BASE MANAGEMENT</td>
</tr>
<tr>
<td>● ASTRONOMY</td>
</tr>
<tr>
<td>● DATA CAPTURE SYSTEM</td>
</tr>
<tr>
<td>● OCEANOGRAPHY</td>
</tr>
<tr>
<td>● DEFENSE SYSTEMS</td>
</tr>
<tr>
<td>● PROTOTYPING</td>
</tr>
<tr>
<td>● CLASSROOM TOOL</td>
</tr>
</tbody>
</table>
FUNDING SOURCES (FY 1985)

- EDC (18.0%)
- CODE (37.0%)
- LAS (7.0%)
- GSFC 520 (37.0%)
- OTHER (1.0%)
FUTURE DEVELOPMENT

- CONTINUE TSO SUPPORT
  - VIDEO TUTORIAL
  - ON-LINE EASE-OF-USE QUESTIONNAIRE/ANALYSIS
  - APPLICATION SOFTWARE CLEARINGHOUSE

- PORT TAE ONTO OTHER COMPUTERS
  - CONVEX
  - IBM PC/AT
  - IBM MAINFRAME

- DEVELOP COMMON FILE I/O (COOP EFFORT WITH EDC)

- EXTEND DMS
  - ENHANCEMENTS
  - INCORPORATE EDC's CONVERSION/ENHANCEMENTS TO UNIX
  - PORT TO OTHER DEVICES

- ENHANCE RCJM TO INCLUDE TCP/IP INTERFACE

- ENHANCE EGNET NETWORK (OSI PROTOCOL)

- DEVELOP GRAPHIC INTERFACE TO TAE

TAE REPRESENTS GSFC'S BEST CHOICE FOR RESPONDING TO USER REQUIREMENTS FOR SOFTWARE/SYSTEM COMPATIBILITY AND TRANSPORTABILITY IN THE SPACE STATION ERA.

NASA/GSFC
5/16/85
TAE VERSION 1.3

FEATURES

- PARAMETER QUALIFIERS
- INTERNAL PROCS
- COMPILED PDFs
- NEW GLOBALS
- MISCELLANEOUS
PARAMETER QUALIFIERS

DEFINITION: QUALIFIERS TO A PROC PARAMETER OR SUB-PARAMETER TO A PARAMETER.

TOPICS DISCUSSED:

- DECLARATION OF PARAMETER QUALIFIERS
- PROC INVOCATION WITH QUALIFIERS
- REFERENCING QUALIFIERS
- TUTOR ON PARAMETERS WITH QUALIFIERS
- RESTRICTIONS ON PARAMETER QUALIFIERS
DECLARATION OF PARAMETER QUALIFIERS

0 USE THE PARM STATEMENT TO DECLARE EACH QUALIFIER

0 HAVE ALL QUALIFIERS TO A PARAMETER DECLARED WITHIN A PROC

0 LINK TOGETHER A PARAMETER AND ITS QUALIFIERS WITH THE QUALS FIELD OF
THE PARM STATEMENT

EXAMPLE:

```
PROCESS
    PARM FILE1 TYPE=STRING QUALS=FILEQ
    PARM FILE2 TYPE=STRING QUALS=FILEQ
    END-PROC

PARMSET
    PARM BANDS INTEGER COUNT=0:7 DEFAULT=--
    PARM FORMAT STRING COUNT=0:1 VALID=(BIL,BSQ) DEFAULT=--
    END-PROC
```
PROC INVOCATION WITH PARAMETER QUALIFIERS

SPECIFY QUALIFIER VALUES IMMEDIATELY AFTER THE PARAMETER VALUE, ENCLOSING IT WITH VERTICAL BARS

EXAMPLE:

```
TAE> IMGPROC FILE1=WASHDC|BANDS=(3,4),FORMAT=BIL|+
TAE> FILE2=NYNY|BANDS=4|
```
TAE VERSION 1.3

REFERENCING PARAMETER QUALIFIERS FROM A PROCEDURE

SYNTAX: PARMNAME.QUALNAME

FILE1.BANDS
FILE2.FORMAT

EXAMPLE:

PROCEDURE
PARM FILE1 TYPE=STRING QUALS=FILEQ
PARM FILE2 TYPE=STRING QUALS=FILEQ
BODY
DISPLAY FILE1.BANDS
IF ($COUNT(FILE2.FORMAT)=0) !I.E., NULL
    LET FILE2.FORMAT = FILE1.FORMAT
END-IF
DISPLAY FILE2.FORMAT
END-PROC

PARMSET
PARM BANDS  INTEGER COUNT=0:7 DEFAULT=--
PARM FORMAT STRING  COUNT=0:1 VALID=(BIL,BSQ) DEFAULT=--
END-PROC
TUTOR ON PARAMETERS WITH QUALIFIERS

0 START WITH TUTOR ON PROC
0 ENTER "QUALIFIER TUTOR" WITH QUALIFY COMMAND
0 TUTOR ON PARAMETER QUALIFIERS
0 EXIT QUALIFIER TUTOR WITH ACCEPT COMMAND

EXAMPLE:

TAE> TUTOR IMGPROC
Demonstration of Parameter Qualifiers

<table>
<thead>
<tr>
<th>parm</th>
<th>description</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE1</td>
<td>File name of 1st image</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[THIS PARAMETER HAS QUALIFIERS]</td>
<td></td>
</tr>
<tr>
<td>FILE2</td>
<td>File name of 2nd image</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[THIS PARAMETER HAS QUALIFIERS]</td>
<td></td>
</tr>
</tbody>
</table>

Enter: parm=value, HELP, PAGE, QUALIFY, SHOW, RUN, EXIT, SAVE, RESTORE; RETURN to page.
? QUALIFY FILE2
TAE VERSION 1.3

Tutor/Qualifiers: parameter "FILE2", proc "IMGPROC"

<table>
<thead>
<tr>
<th>qual</th>
<th>description</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANDS</td>
<td>Selected band numbers</td>
<td>-- (null value)</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Format of the file: BIL or BSQ</td>
<td>-- (null value)</td>
</tr>
</tbody>
</table>

Pg 1.

Enter: qual=value, HELP, PAGE, SHOW, ACCEPT; RETURN to page.
? BANDS=(1,3,5)
Tutor/Qualifiers: parameter "FILE2", proc "IMGPROC"

<table>
<thead>
<tr>
<th>qual</th>
<th>description</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANDS</td>
<td>Selected band numbers</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Format of the file: BIL or BSQ</td>
<td>(null value)</td>
</tr>
</tbody>
</table>

Enter: qual=value, HELP, PAGE, SHOW, ACCEPT; RETURN to page.

? ACCEPT
TAE VERSION 1.3

RESTRICTIONS ON PARAMETER QUALIFIERS

0 NO NESTING OF QUALIFIERS BECAUSE OF SYNTAX PROBLEM

TAE> PROC PARM1 2,3 4161 !AMBIGUOUS IF NESTING ALLOWED

0 CANNOT ADD A PARAMETER QUALIFIER TO A V-BLOCK CREATED BY A VERSION OF TAE PREVIOUS TO 1.3

0 IF THE PARAMETER IN A PROC INVOCATION HAS QUALIFIER VALUES SPECIFIED, THEN THE VALUE FOR THE PARAMETER MUST ALSO BE SPECIFIED

TAE> PROC |HELLO|1 !INTERPRETED AS COMMAND QUALIFIER
INTERNAL PROCS

DEFINITION: A PROC DEFINED ENTIRELY WITHIN THE BODY OR THE DECLARATION SECTION OF ANOTHER PROC

RULES FOR INTERNAL PROCS:

0 IDENTIFIED BY PROCEDURE, PROCESS OR PARMSET STATEMENT USING THE NAME FIELD

0 MAY BE NESTED

0 DECLARED BEFORE INVOKED

0 CANNOT BE INVOKED WITH RUNTYPE=ASYNC OR RUNTYPE=BATCH

0 MAY ACCESS VARIABLES DECLARED WITHIN THE INTERNAL PROC OR IN THE OUTER PROCS

0 CANNOT REQUEST FOR DYNAMIC PARAMETERS USING XQDYNP (MAY USE GETPAR)

0 HELP INFORMATION FOR THE INTERNAL PROC MUST BE PLACED WITH THE MAIN PROC HELP
TAE VERSION 1.3

INTERNAL PROCS (CONTINUED)

EXAMPLE:

```
PROCEDURE
PARAM P1
PARAM P2
PROCEDURE NAME=LOCALP
PARAM LP1
BODY
DISPLAY (LP1, P2)
END-PROC

BODY
DISPLAY P1
LOCALP
!INVOKE PROC "LOCAL"
END-PROC
```
## TAE Version 1.3

### Compiled PDFs

**Definition:** A processed PDF with its declaration statements syntax scanned and symbol table built.

A compiled PDF will execute faster. Designed for PDFs with many parameters or executed often.

**Command Syntax:**
```
compile inproc=input-pdf outproc=compiled-pdf
```

**Format:**
```
<procedure-compiled
<compiled parameters in binary format
body
<contents of body copied
end-proc
<help text
```
TAE VERSION 1.3

COMPiled PDFs (CONTINUED)

RULES FOR ACCESS:

0 MAY TUTOR OR EXECUTE SAME AS SOURCE PDF

0 THE FILE TYPE NEEDS TO BE SPECIFIED EXPLICITLY

OR

USE SETLIB COMMAND TO SPECIFY THE DEFAULT FILE TYPE FOR SPECIFIC LIBRARY

EXAMPLE: TAE> SETLIB (METLIB-CPD, IMAGELIB-PDF)
TAE VERSION 1.3

COMPiled PDFs (continued)

RESTRICTIONS FOR PRE-BODY:

0  NO INTERNAL PROCS
0  NO PARAMETERS WITH QUALIFIERS
0  NO OPTIONS AND NAME FIELDS TO PROCESS OR PROCEDURE STATEMENTS
0  NO DE-REFERENCING ("a") OF GLOBAL VARIABLES EXCEPT IN DEFAULT FIELD
   OF PARM STATEMENT
0  ALL VARIABLES DECLARED OUTSIDE SUBCOMMAND BRACKETS MUST BE EITHER
   BEFORE OR AFTER ALL SUBCOMMAND BRACKETS
0  FILE PARAMETERS WITH DEFAULTS AND ACCESS=IN OR INOUT ARE CHECKED AT
   COMPILe TIME
0  SUBSTITUTION ("&") PERFORMED AT COMPILe TIME
0  COMMANDS DEFINED BY DEFCMD ARE RESOLVED AT COMPILe TIME
NEW GLOBALS

0 $DYNTUT FOR DYNAMIC TUTOR MODE

0 $ECHO, $AECHO, $BECHO
   - MULTI-VALUED
   - POSSIBLE VALUES ARE:
      "FULL", "BODY" OR "YES", "TRACE", "NO"
TAE VERSION 1.3

MISCELLANEOUS FEATURES

FOR ASYNCHRONOUS JOB:
0 COMMAND QUALIFIER ASYNCEND (VALID=("SILENT","NOTIFY"))
0 HOLD COMMAND FOR DYNAMIC TUTOR
0 EMIT COMMAND FOR PROCEDURES

GENERAL:
0 CAN RECALL UP TO 20 PREVIOUS COMMAND LINES
0 COMMAND LINE SIZE LIMIT DOUBLED TO 1024 CHARACTERS
0 PF2 KEY FOR VIEWING VALID LIST FOR STRING PARAMETERS
0 ALL INTRINSIC COMMANDS WITH SUBCOMMANDS NOW HAVE A DEFAULT SUBCOMMAND DEFINED
MISCELLANEOUS FEATURES (CONTINUED)

GENERAL (CONTINUED):

- IMAGE I/O PACKAGE AVAILABLE TO C PROGRAMS
- SECOND COMPONENT TO _ONFAIL FOR ABORT FROM PROC INTERRUPT MODE
- PUTMSG HAS SUBCOMMAND TRACE AND NOTRACE (DEFAULT)
- HELP TEXT FOR VARIABLES MAY BE ASSOCIATED WITH A PARTICULAR SUBCOMMAND

EXAMPLE: 
. VAR X -SUB1
  \HELP TEXT FOR X IN SUB1
. VAR X -SUB2
  \HELP TEXT FOR X IN SUB2

- PROC HIERARCHY SEARCH STARTS WITH INTERNAL PROCS
DISPLAY MANAGEMENT
SUBSYSTEM

A DEVICE-INDEPENDENT INTERFACE FOR INTERACTIVE IMAGE DISPLAY

IMAGE ANALYSIS FACILITY

NASA/GODDARD SPACE FLIGHT CENTER

JUNE, 1985
<table>
<thead>
<tr>
<th>DISPLAY MANAGEMENT SUBSYSTEM</th>
<th>AGENDA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• BACKGROUND</td>
</tr>
<tr>
<td></td>
<td>• USER INTERFACE</td>
</tr>
<tr>
<td></td>
<td>• PROGRAMMER INTERFACE</td>
</tr>
<tr>
<td></td>
<td>• SYSTEM MANAGER INTERFACE</td>
</tr>
<tr>
<td></td>
<td>• CURRENT DMS STATUS</td>
</tr>
</tbody>
</table>
DMS—DISPLAY MANAGEMENT SYSTEM

IMAGE ANALYSIS TERMINAL—
A SCIENTIST’S TOOL

IMAGE PRE-PROCESS

IMAGE DATA

DISPLAY I/O

APPLICATION PACKAGE

USER I/O

IMAGE I/O

NASA
DISPLAY MANAGEMENT SUBSYSTEM

DMS — PROBLEMS AND SOLUTIONS

PROBLEMS WITH IMAGE DISPLAY AND ANALYSIS:

- APPLICATION PROGRAMMERS MUST LEARN THE ARCHITECTURE AND LANGUAGE OF EACH IAT.
- USERS MUST LEARN UNIQUE CHARACTERISTICS OF THE IAT.

SOLUTIONS OFFERED BY DMS:

- APPLICATION PROGRAMS ARE WRITTEN INDEPENDENT OF THE SPECIFIC IAT LANGUAGE.
- USERS REFERENCE IMAGES BY NAMES; NOT BY HARDWARE COMPONENTS. DMS KNOWS THE CAPABILITIES OF EACH DEVICE.
<table>
<thead>
<tr>
<th>tae</th>
<th>DMS—DISPLAY MANAGEMENT SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>APPLICATION UTILITIES — SAMPLE LIST</td>
</tr>
<tr>
<td></td>
<td><strong>ALLOC/DEALLOC</strong>—ALLOCATE/FREE AN IAT</td>
</tr>
<tr>
<td></td>
<td><strong>TOTV</strong>—COPY DISK FILE TO REFRESH MEMORY(S)</td>
</tr>
<tr>
<td></td>
<td><strong>IMGLST</strong>—LIST CURRENT IMAGES</td>
</tr>
<tr>
<td></td>
<td><strong>VIEW</strong>—DISPLAY IMAGE ON MONITOR</td>
</tr>
<tr>
<td></td>
<td><strong>LOOP</strong>—SHOW SEQUENCE OF IMAGES</td>
</tr>
<tr>
<td></td>
<td><strong>ZOOM</strong>—ENLARGE IMAGE AREA</td>
</tr>
<tr>
<td></td>
<td><strong>HISTO</strong>—COMPUTE INTENSITY HISTOGRAM</td>
</tr>
</tbody>
</table>

*NASA*
<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>MODEL#</th>
<th>STATUS</th>
<th>USER NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHINOOK</td>
<td>IIS</td>
<td>75</td>
<td>FREE</td>
<td>HARRIS</td>
</tr>
<tr>
<td>SANTAANA</td>
<td>IIS</td>
<td>75</td>
<td>BUSY</td>
<td>BUSY</td>
</tr>
<tr>
<td>ZEPHYR</td>
<td>IIS</td>
<td>75</td>
<td>BUSY</td>
<td>SMITH</td>
</tr>
<tr>
<td>REF NO</td>
<td>IMAGE NAME</td>
<td>R</td>
<td>G</td>
<td>B</td>
</tr>
<tr>
<td>-------</td>
<td>------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>CHESBAY</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>W1</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>W2</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>W3</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

FOR DETAILED INFORMATION ON AN IMAGE: ENTER REF. NO.[REF # — R/G/B,...], 'HELP', OR 'EXIT'?
<table>
<thead>
<tr>
<th>DISPLAY MANAGEMENT SUBSYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPICAL DMS USER SESSION</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAE &gt; IATSTAT</td>
<td></td>
</tr>
<tr>
<td>TAE &gt; ALLOC CHINOOK ?</td>
<td></td>
</tr>
<tr>
<td>TAE &gt; IATINIT</td>
<td></td>
</tr>
<tr>
<td>TAE &gt; TOTV INFILE = CHESBAY 1, CHESBAY 2, CHESBAY 3</td>
<td>OUTIMAGE = CHESBAY TYPE = RGB</td>
</tr>
<tr>
<td>TAE &gt; TOTV INFILE = WASH 1, WASH 2, WASH 3</td>
<td>OUTIMAGE = W1, W2, W3</td>
</tr>
<tr>
<td>TAE &gt; IMGLST</td>
<td></td>
</tr>
<tr>
<td>TAE &gt; ZOOM INIMAGE = W3</td>
<td></td>
</tr>
<tr>
<td>TAE &gt; HISTO INIMAGE = W3 BANDS = ALL PLANE = 2</td>
<td></td>
</tr>
<tr>
<td>TAE &gt; BPCLEAR PLANE = 2</td>
<td></td>
</tr>
<tr>
<td>TAE &gt; SAVEVIEW OUTIMAGE = WASH ZOOM</td>
<td></td>
</tr>
<tr>
<td>TAE &gt; ILOOP-I INIMAGES = (W1, W2, W3)</td>
<td></td>
</tr>
<tr>
<td>TAE &gt; DEALLOC</td>
<td></td>
</tr>
</tbody>
</table>

NASA
DISPLAY MANAGEMENT SUBSYSTEM

SYSTEM MANAGER UTILITIES

DISPLAY DEVICE TABLE
- DEVADD—ADD IAT DEFINITION
- DEVDEL—DELETE IAT DEFINITION
- DEVEDT—EDIT DEVICE DESCRIPTION

DEVICE NAME TABLE
- NAMADD—ADD NAME TO TABLE
- NAMDEL—DELETE NAME FROM TABLE
- NAMEDT—EDIT NAME IN TABLE

NASA
DISPLAY MANAGEMENT SUBSYSTEM

CURRENT DMS STATUS

- PROTOTYPE VMS VERSION
  - COMPLETED IN APRIL 1985 AT NASA/GODEDARD
  - FULL DOCUMENTATION SET AVAILABLE
  - AOIPS II USING DMS SINCE JAN. 1985
  - CURRENTLY AT 3 USER BETA TEST SITES
    - NORTH CAROLINA STATE UNIVERSITY:
      ICONAS/ADAGE RDS 3000
    - USGS/FLAGSTAFF: DEANZA, RAMTEK, GRINELL
    - NASA/GODEDARD SPACE FLIGHT CENTER: IIS MODEL 75

- UNIX VERSION
  - WILL BE COMPLETED SUMMER 1985 AT USGS/
    EROS DATA CENTER
    - RASTERTEK, DEANZA

NASA
DMS—DISPLAY MANAGEMENT SYSTEM

FUTURE DIRECTIONS

- EXPAND CONTROL OF DISPLAY DATA (E.G., VIEWPORTS)
- SUPPORT IMAGE GROUPING
- SUPPORT NEW DISPLAY FUNCTIONS
- CONVERSION TO NEW DEVICES (I.E. ADAGE)

NASA
REMOTE COMMUNICATIONS JOB MANAGER

RCJM - PROTOTYPE

Dahritri Misra and Phil Miller
Century Computing, Inc.
REMOTE COMMUNICATIONS JOB MANAGER
(RCJM - PROTOTYPE)

- An extension of TAE
- Uses local area network (LAN) connecting two or more HETEROGENEOUS
  nodes
- Allows a TAE user at local node to:
  - Execute a remote proc
  - Request help/tutor on a remote proc
  - Copy text files and TAE 'save' files between local and remote node
<table>
<thead>
<tr>
<th>USER VIEW OF RCJM</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE PROC EXECUTION SIMILAR TO ASYNC JOBS</td>
</tr>
<tr>
<td>FOR REMOTE OPERATION, FIRST CONNECT TO REMOTE NODE</td>
</tr>
<tr>
<td>ENABLE-PATH (VAP, TDM)</td>
</tr>
<tr>
<td>SUBMIT PROC FOR REMOTE EXECUTION USING THE 'NODE' Qualifier</td>
</tr>
<tr>
<td>&gt;P1: NODE = (VAP, TDM); X = 3.5; Y = 1000.</td>
</tr>
<tr>
<td>USE REGULAR ASYNC COMMANDS TO CHECK STATUS OR ABORT A REMOTE JOB</td>
</tr>
<tr>
<td>&gt;SHOW-ASYNC P1</td>
</tr>
<tr>
<td>&gt;ABORT-ASYNC P1</td>
</tr>
<tr>
<td>FINALLY, DISCONNECT FROM REMOTE NODE</td>
</tr>
<tr>
<td>&gt;DISABLE-PATH (VAP, TDM)</td>
</tr>
</tbody>
</table>
RCJM DEVELOPMENT ISSUES

0 IMPLEMENTATION RELATED:
  - PORTABILITY (HETEROGENEOUS NODES)
  - PERFORMANCE
  - RELIABILITY

0 USER CONTEXT RELATED:
  - LOGGING ON TO A REMOTE NODE
  - ESTABLISHMENT OF USER CONTEXT AT REMOTE NODE
  - PERMANENT VS. TEMPORARY REMOTE CONTEXT
PORTABILITY

MAJOR ISSUES ON HETEROGENEOUS NODES:

0 COMPLEX REMOTE LOGIN ALGORITHM

0 HOST FILENAME FORMATS & LENGTHS

[TDM.RCJM.PROTO]MYPROC.PDF (VAX/VMS)
/TDM/RCJM/PROTO/MYPROC.PDF (UNIX)

0 DATA TYPE DIFFERENCES

BINARY --► ASCII CONVERSION REQUIRED
PERFORMANCE

- MAJOR EFFORT EXPENDED IN PERFORMANCE ENHANCEMENT

- PORTABILITY AFFECTS PERFORMANCE

- EMBEDDED LOGIC TO BYPASS DATA CONVERSION AMONG HOMOGENEOUS NODES

- VAX-11/750 (VMS 4.1) BENCHMARK RESULTS (LIGHTLY LOADED SYSTEM):
  - 25/30 SEC TO LOGIN TO REMOTE NODE
  - <15 SEC TO STARTUP REMOTE PROC EXECUTION
    (VS. 10 SEC FOR A LOCAL ASYNC PROC)

- STILL ROOM FOR FURTHER IMPROVEMENT
RELIABILITY

- DIRECTLY RELATED TO COMPLEXITY OF DESIGN

- NET RELIABILITY = RELIABILITY OF RCJM S/W
  AND
  RELIABILITY OF UNDERLYING NETWORK

- BOTH RCJM AND EGNET (NETWORK S/W BY EDC & GSFC) CURRENTLY IN TESTING PHASE

- AN INDEPENDENT VERSION WITH FULL RCJM CAPABILITY, USING DECNET, DEVELOPED AND TESTED ON VAX/VMS SYSTEMS
ESTABLISHING TAE ENVIRONMENT AT REMOTE NODE

TO RUN TAE PROCS AT A REMOTE NODE:

- USER IS LOGGED INTO THE REMOTE NODE UNDER A SPECIFIC REMOTE USER ACCOUNT (VIA ENABLE-PATH)

- REMOTE TM IS STARTED AND INITIALIZED

- AN APPROPRIATE USER CONTEXT (ENVIRONMENT) IS ESTABLISHED BY REMOTE TM

NOTE: REMOTE CONTEXT NOT PERMANENT (MUST BE RE-ESTABLISHED FOR EACH PROC ACTIVATION)
TAE USER CONTEXT

0 DEFINITION:

- VALUES OF TAE DEFINED GLOBALS

- VALUES OF USER DEFINED GLOBALS AND LOCALS AT INTERACTIVE LEVEL

- INCLUDES: LIBRARY SEARCH ORDER
  USER DEFINED COMMANDS

0 USER CONTEXT AT REMOTE NODE:

- VALUES OF CERTAIN TAE GLOBALS AND LOCALS AS ESTABLISHED AT USER'S LOCAL NODE

- VALUES OF ADDITIONAL GLOBALS ESTABLISHED BY RSLOGON/RULOGON PROCS AT REMOTE NODE
ESTABLISHING USER CONTEXT AT REMOTE NODE

- REMOTE CONTEXT IS NON-PERMANENT

- RE-ESTABLISHED BEFORE EACH REMOTE PROC EXECUTION, TO REFLECT THE USER CONTEXT AT LOCAL NODE

- REMOTE CONTEXT ESTABLISHED IN TWO DISCRETE STAGES
  - BEFORE PROCESSING THE JOB AT REMOTE, NODE THE TAE GLOBALS ARE RE-INITIALIZED, AND RSLOGON/RULOGON ARE EXECUTED
  - AFTER THE PROC IS LOCATED (AT REMOTE NODE) AND PROC PARAMETERS ARE CHECKED (AT LOCAL NODE) THE CURRENT CONTEXT IS SAVED BY LOCAL TM AND RESTORED BY REMOTE TM
OPEN ISSUES

- INHIBITING REMOTE PDF COPY AND/OR LOCAL CHECKING OF PARAMETERS
- ASYNCHRONOUS REMOTE LOGIN (ENABLE-PATH COMMAND)
- SYNCHRONOUS EXECUTION OF REMOTE PROCS
- ESTABLISHMENT OF PERMANENT REMOTE CONTEXT
- SPEED ENHANCEMENT
CONCLUSION

- Complex functionality, involving 4 chained processes distributed across heterogeneous nodes, and 7k LOC

- Provided valuable experience in dealing with LAN's and various network protocols

- Current Status:
  - Version using Decnet protocol presently available
  - RCJM with EGNET protocol in integration and test phase

- Future Enhancements:
  - Identification and removal of bottlenecks
  - Improvement based on field tests and user feedback
CATMAN: THE CATALOG MANAGER

USES AND CAPABILITIES

By:

SCOTT ROSS
SCIENCE APPLICATIONS RESEARCH INC.
CATALOG MANAGER USES AND CAPABILITIES

The Catalog Manager (CM) developed by Century Computing Inc. provides a system independent means of building and maintaining catalogs of disk files and related descriptive information. CM does not replace the host system file handling capabilities, but rather complements them by allowing users to store more meaningful information about their files. The catalog is ordered hierarchically, with each user assigned a unique subtree which comprises that user's entire list of data sets. This hierarchy structure facilitates the storage of information in two ways. First, the file naming system allows multilevel names to help describe the file contents and to allow users to organize their files in meaningful ways. Second, the catalog permits users to store additional descriptive information in the form of file attributes.

To describe the uses and capabilities of the catalog manager, three areas will be discussed: 1) Communications between the user and the host computer, 2) Data structures and features supported within the catalog, and 3) Commands used to access the catalog through the Transportable Applications Executive (TAE).
COMMUNICATIONS BETWEEN THE USER AND THE HOST COMPUTER

Figure one illustrates the flow of information between a user and the catalog manager running on the host computer. To access the catalog, the user executes a TAE proc. When the proc requires information from the catalog, it places a request in one of the CM request mailboxes, and transfers control to catalog manager. Catalog manager will then respond by performing the requested operation and, upon completion, returns control to the proc. Notice that the proc may also require the use of various media (disk and tape) to perform the necessary task.
Figure 1: Communications between the user and the host computer
II  Data structures and features within the catalog

There are numerous structures within the catalog to facilitate the storage of information. This information can be maintained by the user in two different ways. First, the file naming system within TAE allows multilevel names to help describe file contents. Second, the catalog contains other descriptive information in the form of file attributes. This section will cover the data structures used within the catalog such as roots, branches, and files. It will define the currently available file attributes, and explain the uses of alias names and wildcarding.

A. Catalog data structures

Data structures within the catalog fall into three categories: roots, branches, and files. Figure 2 illustrates the relationship between these entities. To simplify the discussion, it is helpful to think of the catalog as a tree structure with each user assigned a unique subtree within the catalog. Branches, then, are qualifiers set up by the user to divide data sets into separate groups. The files are then defined at the terminal (or leaf) level.
Figure 2: Catalog data structures
B. File Attributes

File attributes allow the user to store more meaningful information about data set in the catalog. By dividing his catalog into separate branches and then saving detailed information about the files, the user is able to readily identify the contents of the catalog. Figure 3 provides a brief description of each of the currently available file attributes.
<table>
<thead>
<tr>
<th>TAE name</th>
<th>description</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>date as YMMDD</td>
<td>integer</td>
</tr>
<tr>
<td>TIME</td>
<td>time as HHMMSS</td>
<td>integer</td>
</tr>
<tr>
<td>CLAT</td>
<td>center latitude</td>
<td>real</td>
</tr>
<tr>
<td>DLAT</td>
<td>delta latitude</td>
<td>real</td>
</tr>
<tr>
<td>CLON</td>
<td>center longitude</td>
<td>real</td>
</tr>
<tr>
<td>DLON</td>
<td>delta longitude</td>
<td>real</td>
</tr>
<tr>
<td>DT1</td>
<td>data type 1 (8 char max)</td>
<td>string</td>
</tr>
<tr>
<td>DT2</td>
<td>data type 2 (8 char max)</td>
<td>string</td>
</tr>
<tr>
<td>U1</td>
<td>user field 1 (8 char max)</td>
<td>string</td>
</tr>
<tr>
<td>U2</td>
<td>user field 2 (8 char max)</td>
<td>string</td>
</tr>
<tr>
<td>U3</td>
<td>user field 3 (8 char max)</td>
<td>string</td>
</tr>
<tr>
<td>U4</td>
<td>user field 4 (8 char max)</td>
<td>string</td>
</tr>
<tr>
<td>BEST</td>
<td>best version &quot;B&quot;</td>
<td>string</td>
</tr>
<tr>
<td>SOURCE</td>
<td>source number -128 to 127</td>
<td>integer</td>
</tr>
<tr>
<td>INSTR</td>
<td>instrument number</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td>-128 to 127</td>
<td></td>
</tr>
<tr>
<td>VOLSER</td>
<td>volume serial number</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>(12 char max)</td>
<td></td>
</tr>
<tr>
<td>FILE</td>
<td>file &quot;F&quot;=data file</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>&quot;C&quot;=directories</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3: Catalog Manager Attributes**
C. ALIAS NAMES AND WILDCARDING

Alias names and wildcards are available in the catalog to provide greater flexibility and ease of operation.

Alias names are alternate names established within the catalog for a file or a directory. The advantage of alias names is that they allow a user to give shorter names for files and still retain a complex directory structure. For example, if a user named JONES has a file in the catalog named NORTH.EAST.BOSTON.SCALED.BAND5, he can establish an alias name for his file as follows:

```
TAE> CMALIAS-Create NAME="NORTH.EAST.BOSTON.SCALED.BAND5" ALIAS="B5"
```

Once this command is issued, JONES only needs to type in "B5" to refer the file NORTH.EAST.BOSTON.SCALED.BAND5.

Wildcards are specified by entering "*" in place of a qualifier within a given file or directory name. By using wildcards the user is able to control the level at which he accesses information within the catalog. For example, if JONES wants to know which files in his catalog which end with the qualifier ".B5", he can do it in a number of ways. The most efficient method is as follows:

```
TAE> CMLIST-FILE NAME="*.B5"
```

The CMLIST program will then display all file names ending with ".B5".
Commands used to access the catalog through TAE

The following commands are currently available for creating, listing, changing, or deleting information stored within the catalog:

1. CMAILAS
2. CMCAT
3. CMCHGATR
4. CMDEL
5. CMLIST
6. CMRENAME
7. CMSEARCH
8. CMSET
9. CMUNCAT

The following pages show the functions and available options for each of the commands above, and a description of the necessary parameters and their default values, if any.
HELP: PROC "CMALIAS", LIBRARY "CM$USREXE"

CATALOG MANAGER ALIAS FUNCTIONS

THE CATALOG MANAGER ALIAS FUNCTIONS ARE:

CMALIAS-CREATE  ASSIGN AN ALIAS TO A CATALOGED DIRECTORY NAME;

CMALIAS-DELETE  DEASSIGN AN ALIAS FROM A DIRECTORY NAME;

CMALIAS-RENAME  CHANGE THE NAME OF AN EXISTING ALIAS;

CMALIAS-LIST   LIST ALL ALIASES, CM NAMES CORRESPONDING TO AN ALIAS, OR ALIAS NAMES CORRESPONDING TO A CM NAME.
HELP ; PROC "CMALIAS", LIBRARY "CM$USREXE"

THE TAE COMMAND STDOUT MAY BE USED TO DIRECT
A LISTING TO A FILE RATHER THAN TO THE CURRENT
STANDARD OUTPUT.

EXAMPLE: CMALIAS-LIST ISTDOUT=MY.TMP;

THE ABOVE EXAMPLE, ISSUED FROM THE COMMAND MODE,
CAUSES A LIST OF ALL THE ALIASES CATALOGED UNDER
YOUR ROOT TO BE PLACED IN A FILE CALLED MY.TMP.
HELP: Proc "CMALIAS", Library "CM$USREXE"

**PARAMETER SUMMARY:**

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIAS</td>
<td>Alias name</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td>NAME</td>
<td>Catalog Manager directory name</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td>USER</td>
<td>User name</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>OLDALIAS</td>
<td>Old alias to be renamed</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td>NEWALIAS</td>
<td>New alias name</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
</tbody>
</table>
HELP: PROC "CMCAT", LIBRARY "CM$USREXE"

CATALOG A FILE

THE CMCAT PROGRAM PLACES AN EXISTING HOST FILE
IN THE CATALOG. YOU MUST SPECIFY A VALID HOST FILE
NAME AND A CORRESPONDING CATALOG MANAGER FILE NAME.
YOU MAY ENTER ATTRIBUTES TO BE STORED WITH THE CATALOGED
FILE.

IF YOU SPECIFY ATTRIBUTES TO BE STORED WITH
THE FILE, PLEASE NOTE THAT THEY WILL BE ASSOCIATED
WITH THE FILE AND NOT WITH THE DIRECTORY. FOR EXAMPLE,
IF YOU CATALOG A FILE AS:

NAME=A DATE=14-NOV-1984

THE DATE ATTRIBUTE WILL BE ASSOCIATED WITH FILE A:1
AND NOT WITH THE DIRECTORY A.
HELP: PROC "CMCAT", LIBRARY "CM$USREXE"

STORED ATTRIBUTES MAY BE MODIFIED THROUGH THE
CHGATR PROGRAM. CHGATR WILL ALSO ALLOW YOU TO STORE
ATTRIBUTES FOR INDIVIDUAL DIRECTORIES.
HELP: Proc "CMCAT", Library "CM$USREXE"

PARAMETER SUMMARY:

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOSTFILE</td>
<td>Host file name (no wildcards)</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td>NAME</td>
<td>CM file name (no version)</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td>DATE</td>
<td>Date (example: 14-NOV-1984)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>TIME</td>
<td>Time (example: 06:05:22)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>ASSOC</td>
<td>Associated files (up to 30)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>CLAT</td>
<td>Center latitude</td>
<td>REAL</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>DLAT</td>
<td>Delta latitude</td>
<td>REAL</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>CLON</td>
<td>Center longitude</td>
<td>REAL</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>DLON</td>
<td>Delta longitude</td>
<td>REAL</td>
<td>DEFAULT=--</td>
</tr>
</tbody>
</table>
PARAMETER SUMMARY (CONTINUED):

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>STARTLIN</td>
<td>STARTING LINE NUMBER</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>LINES</td>
<td>NUMBER OF LINES</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>LINE INC</td>
<td>LINE INCREMENT</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>STARTSMP</td>
<td>STARTING SAMPLE NUMBER</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>SAMPLES</td>
<td>NUMBER OF SAMPLES PER LINE</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>SAMP INC</td>
<td>SAMPLE INCREMENT</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>BANDS</td>
<td>NUMBER OF BANDS</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>SPECORG</td>
<td>SPECTRAL ORGANIZATION (2 CHARS.)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
</tbody>
</table>
HELP: PROC "CMCAT", LIBRARY "CM$USREXE"

PARAMETER SUMMARY (CONTINUED):

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT</td>
<td>PROJECT NAME (1 TO 30 CHARs.)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>INSTR</td>
<td>INSTRUMENT NUMBER (-128 TO 12)</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>SOURCE</td>
<td>SOURCE NUMBER (-128 TO 12)</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>DATA1</td>
<td>DATA TYPE 1 (8 CHARs. MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>DATA2</td>
<td>DATA TYPE 2 (8 CHARs. MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER1</td>
<td>USER FIELD 1 (8 CHARs. MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER2</td>
<td>USER FIELD 2 (8 CHARs. MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER3</td>
<td>USER FIELD 3 (8 CHARs. MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER4</td>
<td>USER FIELD 4 (8 CHARs. MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>VERSION</td>
<td>BEST FOR BEST VERSION</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
</tbody>
</table>
HELP: PROC "CMCHGATR", LIBRARY "CM\$USREXE"

TO MODIFY DIRECTORY ATTRIBUTES USE:

NAME=#USER.QUAL1.QUAL2

TO MODIFY DATA FILE ATTRIBUTES USE:

NAME=#USER.QUAL1.QUAL2:VERSION

ONLY THOSE ATTRIBUTES SET TO A VALUE OTHER THAN THEIR DEFAULT VALUE WILL BE MODIFIED.

ASSOCIATED FILES ARE STORED FOR FILES ONLY.
AN ASSOCIATED FILE LIST MAY ONLY BE MODIFIED AFTER A DYNAMIC TUTOR SESSION IS STARTED.
HELP: PROC "CMCHGATR", LIBRARY "CM$USREXE"

CHANGE ATTRIBUTES OF A CATALOGED FILE

The CMCHGATR program will modify the attributes of a data file or directory which has been previously cataloged by the Catalog Manager.

Note that the Catalog Manager maintains attributes for files and directories. A file is specified with a version number, and a directory is specified without a version number.

If EDTASSO="YES", you will be allowed to enter dynamic tutor mode. Type TUTOR when you receive the prompt:

TAE-CMCHGATR>
### Parameter Summary:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLAT</td>
<td>Center latitude</td>
<td>REAL</td>
</tr>
<tr>
<td>DCON</td>
<td>Center longitude</td>
<td>REAL</td>
</tr>
<tr>
<td>DLON</td>
<td>Delta longitude</td>
<td>REAL</td>
</tr>
<tr>
<td>CM NAME</td>
<td>CM directory or file name (no wildcards allowed)</td>
<td>STRING</td>
</tr>
<tr>
<td>DATE</td>
<td>Date (example: 14-nov-1984)</td>
<td>STRING</td>
</tr>
<tr>
<td>TIME</td>
<td>Time (example: 06:05:30)</td>
<td>STRING</td>
</tr>
<tr>
<td>ASSOC</td>
<td>Associated file(s) (valid for files only)</td>
<td>STRING</td>
</tr>
</tbody>
</table>

**Help:** PROC "CMCHGTR", LIBRARY "CMUSRXE"
HELP: PROC "CMCHGATR", LIBRARY "CM$USREXE"

PARAMETER SUMMARY (CONTINUED):

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>STARTLIM</td>
<td>STARTING LINE NUMBER</td>
<td>INTEGER</td>
<td>DEFAULT=-999</td>
</tr>
<tr>
<td>LINES</td>
<td>NUMBER OF LINES</td>
<td>INTEGER</td>
<td>DEFAULT=-999</td>
</tr>
<tr>
<td>LINEINC</td>
<td>LINE INCREMENT</td>
<td>INTEGER</td>
<td>DEFAULT=-999</td>
</tr>
<tr>
<td>STARTSMP</td>
<td>STARTING SAMPLE NUMBER</td>
<td>INTEGER</td>
<td>DEFAULT=-999</td>
</tr>
<tr>
<td>SAMPLES</td>
<td>NUMBER OF SAMPLES PER LINE</td>
<td>INTEGER</td>
<td>DEFAULT=-999</td>
</tr>
<tr>
<td>SAMPINC</td>
<td>SAMPLE INCREMENT</td>
<td>INTEGER</td>
<td>DEFAULT=-999</td>
</tr>
<tr>
<td>BANDS</td>
<td>NUMBER OF BANDS</td>
<td>INTEGER</td>
<td>DEFAULT=-999</td>
</tr>
<tr>
<td>SPECORG</td>
<td>SPECTRAL ORGANIZATION (2 CHARs.,)</td>
<td>STRING</td>
<td>DEFAULT=-999</td>
</tr>
</tbody>
</table>
HELP: PROC "CMCHGATR", LIBRARY "CM$USREXE"

PARAMETER SUMMARY (CONTINUED):

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT</td>
<td>PROJECT NAME (1 TO 30 CHARS.)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>INSTR</td>
<td>INSTRUMENT NUMBER (-128 TO 128)</td>
<td>INTEGER</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>SOURCE</td>
<td>Source number -128 to 128</td>
<td>INTEGER</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>DATA1</td>
<td>DATA TYPE 1 (8 CHARS, MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>DATA2</td>
<td>DATA TYPE 2 (8 CHARS, MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>USER1</td>
<td>USER FIELD 1 (8 CHARS, MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>USER2</td>
<td>USER FIELD 2 (8 CHARS, MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>USER3</td>
<td>USER FIELD 3 (8 CHARS, MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>USER4</td>
<td>USER FIELD 4 (8 CHARS, MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>VERSION</td>
<td>BEST FOR BEST VERSION</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>EDIT</td>
<td>EDIT ATTRIBUTES</td>
<td>STRING</td>
<td>DEFAULT=EDIT</td>
</tr>
</tbody>
</table>
HELP: PROC "CMDEL", LIBRARY "CM$USREXE"

DELETE CATALOGED FILE OR HOST FILE

The CMDEL program deletes a single file or all
data files in a branch. The branch delete may be by
file name, by attribute value(s), or a combination of
name and attribute value(s). You may only delete files
for which you have VAX/VMS delete privilege.

A branch delete causes all the files in the branch
to be uncataloged but only those files for which you have
delete privilege are deleted from the host file system.

The DELETE flag allows you to specify DISK,
tape or all.
HELP: PROC "CMDELB", LIBRARY "CM$USRREX"  Pg 2+

THE VERIFY PARAMETER IS SET TO VERIFY BY DEFAULT.  THIS ALLOWS YOU TO VERIFY EACH FILE BEFORE IT IS DELETED.

IF YOU WISH TO DELETE A HOST FILE, YOU MAY SPECIFY NAME AS ANY LEGAL HOST FILE NAME (PRECEDED BY A ^) INSTEAD OF A CATALOG MANAGER FILE NAME. WILDCARDS ARE NOT ALLOWED FOR A HOST FILE DELETE. THE REMAINING PARAMETERS DO NOT APPLY TO HOST FILE NAMES.
HELP: PROC "CMDEL", LIBRARY "CM$USREXE"

PARAMETER SUMMARY:

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>CM OR HOST FILE NAME</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td>TAPEVOL</td>
<td>TAPE VOLUME (6 CHARS.)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>PROJECT</td>
<td>PROJECT NAME (1 TO 30 CHARS.)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>INSTR</td>
<td>INSTRUMENT NUMBER (-128 TO 127)</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>SOURCE</td>
<td>SOURCE NUMBER -128 TO 127</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER1</td>
<td>USER FIELD 1 (8 CHAR, MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER2</td>
<td>USER FIELD 2 (8 CHAR, MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER3</td>
<td>USER FIELD 3 (8 CHAR, MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER4</td>
<td>USER FIELD 4 (8 CHAR, MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
</tbody>
</table>
HELP: PROC "CMDEL", LIBRARY "CM$USREXE"

```
PARAMETER SUMMARY (CONTINUED):

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR_DATE</td>
<td>CREATION DATE</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>LA_DATE</td>
<td>LAST ACCESS DATE</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>VERSION</td>
<td>BEST FOR BEST VERSION</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>VERIFY</td>
<td>VERIFY OR NOVERIFY</td>
<td>STRING</td>
<td>DEFAULT=VERIFY</td>
</tr>
<tr>
<td>DELETE</td>
<td>DELETE DISK, TAPE, OR ALL FILES</td>
<td>STRING</td>
<td>DEFAULT=DISK</td>
</tr>
</tbody>
</table>
```
HELP: PROC "CMLIST", LIBRARY "CM$USREXE"

List Cataloged Information

CMLIST allows you to list cataloged information. The subcommands of CMLIST are:

ATTR List attributes of a specified file or directory.

DIR List files/directories one level below a specified directory name.

CONTEXT List reopen context of cataloged files.

FILE List cataloged files.

HOST List host file name of cataloged files.

TAPE List tapes and tape file information.
HELP:  SUBCOMMAND "ATTR", PROC "CMLIST"

The ATTR program lists the attributes of a file or directory at the specified level. For example, ATTR with NAME=* will list names of the form QUAL1, and ATTR with NAME=*, will list names of the form QUAL1,QUAL2 and QUAL1:VERSION.

- A CM file name contains a version number.
- A CM directory name does not contain a version number.
HELP: Subcommand "DIR", Proc "CMLIST"

The DIR program lists the files/directories one level below a specified directory name. Wildcard substitution may be used for any qualifier in the specified directory name.
HELP: Subcommand "CONTEXT", Proc "CMLIST"  Pg 1.

The CONTEXT program lists the reopen context of a cataloged file. The input CM name must be a fully qualified file name.

For a VAX system, the reopen context is the file ID.
HELP: Subcommand "FILE", Proc "CMLIST"

The FILE program lists a single cataloged file or all files within a specified directory branch. Wildcard substitution may be used for any qualifier in the specified directory name. Both disk and tape files will be listed, including any tape files marked for deletion.
HELP:  SUBCOMMAND "HOST", PROC "CMLIST"

THE HOST PROGRAM LISTS THE HOST FILE NAME OF CATALOGED FILES.
HELP:  SUBCOMMAND "TAPE", PROC "CMLIST"

The LISTTAPE program lists information about a user's tape(s). The FORMAT option controls the type of information listed:

FORMAT=BRIEF   Tape footage information will be listed for the tape label specified by the TAPEVOL parameter.

FORMAT=FULL    The names of the files on the tape will also be listed.

If a wildcard (*) is used for the tape name, information about all of the user's tapes will be listed.
HELP: PROC "CMLIST", LIBRARY "CM$USREXE"

PARAMETER SUMMARY:

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>CATALOG MANAGER FILE OR DIRECTORY NAME</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td>DIR</td>
<td>CATALOG MANAGER DIRECTORY NAME</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td>ATTRIB</td>
<td>LIST ATTRIBUTES OPTION</td>
<td>STRING</td>
<td>DEFAULT=NOLIST</td>
</tr>
<tr>
<td>TAPEVOL</td>
<td>TAPE VOLUME LABEL</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td>FORMAT</td>
<td>LIST OPTION</td>
<td>STRING</td>
<td>DEFAULT=BRIEF</td>
</tr>
<tr>
<td>MARKDEL</td>
<td>TAPE FILES MARKED FOR DELETION</td>
<td>STRING</td>
<td>DEFAULT=NOLIST</td>
</tr>
</tbody>
</table>
HELP: PROC "CMRENAMEx", LIBRARY "CM$USREXEC"

Rename a Cataloged File

The CMRENAMEx Program Renames a Cataloged Catalog Manager File.

PARAMETER SUMMARY:

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Existing CM file name</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td></td>
<td>Without wildcards</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEWNAME</td>
<td>New CM file name</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td></td>
<td>Without wildcards</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
HELP: PROC "CMSEARCH", LIBRARY "CM$USREXE"

Search the Cataloged Files

CMSEARCH allows you to search the catalog for files or directories with specific names and/or attributes. The subcommands of CMSEARCH are:

NAME    Search the catalog for the specified file or directory name. Only PARMs NAME, SEARCH and ATTRIB are used for subcommand NAME.

NAT     Search the catalog for files or directories matching the specified name and attributes.

ATTR    Search the catalog for files or directories matching the specified name and attributes. The search will proceed through the entire branch under the specified name.
HELP: PROC "CMSEARCH", LIBRARY "CM$USREXE"

PARAMETER SUMMARY:

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>CM FILE OR DIRECTORY NAME</td>
<td>STRING</td>
<td>NO DEFAULT</td>
</tr>
<tr>
<td>VERSION</td>
<td>BEST FOR BEST VERSION</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>ATTRIB</td>
<td>ATTRIBUTE LIST OPTION</td>
<td>INTEGER</td>
<td>DEFAULT=NOLIST</td>
</tr>
<tr>
<td>SEARCH</td>
<td>SEARCH DISK, TAPE, OR ALL FILES</td>
<td>STRING</td>
<td>DEFAULT=ALL</td>
</tr>
<tr>
<td>DATE</td>
<td>Date (example: 14-NOV-1984)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>TIME</td>
<td>Time (example: 06:05:22)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>CR_DATE</td>
<td>CREATION DATE</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>LA_DATE</td>
<td>LAST ACCESS DATE</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>CLAT</td>
<td>CENTER LATITUDE</td>
<td>REAL</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>CLON</td>
<td>CENTER LONGITUDE</td>
<td>REAL</td>
<td>DEFAULT=---</td>
</tr>
</tbody>
</table>
HELP: PROC "CMSEARCH", LIBRARY "CM$USREXE"  Pg 3.

PARAMETER SUMMARY (CONTINUED):

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISKVOL</td>
<td>VOLUME SERIAL NUMBER (12 CHARS.)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>TAPEVOL</td>
<td>TAPE VOLUME LABEL (6 CHARS.)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>PROJECT</td>
<td>PROJECT NAME (1 TO 30 CHARS.)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>INSTR</td>
<td>INSTRUMENT NUMBER (-128 TO 12)</td>
<td>INTEGER</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>SOURCE</td>
<td>SOURCE NUMBER (-128 TO 12)</td>
<td>INTEGER</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>DATA1</td>
<td>DATA TYPE 1 (8 CHARS. MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>DATA2</td>
<td>DATA TYPE 2 (8 CHARS. MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>USER1</td>
<td>USER FIELD 1 (8 CHARS. MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>USER2</td>
<td>USER FIELD 2 (8 CHARS. MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>USER3</td>
<td>USER FIELD 3 (8 CHARS. MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
<tr>
<td>USER4</td>
<td>USER FIELD 4 (8 CHARS. MAX)</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
</tbody>
</table>
HELP: Proc "CMSET", Library "CM$USREXE"

CHANGE DEFAULT FOR CURRENT CATALOG ROOT

THE CMSET PROGRAM ALLOWS YOU TO CHANGE YOUR DEFAULT ROOT TO ANY DIRECTORY IN THE CATALOG. IF YOU GIVE A NULL NAME (NAME=--) FOR THE ROOT NAME, YOUR ORIGINAL DEFAULT ROOT IS RE-ESTABLISHED.

PARAMETER SUMMARY:

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>CATALOG DIRECTORY NAME</td>
<td>STRING</td>
<td>DEFAULT=--</td>
</tr>
</tbody>
</table>
HELP: PROC "CMUNCAT", LIBRARY "CM$USREXE"

UNCATALOG A File FROM THE Catalog

The CMUNCAT Program uncatalogs a single file, a list of single files, or all data files in a branch. For a branch uncatalog, files are chosen by directory name, wildcarded directory name, attributes, or a combination. You may only uncatalog files which are cataloged under your default root.

If a copy of the file exists on tape, neither the disk nor the tape copy may be uncataloged by CMUNCAT. In this case, you must use CMDEL.
PARAMETER SUMMARY:

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT</td>
<td>PROJECT NAME (1 TO 30 CHARs.)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>INSTR</td>
<td>INSTRUMENT NUMBER (-128 TO 128)</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>SOURCE</td>
<td>SOURCE NUMBER (-128 TO 128)</td>
<td>INTEGER</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER1</td>
<td>USER FIELD 1 (8 CHARs. MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER2</td>
<td>USER FIELD 2 (8 CHARs. MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER3</td>
<td>USER FIELD 3 (8 CHARs. MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>USER4</td>
<td>USER FIELD 4 (8 CHARs. MAX)</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>VERIFY</td>
<td>VERIFY OR NOVERIFY</td>
<td>STRING</td>
<td>DEFAULT=VERIFY</td>
</tr>
<tr>
<td>CR_DATE</td>
<td>CREATION DATE</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
<tr>
<td>LA_DATE</td>
<td>LAST ACCESS DATE</td>
<td>STRING</td>
<td>DEFAULT=---</td>
</tr>
</tbody>
</table>
ABSTRACT

TAE Window Management

Dorothy C. Perkins
NASA/Goddard Space Flight Center
Greenbelt, Maryland

Goddard’s Data Systems Technology Division, supported by Century Computing, Inc. and the TAE Project Office, is working to define, design and implement a prototype multi-window interface to TAE. The driver behind this work is the desire to use TAE to prototype user interfaces for the next-generation mission operations control centers at Goddard. Because many of the information displays in control centers are multi-window and possibly even multi-screen, it appeared appropriate and highly desirable to augment the power of TAE through the addition of a window interface.

Some of the constraints imposed on this effort include:

- the interface should support both text and graphics; that is, it should facilitate the intermixing of text and graphics for the generation of optimal displays for disseminating information;
- the interface should support monochrome and color devices;
- TAE standards of portability and device independence should be maintained; and
- existing applications must be upward compatible with the new interface.

The primary objective of the effort is to provide a user interface capability which is in harmony with the needs of control centers; that is, one which supports multiple access to display windows by asynchronous processes, the locking of display windows, and control of windows from application programs.

There are two secondary objectives: to extend the potential use of TAE to broader applications (other applications which might benefit from the use of windows); and to augment TAE as a structure for the prototyping of future systems.
TAE WINDOW MANAGEMENT TASK

GOAL:

DEFINE, DESIGN & IMPLEMENT PROTOTYPE
MULTI-WINDOW INTERFACE TO TAE

FUNCTIONAL CONSTRAINTS:

° SUPPORT TEXT & GRAPHICS
° MONOCHROME & COLOR
° TAE STANDARDS OF PORTABILITY/DEVICE-INDEPENDENCE
  (E.G., ALPHANUMERIC AND IMAGING TERMINALS)
° STAGED IMPLEMENTATION FOR ONGOING EVALUATION/MODIFICATION
° UPWARD COMPATIBILITY OF EXISTING APPLICATIONS
TAE WINDOW MANAGER DEVELOPMENT BY:

° THE DATA SYSTEMS TECHNOLOGY DIVISION OF THE GSFC MISSION OPERATIONS AND DATA SYSTEMS DIRECTORATE

° TAE PROJECT OFFICE

° CENTURY COMPUTING, INC.
SOME CHARACTERISTICS OF CONTROL CENTERS

- REAL-TIME, ASYNCHRONOUS EVENTS - SIMULTANEOUS ACCESS
- DISPLAYS NOT ALWAYS USER-DRIVEN
- SOME DISPLAYS CANNOT BE OVERRIDDEN
- MULTIPLE DISPLAY DEVICES
- MULTIPLE FUNCTIONAL AREAS PER DISPLAY DEVICE (INPUT/OUTPUT)
WHY WINDOWS

1. HARMONY WITH NEEDS OF CONTROL CENTERS (AT LEAST)

2. EXTEND POTENTIAL USE OF TAE TO BROADER APPLICATION

3. AUGMENT TAE AS STRUCTURE FOR PROTOTYPING OF FUTURE SYSTEMS
APPROACH

- UNDERSTAND VARIOUS EXISTING WINDOW MANAGEMENT SYSTEMS
  (E.G., SMALLTALK, SUN, IRIS, VAX STATION)

- USE VT220 AND VAX STATION TERMINALS INITIALLY

- BUILD FAST PROTOTYPE (BY FALL, 1985) FOR UNDERSTANDING CONCEPTS
  - POSITIONING
  - SIZING
  - SELECTING
  - REPAINRING
  - ACTIVE WINDOWS
  - ERROR NOTIFICATION
  - SCRAPBOOK
  - APPLICATION UPGRADE
USE OF WINDOWS

° ONE EACH (TEXT): MENU, TUTOR, TCL, HELP
° ONE OR MORE (TEXT/GRAFHICS), EXCLUSIVE OR SHARED, FOR APPLICATIONS
° TAE OR APPLICATION CONTROLS INITIAL PLACEMENT
° USER CONTROLS REPLACEMENT AND SIZE
° PULL-DOWN MENUS FOR WINDOW CONTROL
° ALARM WINDOW
° WINDOWS FOR SYNCHRONOUS AND ASYNCHRONOUS PROCS
° WINDOW CONTROL: USER INTERACTIVE/TCL/APPLICATION (PDF & INTERNAL)
° INPUT/OUTPUT TO ONE OR MORE WINDOWS
° LOCKED WINDOWS
MAJOR ISSUES

° SEPARATE WINDOW MANAGER? WHERE?
  ° HANDLING OF OUTPUT EXTERNAL TO TAE WINDOW MANAGER
    (NON-TAE-WINDOW-APPLICATIONS)
  ° INTERFACE TO HOST WINDOW MANAGERS

° IMPOSE CONSISTENCY ACROSS ALL DEVICES?

° ACCOMMODATING CAPABILITIES AND LIMITATIONS OF TARGET TERMINALS

° SAVING CONFIGURATIONS

° USER CONTROL OF DEFAULT OUTPUT WINDOW

° AMOUNT OF APPLICATION CONTROL (E.G., JUST INITIAL PLACEMENT?)

° HANDLING OF NATIVE I/O
TAE APPLICATIONS
STORAGE, RETRIEVAL, PROCESSING AND DISPLAY
OF TEAL RUBY IMAGERY

Thomas M. Parris and Daniel P. Rice
Environmental Research Institute of Michigan
Ann Arbor, Michigan 48107

4 June 1985
CREDITS

Air Force Space Division

Aerospace Corporation

E-Systems Melpar Division

ERIM
CONTENTS

• What is Teal Ruby?
• What Does Our Software Do?
• How Does Our Software Do It?
• TAE Observations
WHAT IS TEAL RUBY?

- Satellite Mounted Infrared Imaging Sensor Designed to Detect Moving Airborne Vehicles
- Employs a Mosaic of Focal Plane Charge Coupled Device Arrays
- On-board Processor Performs Target Detection Algorithms
- Sophisticated Pointing Allows Sensor to “Stare” at a Fixed Ground Point While Orbiting
- Rockwell International is the Primary Contractor
WHAT IS TEAL RUBY? (Cont'd)

Focal Plane Layout
(Zones)
**What is Teal Ruby? (Cont'd)**

<table>
<thead>
<tr>
<th>1</th>
<th>9</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>8</td>
</tr>
</tbody>
</table>

**Zone Layout (Chips)**

- In-Track
- Cross-Track
WHAT DOES OUR SOFTWARE DO?

- Retrieves User Specified Imagery from Data Base
  - Number of frames
  - Time of center frame in either GMT or vehicle time
  - Level of on-board processing
  - Optional criteria

- Processes Retrieved Imagery to User Defined State
  - Calibration level
  - Bad pixel fill
  - Temporal differencing
  - On-board processor simulation
WHAT DOES OUR SOFTWARE DO? (Cont'd)

- Display Processed Imagery With Interactive Controls
  - Animation
  - Grey Level/Pseudo Color
  - Pan
  - Zoom
  - Viewport
  - Annotation
  - Split Screens

- Provide the Above Within an Interactive Time Frame
How Does Our Software Do It?

- Data Paths
- Data Structures
- I/O Optimization
- Array Processor Utilization
FOUR BASIC DATA PATHS

(1) DATABASE → IMAGE PROCESSOR
(2) DATABASE → ARRAY PROCESSOR → IMAGE PROCESSOR
(3) DATABASE → ARRAY PROCESSOR → APPLICATION
(4) PREVIOUSLY PROCESSED IMAGES → ARRAY PROCESSOR → IMAGE PROCESSOR
TWO DATA STRUCTURES USED
FOR IMAGE DATA

ISB = IMAGE SEQUENCE BLOCK

- IMAGE SEQUENCE IS A TIME-ORDERED SET OF IMAGE FRAMES
- ISB HOLDS A COMPLETE IMAGE SEQUENCE AND ITS ASSOCIATED INFORMATION
- ISB IS USED AS THE BASIC INTERFACE TO THE PRE-PROCESSING DRIVER (CPC #3)

ICB = IMAGE COMMUNICATION BLOCK

- USED TO INTERFACE TO THE IMAGE PROCESSOR
- HOLDS ONE OR MORE ISB'S
- CONTAINS A COMMUNICATION AREA FOR IMAGE PROCESSING APPLICATIONS AND UTILITIES
I/O Optimization

- All images are referenced via virtual memory addresses (pointers)
  + ISB's contain a linked list of such pointers

- Disk resident imagery becomes part of the program virtual address space via use of VAX global sections
  + Disk I/O performs at page fault rate
INNEFFICIENT DATA FLOW

ARRAY PROCESSOR RESIDENT FUNCTIONS

FUNCTION ONE  FUNCTION TWO  FUNCTION THREE

APPLICATION

VAX RESIDENT FUNCTIONS

DATA BASE

ICB
EFFICIENT DATA FLOW
PRE PROCESS DRIVER

ARRAY PROCESSOR STRUCTURE

VAX

UNIBUS

CODE STORAGE

COMMUNICATIONS AREA

INPUT STAGING AREA

OUTPUT STAGING AREA

INTERMEDIATE QUEUES AREA

SCRATCH AREA

DATA STORAGE

ARRAY PROCESSOR
PRE PROCESS DRIVER
- PPD-EXECUTE -
- EXAMPLE ALGORITHM -

INPUT SEQ
QUEUE DIFF1 LIKE SEQ
OUTPUT DIFF2
STEP
  SWDIFF DIFF1 = SEQ
  SWDIFF DIFF2 = DIFF1
END
PRE PROCESS DRIVER
— EXAMPLE —
— SECOND DIFFERENCE —

OUTPUT STAGING AREA

WORKING QUEUE

INPUT STAGING AREA

TEALRUBY CDR III - 94
TAE Experiences

- Overestimated the standardizing effect of TAE
- Tended to over rely on dynamic tutor
- Certain types of input were difficult to handle cleanly via TAE:
  + Parallel Vectors
  + Conditional Prompts
  + High Precision Reals
Customer Reaction to TAE

- Not as friendly as expected
- Low per screen information content
- Too much transition between modes

- Appreciated lower development costs and higher degree of maintainability
ABSTRACT

TAE and BISHOP in a Teaching Environment

Lesley Grove
Imperial College of Science and Technology
London, England

The Centre for Remote Sensing at ICST was set up in February 1983 to serve as an interdisciplinary focus for the development of remote sensing and image processing in a wide variety of fields. Typical users of the facility have backgrounds as diverse as Geology, Atmospheric Physics, Electrical Engineering and the Royal College of Art. Many arrive at the Centre with little or no previous experience of computing or image processing.

The Computing Facility.

The Centre calls upon several computing facilities for the research and teaching it undertakes. The Interactive Planetary Image Processing System (IPIPS) facility consists of a 4 megabyte Digital VAX-11/780 running VMS (currently 3.6) with two specialized I²S image processing workstations, a Calcomp plotter, Printronix and Brother printers and 1.5 gigabytes of disk space. This supports approximately 65 users of whom 40 are active in any one week. A second system of a PDP-11/24 and I²S model 70 running I²S system 500 is also available which supports approximately 7 users.

Why use TAE?

A major element of the Centre's activity is a master's degree course in Remote Sensing offered by the University of London which is undertaken by 20-25 students per year. These students, like many users of Centre facilities, arrive with little or no knowledge of computing and/or remote sensing. This can place a heavy load on both staff and software as the differing user interfaces to database, graphics and VICAR programs can prove confusing even to an experienced user. The first year of the course proved the available documentation to be grossly inadequate. In addition VICAR software had a reputation for user-unfriendliness which did nothing to attract users to the Centre's facilities.
In this environment intensive use has since been made of the user friendly aspects of the program/user interface provided by TAE. Accommodation for a wide spectrum of potential users has been implemented, from the most novice end-users to the expert systems programmer. For MSc students in remote sensing/image-understanding/pattern recognition menus are supplied covering topics such as interactive contrast/brightness stretching, edge detection using convolution, and Fourier spectrum editing for image manipulation using super-ellipses (for example destriping zebras).

BISHOP

Because a large part of our image manipulation software is developed (and then subsequently put in system libraries for general use) by students during their Post-graduate degree a program interface package has been developed called Better Image Software with Help On-line for Programs (BISHOP). BISHOP is an upgrade to VICAR which is very much more programmer friendly, a first draft system is now available to programmers with on-line documentation for callable subroutines. This has been provided through the mechanism of the PDF precisely as for full PROCS. The design philosophy has been to provide an interface to image I/O whereby a bare minimum of parameters are required by subroutines and all other information is determined by the subroutine itself. Portability is not a design goal of the BISHOP system. Source code is well structured and documented to allow programmers on other systems to take full advantage of the target systems facilities.

The most ambitious part of the BISHOP environment involves the setting up of a device independent display package. The model 70 and model 75 can be driven by routines that link to a dummy shareable image at link time and a real shareable image at run time in a manner similar to DMS.
Workstation Activities of the Pilot Land Data System

William Likens

NASA Ames Research Center
Ecosystems Science and Technology Branch
Code SLE: 242-4
Moffett Field, CA 94035

Presented
June 4, 1985
at
5th Annual TAE User's Conference
NASA Goddard Space Flight Center
Greenbelt, Maryland
Pilot Land Data System

Goal

- Make use of computing resources by land scientists easier and more efficient

Objective

- Develop a prototype distributed processing system connecting workstations, minicomputers, and super computers at ARC, GSFC, JPL, NSTL, and related institutions

W. Likens
NASA Ames Research Center
6/4/85
PLDS PROGRAM GOAL

To Establish a Prototype State-of-the-Art Data and Information System to Support Research in the Land Related Sciences Which Will Lead to a Permanent Research Tool
Target Workstation Types

- Project Management
- Field Data Collection
- Image Display
- Image Analysis

W. Likens
NASA Ames Research Center
6/4/85
Target Application Functions

- Image Display, Processing
- GIS Digitizing, Map Overlay
- Statistics
- Instrument Control, Data Collection
- Word Processing
- Data Communications

W. Likens
NASA Ames Research Center
6/4/85
Target Microcomputers

- Colby PC
- IBM PC/XT
- IBM PC/AT
- AT&T Unix PC
- Sun 2
- MicroVAX II
- Future - Other Unix systems
  Ridge 32S
  Apollo 320

W. Likens
NASA Ames Research Center
6/4/85
Target Operating Systems

• MS-DOS
• Unix 4.2 BSD
• Unix System V
• VMS

W. Likens
NASA Ames Research Center
6/4/85
Target Image Analysis Software

- Under MS-DOS: Commercial Packages (ERDAS, PCVision, etc.)

- Under Unix: a) TAE/LAS*
  b) TAE/VICAR2 (MIPL) **
  c) TAE/ELAS **
  d) TAE/EDITOR **
  e) TAE/SPAM **

- Under VMS: a) TAE/LAS
  b) TAE/VICAR2 (MIPL)

* Currently under development at EDC
** To be developed

W. Likens
NASA Ames Research Center
6/4/85
## Potential PLDS Systems

**Original Page is of Poor Quality**

<table>
<thead>
<tr>
<th>Field Data Collection</th>
<th>Image Display #1</th>
<th>Image Display #2</th>
<th>Image Analysis #1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$8500</td>
<td>$10,000</td>
<td>$12,000</td>
<td>$42,000</td>
</tr>
<tr>
<td>System: Colby PC</td>
<td>System: IBM PC/XT</td>
<td>System: IBM PC/AT</td>
<td>System: IBM PC/XT</td>
</tr>
<tr>
<td>Disk Memory: 20MB</td>
<td>Disk Memory: 10MB</td>
<td>Disk Memory: 20MB</td>
<td>Disk Memory: 10MB</td>
</tr>
<tr>
<td>Color Monitor</td>
<td>Color Monitor</td>
<td>Color Monitor</td>
<td>Color Monitor</td>
</tr>
<tr>
<td>Video Memory</td>
<td>Video Memory</td>
<td>Video Memory</td>
<td>Video Memory</td>
</tr>
<tr>
<td>1/2&quot; Tape Drive</td>
<td>1/2&quot; Tape Drive</td>
<td>1/2&quot; Tape Drive</td>
<td>1/2&quot; Tape Drive</td>
</tr>
<tr>
<td>Modem</td>
<td>Modem</td>
<td>Modem</td>
<td>Modem</td>
</tr>
<tr>
<td>Digitizer Pad</td>
<td>Digitizer Pad</td>
<td>Digitizer Pad</td>
<td>Digitizer Pad</td>
</tr>
<tr>
<td>Printer</td>
<td>Printer</td>
<td>Printer</td>
<td>Printer</td>
</tr>
<tr>
<td>Image Display</td>
<td>Image Display</td>
<td>Image Display</td>
<td>Image Display</td>
</tr>
<tr>
<td>Image Analysis</td>
<td>Image Analysis</td>
<td>Image Analysis</td>
<td>Image Analysis</td>
</tr>
<tr>
<td>GIS digitization</td>
<td>GIS digitization</td>
<td>GIS digitization</td>
<td>GIS digitization</td>
</tr>
<tr>
<td>GIS Map Overlay</td>
<td>GIS Map Overlay</td>
<td>GIS Map Overlay</td>
<td>GIS Map Overlay</td>
</tr>
<tr>
<td>Statistics</td>
<td>Statistics</td>
<td>Statistics</td>
<td>Statistics</td>
</tr>
<tr>
<td>Word Processing</td>
<td>Word Processing</td>
<td>Word Processing</td>
<td>Word Processing</td>
</tr>
<tr>
<td>Chart Software</td>
<td>Chart Software</td>
<td>Chart Software</td>
<td>Chart Software</td>
</tr>
<tr>
<td>Instrumentation</td>
<td>Instrumentation</td>
<td>Instrumentation</td>
<td>Instrumentation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image Analysis #2</th>
<th>Image Analysis #3</th>
<th>Image Analysis #4</th>
<th>Image Analysis #5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$45,000</td>
<td>$40,000</td>
<td>$55,000</td>
<td>$60,000</td>
</tr>
<tr>
<td>Disk Memory: 20MB</td>
<td>Disk Memory: 40MB</td>
<td>Disk Memory: 300MB</td>
<td>Disk Memory: 300MB</td>
</tr>
<tr>
<td>Color Monitor</td>
<td>Color Monitor</td>
<td>Color Monitor</td>
<td>Color Monitor</td>
</tr>
<tr>
<td>Video Memory</td>
<td>Video Memory</td>
<td>Video Memory</td>
<td>Video Memory</td>
</tr>
<tr>
<td>1/2&quot; Tape Drive</td>
<td>1/2&quot; Tape Drive</td>
<td>1/2&quot; Tape Drive</td>
<td>1/2&quot; Tape Drive</td>
</tr>
<tr>
<td>Modem</td>
<td>Modem</td>
<td>Modem</td>
<td>Modem</td>
</tr>
<tr>
<td>Digitizer Pad</td>
<td>Digitizer Pad</td>
<td>Digitizer Pad</td>
<td>Digitizer Pad</td>
</tr>
<tr>
<td>Printer</td>
<td>Printer</td>
<td>Printer</td>
<td>Printer</td>
</tr>
<tr>
<td>Image Display</td>
<td>Image Display</td>
<td>Image Display</td>
<td>Image Display</td>
</tr>
<tr>
<td>Image Analysis</td>
<td>Image Analysis</td>
<td>Image Analysis</td>
<td>Image Analysis</td>
</tr>
<tr>
<td>GIS digitization</td>
<td>GIS digitization</td>
<td>GIS digitization</td>
<td>GIS digitization</td>
</tr>
<tr>
<td>GIS Map Overlay</td>
<td>GIS Map Overlay</td>
<td>GIS Map Overlay</td>
<td>GIS Map Overlay</td>
</tr>
<tr>
<td>Statistics</td>
<td>Statistics</td>
<td>Statistics</td>
<td>Statistics</td>
</tr>
<tr>
<td>Word Processing</td>
<td>Word Processing</td>
<td>Word Processing</td>
<td>Word Processing</td>
</tr>
<tr>
<td>Chart Software</td>
<td>Chart Software</td>
<td>Chart Software</td>
<td>Chart Software</td>
</tr>
<tr>
<td>Instrumentation</td>
<td>Instrumentation</td>
<td>Instrumentation</td>
<td>Instrumentation</td>
</tr>
</tbody>
</table>
Future Plans

- Test & Evaluate Commercial Products
- Port Selected Software to Unix
- Integrate Selected Software with TAE
- In 1986, emphasize
  - Field Data Collection Workstation based on Colby PC
  - Image Display System based on IBM PC/XT
  - Image Processing Systems based on:
    IBM PC/AT
    Sun 2
- Continue to survey technology and plan for future migration to more powerful microcomputers

W. Likens
NASA Ames Research Center
6/4/85
Workstation Activities of the Pilot Land Data System

NASA is currently funding several projects that are exploring the development of computer data systems for more efficient and easier processing of science data. The Pilot Land Data System (PLDS), now in its first year, is one of these efforts. The intent is to make computer resources more readily available to and usable by land science researchers. The PLDS is to be a distributed processing system connecting computation resources at a number of NASA Centers and associated universities. Initially to be a small prototype system, PLDS is planned to eventually grow and evolve into a full-scale operational research support tool.

The System Access Capabilities effort, led by the Ames Research Center, is one of several PLDS technical areas. Insuring effective access of computer resources involves addressing a range of technical and organizational issues. The times for initiating the various subelements of PLDS system design and prototyping are staggered. The first System Access Capabilities area in which technical work has begun is that of Workstation Configuration.

For many scientists, the microcomputer workstation will be the principle tool for accessing the PLDS network. Thus, the workstation will not only be a tool for local processing, but will provide the primary user interface to the PLDS. Tasks to be carried out on PLDS workstations include word processing, statistical analyses of field, laboratory, and remotely sensed data, record keeping and data management, and some image display and processing capability.

Rather than engaging in major new system development efforts, the PLDS will emphasize the identification of means for enhancing existing microcomputers that scientists already have, or when acquisition of a new system is desired, identify appropriate systems that may be purchased. Recommended designs will be those identified as already successfully implemented at some cooperating site, plus additional systems configured and tested through the PLDS Workstation Configuration effort.

In the PLDS an emphasis will be placed on the collection and dissemination of technical data about the latest commercial and government workstation activities so that duplication of other efforts may be avoided. A great many workstation development efforts are now in progress within government. These efforts are taking place partly because currently available systems usually do not offer the full range of required functions, especially with regard to narrow science problems. Also, there is limited information exchange between groups conducting workstation development, so that duplication of work may often occur.
We are developing a list of people interested in participating in a Technical Working Group. Members of this group will periodically receive information on PLDS workstation activities, including copies of the Workstation Configurations Data Book, a continuously updated document of technical findings and recommendations. Occasionally, participants in the Technical Working Group will be asked to respond to requests for information about their workstation development activities. If you are involved in NASA associated workstation development and would like to become a member of this group, please write me at the address listed below.

William Likens
PLDS System Access Capabilities Manager
Ecosystems Science and Technology Office/SLE-242-4
NASA Ames Research Center
Moffett Field, CA 94035
TAE AND THE SPACE STATION
USER INTERFACE

TAE USER'S CONFERENCE

JUNE 4, 1985

KAREN MOE
DOLLY PERKINS

GODDARD SPACE FLIGHT CENTER
POTENTIAL SPACE STATION OPERATIONAL INTERFACES

NOTES:
USERS=SCIENCE RESEARCH, SCIENCE OPERATIONS, COMMERCIAL, SUPPORT OPERATIONS PERSONNEL
DB=LOCAL DATA BASES

K. MOE
6/85
BACKGROUND

Recommendations of the Ad Hoc Committee to investigate the need for Space Station User Interface Language (UIL) - 1984

1. Common Interactive System Interface is needed
   - system, payload, instrument integration, test & operations
   - commonality between flight & ground user interfaces for coordinated control of on-board instruments to avoid problems experienced by Shuttle payload experimenters.

2. Development of UIL requires direct involvement of
   - payload scientists, engineers & operations controllers/GSFC
   - integrations & test engineers/KSC
   - flight crew & mission operations controllers/JSC
   in both requirements definition and prototype interface evaluation.

3. Requirements from the above 3 areas need to be integrated to define the specification for common interface elements plus and unique requirements. UIL development & support software should be part of the separate Software System Environment contract planned for Space Station.
WHAT IS UIL?

Purpose
To provide an interface between the User and the System for control and communication operations, and to standardize this interface for use at all facilities through Space Station phases:
- mission planning and scheduling
- subsystem and instrument ground test
- ground integration and test
- flight integration and test
- flight operations and maintenance
- Space Station 'growth' implementation
- ground refurbishment and retest

Philosophy
- Common interfaces where needed
- Common services where cost-effective or helpful
- Do not restrict user flexibility

Elements of UIL
- Language - commands, procedures
- Network interface
- Interface to system data and services
- Interface to planning and scheduling
- Executive for software control and resource management
WHAT IS UIL? (continued)

Functions of UIL
- Established connections to system services, resources, mission planning, scheduling, engineering/ancillary data, command input, data delivery
- Provides common interface where multiple users are involved (e.g. crew operations and payload development, integration, test, operations, servicing)
- Provides optional services to payload users (e.g. interactive interface for applications software, procedure development, control)

Scope of UIL
- A set of software tools for flexible but standard User-to-System interface (and not a programming language)
- Provides English-like language for familiarity, readability
- Provides short form for real-time interactive control
- Includes software utility support for graphics, display, data and dialogue interface and control

Intended Users
- Investigators (science research, science operations, commercial use)
- Operations Personnel (e.g. payload, platform, and Space Station core engineers, controllers, and ground support personnel)
- Flight Crew (both investigators and operations personnel)
KEY FEATURES OF DESIGN: MACHINE INDEPENDENT, MODULAR HW & SW COMPONENTS

COMMON FUNCTIONS REQUIRED BY ALL USERS, E.G. ACCESS TO DATABASES COMMUNICATION OPS LANGUAGE TOOLS AND AIDS

KERNEL FUNCTIONS

USER-SPECIFIC CONCERNS

CREW OPERATIONS

GROUND OPERATIONS

SCIENCE OPERATIONS

DISCIPLINE-SPECIFIC NEEDS

FLIGHT CREW PAYLOAD SPEC.

SHUTTLE SPACE STATION SERVICING PLATFORM PAYLOAD INSTRUMENT

ASTROPHYSICS COMMUNICATIONS EARTH & LIFE SCIENCE MICROGRAVITY PLASMA PHYSICS PLANETARY

K. MOE
6/85
APPRAOCH

UIL Requirements Analysis
- Based on experience with current GSFC facilities and their users, extrapolate a definition of future user interfaces for Customer Data and Operations Systems
- Define customer needs ('kernel functions') and modes of operations (e.g. routine operations vs dynamic interaction)
- Incorporate state-of-the-art user interface concepts
- Iterate requirements with users

UIL Prototyping Lab
- Acquire user interface prototyping tools for developing static and dynamic displays, dialogues
- Acquire state-of-the-art workstation (with UNIX, Ethernet, window managers, graphics, symbolic processing, etc.
- Develop workstation design concept (machine independent, modular hardware/software components)
- Build additional kernel functions onto TAE

UIL Evaluation
- Define language evaluation criteria and evaluator's checklist
- Develop typical user scenario (e.g. payload operations in a Space Station environment)
- Demonstrate TAE, STOL and GOAL in user scenario
- Prototype key UIL/workstation features, demonstrate to users for evaluation and feedback
SPACE STATION UIL ACCOMPLISHMENTS

- Completed preliminary report of *Space Station User Interface Language (UIL): The Goddard Perspective*, dated May, 1985

- Defined language evaluation criteria and checklist

- Sponsored Space Station Workstation Technology Workshop (proceedings available this summer)

- Acquired TAE and STOL systems, developed user scenario for TAE demonstration given to Dr. Bob Parker/JSC Astronaut Office

NEXT STEPS

- Complete acquisition of lab elements (Sep. 1985)

- Complete UIL requirements analysis and workstation functional definition (Dec. 1985)

- Develop and demonstrate UIL prototype (1986-1987)

- Elicit comments from potential users (on-going)
  - Involvement in Software Support Environment (SSE) specification
  - Prototype UIL
    - Visualize, evolve, and verify design
    - Determine where hooks are needed for growth and flexibility
    - Determine requirements for commonality vs common needs for services
  - Determine right level of commonality (more than none, and less than complete standardization)

For more information contact:
Karen Moe
Code 522.2
NASA Goddard Space Flight Center
Greenbelt, MD 20771
Phone: FTS 344-5292
301 344-5292
Telemail: KMOE
DRAFT
UIL EVALUATION CRITERIA

LEXICAL
- What character set?
  Case independent?
  Shift required for symbols? -- hardware dependent

VOCABULARY
- Naming conventions?
  Maximum identifier length?
  Abbreviations?
  Radical difference when one character is dropped?
  Device names?

- Reserved command words?
  Enough?
  Flexible?
  Short, simple, clear?

SYNTAX
- Understandable?
  Column dependent?
  Depends on punctuation, blanks?
  Clear signalling of nesting?
  Prompted for parameters, shown defaults?

SEMANTICS
- Minimum of concepts within user’s frame of reference?
  Different things look different?
  Readable and self-evident?

DIALOGUE

INPUT
- Mode of interaction
  Command?
  Menu?
  Form filling?
  Computer initiated?

- Method of interaction
  Alternate input devices supported?

- Command stacking?

- Moving through menu trees?

- Cursor/active field easily found?

OUTPUT
- Positive feedback acknowledges each command accepted?

- Intelligent response to queries?
  Previous context reasoning?
  Answer to general focus of query?
- Help
  Always available?
  Easy to invoke?
  Context sensitive?
  Applicable commands/procs at this point?

- Error messages
  Brief?
  Positive in tone?
  Specific?
  Comprehensible?

- Error recovery
  Explains how to recover? (or available in help?)
  Repeats command line for editing?
  Undo, cancel, and abort capability?

DISPLAY
- Graphical display of information?
  User configurable?

- Format of information
  Proper labelling and grouping?
  Density kept to manageable level?
  Changes made apparent to user (e.g. highlighting)?

- Scrolling and paging capability?

ENVIRONMENT
- Expandable?
  Define, add, and utilize new commands (procs) and devices?
  Define and utilize variables within a session/procedure?
  Store sequences for replay?

- Configurable?
  Select system help, expert, and prompt levels?
  Able to restrict use of certain commands?
  Can establish tailored user profile?

- Present status information available?
  System: Experiment
  Session: Payload
  Task: Instrument
  Proc

- Past history information available?
  View session log?
  Performance statistics?

- Robust?
  Automatic recovery after crash?

- Quick response time?

- Able to dump screen to printer?
CONCEPTUAL - Type of language?
  Command?
  Query - results desired?
  Functional description?

- Objects language works with?
  Session                              Experiments
  Tasks                                Payloads
  Procs                                Platforms
  Files                                Instruments
  Messages/Mail                        Data
  System (devices)                     Parameters/variables

- What relationships can be established between objects?
- What operations can be performed on objects?
Draft Language Evaluation Checklist

Language being studied: ____________________________
Are you fluent in it? ________

Is the language command-oriented? ________
If not, is it menu-driven? ________
If neither, describe: ____________________________

Does the language support lower-case? ________
Does the language require that special characters be used beyond standard alphanumerics? ________
If so, estimate how many: ________
Are any commands used with shifted characters? ________
If so, is this inconvenient? ________

Is there a maximum variable-name length? ________
If so, what is it? ________

Are abbreviations allowed? ________
If so, are they a) helpful or b) confusing? ________
Do you use them? ________
If no, why not? ____________________________

Please estimate how many reserved command words there are? ________
Rate the amount: (0=too few - 5=too many) ________
Estimate how many are in your everyday use? ________

Rate the command words:
  length: (0=too short - 5=too long) ________
  clarity: (0=not clear - 5=clear) ________

Rate the command structure:
  length: (0=too short - 5=too long) ________
  clarity: (0=not clear - 5=clear) ________
  parameters: (0=too few - 5=too many) ________
  intricacy: (0=simple - 5=complicated) ________

Rate the syntax: (0=very bad - 5=very good) ________
Is it understandable? ________
Is it column dependent? ________
Do you think that is good or bad? ________

When typing commands, if parameters are skipped, what does the system do? ________
How easily are commands confused by one or two characters? (0=easy → 5=hard) ? ______
Is this critical or undesirable in any case? ______
If you use abbreviations, does this increase or decrease mistakes? ______
Examples: ____________________________________________

How readable is the language? (0=not very → 5=very) ? ______

Does the language allow user-defined variables? ______
Does the language type variables (integer, real, string, etc.) automatically? ______
If not, do you have to declare them? ______
If so, how hard is that? ______
What do you use the variables for? ________________________________

Is on line help available? ______
If so, is it easy to get? ______
How helpful is it? (0=minimally → 5=very) ? ______
Is the help context-sensitive? ______
Can you get more if you need it? ______
Can you configure the level of initial help? ______

Rate the system's error messages (0=bad → 5=good) ? ______
Are they short? ______
Are they negative in tone? ______
Do they identify the source of the error? ______
How specific are they? (0=not → 5=very)? ______
Are they informative? ______
How easy are they to comprehend? ______
What would you change about them, given the chance? ______

Answer the same questions for the other non-error system messages? ______
How easy is it to recover from errors? (0=hard 5=easy)?

Does the system allow you to "undo" commands?

What about "abort" or "cancel" them?

Can the use of commands be restricted?

Can you configure:
   a) levels of expertise?
   b) prompt/no prompt?

Does the interface support non-keyboard input?
   If so, what kind?
      a) mouse
      b) touchscreen
      c) puck
      d) light pen
      e) voice
      f) other
   If not, should it?
      If so, which ones?

Does the interface support a variety of outputs?
   (such as graphics, audio (beeps), voice)
   If so, name them:
   If so, which ones should you be able to turn off?

Can you see the user-defined variables' default values?
   If so, can you see current values as well?

Are you able to see which commands are available to you at a given point?

Can you check the system status?
If so, how much is shown (0=very little \( \rightarrow \) 5=too much)? ---------

Can you view the session log? ---------

Is its length restricted? ---------

Can sequences of events be stored for later playback? ---------

Can new functions be defined? ---------

Can new devices be added? ---------

Can you check user-defined entities? ---------

Is it possible to modify old screens or to generate new ones to suit user’s taste? ---------
Unidata, the Next Generation Meteorological Data System in Universities

George J. Huffman
Dept. of Meteorology, U. of Maryland
College Park, MD

1. Introduction

Meteorology has a long history of data-intensive study. The routine surface and upper air observations over North America alone amount to some 5 Megabytes of coded data each day. John von Neumann recognized this problem in the late 1940's and chose weather forecasting as an early problem for the ENIAC. Since then meteorologists have eagerly awaited each new generation of computer.

To date, most of the technological progress in data analysis has been concentrated at a few large centers (for example, NASA/GSFC) due to expense. Now, however, individual Meteorology Programs have the opportunity to take a large step forward in handling and analyzing meteorological data at their home institutions. Both wide-band communication satellite systems and "super" microcomputers are rapidly maturing and dropping in price. Together, these offer the potential of high throughput at a reasonable cost.

2. Unidata Concepts

To help Meteorology Programs tap these potential improvements in data handling, the Unidata Project was initiated in early 1983 by the University Corporation for Atmospheric Research, the National Science Foundation, and Meteorology Programs across the United States. Each participating Program has distinct needs and resources, so Unidata's governing concept is that the Unidata system should have a general design that is configurable to several levels of functionality.

At first the Unidata system was envisioned as only providing access to National Weather Service real-time data and helping standardize data analysis software. Since that point the proposed system architecture has become better-defined and more powerful. In mid-1983 Unidata's scope was expanded to include access to archived data and supercomputers. By mid-1984 the hardware system at each university was envisioned in terms of scientific workstations attached to a Local Area Network (LAN). Next, planning for long-haul communications was split into two segments; a receive-only real-time data channel, and a wideband bidirectional channel for traffic between supercomputers and the Unidata participants. This communications split provides reliable transmission of time-critical data at low-interference wavelengths, and frees the high-volume traffic between Unidata
sites to use the KU band. Although more weather-sensitive, the KU band is desirable for this purpose because a currently-available, moderately-priced satellite communication system supports TCP/IP between LANs over the KU band (Vitalink earth stations with TransLAN interfaces). Design choices currently under discussion include additional data sources, a data ingest/management machine, and optical disks for local data archiving.

The current Unidata Project system design is very ambitious, seeking a flexible solution which addresses all of the major data system components; data sources, data transmission, local site data management, data analysis software, and hardware recommendations (see Figure 1). The system should work as follows (Unidata components underlined): A variety of real-time data is broadcast to the local site, where the database machine formats the data for processing. A researcher or student sitting at a workstation starts an analysis program, which requests data from the database machine across the LAN. Another person at some other workstation sends a request across the LAN, then over the wideband link to a remote supercomputer. In turn, the supercomputer sends model output or archived data back to the workstation. As much as possible, Unidata seeks to employ off-the-shelf resources for each component, but the concept of configurability imposes some limitations.

3. Unidata Software

On the software level, the configurability goal has been translated into several design goals. Briefly, the software should be maintainable, transportable, and friendly. Modular, documented code in standard languages is clearly important. In addition, prior experience suggested the UNIX paradigm of a "toolbox" of application software which the user accesses from an executive "shell". Within the Unidata Project the task of setting functional specifications for the shell was given to the LOcal Hardware-Software System (LOHSS) Working Group. LOHSS reviewed a number of shells, including TAE, studied their design philosophies, and eventually converged on a small number of shell functions desired by Unidata:

1. Provide a uniform interface to the user.
2. Provide interface "hooks" for the application programmer.
3. Protect applications from the host system utilities. In principle the entire "toolbox" of applications can be ported to a new host simply by porting the shell.
4. Provide additional interface services ("environment" or "context" variables, error recovery, etc.).

Concurrently, LOHSS developed a list of shell components needed to meet these functional specifications; a user interface, a display formatter, a parser, environmental variables, a data base interface, an application interface, support utilities, and
error recovery. Finally, LOHSS developed a scorecard for rating shells according to Unidata's perceived needs. In addition to the technical aspects discussed above, the shell's maturity and current support received significant weight because Unidata's funding and timeline are constrained.

The review of existent shells revealed only three viable candidates. Fortunately, TAE had been designed along lines similar to those envisioned by Unidata, and it had already been operational for several years. TAE passed the technical evaluation and received high marks for software maturity and support. An additional advantage for TAE was Unidata's interest in GEMPAK, a meteorological data analysis package developed at NASA/GSFC and running under TAE.

To round out the software discussion, LOHSS is still planning the application tools, but they likely will include GEMPAK, GEMPLT, NCAR Graphics, and various community-generated applications.

4. Hardware Issues

The original prospectus for Unidata referred to Level I, II, and III systems, providing low, medium, and advanced analysis capabilities, respectively. This hierarchy is still being defined, with significant questions remaining as to whether the same shell/application package can serve all levels. LOHSS has agreed that the software shell/tools package will work well on a variety of multitasking, 16- or 32-bit super microcomputers with graphics capability. Such hardware systems have been labeled Level II. Level III is vaguely described as Level II plus image analysis. Vigorous discussions on Level I tend to center around IBM AT types of systems.

The wide variety of equipment which participating Meteorology Programs possess makes the goal of configurability a stringent condition on Unidata. It is impossible to support a perfectly general software package, but Unidata intends to cover a number of operating systems. It is certainly an advantage that TAE already runs under VAX VMS and UNIX. Other operating systems will be added as need, interest, and resources arise.

5. Conclusion

Meteorology Programs around the country, with leadership from UCAR and funding from NSF, are cooperating in the Unidata Project. Unidata is intended to spread modern meteorological data analysis tools throughout the university community, and subsequently enhance the various Programs' ability to carry out research and instruction. As you have heard, Unidata is currently in the planning phase. According to present estimates, a fully functional system will be built by mid-1986.
Figure 1. Schematic diagram of Unidata Project components,
Dynamically Constructing a TAE System of Menus and Procs

Michael Gough
Tamara Weaver

Science Applications Research Inc.
Dynamically Constructing a TAE System of Menus and Procs

by

Michael Gough
Science-Applications-Research
4400 Forbes Boulevard
Lanham, MD 20706
301-794-5200

Tamara Weaver
Science-Applications-Research
4400 Forbes Boulevard
Lanham, MD 20706
301-794-5200

ABSTRACT

A data-independent graphics system for the display of multi-dimensional data sets has been developed at NASA's Goddard Space Flight Center. TAE was used to implement the user interface for this system. Since the system must handle many diverse data sets, the user interface must be extremely flexible. A TAE tutor is used to determine which data set is to be displayed by the system, and the user interface is dynamically constructed according to the structure and contents of the data set. Once built, the customized TAE user interface is invoked, and user interaction begins.

This approach provides user friendliness, because descriptors in the user-selected data set contain information to be used in the menus and procs. The user is therefore immediately familiar with the tutor screens. Appropriate defaults and valid lists are generated for data set dependent parameters throughout the system. This reduces mandatory user interaction, and allows users to get results quickly and easily.

Since the user interface is completely built before it is invoked, users do not have to wait intermittently for tutors to be constructed dynamically, as in previous implementations of this system. This approach provides the benefits of dynamically constructed tutors, without sacrificing interaction speed. Methods for dynamically "compiling" the procs are now being investigated.

A tree structure is used to allow selective invocation of procs, so the user encounters only those procs he desires to enter. The inexperienced user therefore does not have to examine advanced options, while the experienced user can do so easily. User selections are automatically saved when a proc is run, and the user is returned to the top of the tree structure.
All user interaction takes place before the applications program is invoked. When the user is satisfied with all parameter selections, he can invoke the applications program via menu selection. The applications program is executed, and control is returned to the user upon completion. The user may change his parameter selections via the tree structure, and re-invoke the applications program. At no point is it necessary to re-enter the previous parameter selections, since they are automatically restored each time the proc is invoked.

TAE specific software and applications software are not interlaced. This modular design allows for ease of maintenance. Applications software can also be easily transported to a non-TAE environment.
Pilot Climate Data System (PCDS)

Consists of 5 Subsystems:

Catalog Subsystem
Inventory Subsystem
Data Access Subsystem
Data Manipulation Subsystem
Graphics Subsystem
PCDS Graphics Subsystem

Purpose:

To provide interactive graphic displays of scientific data

Design Goals:

Data Independence
Climate Data File (CDF)

Device Independent Graphics
TEMPLATE

User Friendliness
TAE User Interface
User Interface Features

Minimal User Interaction
Valid list function key
Automatic save and restore

Selective Invocation
Beginner sees only essential tutors
Expert has easy access to advanced features

Customized for User’s Data Set
"Dynamic Construction"
Dynamic Construction Data Flow

User Input

CDF Selection Tutor

Selected CDF

CDF name

BUILD

Customized Tutors
Interactive Graphics Data Flow

User Input

Customized TAE System

Parameter Files

Selected CDF

Graphics Applications Program

Graphics Output
procedure help=* 
parm-page FILT1  type=(string,6) count=0:1 default=-- +  
valid=( +  
"YEAR ", +  
"PRESSURE", +  
"SEASON ", +  
"TEMPDEV ", +  
"L_TEMPDEV", +  
"TROPICS ", +  
"NHEMIS ", +  
"SHEMIS ", +  
"GLOBAL ")  
parm RANGE1  type=real count=0:2 default=--  
parm INCL1  type=(string,9) valid=("INCLUSIVE","EXCLUSIVE") default="INCLUSIVE"  
parm-page FILT2  type=(string,8) count=0:1 default=-- +  
valid=( +  
"YEAR ", +  
"PRESSURE", +  
"SEASON ", +  
"TEMPDEV ", +  
"L_TEMPDEV", +  
"TROPICS ", +  
"NHEMIS ", +  
"SHEMIS ", +  
"GLOBAL ")  
parm RANGE2  type=real count=0:2 default=--  
parm INCL2  type=(string,9) valid=("INCLUSIVE","EXCLUSIVE") default="INCLUSIVE"  
parm-page FILT3  type=(string,8) count=0:1 default=-- +  
valid=( +  
"YEAR ", +  
"PRESSURE", +  
"SEASON ", +  
"TEMPDEV ", +  
"L_TEMPDEV", +  
"TROPICS ", +  
"NHEMIS ", +  
"SHEMIS ", +  
"GLOBAL ")  
parm RANGE3  type=real count=0:2 default=--  
parm INCL3  type=(string,9) valid=("INCLUSIVE","EXCLUSIVE") default="INCLUSIVE"  
parm-page FILT4  type=(string,8) count=0:1 default=-- +  
valid=( +  
"YEAR ", +  
"PRESSURE", +  
"SEASON ", +  
"TEMPDEV ", +  
"L_TEMPDEV", +  
"TROPICS ", +  
"NHEMIS ", +  
"SHEMIS ", +  
"GLOBAL ")  
parm RANGE4  type=real count=0:2 default=--  
parm INCL4  type=(string,9) valid=("INCLUSIVE","EXCLUSIVE") default="INCLUSIVE"  
parm-page FILT5  type=(string,8) count=0:1 default=-- +  
valid=( +  
"YEAR ", +  
"PRESSURE", +
WHAT'S A 2GCHAS?

TAE
TAE as the Foundation of a Transportable Executive for Helicopter Analysis

Richard Gemoets
Computer Sciences Corporation/
NASA/Ames Research Center
WHAT?

• To Provide a Computer/Host Independent Helicopter Analysis Environment

HOW?

• COMMON DATA DEFINITIONS
• COMMON ANALYSIS PROGRAMS
• COMMON ANALYSIS PROCEDURES
• COMMON USER INTERFACE

AND
(of course)

• A TRANSPORTABLE, USER AWARE EXECUTIVE
THE ROLE
OF
TAE
- A PARADIGM
- THE USER INTERFACE
- RAPID PROTOTYPE DEVELOPMENT
ROLE OF TAE

USER INTERFACE

USER

PDF

MEC
MODULE EXECUTION CONTROL

MODULE MAIN

DT
DEVELOPER TOOLKIT

WR
REPORT WRITER & GRAPHICS

DM
DATA MANAGEMENT

BI
BATCH INTERFACE

any technology module (APPLICATIONS)

RDB FILES

TAEUC
6/4/85
ADDITIONAL FEATURES
MODULES vs. PROCESSES

DEVELOPER TOOLKIT

RUNTIME DATA MANAGEMENT

- GRAPHICS
- REPORT WRITER
- BATCH
TAE
-GENERAL EXEC

APPLICATION PROCESSES

2GCHAS

TAE
-GENERAL USER INTERFACE

SYSTEM PROCESSES

MODULE A

MODULE B

MODULE C

MODULE D

MODULE E

APPLICATION PROCESSES
FORTRAN:
...
CALL B(I, R, S, STATUS)
...

MODULE B
BODY

B-CALLER
FORTRAN:
CALL A(I,R,S,STATUS)

User Language:

A, I, R, S

Module Execution Control: Invocation of a Module
SUBROUTINE A (X,Y,Z)

....

CALL B (P,Q)

....

END

SUBROUTINE B (R,S)

....

END

STANDARD FORTRAN SUBROUTINE LINKAGE
SUBROUTINE A

CALL B(P,Q)

END

SUBROUTINE B(R,S)

END

DEFMODULE ACALLER

A.OBJ

LINK

A.EXE

DEFMODULE BCALLER

B.OBJ

LINK

B.EXE

EXECUTIVE

2GCHAS TECHNOLOGY MODULE LINKAGE
**Developer Toolkit: The DEFINE-MODULE Operation.**
SUBROUTINE NEWTON (N, COEFS, START, ROOT, STATUS)

C .TITLE
NEWTON -- Roots of a polynomial

C .HELP
NEWTON find the real root of a REAL polynomial with 1 or more
coefficients using NEWTON's method.

C NEWTON finds at most 1 root. If there are multiple roots, the
value of START will determine which one is reached.

C NEWTON will set STATUS to fail if the number of coefficients is
less than 1 or if the method does not yield a root within 10
iterations.

C .PAGE
NEWTON calls EVAL to evaluate the polynomial

C .VAR N INTEGER IN
Number of coefficients in
the polynomial

C .VAR COEFS REAL IN COUNT=1:10
The coefficients of the
polynomial.

C .DETAIL
The coefficients are ordered so that the first entry has the
highest exponent and the Nth entry has the 0th exponent.

C .VAR START REAL IN
Starting value for the
approximation

C .VAR ROOT REAL OUT
The value of the root

C .DETAIL
ROOT will be zero if an error occurs

C .VAR STATUS INTEGER OUT
Completion Status

C .END

C Change Log:
C 10 apr 85 -- create Newton

C INCLUDE 'chas$inc:exsufa.fin'
INCLUDE 'chas$inc:exunit.fin'

C INTEGER status

C DIMENSION coefs (n)
x = start
DO 100 loop = 1, 45, 1
    CALL eval (n, coefs, x, fx, fdotx, instat)
    IF (in-stat .NE. xsucc) THEN
        WRITE (xstout, 20020) loop, x
        20020 FORMAT ('bad status from eval at',i5,13.6)
        GOTO 120
    ENDIF
    WRITE (xstout, 20040) loop, x, fx, fdotx
    20040 FORMAT ('NEWTON',i4,3g14.6)
    IF (ABS (fdotx) .LE. 0.00000001) THEN
        WRITE (xstout, 20060) x, fdotx
        20060 FORMAT ('Slope too small for Newton to converge',2G13.6)
        GOTO 120
    ENDIF
    xnext = x - fx / fdotx
    diff = ABS (x - xnext)
    IF (diff .LE. 0.01 * AMAX1 (ABS (x), ABS (xnext))) THEN
        root = xnext
        GOTO 200
    ENDIF
    IF (diff .LE. 0.005) THEN
        root = xnext
        GOTO 200
    ENDIF
    IF (fx * fxlast .LT. 0) THEN
        xt = x - fx * (x - xlast) / (fx - fxlast)
        WRITE (xstout, 20090) x, xlast, fx, fxlast, xt
        20090 FORMAT ('Ratio build x1, x2, fxl, fx2, res' / 5g13.6)
    ELSE
        xt = xnext
    ENDIF
    xlast = x
    x = xt
    fxlast = fx
100 CONTINUE
    WRITE (xstout, 20100) x, xnext, diff
    20100 FORMAT ('No closure after max iterations', 3g13.6)
    120 status = xfail
    GOTO 220
C
    200 CONTINUE
    status = xsucc
    220 RETURN
END
C  
PROGRAM NEWTON
INCLUDE 'CHASINC:PGMINC.FIN'
INTEGER N
REAL COEFS(10)
REAL START
REAL ROOT
INTEGER STATUS
INTEGER XSTAT, XVLUK(XPRDIM), DMSTAT
INTEGER XC001
INTEGER XC002
INTEGER XC003
CALL XRINIT (XVLUK, XPRDIM, X AoRt, XSTAT)
CALL XCV20M (XVLUK, STATUS)
IF (STATUS .NE. XSUCC) THEN
   CALL XQINTG (XVLUK, 'STATUS', 1, STATUS, XUPDAT, XSTAT)
   CALL XQOUT (XVLUK, XSTAT)
   GO TO 100
END IF
CALL XRINTG (XVLUK, 'N', 1, N, XC001, XSTAT)
CALL XRREAL (XVLUK, 'COEFS', 10, COEFS, XC002, XSTAT)
CALL XRREAL (XVLUK, 'START', 1, START, XC003, XSTAT)
CALL XQTNIT
CALL NEWTON (I, COEFS, START, ROOT, STATUS)
CALL XD2CV (XVLUK, DMSTAT)
IF (DMSTAT .NE. XSUCC) THEN
   CALL XQINTG (XVLUK, 'STATUS', 1, DMSTAT, XUPDAT, XSTAT)
   CALL XQOUT (XVLUK, XSTAT)
   GO TO 100
END IF
CALL XQREAL (XVLUK, 'ROOT', 1, ROOT, XUPDAT, XSTAT)
CALL XQINTG (XVLUK, 'STATUS', 1, STATUS, XUPDAT, XSTAT)
CALL XQOUT (XVLUK, XSTAT)
100 CONTINUE
END
SUBROUTINE EVAL (N, COEFS, X, FX, FDOTX, STATUS)
C .TITLE EVAL -- Evaluate a polynomial
C .HELP EVAL evaluates a REAL polynomial with 1 or more coefficients.
C .PAGE EVAL calculates the value of the polynomial and its first
derivative at the specified point.
C .PAGE EVAL will set STATUS to fail if the number of coefficients is
C less than 1
C .PAGE EVAL uses Horner's scheme to do the polynomial evaluations
C .VAR N INTEGER IN
C .TXT Number of coefficients in
C the polynomial
C .VAR COEFS REAL IN COUNT=1:10
C .TXT The coefficients of the
C polynomial.
C .DETAIL The coefficients are ordered so that the first entry has the
C highest exponent and the Nth entry has the 0th exponent.
C .VAR X REAL IN
C .TXT Where the polynomial is
C to be evaluated
C .VAR FX REAL OUT
C .TXT The value of the polynomial
C at X
C .VAR FDOTX REAL OUT
C .TXT The first derivative of the
C polynomial at X.
C .VAR STATUS INTEGER OUT
C .TXT Completion Status
C .END
C
C Change Log:
C 10 apr 85 -- create EVAL
C
C INCLUDE 'chas$inc\exsufa.fin'
C INCLUDE 'chas$inc\exunit.fin'
C
C INTEGER status
C
C DIMENSION coefs (n)
C--------
C WRITE (xstdout, 20000) x, n, (coefs (i), i=1,n)
C20000 FORMAT (' EVAL', $12.6, I5 / ($13.6))
C Exit with error if there are no coefficients
IF (n .LE. 0) THEN
  status = xfail
  RETURN
ENDIF
C
C Set the values to zero
FX = 0
C fdotx = 0
C Calculate FX
DO 100 i = 1, n, 1
   fx = fx * x + coefs (i)
100 CONTINUE
C WRITE (xstdout, 20100) fx
C20100 FORMAT (' FX =', s13.6)
C Calculate FDOTX
   IF (n .GT. 1) THEN
      DO 200 i = 1, n-1, 1
         fdotx = fdotx * x + coefs (i)
200 CONTINUE
C WRITE (xstdout, 20200) fdotx
C20200 FORMAT (' fdotx =', s13.6)
ENDIF
C
status = xsucc
RETURN
END
RUNTIME DATA MANAGEMENT
VARIABLE:

ARRAY:

MATRIX:

Up to 7 Dimensions

RECORD:

TREE:
### RDB Management Functions

<table>
<thead>
<tr>
<th>STRUCTURES</th>
<th>OBJECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>r</td>
</tr>
<tr>
<td>initial</td>
<td>x</td>
</tr>
<tr>
<td>save</td>
<td>x</td>
</tr>
<tr>
<td>restore</td>
<td>x</td>
</tr>
<tr>
<td>get</td>
<td>x</td>
</tr>
<tr>
<td>put</td>
<td>x</td>
</tr>
<tr>
<td>delete</td>
<td>x</td>
</tr>
<tr>
<td>create</td>
<td>x</td>
</tr>
<tr>
<td>move</td>
<td>-</td>
</tr>
<tr>
<td>push</td>
<td>-</td>
</tr>
<tr>
<td>pop</td>
<td>-</td>
</tr>
<tr>
<td>copy</td>
<td>x</td>
</tr>
<tr>
<td>rename</td>
<td>x</td>
</tr>
<tr>
<td>lock</td>
<td>x</td>
</tr>
<tr>
<td>unlock</td>
<td>x</td>
</tr>
<tr>
<td>invert</td>
<td>-</td>
</tr>
<tr>
<td>transpose</td>
<td>-</td>
</tr>
<tr>
<td>trace</td>
<td>-</td>
</tr>
<tr>
<td>add</td>
<td>-</td>
</tr>
<tr>
<td>subtract</td>
<td>-</td>
</tr>
<tr>
<td>multiply</td>
<td>-</td>
</tr>
<tr>
<td>divide</td>
<td>-</td>
</tr>
<tr>
<td>sin</td>
<td>-</td>
</tr>
<tr>
<td>cos</td>
<td>-</td>
</tr>
<tr>
<td>arctan</td>
<td>-</td>
</tr>
</tbody>
</table>

va = variable  
ar = array  
ma = matrix  
re = record  
tr = tree  
st = stack  
at = attributes  
ss = substructure  
rd = entire rdb
<table>
<thead>
<tr>
<th>CSC/SSD</th>
<th>TAEUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2GCHAS</td>
<td>6/4/85</td>
</tr>
</tbody>
</table>

**DEVELOPMENT SCHEDULE**
ROLE OF TAE

USER INTERFACE

USER

PDFS

MEC

MODULE EXECUTION CONTROL

MODULE MAIN

DT

DEVELOPER TOOLKIT

WR

REPORT WRITER + GRAPHICS

DM

DATA MANAGEMENT

BI

BATCH INTERFACE

RDB FILES

any Technology Module (Applications)
Developer Toolkit: The DEFINE-MODULE Operation.
SUBROUTINE NEWTON (N, COEFS, START, ROOT, STATUS)

C .TITLE
C   NEWTON -- Roots of a polynomial
C .HELP
C NEWTON find the real root of a REAL polynomial with 1 or more
C coefficients using NEWTON's method.
C
C NEWTON finds at most 1 root. If there are multiple roots, the
C value of START will determine which one is reached.
C
C NEWTON will set STATUS to fail if the number of coefficients is
C less than 1 or if the method does not yield a root within 10
C iterations
C .PAGE
C NEWTON calls EVAL to evaluate the polynomial
C .VAR  N   INTEGER IN
C .TXT  C Number of coefficients in
C the polynomial
C .VAR COEFS REAL IN COUNT=1:10
C .TXT  C The coefficients of the
C polynomial.
C .DETAIL
C The coefficients are ordered so that the first entry has the
C highest exponent and the Nth entry has the 0th exponent.
C .VAR START REAL IN
C .TXT  C Starting value for the
C approximation
C .VAR ROOT REAL OUT
C .TXT  C The value of the root
C .DETAIL
C ROOT will be zero if an error occurs
C .VAR STATUS INTEGER OUT
C .TXT  C Completion Status
C .END
C
C Change Log:
C 10 apr 85 -- create NEWTON
C
    INCLUDE 'chas$inc:exsufa.fin'
    INCLUDE 'chas$inc:exunit.fin'

C
C INTEGER status
C
C DIMENSION coefs (n)
C PROGRAM NEWTON
INCLUDE 'CHASING.PUBLISH.FIN'
INTEGER N
REAL COEFS(40)
REAL START
REAL KOUT
INTEGER STATUS
INTEGER XSTAT, XVALUK, XPRIUIM, DMSTAT
INTEGER XCO01
INTEGER XCO02
INTEGER XCO03
CALL XRINIM (XVALUK, XPRIUIM, XABORT, XSTAT)
CALL XCV2UM (XVALUK, STATUS)
IF (STATUS .NE. XSUCC) THEN
  CALL XQINTG (XVALUK, 'STATUS', 1, STATUS, XUPDAT, XSTAT)
  CALL XQUOT (XVALUK, XSTAT)
  GO TO 100
END IF
CALL XRINTG (XVALUK, 'N', 1, N, XCO01, XSTAT)
CALL XRREAL (XVALUK, 'COEFS', 10, COEFS, XCO02, XSTAT)
CALL XRREAL (XVALUK, 'START', 1, START, XCO03, XSTAT)
CALL XQINIT
CALL NEATON (1, "COEFS", START, KOUT, STATUS)
CALL XDN2CL (XVALUK, UMSTAT)
IF (DMSTAT .NE. XSUC) THEN
  CALL XQINTG (XVALUK, 'STATUS', 1, DMSTAT, XUPDAT, XSTAT)
  CALL XQUOT (XVALUK, XSTAT)
  GO TO 100
END IF
CALL XRREAL (XVALUK, 'KOUT', 1, KOUT, XUPDAT, XSTAT)
CALL XQINTG (XVALUK, 'STATUS', 1, STATUS, XUPDAT, XSTAT)
CALL XQUOT (XVALUK, XSTAT)
100 CONTINUE
END
### Structures

<table>
<thead>
<tr>
<th>v a m r t s</th>
<th>a r a c r t</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial</td>
<td>x x x x x x</td>
</tr>
<tr>
<td>save</td>
<td>x x x x x x</td>
</tr>
<tr>
<td>restore</td>
<td>x x x x x x</td>
</tr>
<tr>
<td>get</td>
<td>x x x x x x</td>
</tr>
<tr>
<td>put</td>
<td>x x x x x x</td>
</tr>
<tr>
<td>delete</td>
<td>x x x x x x</td>
</tr>
<tr>
<td>create</td>
<td>x x x x x x</td>
</tr>
<tr>
<td>move</td>
<td>- - - - - - x</td>
</tr>
<tr>
<td>push</td>
<td>- - - - - - x</td>
</tr>
<tr>
<td>pop</td>
<td>- - - - - - x</td>
</tr>
<tr>
<td>copy</td>
<td>x x x x x x</td>
</tr>
<tr>
<td>rename</td>
<td>x x x x x x</td>
</tr>
<tr>
<td>lock</td>
<td>x x x x - -</td>
</tr>
<tr>
<td>unlock</td>
<td>x x x x - -</td>
</tr>
<tr>
<td>invert</td>
<td>- - x - - -</td>
</tr>
<tr>
<td>transpose</td>
<td>- - x - - -</td>
</tr>
<tr>
<td>trace</td>
<td>- - x - - -</td>
</tr>
<tr>
<td>add</td>
<td>- x x - - -</td>
</tr>
<tr>
<td>subtract</td>
<td>- x x - - -</td>
</tr>
<tr>
<td>multiply</td>
<td>- x x - - -</td>
</tr>
<tr>
<td>divide</td>
<td>- x x - - -</td>
</tr>
<tr>
<td>sin</td>
<td>- x x - - -</td>
</tr>
<tr>
<td>cos</td>
<td>- x x - - -</td>
</tr>
<tr>
<td>arctan</td>
<td>- x x - - -</td>
</tr>
</tbody>
</table>

### Objects

<table>
<thead>
<tr>
<th>a s r t s</th>
<th>a s r t s</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial</td>
<td>x</td>
</tr>
<tr>
<td>save</td>
<td>x</td>
</tr>
<tr>
<td>restore</td>
<td>x</td>
</tr>
<tr>
<td>get</td>
<td>x</td>
</tr>
<tr>
<td>put</td>
<td>x</td>
</tr>
<tr>
<td>delete</td>
<td>x</td>
</tr>
<tr>
<td>create</td>
<td>x</td>
</tr>
<tr>
<td>move</td>
<td>-</td>
</tr>
<tr>
<td>push</td>
<td>-</td>
</tr>
<tr>
<td>pop</td>
<td>-</td>
</tr>
<tr>
<td>copy</td>
<td>x</td>
</tr>
<tr>
<td>rename</td>
<td>x</td>
</tr>
<tr>
<td>lock</td>
<td>x</td>
</tr>
<tr>
<td>unlock</td>
<td>x</td>
</tr>
<tr>
<td>invert</td>
<td>-</td>
</tr>
<tr>
<td>transpose</td>
<td>-</td>
</tr>
<tr>
<td>trace</td>
<td>-</td>
</tr>
<tr>
<td>add</td>
<td>-</td>
</tr>
<tr>
<td>subtract</td>
<td>-</td>
</tr>
<tr>
<td>multiply</td>
<td>-</td>
</tr>
<tr>
<td>divide</td>
<td>-</td>
</tr>
<tr>
<td>sin</td>
<td>-</td>
</tr>
<tr>
<td>cos</td>
<td>-</td>
</tr>
<tr>
<td>arctan</td>
<td>-</td>
</tr>
</tbody>
</table>

va = variable  
ar = array  
ma = matrix  
re = record  
tr = tree  
st = stack  
at = attributes  
ss = substructure  
rd = entire rdb
Use of TAE to Facilitate Color Film Generation

Introduction:

Traditionally, digital imagery has been recorded on black and white frames of film which are composited under red, green, and blue filters to form a color composite. The EROS Data Center has purchased a color film recorder, the Color FIRE to automate color film generation. Implementation of TAE as an interface to the color FIRE demonstrates many TAE capabilities. The development of the interface can be broken down into three broad topics.

1. Hardware capabilities of the color FIRE
2. Software requirements defined by the EROS Data Center
3. Development of the TAE modules which form the interface

I. Characteristics of the color FIRE

A. High speed digital film recorder
B. Accepts several types of input image band interleaving
   1. Band interleaved by pixel (BIP)
   2. Band interleaved by line (BIL)
   3. Band sequential (BSQ)
C. Capable of recording images in various pixel spot sizes
D. An image is recorded in one of two modes
   1. Continuous Mode
      a. Images are produced in a real-time environment
      b. Film recorder exposes film at a continuous rate
      c. Required to supply data at a constant line rate (265K per sec)
      d. Input image is required to be BSQ
   2. Triggered Mode
      a. Images are produced in a time share environment
      b. Film recorder exposes film only after data is received
E. Allows eight user defined formats to select image band interleaving, triggered or continuous mode, and spot size of the image to be recorded

II. EDC requirements

A. All images processed from tape
   1. Finite disk space
   2. Disk contention
B. Require images to be BIL and BSQ images
C. Define pixel spot sizes to be in the range of 20.0 - 100.0 micrometers
D. Allow user to select combination of calibration, control, and enhancement LUT files at run-time
E. Two modes of operation
   1. Production mode for fast throughput
   2. Custom mode to process individual requests with annotation included
F. Basic flow of film generation for custom mode
   1. User submits request to generate film
   2. Retrieve image tapes from library as specified by request form
   3. Execute TAE proc to enter image parameters into custom request file
   4. If required, create or modify enhancement, calibration, and control LUT files
   5. Execute TAE proc to read requests from custom request file and expose images onto film. Assessment sheet created for each frame.
   6. Develop roll of film
   7. Inspect film and complete assessment sheets
   8. If film failed inspection
      a. Modify request within custom request file
      b. Reprocess image
   9. If film passed inspection
a. Delete request from Custom Request File  
b. Return developed film to the user  

III. Modules required to satisfy EDC requirements  

A. SETPPARM  
1. Create a configuration file (parameter file for production mode)  
2. Parameter set PDF (not an executable proc) which creates a parameter block on disk  
3. Procedure for creating a configuration file  
a. Tutor the SETPPARM pdf  
b. In tutor mode, assign values to the image parameters  
c. Execute the tutor "SAVE" command, creating a disk file  
4. The file created is accessed by proc MAKEFILM in production mode  

B. PARMENTR  
1. Manipulate the custom request file  
2. Structure of each request within the custom request file  
a. Image parameters including annotation fields  
b. Process flag indicating if the request is to be processed by proc MAKEFILM in custom mode  
c. Priority scheme  
   1. Each request assigned a priority of 1 - 9 (1 = high priority, 9 = low priority)  
   2. Requests with equal priority processed First In First Out  
3. Consists of three subcommands  
a. ADD  
   1. Add an image request to the custom request file in the appropriate priority list  
   2. Creates a sequence number to identify the request  
b. -MODIFY  
   1. Modify an image request within the custom request file  
   2. Inputs sequence number of the request to be modified  
   3. Dynamic tutor session is initiated to modify the request  
      a. Default values of the TAE parameters are the current values of the image request  
      b. All parameter values may be modified including the priority and the process flag  
c. -DELETE  
   1. Delete up to 20 requests from the custom request file  
   2. Inputs sequence numbers of the requests to be deleted  

C. SETFILES  
1. Create or modify enhancement, calibration, and control LUT files  
2. Interactive routine which uses TAE terminal I/O routines to edit the files  
3. LUT Definitions  
a. Calibration file represents the actual film density produced by the film recorder when a brightness value is exposed onto film.  
b. Control file represents the desired film density when a brightness value is exposed onto film.  
c. Enhancement file creates a contrast enhancement of the image data  
d. Files are mathematically combined at run-time to form a LUT which has the effect of applying the enhancement table to the raw image data, and then taking the enhanced image and forcing it to an appropriate film density  

D. MAKEFILM  
1. Expose images onto film  
2. Consists of two subcommands  
a. -PROD  
   1. Process images in production mode  
   2. Algorithm of production subcommand
a. Inputs
1. TAE parameters
   a. Physical tape drive name(s)
   b. Configuration file name
   c. Number of images to be processed
2. Configuration file created by proc SETPPARM
3. Enhancement, calibration, and control LUT files as specified within the configuration file

b. Output
1. Exposed film on the color FIRE
2. Detailed log file

3. Characteristics of production subcommand
   a. Annotation is not generated
   b. Image is not centered vertically
   c. All images of a film session use the same configuration file, and hence also use same LUT files

b. -CUSTOM
1. Process individual image requests
2. Algorithm of custom mode
   a. Input
      1. TAE parameters
         a. Physical image tape drive name
         b. Physical annotation tape drive name
      2. Custom request file
      3. Enhancement, calibration, and control LUT files as specified by the custom request
   b. Annotation is rasterized onto frame of film
      1. Title annotation and bottom annotation block
      2. Annotation is written to a scratch tape
      3. Annotation tape and image tape are synchnorized for data transfer to the color FIRE
   c. Output
      1. Exposed film on the color FIRE
      2. Printed assessment sheets
      3. Detailed log file

3. Characteristics of the custom subcommand
   a. Image is centered onto frame of film
   b. Each request may have a unique set of parameters

E. PREPORT
1. Report printing facility
2. Consists of six subcommands
   a. -LOGFILE print the accumulated log file
   b. -CONFIG print contents of specified configuration file
   c. -CREQ print custom request file
   d. -FDESCRPT print one line LUT file descriptions
   e. -CONTENT print map points of a specified LUT file
   f. -COMULUT compute the LUT of a specified calibration and control file

IV. Generated output of MAKEFILM
A. Assessment Sheet
   1. Contains the date
   2. Unique frame ID
   3. Image parameters

B. Image produced in custom mode
   1. Centered image
   2. Top annotation
   3. Bottom annotation

V. TAE Environment
A. Uniformity
1. Menus help combine related procs
2. Standardize user interface to application programs

B. User Features
1. On-line help
   a. Help sessions for procs, parameters, and TAE commands
   b. Message utility for error messages
2. Access procs from TAE menu, tutor, or command mode
   a. TAE designed for the first time user and the experienced user
   b. Submit batch and asynchronous jobs
3. All input parameters may appear on the screen
4. Parameter values may be saved for later user

C. TAE programming assistance
1. Parameter validation
2. Procs may reduce amount of executable code
   a. Parameter set PDF's
   b. Procedures may use TAE and host computer commands

Conclusion:
In developing the color film recorder interface, many TAE capabilities have been implemented. By using TAE as the interface, we are able to efficiently automate color film generation and still maintain a friendly user environment. Most importantly, this software system demonstrates the usefulness of TAE as an interface to specialized hardware drivers running in a real-time environment.
HELPME: Providing Expert Help from TAE

B. E. Lowrey and P. B. Pease
NASA/Goddard Space Flight Center
Space Data and Computing Division
Greenbelt, Maryland 20771
(301) 344-9513

Abstract

On-line help is generally the fastest method of information retrieval for a user at a terminal. This paper describes a new technique that asks the user to state his/her objective and responds with the name(s) of the function(s) that may satisfy the user's objective. The technique allows the user to interface with a "natural language" of limited vocabulary. The technique implemented is a Transportable Applications Executive (TAE) procedure named HELPME. This procedure will be both effective in helping orient and train the novice and efficient in assisting the expert to recover the exact name of a desired function. The Land Analysis System (LAS) is selected for a demonstration of the technique.

Introduction

Prompt on-line user assistance is an extremely valuable mode of obtaining information about a software system. Generally, the computer can select and present information faster than a person can retrieve a document, read the table of contents, and find the appropriate page. Prompt responses minimize the tendency of the human mind to be distracted from the task to be performed.

For these reasons, the TAE is designed to make considerable on-line information available to the user. Information about a function in a TAE software system is given when the user types TAE - HELP FUNCTION. The user can then choose to tutor on the function TAE - TUTOR FUNCTION. The tutor mode supplies a brief description of the function, a list of input parameters (PARMS), and instructions on how to operate the function. A brief description of each PARM is available, along with the default value, if set. A second level of PARM help is available at the user's request, with a more detailed description of the PARM and its characteristics.

There are two current methods by which TAE helps users locate a function, menus, and PROCS (a function which supplies an alphabetical list of functions on the application system). The primary method for a user to locate an unknown TAE function is to proceed through a series of menus, until the menu level which defines the function is reached. A thoughtfully constructed menu system can assist the novice in understanding the software system and in locating the desired function. However, more complex software systems require many levels of menus. Depending on the user's ability to understand the menus and the amount of activity on the computer, this may be a slow process. It is possible to take a wrong path on a menu series, and end with no usable
functions, a frustrating experience both for novice and expert. Further, an expert who needs only the exact name of a function, is apt to consider a multilevel menu unsatisfactory. Although a number of iterations were made in developing the menu structure for the Land Analysis System (LAS), there are still reports of users who have spent several minutes searching through menus for a function and have given up in disgust or believing that the function does not exist. As more programs are added, the time to find a program using menus will increase.

A second alternative is to get an alphabetical listing of all programs via the PROCS TAE utility program. Searching through pages of an alphabetical list of programs is also a lengthy process, which gets even longer as new functions are added.

This paper proposes an additional method of on-line assistance that supplies a satisfactory function name more quickly than a menu-based system. The basic technique is to give the user a chance to state his or her objective, and then have the computer select the function(s) that allow the user to achieve his or her objective.

Description of the HELPME Procedure

The general expression used by a user wishing to accomplish an objective is "do something". Syntactically, most statements of objectives can be expressed in minimum form as a verb followed by a noun, either as a direct object or as an indirect object. Thus, the strategy employed in this paper was to give a list of objects, or nouns, on the image processing system, and a list of actions, or verbs, to be performed on the objects. The general form of an untutored TAE command is:

```
TAE > FUNCTION PARMI PARM2 (...PARMN)
```

(where the underlined characters are the computer prompt). The command syntax translates into a primitive English syntax by giving an imperative or interrogatory form to the TAE function name. The imperative HELPME is a tidy six-letter function, with high mnemonic value, as it closely corresponds to the universally used HELP (or H) for on-line user assistance. The HELPME command may express the user's feelings! The untutored input is then HELPME VERB NOUN. An example of user input is then HELPME MAKE IMAGE

```
TAE > HELPME MAKE IMAGE
```

(where the input parameters, VERB and NOUN, are allowed only a limited number of values; a choice enforced by the VALID feature of the TAE PROC). The user may tutor on HELPME and thus obtain a list of the VALID verbs and nouns. The level two HELP supplies a list of synonyms for the VALID verbs and nouns. The HELPME procedure calls a Fortran program, which first finds two lists of functions, one that satisfies the input VERB and the second that satisfies the input NOUN. A logical AND is performed on the two functions; only those functions that are in the lists of both VERB and NOUN are output to the user.
The method is implemented under TAE on the LAS, an image processing system built by the Space Data and Computing Division. The HELPME program is expected to be the quickest and most reliable method of finding the name of a desired function, and in a system as large as the LAS should prove to be very beneficial to users.

Utility for Implementing HELPME: the MAKELIST Program

The HELPME program outputs a list of functions that satisfy the given input NOUN and VERB. These functions are selected from the total set of functions in the applications system by the subroutines NOUNTAB and VERBTAB. These subroutines employ the logic IF (NOUN is true) THEN (return subset list of functions). If the applications system served by HELPME is large, the establishment of the lists of functions applicable to each requested NOUN or VERB is time-consuming, and maintenance of the lists is tedious, mainly because Fortran character handling is clumsy.

It was, therefore, decided that the establishment and maintenance of the lists of functions could be streamlined by creating a utility program. The program MAKELIST was developed in order to have a utility capable of generating either the subroutines NOUNTAB or VERBTAB. By operating MAKELIST under TAE, it is possible to develop and save the lists of functions.

Implementation of HELPME on the Land Analysis System

The Land Analysis System (LAS) is a powerful and extensive image processing and analysis system developed by the Space Data and Computing Division of the NASA Goddard Space Flight Center. The LAS consists of about 250 application programs running under the TAE on a Digital Equipment Corporation (DEC) VAX-11/780. LAS was developed initially to assess the performance of the Thematic Mapper (TM) sensor, which was flown on the Landsat-4 and Landsat-5 satellites. It has been extended to become a general image analysis system. Further upgrades to the LAS are planned, which will make it more easily transported to other computers and to include still more application functions.

With a system of the power and complexity of the LAS, locating a desired function can be time-consuming. For example, the menu tree for LAS consists of 46 branches, the alphabetical list from PROCS is 16 screens long, and the User's Guide is about two inches thick. Also, the names of functions are historic in origin and this has resulted in some inconsistency in the names, which adds to the difficulty of locating a desired function. In a system as large as the LAS, the HELPME program is anticipated to be the quickest method of finding the name of a desired function and should prove to be very beneficial to users.
In implementing HELPME on the LAS, the following set of NOUNS and VERBS was selected (the synonyms are those given in the TAE help):

<table>
<thead>
<tr>
<th>VERB List</th>
<th>Synonyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALC</td>
<td>Calculate, Compute, Count, Transform, Mask</td>
</tr>
<tr>
<td>MAKE</td>
<td>Create, Generate, Combine, Train</td>
</tr>
<tr>
<td>CLASSIFY</td>
<td>Theme, Cluster</td>
</tr>
<tr>
<td>LIST</td>
<td>See, View, Display, Print, Plot</td>
</tr>
<tr>
<td>COPY</td>
<td>Move, Print, Subdivide, Save, Write</td>
</tr>
<tr>
<td>STRETCH</td>
<td>Zoom, Contrast, Expand, Contract</td>
</tr>
<tr>
<td>DELETE</td>
<td>Drop, Erase, Remove</td>
</tr>
<tr>
<td>EDIT</td>
<td>Alter, Change, Substitute</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOUN List</th>
<th>Synonyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGE</td>
<td>Pixels, Whole Image</td>
</tr>
<tr>
<td>CATALOG</td>
<td>TAE Catalog</td>
</tr>
<tr>
<td>SITE</td>
<td>Polygon</td>
</tr>
<tr>
<td>STAT</td>
<td>Histograms, Mean, Standard Deviation, Covariances, Inventory, Transforms, Masks</td>
</tr>
<tr>
<td>FILTER</td>
<td>Weights</td>
</tr>
<tr>
<td>LUT</td>
<td>Radiometric Lookup Tables</td>
</tr>
<tr>
<td>LABEL</td>
<td>Header for an Image, History of an Image</td>
</tr>
<tr>
<td>GROUP</td>
<td>File Group</td>
</tr>
</tbody>
</table>

These lists contain eight NOUNS and eight VERBS, a set which allows 64 unique combinations. If the 250 LAS functions were distributed evenly, and if there were no overlap combinations, each request would yield an average of four functions. However, the NOUN "IMAGE" is applicable to a large proportion of the functions; the combinations CALC IMAGE and MAKE IMAGE result in a lengthy list. Increasing the number of NOUNS are VERBS will result in a more cluttered tutor screen, and will increase the number of NOUNS and VERBS for the command mode user to memorize. The pinpointing of requested functions in these categories requires a corresponding increase in complexity. A large number of functions may be all right for the expert who is only seeking the spelling of the function; however, the novice may be overwhelmed. On the other hand, the novice user of a computational function will likely need to consult the User's Guide to understand the calculation properly.

In selecting the NOUNS and VERBS to use for HELPME, several issues arose. First, the NOUN "IMAGE" is applicable to most of the applications; as such, the limit of 50 items allowable for a TAE vector parameter was reached in using MAKELIST to create the NOUN table. Because of this condition, the current implementation of HELPME does not cover all of the applications that exist in LAS. A warning notice to users of HELPME is displayed that states there may be other functions that fit the NOUN & VERB entered that are not listed in the output. A technique to work around this limitation is under development for the LAS system. While the warning message helps to prevent the user from assuming that if the function being looked for does not appear
in the HELPME output, it must not exist, it does limit the usefulness of HELPME on the LAS. The second problem resulting from having a NOUN (or VERB) that is applicable to a large number of functions is that the resulting output may be a large number of matches and thus requires of the user considerable searching to find the one function wanted.

There is an issue concerning the use of MAKELIST to add another application function. While the TAE SAVE and RESTORE commands can be used to recover what was previously entered, adding to the list can be done only by appending the new name to the bottom of the list; this loses the alphabetical order of the output. It is planned to add an alphabetic sorting routine to the MAKELIST utility in the near future.

Another consideration in implementing the NOUN and VERB tables is that some functions may fit more than one NOUN or VERB. For example, the function COPY will copy an image or will copy a group of images; therefore copy should appear in the NOUN table for both the IMAGE and GROUP categories.

The larger the number of functions in a system, the more useful HELPME should be; but at the same time, the selection of the NOUNs and VERBs is more difficult. This situation is not unique to HELPME; in fact, it is probably easier than constructing a good set of menus.

In order to improve the usefulness of HELPME on LAS, the usage of the HELPME will be monitored and users will be surveyed to gain information about how often it is used and whether the user obtained useful results. The evolution of TAE has demonstrated that user feedback is important, and that different users prefer different interaction modes on a system. The LAS in particular contains more knowledge than any one individual is likely to acquire. Therefore, it is important that users communicate their expectations of HELPME in order to achieve the most helpful HELPME.

Discussion

The new version (1.3) of TAE provides several new features which will be considered for further development of the HELPME technique. The new version allows PARMs to be qualified; it may be that implementing HELPME on a large system would be improved with the judicious selection of qualifiers.

Other features worth consideration for a large system are the use of a larger set of VALID values for NOUN and VERB; and the use of subcommands for more precise selection of functions.

A very desirable extension is to add the titles of functions to the function names supplied by the user. This could be done automatically by means of a utility involving the output of the TAE command PROCS. If the technique proves useful, TAE could be enhanced to provide that a PDF or HLP file allow the addition of a NOUN or VERB command in the file followed by a list of NOUNs and VERBs that are applicable to that function. Then the system would return that function if an applicable combination of NOUN and VERB were added.
The LAS implementation was begun well after the system had matured; and even after the upgrade of LAS was well underway. Most definitely, the selection of NOUN(s) and VERB(s) applicable to a function would be more efficient if done as a part of the design and analysis of a function. End-user input on the selection of NOUNs and VERBs with which to view a system would be a value at the design of a system. Further user interaction after the delivery of a system may result in some additions or adjustments to the HELPME response.

Care should be taken that the HELPME PROC does not become overloaded with features that consume machine time. Prompt response is essential.

Conclusion

The HELPME technique can be applied to a wide variety of software systems. Several of the verbs chosen for the LAS system are generic to almost any software system: EDIT, MAKE, DELETE, LIST, and COPY apply to word processing, data base management, or graphic packages. Modern computers allow the installation of Commands or Procedures which accept input parameters, so that the use of TAE, while helpful, is not a requirement for implementation of HELPME.

The list-oriented programming for HELPME is manageable in Fortran, but an artificial intelligence (AI) language such as LISP or PROLOG would definitely streamline the implementation. In the absence of an AI language, a utility is helpful. The present utility MAKELIST, can be augmented by the use of an alphabetic sorting subroutine. A large system under TAE may be desirable to redesign MAKELIST to take input from a file created from output from the TAE procedure names PROCS. This would supply the titles along with selected TAE functions.

The HELPME technique is potentially useful to applications programmers dealing with development in a complex programming environment. As the use of standardized modules has increased in applications programming, the programmer needs to know the names of the subroutines, and the name, meaning, type, and position of the calling arguments for the subroutines. More on-line help is definitely efficient; the HELPME syntax can enable the programmer to pinpoint the routine needed and receive sufficient information to call it without the lengthy interruption needed to consult a manual or a colleague.

Finally, HELPME views an applications software system as a collection of actions and objects. This view may be useful in the design of systems.
Appendix A: HELPME Program and Subroutines

The program structure is shown in Figure 1. Detailed comments are available in the program listing. The versions of NOUNTAB and VERBTAB are exemplary; they are truncated to limit the size of the listing.

This is written in VAX-augmented Fortran 77 using TAE. If the user is in a different environment, these changes are required.

Non-Vax users:

1. Length of subroutine names and variable names may need to be truncated to six characters

2. Exclamation point (!) is used for comments

3. A "D" in Column 1 is used for additional writes for debugging purposes

Non-TAE users:

1. TAE calls are isolated in subroutine GETPARM. This subroutine may be replaced with an equivalent routine for input. Such a routine requires user prompt for input VERB and NOUN. The VERB and NOUN should be read and checked for capitalization and for leading blanks or following blanks; the received strings of the VERB or NOUN must be made identical to the strings as set in the program.

Fortran-66 users:

Because of the extensive use of character data, a redesign of the program is required.

2. The TAE files qualified "PDF" and HLP are not used.
Figure 1. Structure and Data Flow of the HELPME program
HELPME.PDF

PROCESS
PARM NAME=VERB TYPE=STRING VALID=(LIST, DELETE, MAKE, +
    COPY, EDIT, STRETCH, CLASSIFY, CALC)
PARM NAME=NOUN TYPE=STRING VALID=(IMAGE, GROUP, LABEL, LUT, +
    STAT, FILTER, CATALOG, SITE, HISTOGRAM)
END-PROC
User assistance in selecting an LAS routine using "natural language".

HELPME provides on-line assistance to the user who needs to find the name of a routine to execute a task. It has a limited-vocabulary natural language, consisting of a few nouns and verbs, and is intended to provide quick assistance either to the novice or to the experienced user who has forgotten the spelling of a routine.

It does not replace the users' guide, and does NOT contain a comprehensive catalog of all routines on the LAS system.

This routine has the syntax "HELPME VERB NOUN". For example, the user might say "HELPME MAKE IMAGE". The tutor provides a list of synonyms for the allowed VERBS and NOUNS.

Allowed verbs are:
LIST, DELETE, MAKE, COPY, EDIT, STRETCH, CLASSIFY, CALC.

Allowed nouns are:
IMAGE, GROUP, LABEL, LUT, STAT, FILTER, CATALOG, SITE, HISTOGRAM.

A null result does not mean that the routine does not exist — check the LAS Users' Guide. In particular, the IMAGE functions are exemplary, rather than comprehensive.

Synonyms
LIST - See, View, Display, Print, Plot
DELETE - Drop, Erase, Remove
MAKE - Create, Generate, Combine, Train
COPY - Move, Print, Subdivide, Save, Write
EDIT - Alter, Substitute, Change
STRETCH - Zoom, Contrast, Expand, Contract
CLASSIFY - Algorithms associated with Image Classification
CALC - Calculate, Compute, Count, Transform, Mask
(Not: Families of routines may be represented by one member.)
Definitions
IMAGE - Pixels, Whole Image
GROUP - File Group
LABEL - Header for an Image, History of an Image
LUT - Lookup Tables for Radiometric Changes
STAT - Histograms, Mean, Standard Deviation, Covariances,
       Inventory, Transforms, Masks
FILTER - Edge Detector, Gradient, Weighting Function
CATALOG - TAE Catalog
SITE - Polygonal area in an image
HISTOGRAM - Frequency plot of an image or class or site therein.
.END
C******************************************************
C
Name - HELPME
C
Language - FORTRAN 77 (VAX-Augmented) Type - MAIN PROGRAM
C
Version 1.0 Date - 27MAR85 Programmer - B. Lowrey Code 633 GSFC
C
Function - Implements a User Request to find a satisfactory
Function on LAS upon User specification of an
Operator (VERB) and Operand (NOUN)
C
Parameters -
Variable Type I/O Brief Description
------------- ------- ------- -------------------------------
VERB C*9 I Input Specification
NOUN C*9 I Input Specification
VERB_TABLE(50) C*9 I/O Functions that satisfy input VERB
NOUN_TABLE(50) C*9 I/O Functions that satisfy input NOUN
MAX_VERB I I/O # functions in VERB_TABLE
MAX_NOUN I I/O # functions in NOUN_TABLE
ANS_TABLE(50) C*9 I/O Functions that satisfy both input NOUN AND input VERB
MAX_ANS I I/O # of Functions in ANS_TABLE

Subprograms Called - GETPARM, VERBTAB, NOUNTAB, COMPARE

C* Program Description for HELPME
C
Read input NOUN and VERB (using TAE)
C Obtain a Table containing Functions that are under the domain of the NOUN
C Obtain a Table containing Functions that are under the domain of the VERB
C Find the Table which is the Union of the Two Tables
C Output the Union to the User on-line
C******************************************************
PROGRAM HELPME

C Purpose: Assist a user to find a needed LAS Function by returning one
C or more LAS Functions that satisfy a noun and a verb combination
C selected by the user.

CHARACTER*9 NOUN_TABLE(50),VERB_TABLE(50)
CHARACTER*9 VERB,NOUN
CHARACTER*9 ANS_TABLE(50)

C Get Input (FROM TAE)

CALL GETPARM(NOUN,VERB)
WRITE(6,1020) VERB,NOUN

C Get a Table of Functions that satisfy the NOUN

CALL NOUNTAB(NOUN,NOUN_TABLE,MAX_NOUN)
WRITE(6,2001) (NOUN_TABLE(I),I=1,MAX_NOUN)
WRITE(6,2002) MAX_NOUN

C Get a Table of Functions that satisfy the VERB

CALL VERBTAB(VERB,VERB_TABLE,MAX_VERB)
WRITE(6,2003) (VERB_TABLE(I),I=1,MAX_VERB)
WRITE(6,2004) MAX_VERB

C Compare NOUN Table and VERB table (.AND.)

CALL COMPARE(NOUN_TABLE,MAX_NOUN,VERBTABLE,MAXVERB,ANSTABLE,
+ MAX_ANS)

C Output Selected Functions to User

IF (MAX_ANS.EQ.0) THEN
   WRITE (6,1002) !No Functions
ELSE
   WRITE (6,1001) MAX_ANS
   WRITE(6,1000) (ANS_TABLE(I),I=1,MAX_ANS)
ENDIF

C Formats to write to user terminals

1000 FORMAT(5X,A9)
1001 FORMAT('' A LIST OF ',I4,' FUNCTIONS THAT MAY MATCH YOUR NEEDS
+ IS PROVIDED:''
1002 FORMAT('' NO FUNCTIONS WERE FOUND THAT MATCH YOUR REQUEST.',/,
+ '/','' SEE THE LAS USER'S GUIDE FOR MORE COMPLETE INFORMATION.''
1020 FORMAT ('' VERB=',A9,' NOUN=',A9)

C DEBUG FORMATS

2001 FORMAT ('' NOUN_TABLE=',6(5A10,/))
2002 FORMAT ('' MAX_NOUN=',I5)
2003 FORMAT ('' VERB_TABLE=',6(5A10,/))
2004 FORMAT ('' MAX_VERB=',I5)
STOP
END
Name - GETPARM

Language - FORTRAN 77 (VAX-Augmented) Type - SUBROUTINE

Version 1.0 Date - 27MAR85 Programmer - B. Lowrey Code 633 GSFC

Function - Obtain Input Parameters using TAE

Parameters -

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>I/O</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOUN</td>
<td>C*9</td>
<td>I/O</td>
<td>Input Specification</td>
</tr>
<tr>
<td>VERB</td>
<td>C*9</td>
<td>I/O</td>
<td>Input Specification</td>
</tr>
</tbody>
</table>

Subprograms Called - XWINIM, XWINIT, XRSTR

Program Description -

- Calls TAE routines to initialize the BLOCK and check
- Calls TAE routine to find the value of VERB in BLOCK
- Calls TAE routine to find the value of NOUN in BLOCK
SUBROUTINE GETPARM(NOUN, VERB)
INCLUDE 'TAESINC:PGMINC.FIN/NOLIST'
CHARACTER*9 NOUN, VERB
INTEGER BLOCKK(XPRDIM)

C******************************************************************************
C OBTAIN INPUT PARAMETERS FROM TAE & CHECK
C******************************************************************************
CALL XRINIM(BLOCKK,XPRDIM,XABORT,ISTAT)
CALL XBINIT(ISTAT)
CALL XRSTR(BLOCKK,"VERB",1,VERB,ILEN,ICNT,STATUS)
CALL XRSTR(BLOCKK,"NOUN",1,NOUN,ILEN,ICNT,STATUS)
RETURN
END
SUBROUTINE NOUNTAB(NOUN,NOUN_TABLE,MAX_NOUN)
CHARACTER*9 NOUN_TABLE(50)
CHARACTER*9 NOUN
IF (NOUN.EQ.'IMAGE') THEN
MAX_NOUN=24
NOUN_TABLE(1)(1:6 )='ADDPIC'
NOUN_TABLE(2)(1:5 )='BAYES'
NOUN_TABLE(3)(1:5 )='CANAL'
NOUN_TABLE(4)(1:7 )='COMPARE'
NOUN_TABLE(5)(1:7 )='COMPLEX'
NOUN_TABLE(6)(1:6 )='COMPOL'
NOUN_TABLE(7)(1:6 )='CONCAT'
NOUN_TABLE(8)(1:8 )='CONVOLVE'
NOUN_TABLE(9)(1:4 )='COPY'
NOUN_TABLE(10)(1:4 )='FILM'
NOUN_TABLE(11)(1:6 )='HISTEQ'
NOUN_TABLE(12)(1:8 )='ISOCLASS'
NOUN_TABLE(13)(1:6 )='KARLOV'
NOUN_TABLE(14)(1:6 )='LASDEL'
NOUN_TABLE(15)(1:8 )='LINEPLOT'
NOUN_TABLE(16)(1:4 )='LIST'
NOUN_TABLE(17)(1:7 )='MINDIST'
NOUN_TABLE(18)(1:8 )='MULTIPLY'
NOUN_TABLE(19)(1:7 )='MULTPIC'
NOUN_TABLE(20)(1:8 )='NEIGHBOR'
NOUN_TABLE(21)(1:5 )='NOISE'
NOUN_TABLE(22)(1:7 )='TESTGEN'
NOUN_TABLE(23)(1:6 )='XORPIC'
NOUN_TABLE(24)(1:4 )='ZOOM'
ENDIF
IF (NOUN.EQ.'LUT') THEN
MAX_NOUN=3
NOUN_TABLE(1)(1:6 )='DSPRLT'
NOUN_TABLE(2)(1:6 )='DELLUT'
NOUN_TABLE(3)(1:6 )='DSPLUT'
ENDIF
IF (NOUN.EQ.'LABEL') THEN
MAX_NOUN=2
NOUN_TABLE(1)(1:9 )='DSPHISTRY'
NOUN_TABLE(2)(1:6 )='DSPLBL'
ENDIF
IF (NOUN.EQ.'GROUP') THEN
MAX_NOUN=3
NOUN_TABLE(1)(1:5 )='GROUP'
NOUN_TABLE(2)(1:8 )='DSPGROUP'
NOUN_TABLE(3)(1:8 )='DELGROUP'
ENDIF
RETURN
END
*** VERBTAB.FOR ***

**SUBROUTINE** VERBTAB(VERB, VERB_TABLE, MAX_VERB)

**NOTE**: this subroutine produced by MAKELIST program

**NOTICE!!! This example is shortened for publication purposes.**

**SUBROUTINE** VERBTAB(VERB, VERB_TABLE, MAX_VERB)

**CHARACTER*9 VERB Table(50)**

**IF (VERB.EQ."MAKE") THEN**

```
MAX_VERB=14
VERB_TABLE(1)(1:6)='CONCAT'
VERB_TABLE(2)(1:6)='CWTGEN'
VERB_TABLE(3)(1:4)='FILM'
VERB_TABLE(4)(1:5)='GROUP'
VERB_TABLE(5)(1:7)='INSERT2'
VERB_TABLE(6)(1:6)='KARLOV'
VERB_TABLE(7)(1:7)='LINEOFF'
VERB_TABLE(8)(1:8)='MAGPHASE'
VERB_TABLE(9)(1:6)='RADIUS'
VERB_TABLE(10)(1:6)='REDIST'
VERB_TABLE(11)(1:7)='SAMPLET'
VERB_TABLE(12)(1:5)='SCALE'
VERB_TABLE(13)(1:7)='TESTGEN'
VERB_TABLE(14)(1:7)='TEXTURE'
```

**ENDIF**

**IF (VERB.EQ."LIST") THEN**

```
MAX_VERB=8
VERB_TABLE(1)(1:9)='DSPHISTRY'
VERB_TABLE(2)(1:6)='DSPRLT'
VERB_TABLE(3)(1:7)='HISTPLT'
VERB_TABLE(4)(1:8)='LINEPLOT'
VERB_TABLE(5)(1:4)='LIST'
VERB_TABLE(6)(1:7)='LISTCAT'
VERB_TABLE(7)(1:6)='DSPLUT'
VERB_TABLE(8)(1:8)='DSPGROUP'
```

**ENDIF**

**IF (VERB.EQ."DELETE") THEN**

```
MAX_VERB=4
VERB_TABLE(1)(1:6)='LASDEL'
VERB_TABLE(2)(1:7)='CLEANUP'
VERB_TABLE(3)(1:8)='DELGROUP'
VERB_TABLE(4)(1:6)='DELLUT'
```

**ENDIF**

**RETURN**

**END**
Name - COMPARE       Type - SUBROUTINE

Language - FORTRAN 77 (VAX-Augmented)

Version 1.0      Date - 27MAR85      Programmer - B. Lowrey Code 633 GSFC

Function - Select the Functions that are common in NOUN_TABLE
and VERB_TABLE and store the result in ANS_TABLE

Parameters -
Variable      Type   I/O   Brief Description
------------  ------  ----  ----------------------------------------
NOUN_TABLE(50)  C*9   I    Functions that satisfy input NOUN
VERB_TABLE(50)  C*9   I    Functions that satisfy input VERB
ANS_TABLE(50)   C*9   O    Functions that satisfy both input VERB and input NOUN
MAX_NOUN       I   I    # Functions in NOUN_TABLE
MAX_VERB       I   I    # Functions in VERB_TABLE
MAX_ANS        I   O    # Functions in ANS_TABLE
VERBVAL        C*9   O    Temporary - element in VERB_TABLE
NOUNVAL        C*9   O    Temporary - element in NOUN_TABLE
LENV           I         # Characters in VERBVAL
LENN           I         # Characters in NOUNVAL

Subprograms Called -

Program Description -
DOFOR each Verb Function
  DOFOR each Noun Function
    IF the Verb Function is same as Noun Function
      Load Function into Answer Table of Functions
      Increment # of Functions in Answer by 1
    ENDIF
  ENDDO
ENDDO
SUBROUTINE COMPARE(NOUN_TABLE,MAX_NOUN,VERB_TABLE,MAX_VERB,
+ ANS_TABLE,MAX_ANS)

CHARACTER*9 NOUNVAL,VERBVAL
CHARACTER*9 NOUN_TABLE(30),VERB_TABLE(30),ANS_TABLE(30)

MAX_ANS=0
DO I=1,MAX_NOUN
  NOUNVAL=NOUN_TABLE(I)
  LENN=LENGTH(NOUNVAL)
  DO J=1,MAX_VERB
    VERBVAL=VERB_TABLE(J)
    LENV=LENGTH(VERBVAL)
    WRITE(6,1010) NOUN_TABLE(I),VERB_TABLE(J)
    WRITE(6,1011) NOUNVAL,VERBVAL,LENN,LENV
    IF(NOUNVAL(I:LENN).EQ.VERBVAL(I:LENN)) THEN
      MAX_ANS=MAX_ANS + 1
      ANS_TABLE(MAX_ANS)(I:LENN)=NOUN_TABLE(I)(I:LENN)
      WRITE(6,1009) I,J,MAX_ANS
      WRITE(6,1010) NOUN_TABLE(I),VERB_TABLE(J),
+ ANS_TABLE(MAX_ANS)
    ENDIF
  ENDDO
ENDDO

C Debug Formats
D1010 FORMAT(3(4X,A10))
RETURN
END
Appendix B: MAKELIST Program and Subroutines

MAKELIST operates under TAE. Depending on the value of WORD (either NOUN or VERB) a Subroutine NOUNTAB (NOUN, NOUN-TABLE, MAX-NOUN) or a Subroutine VERBTAB (VERB, VERB-TABLE, MAX-VERB) is produced. These subroutines will need to be compiled and linked with the other HELPME modules. The structure of MAKELIST is shown in Figure 2.

Dimension and other parameters that the user might wish to change are isolated in the section of MAKELIST tilted "Definitions". Equivalent changes must be made in the PDF. Currently, the dimension (count) of a TAE array (LIST1.....LIST10) cannot exceed 50; it can be lowered if desired.
Structure of
Makelist Program

Figure 2. Structure of the Makelist Program
PROCESS MAKELIST

5/6/85

A utility to produce either the subroutine NOUNTAB or
the subroutine VERBTAB

Changes which the user might desire to make:

The stringlength of a function in the input LISTs is set not
to exceed 9 because this is the maximum length of LAS PROCs.

The maximum number of elements in a LIST is the current
maximum allowed by TAE: 50.

The number of LISTs may be increased or decreased.

Changes to the above values must be matched by changes
in the program MAKELIST in the "definition" section.

<table>
<thead>
<tr>
<th>PARM NAME</th>
<th>TYPE</th>
<th>VALID</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>STRING</td>
<td>(NOUN, VERB)</td>
</tr>
<tr>
<td>LIST1</td>
<td>STRING, 9</td>
<td>(0:50)</td>
</tr>
<tr>
<td>LIST2</td>
<td>STRING, 9</td>
<td>(0:50)</td>
</tr>
<tr>
<td>LIST3</td>
<td>STRING, 9</td>
<td>(0:50)</td>
</tr>
<tr>
<td>LIST4</td>
<td>STRING, 9</td>
<td>(0:50)</td>
</tr>
<tr>
<td>LIST5</td>
<td>STRING, 9</td>
<td>(0:50)</td>
</tr>
<tr>
<td>LIST6</td>
<td>STRING, 9</td>
<td>(0:50)</td>
</tr>
<tr>
<td>LIST7</td>
<td>STRING, 9</td>
<td>(0:50)</td>
</tr>
<tr>
<td>LIST8</td>
<td>STRING, 9</td>
<td>(0:50)</td>
</tr>
<tr>
<td>LIST9</td>
<td>STRING, 9</td>
<td>(0:50)</td>
</tr>
<tr>
<td>LIST10</td>
<td>STRING, 9</td>
<td>(0:50)</td>
</tr>
</tbody>
</table>

END-PROC

Note: An entire LIST is most easily nullled by setting the
parm value thusly: LIST10=-
MAKELIST.HLP

.TITLE
Creates the subroutine NOUNTAB or VERBTAB for use with HELPME

.HELP
MAKELIST produces a full Fortran subroutine given a set of lists.
The subroutine is either NOUNTAB.FOR or VERBTAB.FOR, according
to which value is given to WORD.

If WORD=NOUN, then the first value in each list (LIST1,. . .,LIST10)
must be a VALID value of NOUN in the HELPME PDF. The following
values in that list are LAS procedures which satisfy the NOUN.
If WORD=VERB, the above applies for VERBs.

There are up to 10 lists, each of which contains one NOUN/VERB, and
up to 49 LAS names in each list.

If there is no value in the first value of a LIST, the rest of the list
will have no meaning. If a value in a LIST after the first value is
null (EX: LIST2(5)="---"), any following non-null values will be
processed. An entire LIST may be null by typing: LIST10---

The subroutine produced by this PROC must be compiled and linked to
the HELPME program.

.LEVEL1
.VAR WORD
NOUN or VERB
.VAR LIST1
First value must be a NOUN/VERB, LAS PROC names follow (<49)
.VAR LIST2
First value must be a NOUN/VERB, LAS PROC names follow (<49)
.VAR LIST3
First value must be a NOUN/VERB, LAS PROC names follow (<49)
.VAR LIST4
First value must be a NOUN/VERB, LAS PROC names follow (<49)
.VAR LIST5
First value must be a NOUN/VERB, LAS PROC names follow (<49)
.VAR LIST6
First value must be a NOUN/VERB, LAS PROC names follow (<49)
.VAR LIST7
First value must be a NOUN/VERB, LAS PROC names follow (<49)
.VAR LIST8
First value must be a NOUN/VERB, LAS PROC names follow (<49)
.VAR LIST9
First value must be a NOUN/VERB, LAS PROC names follow (<49)
.VAR LIST10
First value must be a NOUN/VERB, LAS PROC names follow (<49)

.LEVEL2
.VAR WORD
IF WORD=NOUN, a subroutine named NOUNTAB is produced and stored in
a file named NOUNTAB.FOR.
IF WORD=VERB, a subroutine named VERBTAB is produced and stored in
a file named VERBTAB.FOR.
.VAR LIST1
The first element must be a VALID NOUN (if WORD=NOUN), or a VALID VERB
(if WORD=NOUN) in the TAE PROCEDURE named HELPME.
An unwanted LIST may be null: LIST1---; LISTS following will be used.
An unwanted element in a LIST may be deleted: LIST5(3)---; the
following elements in the LIST will be used.
.END
Name - MAKELIST

Language - FORTRAN 77 (VAX-Augmented) Type: MAIN PROGRAM

Version 1.0 Date - 27MAR85 Programmer - B. Lowrey Code 633 GSFC

Function - A utility to create either the Subroutine NOUNTAB
OR the Subroutine VERBTAB for calling by the HELPME
program.

Parameters -

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>I/O</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>C*4</td>
<td>I/O</td>
<td>either NOUN or VERB</td>
</tr>
<tr>
<td>LIST1</td>
<td>C*9</td>
<td>I/O</td>
<td>Each LIST is headed by</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td></td>
<td>a VALID NOUN or VERB</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td></td>
<td>in the HELPME.PDF. The</td>
</tr>
<tr>
<td>LISTx</td>
<td>.</td>
<td></td>
<td>Following elements in the</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td></td>
<td>lists are LAS PROCnames</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td></td>
<td>which may satisfy the</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td></td>
<td>VERB or NOUN.</td>
</tr>
<tr>
<td>LIST10</td>
<td>C*9</td>
<td>I/O</td>
<td></td>
</tr>
</tbody>
</table>

Subprograms Called - XRINIM, XBINIT,XRSTR,DATE,string_write,CHAR4

--------------------------------------------------------------------------
** Program Description: **

GET the string value assigned to WORD from TAE: either NOUN or VERB
The value of WORD is part of the subroutine Name and of
the output file containing the subroutine.
WORD is NOUN or VERB; if NOUN, the NOUN lists are input and
Subroutine NOUNTAB is created, if VERB, the subroutine VERBTAB
with VERB lists is generated
OPEN: Output File to contain Fortran Subroutine being created
WRITE: Header for SUBROUTINE and Comments
WRITE: Variable type and dimensions in Subroutine NOUNTAB or VERBTAB
DOFOR: LIST1 to LIST10
GET: LISTx from TAE
MAKELIST Program p2

C CHECK: LISTx is not nulled; LISTx(1) is not nulled.
C SET: first value of LISTx to a TAE HELPME input NOUN or VERB
C WRITE: " if NOUN is equal LISTx(1) then 
C WRITE: " number of values in LISTx array"
C DOFOR: 2nd value in LISTx(J) to last value in LISTx
C SET: either NOUN_TAB(J-1) or VERB_TAB(J-1) to LISTx(J)
C (* these values are names of LAS procedures which satisfy the 
C NOUN or VERB in LISTx(1) *)
C WRITE: "NOUN_TAB(J-1) = 'LASNAME' " LASNAME is LISTx(J)
C ENDDO elements in LISTx
C ENDDO LISTs
C WRITE: RETURN and END statements
C*** END PDL

INCLUDE 'TAE$1NC:PGMINC.FIN/NOLIST' !standard TAE include
INTEGER BLOCK(XPRDIM) !standard TAE BLOCK size

**********************************************************************
* Definitions:
**********************************************************************
DIMENSION LENLIST(50)
CHARACTER*9 LIST1(50),LIST2(50),LIST3(50),LIST4(50),LIST5(50),
+ LIST6(50),LIST7(50),LIST8(50),LIST9(50),
+ LIST10(50)
CHARACTER*9 LISTVAL,OUTV(LIST8(50))
DATA NUM__OF_LISTS/10/ !Maximum number of Strings, must be 
+ compatible with LIST1...LIST10 definitions
DATA NUM_IN_LIST/50/ !maximum number of values in the Lists
DATA MAX_LENGTH/60/ !maximum # of characters in a string 
+ to be written out
DATA OUT/10/ !output unit

**********************************************************************
* End definitions
**********************************************************************
C* Set Blank Character Strings and Give Names to Special Characters
C
CHARACTER*6 B6"" / 16 blanks
CHARACTER*9 B9"" / 19 blanks
CHARACTER*12 B12"" / 142 blanks
CHARACTER*1 IQUTE"" / "quote mark

C

CHARACTER*2 ICHAR,JCHAR,CJAXJ,CLEN !Character val of I,J,CJAXJ,CLEN
CHARACTER*4 WORD
CHARACTER*12 CFILE !Name of File for subroutine
CHARACTER*2 CHAR_NUM_IN_LIST
CHARACTER*10 DATEX
CHARACTER*60 SUBNAME
CHARACTER*60 STRING !OUTPUT DUMMY

C
C* Initialize TAE; Get WORD from TAE; either =NOUN or =VERB
C
CALL XRINIM(BLOCK,XPRDIM,XABORT,ISTAT)
CALL XBINIT(ISTAT)
Call XRSTR(BLOCK,‘WORD’,1,WORD,ILEN,ICNT,ISTAT)
C
C* SETUP Output File name, and OPEN Output File
C
CFILE=WORD//'TAB'//'.FOR'
WRITE (6,1011) CFILE

A2-6
OPEN (UNIT=OUT,NAME=CFILE,STATUS='NEW')
CALL DATE(DATEX)

C* WRITE out Comments for Subroutine Header
C
WRITE (OUT,1099) CFILE,DATEX
WRITE (OUT,1100) WORD
WRITE (OUT,1101)
WRITE (OUT,1102)

C* WRITE out Subroutine Name and Calling Arguments
C
SUBNAME=WORD//"TAB(//WORD//",//WORD//"_TABLE,MAX_//WORD//")"
STRING=B6//"SUBROUTINE //NAME=SUBNAME
CALL STRING_WRITE(STRING,MAX_LENGTH,OUT)

C* Write out Variable types and dimensions
C
CALL CHAR4(NUM_IN_LIST,CHAR_NUM_IN_LIST)
STRING=B6//"CHARACTER*9 //WORD//"TABLE(//char_num_in_list//")"
CALL STRING_WRITE(STRING,MAX_LENGTH,OUT)

C* Obtain list values and write out equalities of LAS names
C
DO I=1,NUM_OF_LISTS
CALL CHAR4(I,ICHAR)
LISTVAL="LIST"//ICHAR 1obtains list name
WRITE (6,2090) LISTVAL,ICHAR
CALL XRSTR(BLOCK,LISTVAL,NUM_IN_LIST,OUTVAL,LENLIST,MAXJ,ISTAT)
Write (6,2096) maxj
IF(MAXJ.NE.0) THEN
   IF (OUTVAL(1).NE."--") THEN
      WRITE (6,1095) LISTVAL,(OUTVAL(KK),KK=I,MAXJ)
      STRING=B6//"IF //WORD//".EQ.//QUOTE//OUTVAL(1)(I:LENLIST(1))//QUOTE//" THEN"
      CALL STRING_WRITE(STRING,MAX_LENGTH,OUT)
   END IF
C
C Obtain number of Functions that have non-zero values
C and write out the number
C
K=0
DO J=2,MAXJ
   IF (LENLIST(J).NE.0) K=K+1
ENDDO
CALL CHAR4(K,CMAXJ)
STRING=B9//"MAX_//WORD//"MAX_//CMAXJ
CALL STRING_WRITE(STRING,MAX_LENGTH,OUT)
K=1

C Set NOUN_TABLE (or VERB_TABLE) elements equal to each name in list
C
DO J=2,MAXJ
   IF (LENLIST(J).NE.0) THEN
      CALL CHAR4(K,JCHAR)
      CALL CHAR4(LENLIST(J),CLEN)
      STRING=B12//"_TABLE("//JCHAR//"(1:"+"
      //CLEN//")="//QUOTE//OUTVAL(J)(1:LENLIST(J))//
      +"QUOTE
      K=K+1
      CALL STRING_WRITE(STRING,MAX_LENGTH,OUT)
ENDIF
ENDDO
STRING=B6//'ENDIF"
CALL STRING_WRITE(STRING,MAX_LENGTH,OUT)
ELSE
ENDIF
ELSE
ENDIF
ENDDO ! END OF LIST LOOP OVER I

C Put out last statements in subroutine
C
WRITE (OUT,1003) !RETURN
D WRITE (6,1003)
WRITE (OUT,1005) !END
D WRITE (6,1005)
CLOSE(UNIT=OUT)

1002 FORMAT (A)
1003 FORMAT (T7,'RETURN')
1005 FORMAT (T7,'END')

C MISCELL FORMATS
1011 FORMAT (IX,A20," : =FILE NAME OF SUBROUTINE BEING CREATED")
1095 FORMAT (" Values stored in ",A10," are:",/4(bA10,/))

C HEADER FORMATS
1099 FORMAT("C",71("**"),/,
+ "C",/,
+ "C Name - ",A20," Type - SUBROUTINE",/,
+ "C",/,
+ "C",/,
+ "C",/,
+ "C DATE - ",A10," PROGRAMMER - B.LOWREY,CODE 633,
+ GSFC")
1100 FORMAT("C PURPOSE - Returns Table of Functions that satisfy ",A4)
1101 FORMAT("C*",12X," Note: this subroutine produced by MAKELIST
+ program")
1102 FORMAT("C",71("**")

C DEBUG FORMATS
2090 FORMAT (" LISTVAL=",A6," ICHAR=",A5)
2096 FORMAT (" Number of names in list is=",i4)

STOP
END
C Subroutine CHARGET
C*******************************************************************
SUBROUTINE CIIARGET(IDIGIT,ICCHAR)
C
C Function: To convert a single input INTEGER digit IDIGIT to an output
C CHARACTER ICHAR
C
CHARACTER*(*) ICHAR
IF (IDIGIT.EQ.0) ICHAR='0'
IF (IDIGIT.EQ.1) ICHAR='1'
IF (IDIGIT.EQ.2) ICHAR='2'
IF (IDIGIT.EQ.3) ICHAR='3'
IF (IDIGIT.EQ.4) ICHAR='4'
IF (IDIGIT.EQ.5) ICHAR='5'
IF (IDIGIT.EQ.6) ICHAR='6'
IF (IDIGIT.EQ.7) ICHAR='7'
IF (IDIGIT.EQ.8) ICHAR='8'
IF (IDIGIT.EQ.9) ICHAR='9'
RETURN
END
**SUBROUTINE CHAR4**

**Function** - To convert the INTEGER*4 value INUM to an output CHARACTER data ICHAR

```fortran
SUBROUTINE CHAR4(INUM, ICHAR)
  INTEGER*4 INUM
  CHARACTER*(*) ICHAR
  CHARACTER*1 ICHARA
  ICHAR = ' '  
  ISTRIP = INUM
  20  IDIGIT = MOD(ISTRIP, 10)
      CALL CHARGET(IDIGIT, ICHARA)
      ICHAR = ICHARA//ICHAR
      ISTRIP = (ISTRIP - IDIGIT)/10
      IF (ISTRIP .NE. 0) GOTO 20
  WRITE(6,2000) ICHAR
2000  FORMAT(3X,A80)
RETURN
END
```
NEW ENHANCEMENTS TO THE LAS IMAGE PROCESSING SYSTEM

by

Jon W. Robinson PhD.

Science Application Research Inc.
4400 Forbes Blvd.
Lanham, Maryland 20706

Introduction

The Landsat Analysis System (LAS) which runs under TAE is an image processing system that has been developed primarily at Goddard Space Flight Center (GSFC). LAS is a collection of programs and procedures that can be invoked from TAE that permit a wide range of image processing functions to be carried out on digital images. This includes functions for loading standard CCT's, functions for registering images to each other or to maps, functions for classifying images, functions for changing color and grey level mappings and many utility functions.

During the past year, the Landsat Analysis System at Goddard Space Flight Center has been enhanced in several ways. The nature of these enhancements and their advantages for the analyst will be described in what follows.

The major additions to the LAS have been retro coding old procedures, the addition of new procedures to provide IDIMS equivalency and the integration of International Imaging Systems displays and software into the LAS system.

Retrocoding

The retrocoding of the old procedures was carried out to provide a more uniform user interface and to provide a friendlier user interface. The friendlier interface includes several features. These are: 1). Each image now has a history associated with it; 2). At the conclusion of each process the names of the output files created by the process are listed for the analyst; 3). Most of the VAX system errors are intercepted by the program and explained in understandable terms; 4). The window specification for an input image is now included in the specification of the input image, not as a separate parameter; 5). The input increment of the input image can be specified with the input image name for the LASCI display functions; 6). If the input image is a multiband image, then sub bands can be selected in the specification of the input image name; 7). The analysis and manipulation of multiband images has been facilitated by modification of the LAS naming convention and the addition of several utility procedures; 8). A session log of all the user entries and program messages can be kept and can be printed at the end of a session; 9). Many of the old codes had new features added to them when they were retrocoded.
Session Histories

When a LAS user executes a function that produces an output image, the values entered for the mandatory parameters and nondefault parameter values are recorded in the output image's label along with the histories of the input images if any. The program DSPhistory will allow the user to view at his terminal or send to the printer, the history of a specific image. This ability allows an analyst to review how a particular image was created. This is especially useful when several analysts are working together and it is necessary to check on what happened several steps prior to an image that is currently presenting a problem to the analyst.

Procedure Wrap Up Messages

At the conclusion of an LAS program, the names of the files that were created are listed along with the message that the program completed successfully. This has replaced the FORTRAN STOP message that users use to receive. When using the command mode, this leaves the names of the most recently created images on the screen so that the analyst can see them for use in the next function he elects to use.

Interception of VAX System Error Messages

In the past, when a program failed during execution, a variety of obscure system error messages were written to the terminal. In the present release of LAS most of these messages are trapped and sent to the user in a more clearly understood form.

Input Image Specification

In the old version of LAS, those programs that allowed windowing of input images, had a specific parameter, WINDOW, that allowed the window to be specified. In the current version, any LAS program that takes an input image can accept windowing as part of the input image parameter. The format is to provide the name of the input image followed by an open parenthesis followed by the following parameters separated by spaces: start sample, start line, number of samples, number of lines; then a close parenthesis. Most LASCI functions allow two more parameters, the sample increment and the line increment. If the input image is a multiband image and only a subset of the bands is to be processed, then, the bands to be processed are listed after a colon which separates the window parameters from the sub band parameters. Thus a complete input specification would look like:

"INPUT. IMAGE( SS SL NS NL SI LI : SB1 SB3 SB4 )"

note that the sample and line increment parameters are not available in the mainline LAS functions but only in the LASCI
functions.

Multiband Images

In the old LAS each band of a multiband image had to be listed in the input parameter. Now these bands can be placed together in a file group using the command GROUP and only the single group name is needed as an input parameter to subsequent functions. In addition the restrictions on the form of the group names have been dropped.

The function DSPGROUP will list the image files included under a group name. Thus the identities of the bands in a multiband image can be determined.

If a multiband image is being input to the LAS system or is being produced by a function, its group file is automatically produced and the members are stored in it. This makes work with multiband images as easy as working with single band images was under the old system.

Session Log

The user may choose to keep a log of his session. This allows him to keep track of the exact sequence of procedures that he followed. This is particularly useful in the interactive environment of LAS where an analyst may make several false starts before choosing the final set of procedures and parameters that produce the result he is satisfied with. There is also a provision to keep the tutor screens as a separate option.

New Options

A number of the old codes had new options added to them. For example, the program BAYES now has the option to produce a CHI SQUARED image that can be used as input to UNKNOWN in order to set thresholds for pixels that are unlikely to be classified correctly. Using the CHI SQUARED image as input, the analyst can pick probability levels as thresholds for each class in a classified image produced by BAYES.

It is now easier to edit a statistics file with EDITSTAT. The program ADD2STAT allows statistics files to be combined. STATPLOT plots the mean and V-CV ellipses for any two bands selected from a statistics file.

There are a variety of new functions that facilitate supervised classification. ZIP allows a band mask of training areas to be created. This program in combination with STATCUT, MASKSTAT and TRANSDIV (transformed divergence between pairs of classes) can be used to evaluate and refine a supervised classification. With the variety of supervised classifications available (BAYES, DISCRIM, MINDIST, CANAL (canonical analysis)) the analyst has many options available, not to mention the unsupervised procedures (ISOCLASS, KMEANS and HINDU).
For filtering there are a host of methods available. LOWCAL has 13 spatial domain filters. These are MEAN, STDEV, VARIANCE, QUANT (quantitized mode window), MINIMUM, MAXIMUM, MEDIAN, MABS (median absolute deviation window), SHARP, KNN (k nearest neighbor window), SELECT (selective average), MSUB (mean subtraction) and CONV (separable convolution).

CONVOLVE allows the analyst to apply any spacial convolution filter he wishes to an image by using CWTGEN to create a matrix of filter weights.

FFT2 and IFFT2 provide gateways to and from the frequency domain. DEBLUR applies deblurring functions to frequency domain images. FT2FL performs a two dimensional Fourier transform filter. FT2PIX allows an image of both the magnitude and the phase to be extracted from the complex image and displayed on the display monitor. CROSSCOR calculates the cross correlation in the frequency domain. There are other programs for creating complex images by combining two images, COMPLEX, and taking complex images apart into their real and imaginary parts COMSEP.

To facilitate terrain analysis, digital terrain data can be entered into LAS using the DEMENTER program. AMSCNVT converts Analytic Mapping System (AMS) files to LAS file format. BANGLE calculates the Beta angles from an image of elevation data and creates an image in the form of a shaded relief map. SURFACE takes in data as triplets (X,Y, Z) and creates a surface image by using any of three interpolation methods.

For the analysis of geographic information stored as images there are programs like GETBLOB which identifies the regions in the input image which comprise a given class and assigns a unique number to each contiguous area while assigning zeros to the rest of the output image. There is the program INTERSECT which creates an image which represents the unique combinations of grey levels from two input images; the program SPREAD which replaces each pixel in an image with its distance to the nearest pixel that contains the target value.

There are many other LAS capabilities for registering images to each other and to maps, edge detection algorithms etc. In addition there is a new sub system, LASCI, which allows the LAS user to display images on the IIS display and make use of the IIS display commands. Among the procedures in the LASCI subsystem are programs to display images from LAS image files, DISPLAS, and IIS image files, DISPLAY. The procedure LASPOINTS allows the analyst to use the cursor to identify the pixel values and image and screen coordinates. It also allows the analyst to change the pixel value at any selected coordinate. LPALETTE provides the ability to color single band images such as class images using a variety of methods. LTLM implements linear track ball mapping and STASH saves these mappings to a IIS stash file. LSTASH converts an IIS stash file to an LAS LUT file for use in various LAS functions. PROCLAS allows the mappings saved in a STASH file
to be applied to an LAS image that is larger than 512 by 512.
L2I converts LAS format images to IIS format images making
available all of the IIS procedures. IIS images can be
transfered back to LAS by either putting them in the display and
using SAVLAS or by writing the image to a tape with TRANSFER and
then using the LAS program IENTER to read the image.

There are many other LAS functions that cover almost every aspect
of image processing. All of the functions are described in the
LAS Users Manual which is available from Goddard Space Flight
Center, Greenbelt, Md. 20771, Attn: Larry Novak. If an IIS
terminal is available LAS offers entre into the IIS analysis
system which adds new options for the image analyst. For those who
wish to expand on LAS there is the LAS Programmers Manual and the
TAE Programmers Manual.

LAS offers a complete image analysis capability running under
TAE. The advantages of TAE as a user interface have been
discussed in other papers in this conference and will not be
repeated here.
THE COMMUNICATIONS NETWORK FOR
NASA'S PILOT LAND DATA SYSTEM (PLDS)

HARRY JONES
NASA AMES RESEARCH CENTER

FIFTH ANNUAL TAE USERS' CONFERENCE
NASA GODDARD SPACE FLIGHT CENTER
JUNE 5, 1985
COMMUNICATIONS NETWORK FOR PLDS

PLDS - PILOT LAND DATA SYSTEM

* A NASA PROGRAM DIRECTED BY GSFC
* SYSTEM CONCEPT DEVELOPED BY LAND SCIENTISTS AND TECHNICAL EXPERTS
* TO FACILITATE LAND SCIENCES RESEARCH
  - LOCATION, ACQUISITION, PROCESSING, AND ANALYSIS OF DATA
  - RELATED TO PLANETARY, OCEAN, AND CLIMATE PILOTS

* SUPPORT NASA LAND SCIENCE PROJECTS
  - SEDIMENTARY BASINS (JPL)
  - LAND SURFACE CLIMATOLOGY (GSFC)
  - FUTURE PROJECTS

* TO UTILIZE ADVANCED INFORMATION SCIENCE TECHNOLOGY
  - IMAGE AND DATA PROCESSING
  - DATA BASE SYSTEMS
  - COMPUTER ACCESS WORKSTATIONS
  - NETWORKING AND COMMUNICATIONS
COMMUNICATIONS NETWORK FOR PLDS

PLDS AND TAE

* PLDS WILL USE TAE AND LAS, THE LAND ANALYSIS SYSTEMS
* PLDS WILL USE TCP/IP NETWORKING SOFTWARE
  - WIDELY AVAILABLE
  - INCLUDED IN BSD 4.2 UNIX
* TAE SHOULD INCORPORATE TCP/IP NETWORK ACCESS
  - TAE & LAS BEING CONVERTED TO UNIX
COMMUNICATIONS NETWORK FOR PLDS

OBJECTIVES

* PROVIDE STANDARD NETWORK SERVICES
  - ELECTRONIC MAIL
  - REMOTE LOGIN
  - FILE TRANSFER

* PROVIDE ACCESS TO SPECIALIZED PLDS SERVICE
  - DATA BASE
  - REMOTE PROCESSING
  - SOFTWARE LIBRARY
COMMUNICATIONS NETWORK FOR PLDS

REQUIREMENTS

• WIDE AREA CONNECTIVITY
• INTERFACE FOR VAX, SUN, IBM, PC, AND OTHER COMPUTERS
• MAXIMUM USE OF EXISTING COMPUTERS, NETWORKS, AND SOFTWARE
A protocol is layered software that provides communications functions

- Blocking, addressing, error check and retransmit, network routing, etc.

Some protocols are:

- TCP/IP - developed for Arpanet, widely available
- DECnet - DEC's system network architecture
- X.25 - international standard used by Telnet
- ISO model - the international standards organization (ISO) model for open systems interconnection

It incorporates DECnet, and is developing part of the ISO model.
COMMUNICATIONS NETWORK FOR PLDS

ALTERNATIVES

<table>
<thead>
<tr>
<th>PROTOCOL SOFTWARE</th>
<th>NETWORK</th>
<th>LINK</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/IP</td>
<td>ARPANET</td>
<td>LEASED LINES</td>
</tr>
<tr>
<td>X.25</td>
<td>TELENET (NPSS)</td>
<td>&quot;</td>
</tr>
<tr>
<td>DECENT</td>
<td>SPAN</td>
<td>&quot;</td>
</tr>
<tr>
<td>ISO</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>MANY</td>
<td>MANY</td>
<td>DIAL-UP</td>
</tr>
</tbody>
</table>

HJONES
NASA ARC
COMMUNICATIONS NETWORK FOR PLDS

ADVANTAGES OF TCP/IP

* MANY IMPLEMENTATIONS CURRENTLY AVAILABLE, SOME FREE
  - BSD 4.2 UNIX (VAX, SUN, etc)
  - VAX ULTRIX, VAX VMS PACKAGES
  - ARPA NET
  - ETHERNET LAN’S

* DECNET AND SNA ARE SINGLE VENDOR SYSTEMS, USING DEC AND IBM HARDWARE ONLY

* THE ISO MODEL IS NOT YET FULLY STANDARDIZED OR COMMERCIALY AVAILABLE

* TCP/IP (AND DECNET) WILL PROBABLY BE REPLACED BY ISO MODEL PROTOCOLS WHEN AVAILABLE
COMMUNICATIONS NETWORK FOR PLDS

APPROACH

* PROTOCOL: TCP/IP

* NETWORKS: COMBINATION OF ARPANET AND TELNET
  - TCP/IP ENCAPSULATED IN X.25
  - IMPLEMENTATION DESIGNED FOR CSNET

* LINKS: ARPANET AND TELNET AUGMENTED BY LESED LINES, LOCAL AREA NETWORKS, DIAL-UP LINES, AND WIDE BAND SATELLITE LINKS
COMMUNICATIONS NETWORKS FOR PLDS

CONCLUSIONS

* THE TCP/IP PROTOCOL HAS BEEN SELECTED FOR PLDS.

* THE TAE SHOULD INCORPORATE TCP/IP, IN ADDITION TO DECNET AND ISO PROTOCOLS.
INTERACTIVE FORMAT CONVERSION SYSTEM (IFCS)

Steven J. Kessler, NASA/GSFC

ABSTRACT

The Interactive Format Conversion System (IFCS) is a package designed to facilitate the transfer of data between heterogeneous computers. The system has the generalized capability of: 1) accepting input data from a number of devices (disk, tape, data line); 2) performing useful data conversions, and; 3) producing output on a variety of devices. The structure of the data conversion subsystem simulates a subset of the presentation layer in a network communications link by converting input data into an internal machine independent format and then converting the internal data to the output format. This conversion subsystem is derived by the inputs of the application. The Transportable Applications Executive (TAE) is used to provide a consistent user interface and tie the various subsystems together.

1.0 INTRODUCTION

1.1 BACKGROUND

The transfer of space-derived data between computers, both heterogeneous and homogeneous, have become more common and increasingly desirable. The International Organization of Standardization (ISO) has established a seven layer model for networking. (Tanenbaum describes this in detail in his book Computer Networks.) See Figure 1. The five lower layers of the networking model are being addressed by various organizations, including the National Bureau of Standards (NBS), International Standards Organization (ISO), and the IEEE. Currently, the sixth layer, the Presentation layer, is being developed on a case-by-case basis many times over for a variety of space-related data. This layer specifically performs transformations on data, such as text compression, format conversions, encryption, etc.

1.2 CURRENT EFFORT

Problem: Currently, Presentation layer, specifically format conversion software, is being developed
on a case-by-case basis many times over for a variety of space-related science data, leading to much duplication of effort and code.

Solution: The Interactive Format Conversion System (IFCS) is a subset of Presentation Layer Software. It generalizes format converting by interactively generating software that transforms data from and to the desired machine formats. The generated code is transportable and can be generated for a particular application and used on a number of different computers. Such a system solves the problem for the users of space-derived data that has not been attacked in any general sense up to now.

User inputs to the system include global variables which are defined and used throughout the IFCS session. After IFCS is launched (Figure 2), the CVTGEN executes by receiving communications (inputs) from the user: data definition files, file and machine names. CVTGEN generates a stand alone conversion routine, that may be linked to a user supplied program or link to the General Format Conversion Utility (CVTLINK). To execute the General Format Conversion Utility (IFCUTIL), the data input devices (containing the data to be converted) and data output device, and device attributes must be communicated (input) to the utility, as well as the number of records to convert. The result is converted data.

IFCS presently resides on the Laboratory of Extraterrestrial Physics (LEP) (Code 690) VAX 11/780, in Building 2 at Goddard Space Flight Center. This computer supports a wide range of scientific space-related data and the analysis of this data. Included are data from Mariner, Voyager, ISEE and IMP satellites. In addition, the LEP VAX supports many data analysis packages and numerical libraries. Therefore, the need has grown to transfer data on the LEP VAX to other computers as well as visa versa, to support the needs of scientific data analysis in familiar environments. This need is not limited by any means.

1.3 DESIGN CONSIDERATIONS

The primary objective in designing and implementing IFCS was to develop a system that allows characteristics of the source and target data streams, along with identifier information, to be easily specified interactively. IFCS utilizes this input to produce transportable computer code that maintains the semantics of the data as they are transformed from one computer to another. In addition, IFCS was developed to be friendly and flexible. The user need not supply more information than is absolutely necessary for the function to be performed. Also, interactive input requirements must be unambiguous and check for invalid inputs. IFCS is made flexible enough to handle a wide spectrum of possible data inputs. Finally, the system was designed to isolate host specific code so that it may be transported with minimal change.
2.0 CAPABILITIES

2.1 FUNCTIONAL CAPABILITIES

The first step of IFCS is the conversion generator program. Required inputs include input and output record definition and optional namelist file names, machine format associated with the data and the name of the output conversion routine. The conversion generator analyzes the input and output record definitions and builds a file (a subroutine) that contains all the actual field conversion routines in exact order as defined by the record definition. The library containing these lower level conversion routines is the heart of IFCS. Each routine performs a different function (i.e. convert DEC R*4, convert to IBM R*8, etc. See PERFORMANCE CAPABILITIES for a further discussion on converting.) The conversion generator creates the routine which in turn accesses these pre-existing routines when run. The second step in IFCS is to compile and link the conversion routine. It may be linked to a user developed program or it may utilize the General Format Conversion Utility (GFCU). This utility will perform all general input of data, convert the data using the conversion routine and output the results. Finally, GFCU or the user application program is executed to perform the format conversion.

In addition, IFCS includes a Machine Definition program for when it becomes desirable to add new machines to the system (IFCS presently supports DEC VAX and IBM). New conversion library routines will also need to be implemented.

IFCS utilizes TAE to enhance its functional capabilities as well as fulfill design objectives. Inputs are entered using TAE standards. This provides ease for the experienced user and support for the less experienced user. Menus and help files provide information for the first time user. In addition, IFCS can be easily transported to any installation that maintains TAE.

2.2 PERFORMANCE CAPABILITIES

IFCS has several important performance capabilities. A primary capability is its use of namelists. This provides the user with the ability to convert only certain fields of data from the input record. GFCU is capable of inputting data from up to three input sources, converting the data and outputting to a single sink. Also, GFCU can input and output to tape or disk. Most important is that IFCS uses an intermediate data format when converting data (Figure 3). That is, every field transformed is actually converted twice (i.e. IBM -> intermediate form -> VAX). This design was implemented so that when additional machines are added, source code will increase at a much smaller rate. (All new routines will convert to
3.0 OPERATION

IFCS is able to perform six basic operations through the use of the TAE menus (Figure 4). IFCS68L is a global procedure that allows the user to define certain variables. CVTGEN, CVTLINK and GFCUTIL are the three steps for developing and executing a data conversion program (Figure 5). The procedure, IFCS, combines the previous three PDF's in one procedure. Finally, MACHDEF allows the programmer to implement additional machines.

3.1 IFCS68L

Two variables set in this global are used throughout IFCS.
- the name of the IFCS generated conversion routine to be linked and executed.
- the number of input sources (up to three).

3.2 CVTGEN

CVTGEN represents the first step if IFCS (Figure 6). Using information recieved from user created internal files and user input, CVTGEN creates a FORTRAN routine that, when executed, will receive data according to the specified format, convert the data to the desired machine and output only the data fields within the record that are of interest. The conversion routine is made up of a series of calls to pre-existing lower level routines. For each data field, a lower level routine is accessed to perform the correct bit manipulations to move that field into and out of the intermediate format (as described earlier).

3.2.1 Inputs

The required internal files include:
- the record definition files which contain the exact field format of the data to be input (one is required for each input source).
- the record definition file which contains the exact field format of the data to be output.
- the optional namelist files which contain a name that corresponds to each field in the record definition files (one for each input source) of the data to be input.
- the optional namelist file which contains only the names of the fields that are to be converted and output. If namelist files are not specified or the input data and output data namelists are exact, then all fields are converted.

The user inputs include:
- the name of the file that contains the record definitions (one for each of up to three input sources).
- the name of the file that contains the input namelists (up to three). This is optional.
- the name of each machine which the input data was generated on (up to three).
- the name of the file that contains the record definition for the output.
- the name of the file that contains the output namelist. This is optional.
- the name of the machine which the output data is generated for.
- the name to be given to the file that will contain the newly generated conversion routine.

3.2.2 Outputs
The CVTGEN output is the reusable conversion routine created to user specification.

3.3 CVTLINK

At this point the conversion routine may be utilized with a user application or it may be linked to the General Format Conversion Utility (GFCU). This procedure compiles and links a conversion routine to the GFCU. The name of the object and load modules will be the same as the source file, which are defined in IFCSGBL. No interactive inputs are required for this step.

3.4 GFCUTIL
The third step, GFCUTIL, actually converts the specified data (Figure 7). This procedure, using the tape I/O library, provides a means for reading tapes created on and writing tapes for other machines in any format, as well as reading and writing to disk. Generally, GFCU acquires the input data, performs the specified conversions, and outputs the results.

3.3.1 Inputs
To operate GFCU, the following inputs are requested:
- the type of device in which the input is received from (TAPE or DISK). One for each input source.
- the type of device in which the output is to be sent (TAPE or DISK).

For each TAPE used, the following must also be provided:
- the name of the tape drive.
- the tape label if the tape is labeled.
- the machine format of the tape (presently, IBM or VAX).
- the record type of the tape file.
- the logical record size.
- the tape block size.
- the file name of the data (for input data tapes and output data tapes), or the file number of the data (for input data tapes).
- the starting record number in the file of the tape where data conversion is to commence (for input data tapes).

For each DISK file used, the following must also be provided:
- the organization of the disk.
- the disk access method.
- the record size of the disk.
- the name of the disk file.
- the starting record in the file where data conversion is to commence (for input data).

In addition, these parameters are also required:
- the number of data records to be converted.
- whether GF Cut Uil is to be executed in interactive mode or batch mode.

3.3.2 Outputs
The output generated is a file containing the desired converted data.

3.5 IFCS

The purpose of this operation is to combine the three steps of IFCS into one procedure. This provides much convenience when it is desired to create, link and execute IFCS all at once.

3.6 MACHDEF

This operation is primarily used by the IFCS manager. When a new machine is implemented into IFCS, the manager must: create lower level conversion algorithms that convert data to and from the intermediate format; provide for any tape formatting dissimilarities that the new machine has to the existing machines (in the tape I/O library) and; execute MACHDEF. MACHDEF is a software maintenance program that implements the characteristics of any newly added machine to IFCS. As mentioned, only the IBM and VAX are presently supported. This software receives the machine characteristics and places them in a machine definition file.

3.6.1 Inputs
The inputs include:
- the name that identifies the machine whose attributes are being entered (ex. VAX).
- a two character machine identifier (ex. VX for VAX).
- the number of bits per byte of this machine.
- the default Hollerith code to be assigned to this machine (ASCII or EBCDIC).
- up to 20, two character data format identifiers (ex. R4 for REAL*4).
- up to 20, integers describing the length in bytes of each data format identifier entered.
- up to 20, one character data type identifier for each data format identifier entered (I, F, H or Z).

3.6.2 Outputs
The output of this process is the addition of the new machine specifications in the machine definition table.

4.0 FUTURE CONSIDERATIONS

Plans exist for IFCS on all fronts. Enhancements to the system include adding a provision for special data types (for example, spacecraft telemetry). Enhancements for I/O include implementing electronic communication into GFCU. Adding other machines to the system is another immediate consideration, as well as implementing IFCS on other machines. From an operational point of view, optimizing the speed of IFCS is being addressed.

ACKNOWLEDGEMENTS

I wish to acknowledge William Mish, Thurston Carleton and Jack Yambor for their design and implementation of IFCS.

REFERENCES

**Figure 1**

This diagram illustrates the communication process between two hosts, Host A and Host B, using the OSI model. The layers involved are as follows:

1. **Layer 1 (Physical Layer)**: Handles physical characteristics of the transmission medium.
2. **Layer 2 (Data Link Layer)**: Manages data transmission over the physical medium.
3. **Layer 3 (Network Layer)**: Connects networks to form a global communication system.
4. **Layer 4 (Transport Layer)**: Provides end-to-end connection services.
5. **Layer 5 (Session Layer)**: Establishes, maintains, and terminates sessions.
6. **Layer 6 (Presentation Layer)**: Formats data for higher layers.
7. **Layer 7 (Application Layer)**: Interacts directly with users or processes.

Each layer in Host A communicates with a corresponding layer in Host B, with protocols specified at each interface. The layers are interconnected to facilitate the exchange of data between the two hosts.
USER INPUTS FOR IFCS

GLOBAL VARIABLES

DATA DEFINITION FILES; FILE AND MACHINE NAMES

IFCSGBL

CONVERSION ROUTINE

TO USER SUPPLIED PROGRAM

CVTGEN

CVTLINK

GFCUTIL

DEVICES USED AND THEIR ATTRIBUTES; NUMBER OF RECORDS TO CONVERT

FIGURE 2
NUMBER OF CONVERSION ROUTINES REQUIRED AS A FUNCTION OF THE NUMBER OF MACHINES IN IFCS

CONVERTING ONE DATA TYPE TO THE SAME DATA TYPE:

<table>
<thead>
<tr>
<th>NUMBER OF MACHINES (M)</th>
<th>ONE ROUTINE DOES CONVERSION (M*(M-1))</th>
<th>WITH IFCS INTERMEDIATE FORMAT (M*2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2 (1 IN EACH DIRECTION)</td>
<td>4 (2 IN EACH DIRECTION)</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>

CONVERTING ONE DATA TYPE TO ANY OF 4 DATA TYPES:

<table>
<thead>
<tr>
<th>M</th>
<th>16 * M * (M-1)</th>
<th>4 * M * 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>32 (16 IN EACH DIRECTION)</td>
<td>16 (8 IN EACH DIRECTION)</td>
</tr>
<tr>
<td>3</td>
<td>96</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>192</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>320</td>
<td>40</td>
</tr>
</tbody>
</table>

ORDINARILY, IT TAKES 32 ROUTINES TO BE ABLE TO CONVERT ANY 4 DATA TYPES (R*4, R*8, I*2, I*4) FROM ONE MACHINE TO ANY OF THOSE DATA TYPES ON ONE OTHER MACHINE.

USING IFCS INTERMEDIATE FORMAT IT TAKES ONLY 16.

FIGURE 3
INTERACTIVE FORMAT CONVERSION SYSTEM

1) TO ALTER IFCS GLOBAL VARIABLES (IFCSGLB)
2) TO GENERATE AN IFCS CONVERSION ROUTINE (CVTGEN)
3) TO LINK THE CONVERSION ROUTINE TO THE GENERAL FORMAT CONVERSION UTILITY (CUTLINK)
4) TO EXECUTE THE GENERAL FORMAT CONVERSION UTILITY (GFCUTIL)
5) TO EXECUTE PROCS CVTGEN, CUTLINK AND GFCUTIL (IFCS)
6) TO IMPLEMENT A NEW MACHINE'S ATTRIBUTES (MACHDEF)

Enter: selection number, HELP, BACK, TOP, MENU, COMMAND, or LOGOFF.
?

FIGURE 4
INTERACTIVE FORMAT CONVERSION SYSTEM (IFCS)

STEP 1

RUN CONVERSION ROUTINE GENERATOR

DATA DEFINITION INPUT

STEP 2

CONVERSION LIBRARY

EXECUTABLE CONVERSION PROGRAM

DATA DESCRIPTION INPUT

STEP 3

COMPILE AND LINK TO GFCU

I/O LIBRARY

RUN GFCU

CONVERTED DATA

FIGURE 5
CONVERSION ROUTINE GENERATOR

INTERACTIVE SPECIFICATION

CONVERSION ROUTINE GENERATOR

STORED DESCRIPTIONS
INTERNAL REPRESENT

TRANSPORTABLE FORMAT CONVERSION SOFTWARE

FIGURE 6
GENERAL FORMAT CONVERSION UTILITY (GFCU)

FIGURE 7
TAE and
Interactive Research Imaging System - IRIS

by

D. Neil Allen
Cooperative Institute for Research in the Atmosphere
June 4-6, 1985
ABSTRACT

TAE and Interactive Research Imaging System - IRIS

The Cooperative Institute for Research in the Atmosphere (CIRA), and the Department of Atmospheric Science, at Colorado State University, Ft. Collins, Colorado, have been active users of TAE for the past three years. TAE is an integral part of our Interactive Research Imaging System (IRIS).

IRIS is used by our research scientists, staff, and graduate students to process weather satellite imagery for research programs and education. In addition to the weather satellite data, other weather data is used. An example is the data received through United Video satellite transmission services. These data are input to the GEMPAK system that was also developed by NASA Goddard and is also a component of IRIS. An overview of IRIS, its components, data, and applications will be presented.

IRIS runs on a DEC VAX 11/780 under VMS V4.1. A conversion to VMS V4.1 from V3.7 was made recently. This conversion will be reported to inform other TAE users of our methods and to identify those problems we encountered and how they might be avoided.
TAE and
Interactive Research Imaging System - IRIS

The Cooperative Institute for Research in the Atmosphere (CIRA), and the Department of Atmospheric Science, are located on the Foothills Campus of Colorado State University, Ft. Collins, Colorado.

In the mid-1970's, the Department of Atmospheric Science developed its capability to receive directly GOES Satellite imagery, with the installation of a Direct Readout Satellite Earth Station. Computer systems were installed and image processing capabilities were built by engineers from the Electrical Engineering Department. Since these first developments several upgrades to computers and image processing have occurred. In 1981, the third generation of satellite image processing capability was installed. A Digital Equipment Corporation VAX 11/780, running the VMS operating system, and a Comtal Vision One/20 were integrated. An extensive project was initiated that year to bring together the image processing programs into one unified system. This system became known as IRIS, Interactive Research Imaging System. The Satellite Earth Station and Image Processing Facility is now under the direction of the Cooperative Institute for Research in the Atmosphere (CIRA). Dr. Thomas VonderHaar is the Director of CIRA.

IRIS is a system of several software components. They are a relational data base management system, application programs libraries, DI3000 Graphics, device drivers and image processing
libraries for the Comtal. The Transportable Application's Executive (TAE) developed by NASA Goddard, is a major part of IRIS. Starting with IRIS's development, TAE was evaluated using the prototype releases. TAE was found to be the best method of providing user interface with the many programs in the satellite image processing libraries. TAE Version 1.2 is now in use.

The users of the system are frequently short-time users, usually for a two year period or less. TAE provides a method for these users to become familiar with IRIS quickly, which allows more time for research and educational studies. TAE also provides an efficient method to add new applications to IRIS. As an example, the GEMPAK and GEMPLT systems developed at Goddard were added to IRIS this year.

A major event that is now in progress at CIRA is the development of a real time weather laboratory. This laboratory will receive and process real time satellite and weather observation data. Satellite data being received at CIRA are VISSR (Mode A), and VAS (Mode AA). Work is now in progress on a frame synchronizer that will receive Mode AAA when transmission in this format begins in the fall of 1986. Real time weather observation data now being received and ingested into the VAX 11/780 are GOES DCS data; FAA604; NMC LFM Gridded data; PROFS Mesonet; APOS; and Limon, Colorado, and Cheyenne, Wyoming, Radar.

The FAA604 data real time ingest into the VAX 11/780 has been interfaced to the GEMPAK and GEMPLT systems making it possible to produce real time surface and sounding graphics products on the Comtal and other graphics devices. This is a significant step
towards providing real time weather data to the weather laboratory. In addition to providing real time weather data, a major upgrade to the hardware systems is also in progress.

Purchase orders have just been submitted for more than $1,000,000 worth of new computer systems, storage, local area networks, and work stations. Three VAX 11/750's will be added to the present VAX 11/780 by October 1985, in a Star Cluster configuration, with Ethernet and VAX VMS V.1 Workstations providing the first phase of the real time weather laboratory, which will become the fourth generation of weather processing capabilities at CIRA and the Department of Atmospheric Science.

TAE will continue to play a significant role in the future developments of systems for research and education at CIRA and the Department of Atmospheric Science. Significant research in weather satellite data and real time weather will be conducted here in the next several years. Students will be given opportunities to use the latest state-of-the-art technology in their studies and research projects. Systems like TAE, GEMPAK, and GEMPLT will make this possible. Without them, funds for research programs would have to be used for software development not directly related to pure research objectives, thus increasing the cost of research while at the same time decreasing research effectiveness. CIRA and the Department of Atmospheric Science look forward to a long-term continuation of cooperative efforts with NASA in the further development and use of the Transportable Application's Executive, TAE.
UNIX EMPHASIS
TRANSPORTING TAE TO UNIX ENVIRONMENT

Outline

- UNIX History
- UNIX Standardization
- UNIX Influence on TAE Design
- UNIX-specific TAE features.
- Major Porting Issues
- Current TAE/UNIX Roll Call
UNIX STANDARDIZATION EFFORTS

o TAE standard:
  - UNIX Version 7 (library calls and utilities)
  - Installation with (Bourne) `sh`
  - ASYNC feature requires BSD 4.1 or 4.2
  - `tar` tape format
  - Goal: one tape fits all

o /usr/group proposed standard:
  - Based on System III+
  - Expecting IEEE, ANSI, ISO adoption in 1985
  - System calls only (no shell commands)
  - No termcap or ioctl details for terminals
  - No intertask communication (other than pipes)

o AT&T Standard: System V, Release 2
UNIX INFLUENCE ON TAE DESIGN

- Hierarchy search for TAE procs.
- Symmetry of procedure and process.
- File name as command name (i.e., no "RUN" command).
- C language.
- Use of sequential text files (PDFs, MDFs, and Help).
- Documentation (negative influence).
- Error Messages (negative influence).
UNIX-SPECIFIC FEATURES IN TAE

- TAE> ush any-shell-command
- Shell symbol substitution in file names.
- Use of termcap for terminal independence.
- Conditional help files:
  .if UNIX
  this is a line of UNIX help
  .if VMS
  this is a line of VMS help
- De-bugger activation under TAE.
MAJOR PORTING ISSUES

Problem areas:
- Availability of development machines.
- UNIX incompatibilities.
- Development of one "UNIX portable" installation tape.
- Re-porting of each new TAE release (6 weeks).
- Other details:
  - Interprocess communication.
  - Signal catching by the de-bugger.
  - Terminal handling (forced into loss of type-ahead).

The following tended not to be problems:
- Bugs in "portable" code.
- UNIX learning curve.
TAE/UNIX ROLL CALL

- VAX BSD 4.1 (EDC).
- SUN workstation BSD 4.1 (EDC and GSFC).
- VAX BSD 4.2 (EDC).
- GOULD/SEL 4.1C (EDC).
- Apollo Domain "Version 7" (University of Maryland).
- Unknown installations?
- Planned: IRIS System "5" (GSFC).
- Planned: CONVEX BSD 4.2 (GSFC)
IMPLEMENTATION OF THE ON AN APOLLO NETWORK

JAMES N. COOPER
DEPT. OF METEOROLOGY, U. OF MARYLAND
COLLEGE PARK, MD 20742

MAY, 1985
1. Introduction

The Department of Meteorology at the University of Maryland is currently developing a microcomputer-based system to process and display meteorological data. After considering various options, we chose to implement the NASA/GSFC GEMPAK software, and therefore TAE, on Apollo super-microcomputers.

The Department’s present network of Apollo computers consists of two DN320’s, one of which is equipped with a 70MB disk. The DN320’s are a Motorola 68000-based machine with 1.5 to 3 MB of memory connected by a token-ring local area network. The Apollo network is incorporated into the operating system so its presence is transparent to the user. The network will be expanded in the near future with the addition of two more DN320’s and a file server.

Previously, the UNIX version of TAE had only been ported to machines running Berkeley 4.2 UNIX. In this paper, we will discuss our transport of TAE to a machine running a non-standard version of UNIX. First, we describe problems we overcame to implement TAE on our Apollo network. Then, we discuss why the initial implementation was unsatisfactory, and what steps we took to improve it.

The work described in this paper was partially supported by the University of Maryland and National Science Foundation Grant ATM-8409457.

2. Initial Port of TAE to the Apollo

The Apollo runs a version of UNIX called AUX, so we chose to implement the UNIX version of TAE. AUX co-exists with Apollo’s proprietary operating system AEGIS, and is a mix of Berkeley 4.1, Version 7, and System III UNIX. At the user level, AUX is similar to UNIX, but our attempt to install TAE under AUX uncovered two major systems-level differences.

2.1 Problems Caused by Compiler Differences

The first problems involved the compilers, because AUX does not contain any of the usual UNIX compilers. Instead, the AEGIS
compilers are used. These compilers, particularly the AEGIS FORTRAN 77 compiler, caused a great many of our TAE implementation problems.

There were four major incompatibilities between the AEGIS FORTRAN 77 compiler and a UNIX-standard compiler. Apollo FORTRAN 77

1. does not expect underscores after the names of FORTRAN 77 callable C subroutines,

2. passes character strings to subroutines differently,

3. has a different form of the INCLUDE statement, and,

4. sets true and false values in LOGICAL variables differently.

All the differences force modifications to the TAE source code, and the fourth difference creates a restriction for the FORTRAN applications programmer.

Under standard UNIX FORTRAN, when a FORTRAN program calls a C subroutine, the sequence works like this:

The FORTRAN program says
CALL XWRITE
but links to the C subroutine
XWRITE

TAE places a C bridge subroutine between the FORTRAN program and the actual C subroutine. Normally, the bridge subroutine performs no other function than to call the real C subroutine. However, for machines like the Apollo, this bridge structure provides a method to compensate for non-standard argument passing.
The normal TAE calling sequence looks like this:

The FORTRAN program says
   CALL XUWRIT
but links to the C bridge routine
   xuwrit_
which calls the actual C subroutine
   xuwrit.

Because AEGIS FORTRAN 77 does not append underscores when linking FORTRAN-callable C subroutines, the standard TAE bridge subroutines will be bypassed.

The solution to this problem is to remove the underscores from all the bridge subroutine names so they match exactly the name used by the FORTRAN program. Then, the real C subroutine names must be differentiated from the bridge names to avoid having two library subroutines with the same name. Our convention was to add an "_r" (for "real") to the names of the actual C subroutines.

The TAE subroutine calling sequence on the Apollo therefore looks like this:

The FORTRAN program says
   CALL XUWRIT
which links to the C bridge subroutine
   xuwrit
which calls the actual C subroutine
   xuwrit_r.

The bridge subroutine structure just described is also used to compensate for Apollo's non-standard method of passing character strings to a C subroutine. Both UNIX and AEGIS FORTRAN 77 normally pass subroutine arguments by reference. If, however, one of those arguments is a character string, then an extra argument is tacked onto the C subroutine parameter list. In UNIX FORTRAN 77, the extra argument is the actual length of the character string in a 32-bit integer. In AEGIS FORTRAN 77, however, the extra argument is a 16-bit integer (pointer) holding the address of a 32-bit integer containing the length of the character string.
string. Thus, the bridge subroutines must be changed to compensate for this difference before the actual C subroutine is called.

The final AEGIS/UNIX FORTRAN 77 incompatibility involves the way Boolean variables are set to true and false. AEGIS reserves 4 bytes for logical variables, but only sets the leftmost byte to true or false (00 or FF, respectively). When a TAE subroutine determines if a variable is true or false, it checks to see if all bits are set to zero or if any bit is set to 1. Thus, if an AEGIS FORTRAN 77 logical variable is set to true (00), but has garbage in any of the three righthand bytes, the TAE subroutine will interpret the value as false. To overcome this problem, all bits in a FORTRAN 77 logical variable are set to zero before being passed to a TAE subroutine.

2.2 Problems Caused by Missing Library Utilities

Aside from the compiler differences AUX also lacks the UNIX library utilities "lorder" and "ranlib". Furthermore, although the basic library utility "ar" exists, it does not produce a library format that the AUX "ld" command can use. These differences forced us to modify the entire TAE library-building command sequence in the installation script so that the AEGIS command "lbr" could be used. Unfortunately, "lbr" does not make a random access library. The proper order for subroutines in the library therefore becomes important, since a subroutine cannot be linked to another located before it. The existing alphabetical list in the installation script was replaced with a properly ordered list.

According to Apollo, the next release of their UNIX will contain the missing utilities, and "ar" will work properly.

3. Producing an Apollo TAE

Once we overcame these differences, we could successfully install TAE on our Apollo network. In a short time, however, we realized that the implementation was not satisfactory. The TAE display was operating with very slow throughput (about 1200 baud). Compared to the normal throughput of the Apollo display (about 19200 baud), we found the slow rate unacceptable and
One of the strengths of the Apollo is the user interface, known as the Display Manager. The Display Manager provides an excellent work environment with windows and a full screen editor. Unfortunately, AUX is configured so that a program cannot use the Display Manager and the Berkeley UNIX termlib/termcap facility at the same time. (See Appendix A for a description of the termlib/termcap facility.) TAE uses the termlib subroutine “getent” to obtain the ASCII escape sequences needed for screen control from the termcap file. In AUX, “getent” has been modified to automatically startup a vt100 terminal emulator. The vt100 emulator puts an additional layer between TAE and the display, slowing the throughput. In addition, the Display Manager loses control of that window, so its “nice” features cannot be used.

Our next step was clear: Develop an extension of TAE that is able to more fully utilize the power of the Apollo display. The extension of TAE was developed in two phases. The goal of Phase I was to improve the display throughput of TAE on the Apollo. The goal of Phase II was to use the windowing capabilities of the Apollo. The guiding principle in both phases was to achieve maximum functionality with a minimum of TAE source code modifications.

There were three main steps required to meet the Phase I goal:

1. Use TAE’s internal termcap/termlib facility instead of the AUX version to prevent the startup of the vt100 terminal emulator.

2. Configure a standard Apollo window to allow cursor control, line clearing, etc, via ASCII escape sequences.

3. Undo the TAE design choice of unbuffered output to increase the display throughput and permit recognition of the ASCII escape sequences.

Fortunately, the UNIX version of TAE comes with its own termcap/termlib facility that has the same subroutine calls and
performance as the Berkeley version. This facility was originally provided so that TAE could be ported to systems running Bell instead of Berkeley UNIX. By building Apollo-TAE with TAE's internal termlib/termcap facility, the call to "getent" was linked to the TAE version instead of the AUX version thereby eliminating the vt100 emulator startup.

The next problem was to get the standard Apollo window configured to accept ASCII escape sequences for cursor movement, screen clearing, etc. Figure 1 shows how an Apollo window is divided into output and input windows. Normally, the output window does not permit cursor control. In Apollo terminology, it is in "line mode". However, by calling an Apollo system subroutine, the window can be put into "frame mode". This configuration enables recognition of ASCII escape sequences for cursor control. Frame mode provides enough capability to implement the full screen version of TAE. Thus, the TAE source code was modified to shift the output window into frame mode whenever a menu, help information, or full screen tutor were displayed. When the user returns to command mode or invokes the noscreen tutor, the output window is returned to line mode.

After completing the first two steps, TAE functioned properly only with the output window in line mode. The display throughput was still slow compared to normal Apollo speed and the escape sequences issued by TAE for screen control were not being recognized. Further testing showed us that TAE performs terminal output one character at a time (unbuffered). This was a conscious design choice made by the developers of TAE to avoid having to flush a buffer to produce output. With the unbuffered output, throughput is slowed since the Apollo generates I/O overhead for each character. Also, the Apollo cannot correctly interpret the ASCII escape sequences since it sees each individual character instead of a single command. By eliminating the UNIX subroutine call that unbuffered the output, the display speed increased dramatically and the ASCII escape sequences were interpreted correctly. For example, the TAE acceptance test on the initial version took about 13 minutes to complete. On the new version, the test was finished in about 7 minutes.

Because of the modular nature of TAE and the existence of its own termcap/termlib facility, the steps outlined above were
$ date
Monday, May 6, 1985  11:04:00 am (EDT)
$ lcnnode

The node ID of this node is 1614.
1 other node responded.

<table>
<thead>
<tr>
<th>Node ID</th>
<th>Boot time</th>
<th>Current time</th>
<th>Entry Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1614</td>
<td>1985/05/01 14:06:44</td>
<td>1985/05/06 11:04:49</td>
<td>//huis</td>
</tr>
</tbody>
</table>
| 3680   | 1985/05/01 15:22:58 | 1985/05/06 11:04:55 | *** DISKLESS *** partner node: 1614

$ RUNITM

Figure 1. An Apollo window
implemented with only minor source code changes. The resulting version of TAE maximized the performance improvement while minimizing source code disruption. Table 1 gives a detailed breakdown of the modifications required in Phase I.

The second phase of TAE improvements on the Apollo was to utilize the multiple window capability of the Apollo display. The TAE development group at NASA has been working with Century Computing to produce a version of TAE that operates with multiple windows. A prototype has been developed that runs under VMS on a VAX vt220 terminal and a VAX STATION. Although the two display devices are quite different (the VAX STATION has a bit-mapped screen, the vt220 does not), the user sees the same display format. While the code inside the window subroutines is different for each device, the location of the subroutine calls in the TAE source code is device-independent. Since we wanted the Apollo version of TAE to follow any standards set by NASA, our window version of TAE was created by inserting the window subroutine calls at those same locations, but replacing the subroutines' "guts" with Apollo system calls. As a result, the format of Apollo-TAE using windows is the same as the NASA prototype. An example of this version of Apollo-TAE is given in Figure 2.

4. Conclusion

Our transport of TAE to the Apollo was the first attempt to put the UNIX version on a machine running a non-standard UNIX. A moderate amount of work and modification to TAE was required for a successful installation. Most of the problems were caused by the non-standard compilers and missing library utilities in Apollo UNIX. The initial Apollo-TAE functioned correctly, but with unacceptable display throughput. We therefore developed an extension of TAE that utilized some Apollo system subroutines. This version of Apollo-TAE has an improved display speed, and uses the windowing capability of the Apollo. Because of TAE's modular nature, this new version was produced with a minimum of source code disruption.
<table>
<thead>
<tr>
<th>CHANGE DESCRIPTION</th>
<th>SOURCE MODULE AND SUBROUTINE NAME</th>
<th>LINES REMOVED (-) OR ADDED (+)</th>
</tr>
</thead>
</table>
| Switching in and out of frame mode | MENU.C/menmod
DYNAMIC.C/dynget
TUTGR.C/sub_tutor
MISCCMD.C/getparm_do | +4
+3
+3
+3 |
| Switching into frame mode | TUTOR.C/tutscreen | +3 |
| Switching out of frame mode | TUTGR.C/tutnoscreen | +3 |
| Removed C system call that unbuffered output | TERMINAL.NP.C/t_init | -1 |
| Removed C I/O call that echoed input | TERMINAL.NP.C/read_terminal | -1 |

TABLE 1. Breakdown of TAE source code modifications required to meet Phase I goals
Figure 2. Display format of Apollo-TAE using windows
Appendix A

A Description of the Berkeley UNIX termcap/termcap Facility

The Berkeley termcap/termcap facility consists of a terminal capability database (/etc/termcap) and a library of subroutines (/usr/lib/libtermcap.a). The database contains a coded description of the capabilities of a wide variety of terminals plus the escape sequences used to execute those capabilities. Additional entries can be added by users. The library subroutines perform such functions as obtaining information about a specific terminal from the termcap file and encoding cursor movement strings. The presence of this facility enables Berkeley UNIX to contain programs that require screen control (i.e., the full screen editor "vi") without restricting users to a single terminal type.
NEW TRANSPORTABLE LAND ANALYSIS SYSTEM

Software Development Section
Computer Services Branch

EROS Data Center

Tony Butzer
**TABLE OF CONTENTS**

1 Introduction .......................................................... 1
  1.1 Why UNIX ...................................................... 1
  1.2 Document Organization ......................................... 2

2 Overall Design Philosophy ........................................... 3
  2.1 Modularity .................................................... 3
  2.2 Isolation and Elimination of System Dependencies .............. 3

2.3 Simpler and More Flexible Designs ................................ 4
  2.4 UNIX Tools ................................................... 4
  2.5 Concurrent Development on Heterogeneous Machines ............ 5

3 Design Issues .......................................................... 5
  3.1 Imageio ....................................................... 5
    3.1.1 Modularity .............................................. 5
    3.1.2 System Dependencies ................................... 6
    3.1.3 Simpler Design ........................................ 6
    3.1.4 UNIX Tools and Concurrent Development ................. 6
    3.1.5 Performance ............................................ 7
    3.1.6 Random I/O ............................................. 7

  3.1.7 Compatible File Format ................................... 7
  3.2 Label Services or Related Files .............................. 8
  3.3 Pixel Manipulation .......................................... 8
  3.4 Catalog Manager ............................................. 9
    3.4.1 VMS .................................................... 9
    3.4.2 UNIX .................................................. 9
  3.5 Tapeio ....................................................... 10
  3.6 Error Handling ............................................... 10

4 Overall LAS Project .................................................. 11
  4.1 Production Control Applications ............................ 11
  4.2 Application Branch Functions ............................... 12
  4.3 Current Status ............................................... 14
  4.4 Application Conversion Effort ................................ 16
  4.5 Stages of the Software ..................................... 16
CHAPTER 1
INTRODUCTION

1. Introduction

This paper describes the conversion of the Land Analysis System (LAS) currently being implemented at the EROS Data Center. The goal of the conversion effort is to write a set of library and application modules that perform the same functions on both VMS and UNIX.

In writing this software several items had to be considered. First, since there are over 200 programs written in Fortran for VMS, how the new libraries would impact this code and how they would interface to Fortran on UNIX and VMS had to be looked into. Second, the impact of operating system limitations imposed by UNIX such as the ability to open only 20 files at a time had to be considered in the design. Third, compatibility with other image processing tasks such as LAMS and TAE based systems was considered to be a primary objective to avoid unnecessary data conversions between image processing systems. Fourth, the system had to be optimized for performance because of the amount of processing and I/O that is done in image processing tasks. And finally, the system had to be accepted by its user community in order for it to be considered a success, so considerable time was spent talking to the users of LAS to find out what they needed in a Land Analysis System.

1.1. Why UNIX

No one will question the value of running the same software on a variety of machine architectures without changing a single line of code. Imagine running a task on a workstation or the same task on a supermini with a larger data set. In addition what happens in three years when I can buy ten times the processing power at half the cost and half the floor space, but I have all this software that will only run on processor "X". It is precisely these reasons why UNIX becomes so attractive. UNIX allows users and programs to be move from one system to another with minimal impact on
either. It also allows for the computer power of the future to run the well designed software of today.

1.2. Document Organization

This document will discuss the Land Analysis System project following areas:

Design;
Implementation Approach of the Overall Project;

DRAFT May 29, 1985
2. Overall Design Philosophy

The design of the LAS Libraries discussed in the following paragraphs reflects these techniques in writing transportable code:

1. Modularity;
2. Isolation and Elimination of Operating System Dependencies;
3. Simpler and more Flexible Designs;
4. UNIX Tools;
5. Concurrent Development on Heterogeneous Machines.

2.1. Modularity

Modularity is nothing new in the computer science discipline, whether it is called Top Down Development or Structured Programming it really boils down to writing modules that perform one very specific function. What is also important is at what level unrelated functions are bound to form the program. In design of the support libraries we attempted to segregate these functions as much as possible. This avoids the all or nothing syndrome that occurs because your libraries depend on having other libraries already written, so you don't have anything until you have everything.

The binding of the new LAS libraries will be at the Application Level or an intermediate level wherever possible.

2.2. Isolation and Elimination of System Dependencies

In designing the LAS libraries we eliminated the use of assembly language routines to perform functions that were
not time critical. With todays optimized compilers many functions written in high level languages such as C can generate as efficient code as the most scrutinizing assembly language programmer and not have the portability problems innate to assembly language.

In addition, through the use of base TAE I/O routines we constructed a Imageio Library that is the same source code on both VMS and UNIX, thus isolation of the operating system dependent I/O was already done for us by TAE.

We also avoided use of elaborate ISAM files when a simple sequential file format would be more than adequate. This eliminates the use of system dependent file formats. Along the same lines attempts were made to keep the files all one data type to avoid transfer problems between different systems.

Through the use of an error routine, system dependent error messages are also replaced by portable defines which can then be interpreted the same on all systems.

2.3. Simpler and More Flexible Designs

The subject of simpler and more flexible designs includes the above two topics. In addition means keeping the amount of entry points and the complexity of the routines as simple as possible to provide the functionality required.

Communication with the user is of great value in designing the system. This eliminates problems of both over and under design of the project.

2.4. UNIX Tools

The use of UNIX tools in testing functions and building the libraries can greatly decrease the development and debugging time for the project.

Lint is a UNIX tool that will type check variables being passed to subroutines and make sure that there are the right number. It also checks for unused variables and variables which are not set but are used.
The C-Shell provides features such as history and aliases that help the programmer avoid unnecessary keystrokes in the development process.

The utility Make allows for programs to be written in a highly modular fashion and only compiles those modules which have been modified, thus decreasing compile time and the amount of CPU cycles used.

The Dbx debugger is of great value when testing C code. Among its most attractive features are the ability to list out data structures and lines of source code.

2.5. Concurrent Development on Heterogeneous Machines

One of the best ways to really write transportable code is to develop it on all of the target systems at the same time. This concept is considerably easier if you happen to have a local area network. In developing the LAS libraries, I would code a module in C on the Gould 32/87 and test it. Then ship it via the Network Copy Program to the VAX VMS machine and test it there. It most cases the routine will work as expected on the second system, but in the case where a portability issue was overlooked it will quickly show up when this actual port is done.

3. Design Issues

3.1. Imageio

In designing the Imageio Library all of the above issues were addressed as well as emphases on performance and how to best interface to both Fortran and C in a VMS and UNIX domain. Compatibility in image file formats was also a concern.

3.1.1. Modularity

Each entry point is located in a separate file by the same name and calls further subroutines to perform specific tasks it might need. An application does not require any other libraries to use the Imageio Library.

DRAFT May 29, 1985
3.1.2. System Dependencies

The same source code runs on both VMS and UNIX systems. The system dependencies were already isolated by using the existing TAE I/O subsystems on each respective system. (see diagram below)

```
* LAS IMAGEIO LIBRARY *
*
* UNIX TAE I/O   VMS TAE I/O *
*
* UNIX I/O       VMS QIO's *
*
* UNIX 4.2 BSD   VMS 3.7 *
*
```

3.1.3. Simpler Design

The Current LAS Imageio required the calling program to use some combination of 21 subroutine calls. These 21 calls were replaced by four calls that will always be executed in the same order in each application.

3.1.4. UNIX Tools and Concurrent Development

The use of the UNIX tools mentioned above will be done throughout all of the new LAS software, as well as use of the network for shipping program and data files between the systems.

DRAFT May 29, 1985
3.1.5. Performance

The designers of the current VMS LAS Imageio Library did many things to make LAS Imageio perform extremely well and also take some of the burden off the application in the data conversion area. We felt that we could learn a great deal in the area of performance from looking at some of the techniques used in the current LAS Imageio.

The use of a large data buffer in the users space to be filled by Imageio was done in the current and the new Imageio Library so that whenever possible data could be read directly into the users buffer without having to be moved in memory. In addition the new LAS I/O has a define which can be changed to read a number of lines ahead of the application in anticipation of future need for these lines. These two things alone greatly increased the amount of data throughput seen by the application over just doing line by line processing with TAE.

Another advantage of looking at the current LAS Imageio was to provide a similar interface from a functionality standpoint to the application. This will allow most of the algorithms to remain the same and only a change in the I/O calls will necessary.

3.1.6. Random I/O

In addition to providing the current capabilities of the Imageio, the addition of random I/O was also made for those applications that may require this capability. However use of random I/O for a sequential application will be accompanied by a severe degradation in performance.

3.1.7. Compatible File Format

We also are striving for a single file format, that of TAE, to underly as many different applications as possible, so we can share data files without conversion among various image processing systems. (LAMS, LAS, DMS, etc) Another reason for using TAE I/O was to store multiple bands per file due to a limit of 20 open files per task under UNIX.
3.2. Label Services or Related Files

The design of the Label Services Library was approached as a series of related files that will be associated with the main image data set.

The VMS Label Services placed every type of related information for all image files in a persons directory in one large ISAM file that had to be searched to find the history records or the look up table record or any other records that were needed to process a given image. This was especially a problem when the image was to be stored offline since the related records for that image would have to be extracted and deleted and then merged again when the image was brought back online.

As an example user Quirk might have the following Catalog Manager files.

```
#QUIRK.NEWYORK.DAT

----> #QUIRK.NEWYORK.DAT.HIS

----> #QUIRK.NEWYORK.DAT.LUT

----> #QUIRK.NEWYORK.DAT.STT
```

The the first file is the main image file that might be a three band image of New York and the following files would be associated data files for history, look up table and statistics.

Each type of associated file will be managed by a service well suited to process that type of data.

3.3. Pixel Manipulation

These routines perform arithmetic and logical operations at the pixel level. The VMS Pixel Manipulation Library consists of over 50 small subroutines performing these
operations. Many of these functions are never called and others can be combined into a single routine where there are now four. (ie. one for each data type)

Our approach to converting this Library will be to write these functions as applications that depend on them are written. This will assure that only needed functions are written.

3.4. Catalog Manager

The need for the Catalog Manager directly by the LAS Library routines has been greatly reduced by the elimination of image file specific attributes from the Catalog Manager. However the interface for creation of catalog files and getting the host name functions will still be needed for the Label Services Routines.

3.4.1. VMS

The VMS domain currently has a working Catalog Manager and the calls to retrieve and store files by the catalog manager will be utilized on VMS.

There is however a deficiency in the way the Catalog Manager archives files. Currently the user is responsible for archiving any associated files that an image file might have. This manual process should be eliminated by the Catalog Manager either automatically archiving these files or at least prompting the user as to whether he wants these files archived.

In addition no program callable subroutines have been provided in the Catalog Manager for associating files or finding out which files are associated with a given file. These calls are necessary for the Label Service Routines as well as other applications.

3.4.2. UNIX

Since we do not yet have a Catalog Manager on UNIX, we are writing the necessary routines to be used by the applications and Label Services in abbreviated form. These routines are xbfile, xbcat, xbhost. While these routines
will not provide full Catalog Manager capabilities they will provide the bases for development of LAS and can be later replaced when Catalog Manager phase II is converted to UNIX.

We are also writing the callables for associations that are not currently provided by the Catalog Manager on VMS and are not available under UNIX either.

3.5. Tapeio

The new tapeio provides a system independent interface to tape. In addition the new Tapeio Library does not require the user to know the tape drive his job is using. A message is sent to the operator on the open asking for a tape drive and the operator responds directly to the job the tape unit number or that no tape drives are available at this time.

3.6. Error Handling

The error reporting allows for both fatal and warning messages to be printed on fatal error the program will abort after the message is printed. The message consists of the date and time, a key, and a short error message. The key allows for a search of another more detailed error file which can provide additional information if needed.
CHAPTER 3

IMPLEMENTATION

4. Overall LAS Project

Because of the scope of this project and the urgency to move to UNIX at EDC the project is being broken down in subsets of applications that provide an overall capability. Currently Applications Branch and Production Control have submitted a list of applications that each considers a useful subset.

4.1. Production Control Applications

The following is a list of applications that the Production Analysts require to do Custom Image Processing Tasks. They would also like some sort of display capabilities.
### Production Analyst's List for Custom Image Processing (CIPS)

<table>
<thead>
<tr>
<th>Priority</th>
<th>Module Name</th>
<th>Equivalent AB Function</th>
<th>Manpower Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fmount</td>
<td>Fmount</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>Dmount</td>
<td>Dmount</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>Whatisit</td>
<td>Whatisit</td>
<td>1.0 week</td>
</tr>
<tr>
<td>4</td>
<td>Rewind</td>
<td>Dalin</td>
<td>0.0 weeks</td>
</tr>
<tr>
<td>5</td>
<td>Dalin</td>
<td>Dalin</td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>6</td>
<td>Dalout</td>
<td>Dalout</td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>7</td>
<td>Histplt</td>
<td>&lt; Pixcnt</td>
<td>6.0 weeks</td>
</tr>
<tr>
<td>8</td>
<td>List</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>9</td>
<td>Map</td>
<td>Map</td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>10</td>
<td>Scale</td>
<td>Scale</td>
<td>2.0 weeks</td>
</tr>
<tr>
<td>11</td>
<td>Butter</td>
<td>Butter</td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>12</td>
<td>Edipsedge</td>
<td>&lt; Filter</td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>13</td>
<td>Insert2</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>14</td>
<td>Destripe</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>15</td>
<td>Fixline</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>16</td>
<td>Cam/proj</td>
<td></td>
<td>2.0 weeks</td>
</tr>
<tr>
<td>17</td>
<td>Geom</td>
<td></td>
<td>6.0 weeks</td>
</tr>
<tr>
<td>18</td>
<td>Makeimage</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>19</td>
<td>Testgen</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>20</td>
<td>Edipsin</td>
<td></td>
<td>4.0 weeks</td>
</tr>
<tr>
<td>21</td>
<td>Edipsout</td>
<td></td>
<td>4.0 weeks</td>
</tr>
<tr>
<td>22</td>
<td>Dementer</td>
<td>Dementer</td>
<td>4.0 weeks</td>
</tr>
<tr>
<td>23</td>
<td>Ccttipsp</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>24</td>
<td>Plenter</td>
<td></td>
<td>4.0 weeks</td>
</tr>
</tbody>
</table>

Total 69.0 weeks

In addition to the above LAS functions the PA's require a display capability such as the DMS Functions.

### 4.2. Application Branch Functions

The following is a list of modules put together by Bruce Quirk that would provide a minimal Image Processing Environment under UNIX. Bruce also emphasized that because
of the problems with transferring the data between systems and the lack of the full capabilities that the VMS utilities would provide, he did not anticipate a large number of scientists using UNIX immediately.

Application Branch's List

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Equivalent PA Function</th>
<th>Manpower Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addpic</td>
<td></td>
<td>0.5 weeks</td>
</tr>
<tr>
<td>Areafilter</td>
<td></td>
<td>4.0 weeks</td>
</tr>
<tr>
<td>Bangle</td>
<td></td>
<td>4.0 weeks</td>
</tr>
<tr>
<td>Butter</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>Convert</td>
<td></td>
<td>2.0 weeks</td>
</tr>
<tr>
<td>Convolve &amp; Cwtgen Filter</td>
<td>&gt; Edipsedge</td>
<td>6.0 weeks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.0 weeks</td>
</tr>
<tr>
<td>Dalin</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>Dalout</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>Dementer</td>
<td></td>
<td>4.0 weeks</td>
</tr>
<tr>
<td>Divpic</td>
<td></td>
<td>0.5 weeks</td>
</tr>
<tr>
<td>Dmount</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Edgedet</td>
<td></td>
<td>4.0 weeks</td>
</tr>
<tr>
<td>Fmount</td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Intersect</td>
<td></td>
<td>4.0 weeks</td>
</tr>
<tr>
<td>Map</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>Max</td>
<td></td>
<td>2.0 weeks</td>
</tr>
<tr>
<td>Min</td>
<td></td>
<td>2.0 weeks</td>
</tr>
<tr>
<td>Minmax</td>
<td></td>
<td>2.0 weeks</td>
</tr>
<tr>
<td>Mulitpic</td>
<td></td>
<td>0.5 weeks</td>
</tr>
<tr>
<td>Pixcmt</td>
<td>&gt; Histplt</td>
<td>4.0 weeks</td>
</tr>
<tr>
<td>Rasterize</td>
<td></td>
<td>3.0 weeks</td>
</tr>
<tr>
<td>Scale</td>
<td></td>
<td>2.0 weeks</td>
</tr>
<tr>
<td>Slap/Topo</td>
<td></td>
<td>4.0 weeks</td>
</tr>
<tr>
<td>Spread</td>
<td></td>
<td>5.0 weeks</td>
</tr>
<tr>
<td>Whatisit</td>
<td></td>
<td>1.0 weeks</td>
</tr>
</tbody>
</table>

Total 67.5 weeks
Comparing both sets of functions there are 9 duplicated functions comprising 27 man weeks of programming effort. Thus 69 + 67.5 - 27 yields 109.5 man weeks or a little over two man years of effort to provide the above subsets of functions.

4.3. Current Status

The following chart illustrates the progress of the LAS project so far.
NEW PORTABLE LAS STATUS

<table>
<thead>
<tr>
<th>MODULE</th>
<th>PRELIM</th>
<th>DESIGN</th>
<th>CODE</th>
<th>TEST</th>
<th>DOCUM</th>
<th>PROGRAMMER</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageio</td>
<td>L</td>
<td>FF</td>
<td>FF</td>
<td>CC</td>
<td>CC</td>
<td>T. Butzer</td>
</tr>
<tr>
<td>history</td>
<td>L</td>
<td>FF</td>
<td>CC</td>
<td>CC</td>
<td></td>
<td>T. Bodoh</td>
</tr>
<tr>
<td>pixel manip</td>
<td>L</td>
<td>FF</td>
<td>CC</td>
<td></td>
<td></td>
<td>T. Butzer</td>
</tr>
<tr>
<td>tapeio</td>
<td>L</td>
<td>FF</td>
<td>FF</td>
<td>CC</td>
<td>CC</td>
<td>T. Butzer</td>
</tr>
<tr>
<td>UNIX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T. Butzer</td>
</tr>
<tr>
<td>catalog</td>
<td>L</td>
<td>FF</td>
<td>CC</td>
<td></td>
<td></td>
<td>T. Bodoh</td>
</tr>
<tr>
<td>emulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T. Butzer</td>
</tr>
<tr>
<td>mount</td>
<td>A</td>
<td>RR</td>
<td>RR</td>
<td>RR</td>
<td>RR</td>
<td>T. Bodoh</td>
</tr>
<tr>
<td>dismount</td>
<td>A</td>
<td>RR</td>
<td>RR</td>
<td>RR</td>
<td>RR</td>
<td>T. Bodoh</td>
</tr>
<tr>
<td>whatisit</td>
<td>A</td>
<td>CC</td>
<td></td>
<td></td>
<td></td>
<td>T. Butzer</td>
</tr>
<tr>
<td>rewind</td>
<td>A</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>T. Butzer</td>
</tr>
<tr>
<td>dalin</td>
<td>A</td>
<td>FF</td>
<td>CC</td>
<td></td>
<td></td>
<td>P. Mumford</td>
</tr>
<tr>
<td>dalout</td>
<td>A</td>
<td>FF</td>
<td>CC</td>
<td></td>
<td></td>
<td>P. Mumford</td>
</tr>
<tr>
<td>PA's 7-24</td>
<td>A</td>
<td>FF</td>
<td>CC</td>
<td></td>
<td></td>
<td>A. Programmer</td>
</tr>
</tbody>
</table>

FF ---> Finished  
CC ---> Current work  
RR ---> Replaced  
L ---> Library  
A ---> Application

DRAFT May 29, 1985
4.4. Application Conversion Effort

The design changes that have been made in the LAS Libraries are significant and the calling programs will be greatly affected. However the basic functions performed by the libraries will remain the same. This means that the algorithms used in the LAS Applications should remain the same and that much of the code that was raw Fortran 77 can go untouched.

The things that will change are how the new LAS routines will be called and what an image file looks like underneath. There will no longer be file groups but multiple bands will be stored in a single file.

4.5. Stages of the Software

The eventual goal of this project is to have one version of LAS a portable one that runs under both VMS and UNIX with the same set of applications and the same data file format. The following diagram shows the major steps required to achieve this goal.

<table>
<thead>
<tr>
<th>UNIX</th>
<th>Transferring Data</th>
<th>VMS</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>New LAS Libraries and CIPS or AB Applications. (without Catalog Manager)</td>
<td>Transfer and convert.</td>
<td>Existing LAS Libraries and ALL LAS applications</td>
<td>New LAS Libraries &amp; applications in testing on VMS.</td>
</tr>
<tr>
<td>New LAS Libr. CIPS and AB Applications (with Catalog Manager Phase II)</td>
<td>Transfer and convert.</td>
<td>Existing LAS</td>
<td>New LAS in test stage on VMS.</td>
</tr>
<tr>
<td>All of New LAS Completed.</td>
<td>Transfer</td>
<td>New LAS Completed.</td>
<td>Existing LAS Replaced.</td>
</tr>
</tbody>
</table>

DRAFT May 29, 1985
1. Introduction
   A. Conversion
   B. VMS and UNIX (same source and functionality)
   C. Considerations
      i. Fortran
      ii. Operating System Limitations
      iii. Compatibility and Networking
      iv. Performance
      v. User Acceptance

1.1 Why UNIX
   A. Portability
   B. Architectural Flexibility
   C. Future Machines
   D. User Environment

1.2 Why VMS
   A. Existing Hardware
   B. Wide Variety of Peripherals

2. Design

2.1 Modularity
   A. Binding at Application Level
   B. vs. Binding at Library Level

2.2 System Dependencies
   A. Isolation
   B. Elimination
      i. File Format
      ii. Assembly Language

2.3 Simpler Designs

2.4 UNIX Tools
   A. Lint
2.5 Concurrent Development

A. Network
B. Portability Questions

3. Design Issues

3.1 Imageio

A. Modularity
B. System Dependencies
   i. C (instead of assembly language)
   ii. TAE I/O
   iii. Fortran Friendly (string descriptors)
C. Simpler Design
   i. 21 vs. 4 Entry Points
D. UNIX Tools (aforementioned)
E. Performance
   i. Current LAS Considerations
   ii. Large Data Buffer
   iii. Predictive I/O
F. Random I/O
G. Compatible File Format
   i. Among Different Application Subsystems. (TAE, LAMS, etc)
   ii. Across Machine Architectures (DEC, SUN, GOULD, etc)

3.2 Label Services

A. Flexibility
B. Associated Files
   i. History
ii. LUT
iii. Statistics
iv. etc.

C. File Format.
i. Sequential
ii. Single Data Type

D. Communication with User
i. Over Design
ii. Under Design

3.3 Pixel Manipulation
A. Small
B. Combine Routines

3.4 Catalog Manager
A. XB's
B. Associations
C. VMS Phase II
D. UNIX Phase 0 (used so application development could begin)

3.5 Tapeio
A. User Interface
B. Operator Interface

3.6 Error Handling
A. Keys
B. Messages

4. LAS Project
4.1 Production Control
4.2 AB (minimum subset)
4.3 Application Conversion Effort
A. Significant
B. Algorithms Remain Intact
C. Calling Sequences

4.4 Stages of Software
Global Imaging has developed an interactive digital image processing workstation for Earth remote sensing applications. This turn-key system provides the capability to process imagery from commonly used Earth observation spacecraft in conjunction with in situ data sets. We have extended NASA's Transportable Applications Executive (TAE) to provide essential image processing capabilities not available in its original version.

The system hardware is based on the Hewlett-Packard 9000, a high-performance 32-bit processor (CPU), with a direct address range of 500 megabytes. A separate input/output processor (IOP) frees the CPU from functions associated with direct memory access by peripherals such as disk drives and displays. The modular design of this computer permits multiple CPUs and IOPs to reside on the same bus to provide increased performance when necessary.

The Metheus Omega 500 display controllers drives the color CRT display. The controller memory may be configured to hold 1280x1024x8 or 640x512x32-bit images. Images are displayed at 60 Hz non-interlaced refresh rate using bright color monitors. The custom bit-slice processor contained in the Omega 500 has a cycle time of 167 nanoseconds and can flash-fill rectangles at 35 million pixels per second.

Workstation software includes the Global Applications Executive (GAE), a library of general purpose applications functions, and software packages for analyzing data from the Advanced Very High Resolution Radiometer, Coastal Zone Color Scanner, and the Visual and Infrared Spin Scan Radiometer. GAE, which standardizes the link between the user and the applications program, runs under UNIX and is an extension of TAE version 4.1. All TAE commands except Disable Log and Vicar are available under GAE. Unlike TAE 4.1, when a command is not recognized as a GAE command, GAE assumes the command is a reference to an executable file. The UNIX operating system searches for this file via the
path specified by the user. The session log is enabled at GAE log on and cannot be disabled. A session history can be obtained at any time by typing "session" or "pr session.tsl | lpr".

Along with GAE, Global Imaging supplies utilities for parameter manipulation, terminal i/o, sequential text file i/o, display controller i/o, string manipulation, and image file access. The calling sequences for all routines, except those included in the image file access and display controller i/o packages, are identical to those supplied with TAE 4.1. Parameters of type infile and outfile are not supported under GAE. Instead the multi-valued strings "in" and "out" are used to specify input and output images. Input images are specified as follows:

```
in = image name [bandlist] (sl ss nl ns)
```

where sl is the starting line; ss, the starting sample; nl, the number of lines; and ns is the number of samples. Image bandlist and subsection are optional modifiers. Images can be of any size, any data type from 8 bits per pixel to 64 bits per pixel, and can contain up to 64 bands.

Applications programmers refer to images by number. The first image specified in the list of input images is number 1, whereas the last image in the output image specification is nids + nods. Nids is the number of input images and nods is the number of output images. Utility subroutines are available for opening and closing input and output images, reading from input images, reading from and writing to output images, and checking image file i/o errors. Other routines return the number of input and output images, the name of the image, and the bandlist and subsection information specified by the user. When reading or writing to images, the system automatically converts data to the type specified by the programmer.

GAE utilities are also available for creating and maintaining image related files and for handling display controller i/o. Examples of image related files are history, navigation, and calibration files. Any number of image related files can be created by a programmer. Display controller utilities include routines for reading data from and writing data to the display, positioning the graph pen, drawing character strings on the display, drawing lines on the display, and reading the graph pen position.

Global Imaging offers a growing library of general purpose applications and display controller functions for black and white and pseudocolor image analysis. These functions include the display of monochrome and true color images, the display of graphics data, image arithmetic, creation of synthetic images, edge detection, and histogram computation. Graphics data, such as contour maps, can be superimposed on images in different
colors. With the Hewlett-Packard 9111A graphics tablet it is possible to interactively manipulate the contrast and brightness of an image. Other interactive functions include the display of image intensity, pan, and image expansion.

Software packages are also available for analyzing data from the Advanced Very High Resolution Radiometer (AVHRR), Coastal Zone Color Scanner (CZCS), and the Visual and Infrared Spin Scan Radiometer (VISSR). All of these packages include programs for geometric correction, navigation, and registration of the data. The AVHRR package also includes a program for radiometrically calibrating the data.
A

Transportable Display Management Subsystem

May 30, 1985

By: Kenneth P. Johnson

EROS Data Center
Computer Services Branch
Software Development Section
Sioux Falls, SD 57198
# Table of Contents

1 Introduction ........................................ 1
2 Porting DMS ........................................ 2
  2.1 Shared Memory .................................... 3
  2.2 String Descriptors ............................... 5
  2.3 Variable Argument Lists ......................... 6
  2.4 System Service Functions ......................... 7
  2.5 File System Interface ............................ 7
  2.6 Global Data Structures ........................... 8
  2.7 Lookup Table Functions ......................... 8
  2.8 Cursor, Function Button and Trackball Functions .... 9
  2.9 Device Dependent Functions ....................... 10
  2.10 Device Independent Functions .................... 12
3 Functional Enhancement .............................. 12
4 DMS Modifications .................................. 14
  4.1 DD Functions .................................... 14
  4.2 DM Functions .................................... 17
  4.3 DO Functions .................................... 22
  4.4 DT Functions .................................... 24
  4.5 XD Functions .................................... 26
  4.6 XL Functions .................................... 34
  4.7 XO Functions .................................... 36
  4.8 XU Functions .................................... 38
5 DMS Utilities ....................................... 39
  5.1 Arithmetic Operations ............................ 39
  5.2 Color Display Control ............................ 39
  5.3 Cursor Manipulation .............................. 41
  5.4 Graphics Overlay ................................ 41
  5.5 Intensity Mapping ............................... 41
  5.6 Mensuration Capabilities ......................... 43
  5.7 Pseudo Color .................................... 44
  5.8 Timing Functions ................................ 45
1. Introduction

The Display Management Subsystem (DMS) of the Transportable Applications Executive (TAE) was originally designed to support the development and use of image display software. DMS provides facilities that allow display programs to be easily developed and portable among raster imaging devices by providing a device independent interface.

DMS consists of the following components:

* program-callable functions for the transfer of image, graphic overlay and lookup table data to and from a raster imaging device

* program-callable functions for device control, and data display and manipulation

* program-callable functions which access disk-based image and lookup table data

* utility programs for device manipulation by an end user

The primary goal of DMS is to permit both programs and end users to access raster imaging devices without having to understand the hardware or particular configuration of a device. This paper outlines the development stages required
to make the DMS environment portable across operating systems, and to provide for extended functionality in support of the EROS Data Center (EDC) mission requirements.

EDC has been involved with DMS since the fall of 1983, when a former member of our staff participated in its initial development. Deliveries of DMS to EDC have been the following:

* First preliminary version of prototype in Oct. 84'
* Second preliminary version of prototype in Dec. 84'
* Final version of prototype in Apr. 85'

2. **Porting DMS**

In December of 1984 we decided to start the port of DMS based on the second preliminary version of the prototype. Portability deficiencies in DMS were first noted while writing device dependent code for a DeAnza IP8500 under the VMS operating system. EDC requirements dictated that the DMS environment be ported to SUN Microsystems Workstations running Berkeley UNIX (4.2 BSD) with Raster Technologies Model One/25 displays. To comply with these requirements, extensive modifications to DMS were necessary to eliminate operating system dependencies and to achieve portability.
Guidelines followed during the port of DMS were:

* **Simplify** the design.
* **Optimize** performance.
* **Minimize** the effort for implementing new raster imaging devices.

2.1. **Shared Memory**

The DMS prototype retained display control information in a shared memory area. These tables kept information describing the current state of the display. When a program terminated, the shared memory retained the information for use by the next program. Global files and a lock file are being used in place of shared memory.

One function of the shared memory was to prevent multiple allocation of any particular device. When a device is allocated, a field is set to one flagging that particular device as allocated. Other users attempting to allocate that device will be denied. These allocation flags were the key reason for the shared memory scheme. The allocation flags were moved to a global file that all users have access to. The user opens the shared file, checks a device allocation flag and if the device is not already flagged as allocated,
sets the flag. The device is then allocated to that user. Mutually exclusive access to the global files is accomplished via the creation of a lock file.

The second reason for the use of shared memory was to store information describing the current state of the device. For instance when a user displayed an image, a logical name for the image, which memories the image occupied, the type of image (bw/rgb), and other information defining that image was stored in the Display Memory Tables (DMT). This information was kept in the shared memory and could be accessed by subsequent programs run by the user. These tables need not be shared between users. These tables are placed on disk and read into memory at the start of each utility. They are accessed and updated by the utility, and rewritten to disk at the end of the utility. All other tables used in DMS are handled in the same way.

Though several systems support a shared memory facility, the shared memory concept is nonstandard, and nonportable. The shared memory tables were replaced with a set of files. At allocation time, the tables referring to the device being allocated are copied into local table files in the user's directory, and the global table files are modified to show the device is allocated (i.e. the Display Device Tables Transportable 4 DMS
(DDT) global entry for the device is modified). The local files are then used to maintain the display parameters until the user deallocates the device. At deallocation time, the local table files are copied back out to the global tables files(?), and the device is flagged as free. Since the user only requires access to one device at a time, the device dimensions of the DDT, DMT, and Image Configuration Tables (ICT) tables were eliminated. The files are used to retain information from program to program, and from user-session to user-session.

2.2. String Descriptors

The prototype was a mixture of FORTRAN and C functions and programs. When VMS-FORTRAN stores text information in memory, it constructs a small block of information called a string descriptor which describes the size and location of this character data. When a function is called by VMS-FORTRAN, the descriptor is passed (not the actual data). The DMS functions were written to accept this descriptor and decode it to find the character data. This resulted in undesirable overhead and complexity in forming and decoding string descriptors.
The prototype code was 65% C, 33% FORTRAN, and 2% VAX-VMS Macro Assembler. Unlike the prototype, the programming language assumed for DMS applications development is C rather than FORTRAN. Therefore all FORTRAN functions were rewritten in C due to a higher degree of portability inherent in C code. All FORTRAN string descriptor parameters were replaced with C pointers in all calling sequences. Wherever possible, character strings were replaced by defined constants, e.g. colors are now referred to using defined integer values. All blank filled character strings were replaced with null terminated character strings.

If the ability to write FORTRAN utilities is desirable, a set of "bridge" routines could be written for the XD and XO functions.

2.3. Variable Argument Lists

Many systems provide no method for a function to determine the number of parameters passed to it by the calling function. There were many places where DMS permitted the addition of "optional" parameters. Either the argument list was simplified or all parameters are now required to eliminate this problem.
2.4. **System Service Functions**

There were several places where operating system dependent functions were called. Wherever possible operating system dependent function calls were replaced with standard UNIX-C library and system calls. Some DT functions were eliminated since their functionality was redundant with standard C library functions.

2.5. **File System Interface**

The file handling system has traditionally been a difficult area to address. Each installation usually has its own method of reading and writing data. The prototype DMS system included an elaborate file handling system. The XL functions were used to read and write image and table data to disk in an AOIPS/2 format. The data was packed into TAE files in a nonconventional way. This made a reformat necessary before TAE files could even be displayed. Only images that were 512x512 could be displayed.

In order to eliminate the debate over which file system should be used (XL's or the local system), the XL file handling system was completely eliminated. It was replaced by four DF functions which open, read, write and close image files. Each installation will be required to rewrite these
four small functions to attach to their own file system. At EDC the DF functions are based on standard TAE I/O and are also used to manipulate the table files. This solution achieves the highest degree of portability and compatibility with existing systems.

2.6. Global Data Structures

All unused members in global data structures were eliminated. The philosophy of accessing members in global data structures was changed. The prototype version of DMS contained two application-callable functions, XDGETF and XDPUTF, to access and update members in global data structures. These functions called DMGETF, DMPUTF, and DMINKY to do the actual access and update. Some of the XD functions used the XD callables, some used the DM callables, and some used both. To avoid the overhead of multiple function calls (8 minimum per member access), and to simplify the code, the XD and XO functions access the global data structures directly. The functions XDPUTF, XDGETF, DMPUTF, DMGETF and DMINKY used for accessing members were eliminated.

2.7. Lookup Table Functions

Based on EDC requirements, the XU routines released with the DMS prototype are not currently being utilized.
Rather than saving the lookup table data of a viewed image in a data base of lookup tables, all parameters necessary to define a particular view of an image are saved in a file directly associated with the image data. The parameters saved include the entry name, the file window the data was loaded from, the bands that were loaded, the shift and zoom factors being applied, and the lookup table data. Along with these parameters, an 80 character description is stored to allow the user to further describe the entry. This scheme allows the user to specify an image name along with a saved entry name and the load routine (TODSP) can then determine the window and bands to be loaded, the zoom and shift factors and the lookup table data to be applied. Using this sequence of events, the user may recreate a DMS session at a later date using the saved parameter entries.

2.8. Cursor, Function Button and Trackball Functions

Redundant cursor, function button and trackball functions were eliminated from the graphics overlay functions. The functions XOCRRD, XOCRWR, XOWTIR, and XORINP were eliminated. The XO functions returned coordinates that were screen relative and the XD functions with the same names returned coordinates that were image relative. Our utilities and functions constrain the cursor coordinates to the Transportable 9 DMS
current image window, therefore the XO functions were not required. The supporting device dependent routines (DOCRRD, DOCRWR, DORINP, DOWTFB, DOWTIR, and DOWTTB) were moved to the DD functions. Support for cursors and trackballs is done in the XD and DD functions.

2.9. Device Dependent Functions

The SIGCORE Graphics Package available on SUN Microsystems Workstations is implemented with twelve device dependent functions and provides very sophisticated raster and graphics operations. From this we have learned that the number of device dependent functions should be kept to a minimum and each should be a single purpose function. In general, we attempted to make the device dependent functions as simple as possible, to minimize the effort of rewriting these functions for new devices. Some of the device dependent functions contained device independent code as well as parameter checking and coordinate conversions. By putting functionality at the device independent level some of the device dependent functions were eliminated (DDTOTV, DOBOX).

The prototype version of DDTOTV

(1) received the name of the image to be transferred to the display memory,
(2) selected a logical unit number for the display memory,

(3) received information about the size and type of the disk file,

(4) set up the memory window for the display memory,

(5) set up the control loop for reading and writing the image data,

(6) read the image data from the disk file, and

(7) called a display device function to write the image data to the device.

Most of the code is not device dependent, and should not be rewritten for a new display device. The functionality of DDTOTV is now performed by XDDROP, which calls a very simple device dependent routine, DDWRIW, to write the data to the display device.

The prototype version of DOBOX called DOGPDR four times to draw the four sides of the box. This functionality is now performed by XOBOX which calls DOGPDR four times.
2.10. Device Independent Functions

The device independent functions should also be single purpose functions that provide a standard interface to the device dependent functions. Device independent functions that appear to be partially functionally redundant or multi-purpose functions should be cleaned up (XODRAW vs. XOGPDR, XOSHFT vs. XOZOOM, XDSHFT vs. XDZOOM). Functions that combine single purpose functions (XDALIN, XDBRIT, XDCONT, XDDROP, XDENGR, XDFADE, XDFRTV, XDLOOP, and XDSLIC) should be moved to another layer of functionality.

3. Functional Enhancement

While making DMS portable was our main goal, several enhancements were also made. Functional extensions to DMS were added to support

* split screen capability via windowing. This allows the application to partition up one memory to contain several different images. To support this, some of the DMT fields were moved to the ICT (window, zoom, shift, lookup tables, source file names).

* histogram processing of any window.
logical and arithmetic operators (AND, NOT, OR, XOR, ADD, DIV, MULT and SUB).

support for displaying images larger than 512x512 with automatic subsampling.

cursor support to relate screen coordinates back to original file coordinates.

During development, particular attention was given to portability, modularity, and placing functionality at the proper levels in the software to achieve clarity and optimal performance.
4. **DMS Modifications**

4.1. **DD Functions**

**DDCLR**
- Added parameters ictindex and clrwind (clrwind is a flag indicating whether to clear the image window or the entire memory)

**DDCOMP**
- Functionally unchanged.

**DDCRDF**
- Functionally unchanged.

**DDCROF**
- Added parameter cursor, to indicate which cursor to turn off.

**DDCRON**
- Added parameter cursor, to indicate which cursor to turn on.

**DDCTRN**
- Eliminated.

**DDENGR**
- Functionally unchanged.

**DDFADE**
- Functionally unchanged.

**DDGCOM**
- Eliminated.

**DDGETD**
- Functionally unchanged.

**DDGFCB**
- Eliminated.

**DDICOM**
- Eliminated.

**DDILUT**
- Functionally unchanged.

Transportable 14  DMS
DDIM2M - Functionally unchanged.

DDIM2S - Functionally unchanged.

DDLCOD - Eliminated.

DDLLOOP - Not implemented.

DDLSHP - Eliminated.

DDLUTI - Functionally unchanged.

DDLUTR - Eliminated.

DDLUTW - Parameter list changed from (memid, rgbmsk, nsegs, lshape, ldata, start, length) to (memory, lutcod, data).

DDMNDF - Functionally unchanged.

DDRDIW - Parameter list changed from (memid, buffer, npix) to (memid, line, buffer0, buffer1, buffer2).

DDRDZM - Eliminated.

DDRECD - Added parameter ictindex.

DDRINV - New routine - replaces dorinp. This routine is functionally equivalent to dorinp, except that it returns memory-relative coordinates instead of screen-relative coordinates.

DDRSET - Not implemented.

Transportable 15 DMS
DDRSHF -
   Eliminated.

DDS2IM -
   Replaced memid parameter by ictindex; reordered parameter list.

DDS2M -
   Replaced memid parameter by ictindex.

DDSDIW -
   Not implemented - applies only to IIS.

DDSDMW -
   Not implemented - applies only to IIS.

DDSHFT -
   Functionally unchanged.

DDSYSAL -
   Functionally unchanged.

DDTOTV -
   Eliminated - see xddrop.

DDUNZM -
   Eliminated.

DDVIDE -
   Added parameter display type, to facilitate handling more display types.

DDWRIW -
   Parameter list changed from (memid, buffer, npix) to (memid, line, buffer0, buffer1, buffer2).

DDXCOL -
   Eliminated - see xdxcol.

DDZMRN -
   Functionally unchanged.

DDZOOM -
   Functionally unchanged.
4.2. DM Functions

CHLISTN -
   eliminated - formerly used only by dmchar (?).

DMAGEI -
   not used.

DMALOC -
   dmalloc now reads in the global names, verifies the device, and copies the global data into a set of local table files. It also flags the device as allocated in the global files.

DMCHAR -
   call to chlistn replaced by inline code.

DMCTPC -
   not used.

DMCTYP -
   not used.

DMDCLR -
   dmdclr reinitializes the devices memories specified in the mask. Argument usrid removed; references to usrid replaced by references to global variable. Added code to read from and write to disk files instead of shared memory.

DMDCRM -
   Image name no longer converted to a FORTRAN string descriptor.

DMDEFI -
   descriptor parameter imagep was changed to a pointer. Also descriptor parameter dispp and type were changed to defines. descriptor parameter curimg was added to

Transportable 17 DMS
allow multiple images in each memory, and filwdw was added to support file coordinates. Calling sequence changed - memwdw, filwdw removed from argument list; curimg, ictind added - dmdefi will now return the ict index of the new image. In addition, dmdefi will no longer update the ict slot - the caller of dmdefi (e.g., xddrop) will now do this, instead of passing all the information to dmdefi. The curimg implementation replaces the use of the "scratch" field - this allows more flexibility in using windows in the display memories, but still allows the user to specify an an existing image. If a current image is specified, the same memories will be re-used for the new image, if not locked. If the current image == the new image, the same ict slot will also be reused.

DMDELI -
Parameter imgnam changed from a FORTRAN string descriptor to a pointer.

DMDET -
unchanged.

DMDFRE -
Added code to read ddtable from a disk file instead of shared memory.

DMDID -
descriptor parameter devnam was changed to a pointer. Added code to read table information from disk files instead of shared memory. Fortran descriptors eliminated.

DMDISD -
FORTRAN string descriptors were changed to pointers. Added code to read DDT from a disk file instead of shared memory.

DMEMMLK -
unchanged.

DMFNM -
not used.

DMGDDT -
not used.
DMGETF - descriptor parameter keyword was changed to a define. Support should be added to get DDT and DMT information. This routine is no longer called at all. DM and XD level routines that need to access table fields now do so directly.

DMGETI - not used.

DMGETM - FORTRAN string descriptors were changed to pointers.

DMGICT - FORTRAN string descriptors were changed to pointers.

DMGITYP - FORTRAN string descriptors were changed to pointers. Display type names were put into a static character array.

DMICTI - Removed devid from argument list - not needed. Strings changed from blank-padded to null-terminated. Changed to reflect changes in the image configuration table.

DMIFRE - unchanged.

DMIMNM - descriptor parameter imgnam was changed to a pointer. Added argument - ict index will be returned in addition to the image name for the currently displayed image.

DMINKY - descriptor parameter keyword was changed to a define. Support should be added to use DDT and DMT keywords. This routine is no longer used. DM and XD level routines that need to access table fields now do so directly.

DMLDGB - new routine to read tables from disk files.

DMLOAD - new routine to read tables from local disk files.
DMLPCK -
not used.

DMLSTI -
dmlsti - descriptor parameter blank was changed to a define, descriptor array parameter imgnam was changed to an array of pointers.

DMMFRE -
not changed.

DMMICT -
not changed.

DMMKCV -
Changed parameter max to maxx to avoid conflict with max function.

DMNAGE -
unchanged.

DMNCON -
not used.

DMNMAT -
descriptor parameter devnam was changed to a pointer. Parameter session was removed - not needed. Added code to read DDT from a disk file instead of shared memory.

DMNMQV -
FORTRAN string descriptors were changed to pointers.

DMNVER -
descriptor parameter devnam was changed to a pointer.

DMPRDA -
descriptor parameter lognam was changed to a pointer. It is not possible to allocate/deallocate devices on UNIX, so the UNIX version of this routine doesn't do anything - just returns.

DMPUTF -
descriptor parameter keyword was changed to a define. This routine is no longer used at all. DM and XD level routines that need to access table fields now do so directly.
EROS Data Center

DMSERR -
new routine - combines the functions of xlperr, xdperr, and dmserr.

DMSETM -
unchanged.

DMSTGB -
new routine to write tables to disk files.

DMSTORE -
new routine to write tables to local disk files.

DMSYSAL -
It is not possible to allocate/deallocate devices on UNIX, so the UNIX version of this routine doesn't do anything - just returns.

DMSYSCK -
It is not possible to allocate/deallocate devices on UNIX, so the UNIX version of this routine doesn't do anything - just returns.

DMSYSDA -
It is not possible to allocate/deallocate devices on UNIX, so the UNIX version of this routine doesn't do anything - just returns.

DMSYSPD -
not used.

DMTBLA -
this routine now locks a set of global files.

DMTBLF -
this routine now unlocks a set of global files.

DMTINI -
Changed to reflect changes in the image configuration table.

DMUSER -
unchanged.

Transportable 21 DMS
4.3. DO Functions

DOBLCH - Not implemented.

DOBOX - Eliminated.

DOCINI - Functionally unchanged.

DOCRRD - Eliminated (functionally replaced by DDCRRD).

DOCRWR - Eliminated (functionally replaced by DDCRWR).

DODRAW - Eliminated (functionally redundant to DOGPDR and DOGPER).

DOGPAN - Eliminated CENTR, LENGTH, and WARN parameters from DOGPAN.

DOGPCL - Functionally unchanged.

DOGPCO - Modified to receive integer values for red, green and blue from XOGPCO.

DOGPDR - Functionally unchanged.

DOGPER - Functionally unchanged.

DOGPOF - Functionally unchanged.

DOGPOPON - Functionally unchanged.

DORDZM - Eliminated (information returned stored in DMS tables).

Transportable 22 DMS
DORINP -
   Replaced by ddrinp.

DORSET -
   Functionally unchanged.

DORSHM -
   Eliminated (information returned stored in DMS tables).

DOSHFT -
   Eliminated (functionally redundant to DOZOOM).

DOWTFB -
   Eliminated (functionally replaced by DDWTIR).

DOWTIR -
   Eliminated (functionally replaced by DDWTIR).

DOWTTTB -
   Eliminated (functionally replaced by DDWTIR).

DOZOOM -
   Functionally unchanged.

Transportable 23 DMS
4.4. DT Functions

BLANKS -
    not used.

CTOFOR -
    not used.

DTATOI -
    not used.

DTBCPY -
    not used.

DTBITS -
    not used.

DTCKSP -
    changed FORTRAN string descriptors to pointers. Removed length parameter. (All strings are now null-terminated rather than blank padded, so the length parameter is no longer needed.)

DTDONE -
    not used.

DTFNDC -
    not used.

DTI2A -
    not used.

DTIUNQ -
    unchanged.

DTMONS -
    not used.

DTR2A -
    not used.

DTRLEN -
    Changed FORTRAN string descriptors to pointers.

DTRPLY -
    Changed FORTRAN string descriptors to pointers. Eliminated gotos
DTSCMP -
  Removed length parameter. (All strings are now null-terminated rather than blank padded, so the length parameter is no longer needed.)

DTSELH -
  unchanged.

DTSUBS -
  Changed FORTRAN string descriptors to pointers.

DTTURN -
  not used.

DTUNIQ -
  unchanged.

FOR2C -
  not used.

GETBIT -
  not used.

LENSTR -
  not used.

MOVC2I -
  not used.

UPCASE -
  not used.
4.5. XD Functions

LMHELP - Converted from FORTRAN to C.

XDALIN - Converted from FORTRAN to C. changed the descriptor array parameter images to an array of pointers.

XDBRIT - Converted from FORTRAN to C. changed the descriptor array parameter imgnam to an array of pointers.

XDCELM - Converted from FORTRAN to C. changed the descriptor array parameters inrgb and outrgb to arrays of pointers.

XDCLR - Changed FORTRAN string descriptors to pointers. Added argument clrwind - indicates whether to clear window only, or whole screen.

XDCONT - Converted from FORTRAN to C. Changed the descriptor parameter imgnam to a pointer and changed the descriptor parameter exoption to a defined integer OLD or NEW.

XDCOTR - This routine just decided which of several coordinate conversion routines to call. The conversion routines are now called directly; xdcotr is no longer used.

XDCRDF - Converted from FORTRAN to C. changed the descriptor array parameters colors and rates to arrays of pointers.

XDCROF - Converted from FORTRAN to C.

XDCRON - Converted from FORTRAN to C.

XDCRRD - Converted from FORTRAN to C. Removed parameter image - use currently displayed image.
EROS Data Center

May 30, 1985

XDCRWR -
Converted from FORTRAN to C. Removed parameter image - use currently displayed image. Added code to validate the specified position.

XDDEFC -
Changed descriptor parameters config, rname, gname and bname to pointers. This routine is no longer used.

XDDEFG -
This routine has not been written for UNIX yet.

XDDEFI -
changed the descriptor parameter imagep to a pointer and changed the descriptor parameter dispp to a define. The luns parameter was eliminated. A parameter defining the original file window was added to support file coordinate translation. A parameter called curimg was added. It provides a means for a new image to be placed in the same memories as a currently existing image.

XDDELG -
This routine has not been written for UNIX yet.

XDDELI -
Changed FORTRAN string descriptors to pointers.

XDDETL -
Changed FORTRAN string descriptors to pointers.

XDDROP -
changed the descriptor parameter image_name to a pointer and changed the descriptor parameter protection to a define. The luns parameter was replaced with an array of pointers to diskfile names. An array of integers was added to specify which bands are to be used for each diskfile. A parameter defining the original file window was added to support file coordinate translation. A parameter called curimg was added. It provides a means for a new image to be placed in the same memories as a currently existing image. The descriptor parameter blankflg was changed to a define. xddrop will now load the appropriate info into the ict slot - this was previously done by dmdefi.

XDENGR -
Converted from FORTRAN to C.

Transportable 27 DMS
XDEEXIT -  
Changed FORTRAN string descriptors to pointers.  
Changed to use C-callable TAE routines.  Added code to 
write tables to local disk files instead of shared 
memory.

XDFADE -  
Converted from FORTRAN to C.  changed the descriptor 
parameters imagel and image2 to pointers.

XDFILI -  
changed the descriptor parameters image_name and 
file_ary to pointers.  Removed call to dmgetf - xdfili 
now accesses the table directly.  Changed so the files 
array returned contains unique filenames, and the 
nfiles parameter returns the count of unique filenames.

XDFILCKR -  
New routine.

XDFRTV -  
This routine has not been written for UNIX yet.

XDGERR -  
Converted from FORTRAN to C.

XDGETD -  
Replaced the union parameter with separate parameters.  
Changed FORTRAN string descriptors to pointers.  For 
the sake of simplicity, keyword strings are not used in 
parameter passing - all parameters must be supplied, 
and in the right order.

XDGETF -  
changed the descriptor parameter image_name to a 
pointer.  Also the keyword was changed from a descriptor 
to a define.  Support must be added to allow access to 
DDT and DMT fields.  This routine is no longer used.  
XD level routines that need to access table fields now 
do so directly.

XDGETU -  
not used.

XDHIST -  
This routine has not been written for UNIX yet.

Transportable 28 DMS
XDILUT -
Converted from FORTRAN to C. Added code to save the lut in the ICT entry for the currently displayed image.

XDIM2F -
Converted from FORTRAN to C.

XDIM2M -
Converted from FORTRAN to C. changed the descriptor parameter image to a pointer. Because of changes to the DMT and ICT, this routine no longer calls dmgetm and dtiunq to get an image's memids; instead, it calls dmgict to get the image's ict index.

XDIM2S -
Converted from FORTRAN to C. changed the descriptor parameter image to a pointer. Because of changes to the DMT and ICT, this routine no longer calls dmgetm and dtiunq to get an image's memids; instead, it calls dmgict to get the image's ict index.

XDIMNM -
Converted from FORTRAN to C. Changed to return image name and ict index of the currently viewed image.

XDIMRD -
Converted from FORTRAN to C. changed the descriptor parameter imgnam to a pointer. Also changed the descriptor luton to a define.

XDIMRT -
not used.

XDIMVR -
changed the descriptor parameter imgnam to a pointer.

XDIMWR -
Converted from FORTRAN to C. changed the descriptor parameter imgnam to a pointer. Also changed the descriptor luton to a define. Unused parameter inttype removed.

XDINI -
new routine - replaces iatinit - initializes a device's image and/or graphics memories.
XDINTI -  
Converted from FORTRAN to C. changed the descriptor parameter imgnam to a pointer.

XDLCSPR -  
This routine has not been written for UNIX yet.

XDLCSPW -  
This routine has not been written for UNIX yet.

XDLOOP -  
changed the descriptor array parameter images to an array of pointers. Also changed descriptor direction to an integer (it can passed a defined value).

XDLSSTI -  
Changed FORTRAN array descriptor parameter imgnams to an array of pointers.

XDLSUTI -  
Converted from FORTRAN to C. Changed the descriptor parameter imgnam to a pointer. Also Changed descriptor lutcod to an integer array indicating which luts (r,g,b) to initialize.

XDLSUTR -  
Converted from FORTRAN to C. Changed the descriptor parameter imgnam to a pointer. Parameters lutcod, Ishape, ldata, start, length were removed and replaced by 3 arrays containing the red, green, and blue lut values.

XDLSUTW -  
Converted from FORTRAN to C. Changed the descriptor parameter imgnam to a pointer. Parameters lutcod, Ishape, ldata, start, length were removed and replaced by 3 arrays containing the red, green, and blue values.

XDMEMW -  
Changed the descriptor parameter imgnam to a pointer. Because of changes to the DMT and ICT, this routine no longer calls dmgetm and dtiunq to get an image's mem- ids; instead, it calls dmsgict to get the image's ict index.

XDMNDFF -  
Converted from FORTRAN to C. Changed the descriptor

Transportable 30 DMS
array parameter defs to an array of pointers. Parameter fcodes removed. Rather than calling a dd level routine to print the menu, xdmndf now prints the menu directly.

XDNMEM -
Converted from FORTRAN to C. changed the descriptor parameters image and clean to pointers and descriptor array parameters rgb and qname to arrays of pointers.

XDPROT -
Converted from FORTRAN to C. Changed the descriptor parameter imgnam to a pointer. Also changed descriptor prot to a define. Changed to access the dmtable directly to set protection; no longer calls xdputf.

XDPUTF -
changed the descriptor parameter image_name to a pointer. Also the keyword was changed from a descriptor to a define. Support must be added to allow access to DDT and DMT fields. This routine is no longer used. XD level routines that need to access table fields now do so directly.

XDQKLK -
changed the descriptor parameters r_elements, g_elements and b_elements to pointers. Changed to pass the appropriate Image components to dmdefc indicating which component of the ict slots specified are to used for the red, green, and blue components of the new image.

XDRDZM -
Converted from FORTRAN to C. Changed the descriptor parameter image to a pointer. No longer calls a dd level routine to access the tables; it now accesses the tables directly.

XDRECD -
This routine has not been written for UNIX yet.

XDRINP -
Replaced call to dorinp (which returned screen-relative coordinates) by a call to ddrinp (new routine - returns memory-relative coordinates).
XDRSET - Converted from FORTRAN to C.

XDRSHF - Converted from FORTRAN to C.

XDS2IM - Converted from FORTRAN to C. Changed the descriptor parameter image to a pointer. Because of changes to the DMT and ICT, this routine no longer calls dmgetm and dtiunq to get an image's memids; instead, it calls dmgict to get the image's ict index.

XDS2M - Converted from FORTRAN to C. changed the descriptor parameter image to a pointer.

XDSAVE - new routine. Not written yet. GSFC has written a similar routine that may suffice.

XDSELD - new routine. Not written yet. It must save current local tables into global files, find the requested device and insure that it has been allocated to this user, then read in the context for that device.

XDSELI - Converted from FORTRAN to C. changed the descriptor array parameters prompt and selnam to arrays of pointers.

XDSERR - Converted from FORTRAN to C. changed the descriptor parameter mode to a pointer.

XDSHFT - Converted from FORTRAN to C.

XDSLIC - Converted from FORTRAN to C. changed the descriptor parameters imgnam and msg to pointers. Also changed descriptor parameters colcod and keep to defines.

XDUNZM - Converted from FORTRAN to C. Changed the descriptor parameter imgnam to a pointer. Accesses ictable

Transportable 32 DMS
directly instead of calling xdputf.

**XDVIEW**
- changed the descriptor parameter image name to a pointer. Removed coded to check for qualifiers and call xdqklk - this will be done by the utility before the call to xdview.

**XDWAIT**
- Eliminated VMS system service calls.

**XDWTFB**
- Converted from FORTRAN to C. Removed parameter image - uses currently displayed image.

**XDWTIR**
- Converted from FORTRAN to C. Removed parameter image - uses currently displayed image.

**XDWTTB**
- Converted from FORTRAN to C. Removed parameter image - uses currently displayed image. Added parameter fcode for the function key value.

**XDXCOL**
- Converted from FORTRAN to C. Changed descriptor color to a defined integer value. Changed red, green, and blue components to integer values instead of normalized real values. Color components are now in a local value array - it is not necessary to call a dd level routine to get color components.

**XDZMPN**
- changed the descriptor parameter image to a pointer.

**XDZMRN**
- Converted from FORTRAN to C.

**XDZOOM**
- Converted from FORTRAN to C. changed the descriptor parameter image to a pointer. Changed to reflect changes in the dmtable and ictable.
4.6. XL Functions

The XL functions were not implemented.

XLADD -
XLCLSR -
XLCOPY -
XLFIBI -
XLFTCH -
XLGADD -
XLGET -
XLIMKY -
XLIN -
XLINFO -
XLLUKY -
XLMEMU -
XLOPEN -
XLOUT -
XLPERR -
XLPUT -
XLREAD -
XLRLAB -
XLRMLB -
XLSET -
Transportable 35 DMS
4.7. **XO Functions**

**XOBLCH** - Not implemented.

**XOBOX** - Converted from FORTRAN to C. Modified to invoke DOGPDR four times instead of calling DOBOX.

**XOCINI** - Converted from FORTRAN to C.

**XOCCLR** - Modified to invoke XOZOOM to center and unzoom the graphics planes.

**XOCRRD** - Eliminated.

**XOCRWR** - Eliminated.

**XODRAW** - Eliminated (functionally redundant to XOGPDR and XOGPER).

**XOGPAN** - Converted from FORTRAN to C. Eliminated CENTR, LENGTH, and WARN parameters.

**XOGPCL** - Converted from FORTRAN to C.

**XOGPCO** - Converted from FORTRAN to C. Modified to pass integer values for red, green and blue to DOGPCO.

**XOGPDR** - Converted from FORTRAN to C.

**XOGPER** - Converted from FORTRAN to C.

**XOGPOF** - Converted from FORTRAN to C.

Transportable 36 DMS
EROS Data Center

May 30, 1985

XOGPON -
  Converted from FORTRAN to C.

XORDZM -
  Eliminated (information returned stored in DMS tables).

XORINP -
  Eliminated.

XORSET -
  Converted from FORTRAN to C.

XORSHM -
  Eliminated (information returned stored in DMS tables).

XOSHFT -
  Eliminated (functionally redundant to XOZOOM).

XOWTIR -
  Eliminated.

XOZOOM -
  Converted from FORTRAN to C.

Transportable 37 DMS
4.8. **XU Functions**

The XU functions were not implemented.

- XUCCLS
- XUCCRE
- XUCCTG
- XUCDEL
- XUCOPN
- XUCREN
- XUCREN
- XULDEL
- XULGET
- XULHDR
- XULLST
- XULPUT
- XUUTLS
5. **DMS Utilities**

5.1. **Arithmetic Operations**

(H) **ARITH-ADD** *
   Adds two images together.

(H) **ARITH-DIV** *
   Divides one image by another.

(H) **ARITH-MULT** *
   Multiplies two images together.

(H) **ARITH-SUB** *
   Subtracts one image from another.

(H) **CONVOL**
   Applies an NxN kernel convolution to an image.

(M) **FLIPD**
   Flips an image top to bottom.

(L) **HEIGHT**
   Calculates the height of an object by its shadow.

(H) **LOGIC-AND**
   Performs a logical AND of two images.

(L) **LOGIC-NOT**
   Performs a logical NOT of an image.

(H) **LOGIC-OR**
   Performs a logical OR of two images.

(L) **LOGIC-XOR**
   Performs a logical XOR of two images.

(M) **MIRROR**
   Creates the mirror image of an image.

(M) **ROTAT**
   Rotates an image by a specified amount.

5.2. **Color Display Control**

(H) **ALLOC** *
   Reserves/allocates a display for use during an

Transportable 39 DMS
interactive session.

(H) DALLOC *
Frees/deallocates all or any specified display device, that was previously allocated to the user, during an interactive session.

(H) DELIMG
Deletes entries from the display parameter table.

(H) DSTAT
Displays either summary or detailed information on a specified device or on all devices in the specified category.

(H) FRMDSP *
This utility copies single or multiple images from the display device refresh memory to a disk file or to a refresh memory.

(H) INIT *
Initialize all refresh memories (and hence LUT's) and/or graphics planes of the "active" allocated display.

(H) LSTIMG *
Lists information for all the images currently in the display parameter table.

(H) NETDSP
This utility copies single or multiple images or image windows from a local area network directly to the display device refresh memory.

(H) PROTEC
Changes the protection of the refresh memories used by an image.

(H) SAVIMG
Saves the displayed image in the display parameter table.

(L) SETDSP
Switches activity between multiple allocated devices.

(H) SHOIMG *
Displays an image using combinations of display Transportable 40 DMS
parameter table entries.

(M) SPLIT
Displays two or four images in split screen mode.

(H) TODSP *
This utility copies single or multiple images or image windows from disk to the display device refresh memory.

(H) ZOOPAN
Zooms and pans over an image using the function buttons to zoom and the trackball/ joystick to pan over the image.

5.3. Cursor Manipulation

(H) CURPOS *
Displays the cursor position and image intensity.

(H) CURSOR *
Turns the cursor on or off.

(M) DEFCUR
Defines a new cursor shape, size, and color.

5.4. Graphics Overlay

(H) HISTO-IMAGE *
Displays a histogram for one or all bands of the image currently being displayed.

(M) HISTO-LINE
Displays a histogram along a specified line of one or all bands of the image currently being displayed.

(L) TIC-OFF
Erase displayed tic marks.

(L) TIC-ON
Surrounds the image with reference tic marks.

5.5. Intensity Mapping

(H) ADJUST
Allows the user to interactively adjust the brightness (BIAS) and/or the contrast (GAIN) of a displayed image through movement of the trackball/ joystick.
(L) BITS
Temporarily modifies the mapping to simulate suppression of any of the eight bits of an image's pixels.

(M) DELDPT
Delete an entry from a source file's associated display parameter file.

(L) DSPCOR
Compensates for the exponential decay in intensity due to the cathode ray tube.

(H) INMAP *
Replaces a mapping of an image with a linear mapping.

(H) LODDPT *
Loads a display parameter table entry from an associated source file.

(M) LSTDPT
Lists all entries in a source file's associated display parameter file.

(L) MAPP-EXP
Applies an exponential mapping and scaling to an image.

(M) MAPP-HISTEQ
Apply a histogram equalization mapping to an image.

(M) MAPP-HISTO
Allows the user to define an arbitrary piecewise linear mapping using the cursor to select breakpoints while a histogram is displayed as a reference.

(L) MAPP-LOG
Applies a logarithmic mapping and scaling to an image.

(H) MAPP-MAN *
Allows the user to build a mapping with a piecewise linear fit of breakpoint pairs for a RGB or B/W image.

(H) SAVDPT *
Saves a display parameter table entry in a source file's associated display parameter file.

(M) SHODPT
Displays all information from an entry in a source
The file's associated display parameter file.

**H** SHOMAP
Displays all pairs of values for the current mapping and/or displays a graphics representation of the function mapping.

**L** ZONE-TB
Views a progression of brightness levels by use of the trackball.

**L** ZONE-TIME
Views a progression of brightness levels with the levels in each zone stretched from black to white.

### 5.6. Mensuration Capabilities

**M** CHGMEN
Allows the ability to change a record in the mensuration workfile.

**M** DELCUR
Delete annotation, point or line by cursor position.

**H** DELLAB *
Deletes a record from the mensuration work file by the label.

**M** DELREC
Deletes a record from the mensuration work file by the element number.

**M** EXTRAC
Searches a specified mensuration file for elements with specified label and/or attributes and adds them to the mensuration work file.

**L** FIT-CONIC
Fits a conic section to a number of specified points and saves it to the mensuration workfile with a label and attributes.

**H** FIT-LINE *
Fits a line to a set of points and saves it in the mensuration workfile along with a label and attributes.
(H) **INMEN** *
Clears the graphic bitplanes and/or initializes the mensuration work file.

(H) **LODMEN** *
Appends a specified file to the mensuration work file.

(H) **LSTMEN**
Displays and/or prints a mensuration file.

(L) **MEASUR-CONIC**
Finds the characteristics of a conic.

(M) **MEASUR-LINE**
Find the length of a line.

(M) **MEASUR-POINT**
Find the location of a point.

(M) **MEASUR-POLY**
Find the area of a polygon.

(H) **PUT-ANNOT** *
Places annotation at a given point.

(H) **PUT-POINT** *
Allows the user to pick a point in a specified bitplane and optionally associate with it a label and attributes.

(H) **PUT-POLY**
Allows the user to pick sets of end points that describes each side of a polygon and optionally associate with it a label and attributes.

(H) **SAVMEN** *
Saves the mensuration work file to a specified file.

(H) **SHOMEN**
Displays a bitplane from the work file in a specified color.

5.7. **Pseudo Color**

(H) **PSD-MAN** *
Manually assigns pseudo colors to grey level ranges.
(M) PSD-PALLAT
Allow the user to assign colors for grey level ranges from a pallatte using the cursor.

(M) PSD-PIECE
Defines a pseudo color mapping by selection of breakpoints.

(H) SLICE
Does grey level slicing of the specified image.

(L) ZONE-COLOR
Zone a color contour through an image.

5.8. Timing Functions

(M) FLICKR-DPT
Applies and displays two to ten mappings (derived from a source file's associated display parameter file) to the currently displayed image alternately in a timed sequence in interactive or independent (asynchronous) mode.

(H) FLICKR-IMAGE *
Displays two to ten images alternately in a timed sequence in interactive or independent (asynchronous) mode.

(H) FLICKR-STOP
Stops an independent (asynchronous) process initiated by FLICKR-IMAGE or FLICKR-DPT.
WORKSHOP SUMMARIES
The open discussion session on the Land Analysis System (LAS) focused on the three TAE/LAS related areas of Image I/O, Catalog Manager, and the Display Management Subsystem (DMS).

**Image I/O:**

A review of the existing implementations of image data structures and interfaces revealed that there are several different image I/O approaches currently used in the TAE/LAS environment. In addition to the original TAE image I/O structure and interface (commonly referred to as the Xi's), there is the LAS image I/O implemented under VAX/VMS and a new I/O being implemented under UNIX incorporating the basic data structure of TAE image I/O, the functionality of LAS I/O, and a simplified callable interface of open/close and read/write. The differences in these approaches resulted from development history, expanding functional requirements and concerns over processing efficiency. It was generally agreed that the development and wide endorsement of a common image I/O data structure and interface would be difficult to achieve due primarily to concerns for processing efficiency on the various TAE host computer systems. However, it was felt that the development of a common application callable interface for image I/O, modeled after the interface being developed for LAS image I/O under UNIX, could be achieved facilitating the exchange of image processing applications software among TAE users.
Catalog Manager:

Originally implemented to provide image file management support to users of the Landsat Assessment System (predecessor to the current Land Analysis System, LAS), the Catalog Manager has subsequently undergone significant enhancement in the areas of tape library support, associated file handling, performance improvements, and recovery support. This "new" Catalog Manager is currently available under the VAX/VMS operating system. Implementation of a full 'C' version of this Catalog Manager under the UNIX operating system is scheduled for completion in November, 1985. Several in the discussion group expressed concern over the design and implementation of the Catalog Manager within the TAE environment, and whether a redesign and implementation more closely integrated with TAE might be appropriate. It was noted that even though the primary functional requirements definition and funding for this Catalog Manager enhancement was provided by LAS, efforts have been made to maintain the Catalog Manager as generic as possible for use by applications other than LAS. It was also noted that more analysis is needed in the area of the Catalog Manager's functionality within a local-area network of TAE host systems especially as it relates to the TAE Remote Communications Job Manager (RCJM) implementation.
Display Management Subsystem (DMS):

A prototype VAX/VMS version of the TAE Display Management Subsystem (DMS), developed at NASA Goddard Space Flight Center, has been provided to several TAE Beta test sites. The EROS Data Center (EDC) is currently involved in implementing DMS under UNIX with some enhancements. NASA and EDC are cooperating in the development of a single VAX/VMS and UNIX version of DMS which will hopefully be available to TAE users in January. The EDC is also developing more than 50 LAS display application utilities interfaced to DMS. A question was raised as to the plans and provisions in DMS for supporting the "dumb" display with minimal hardware functionality. Although some software emulation of capabilities provided in hardware on other displays is taking place, there is no plan at this time to "fully" emulate in software the basic features which are "normally" furnished in display hardware. These software capabilities will more than likely evolve as new displays are interfaced to DMS. To date, IIS, DeAnza, and Raster Tech displays have been interfaced to DMS.
Window Management

- Century explained that there are about 25 lines (i.e., hooks) in the TAE TM which reference the window management subroutines. A site that wanted to install TAE with windows on a non-supported workstation would easily be able to plug the device's window manager interfaces at these hooks. Jim Cooper (University of Maryland) discussed having done this on his APOLLO workstation.

- Interest by several individuals was expressed for a SUN workstation implementation. GSFC will see about getting the prototype installed on the SUN.

- It was suggested that a means of soliciting ideas for graphic input requirements from the user community be developed.

- The question of whether windows make TAE run more efficiently or are only a 'toy' was discussed. Analysis of the use of the prototype should give some insight into this question.

- A means for switching between TAE window mode and 'plain old' TAE (i.e., global variable) was discussed. This will be implemented.

UNIX Implementation

- The question of whether TAE would run under ULTRIX was brought up. It is a 4.2 bsd version and, therefore, should port quite easily.

- An interest to port TAE onto an AT&T with System 5 (or System 3) UNIX was expressed. Probable targets are the IRIS and the MASSCOMP.

- The issue of whether TAE should be implemented with a slightly different 'flavor' - to take advantage of each computer's environment - or remain totally generic between machines was discussed.

The premise that users usually work within one workstation environment was disputed by some users who have a facility environment in which end-users swap between many different computers within a single day.
Discussions on "standards" versus "tailoring features were held. Typically, tailoring causes PDFs to not be portable, while without the ability to tailor, users complain—loudly.

RCJM/Networking

- The RCJM model, in which the user invoking jobs on other nodes must understand the context of that node, was discussed. The point was made that this does not represent the true "spirit" of a network, where a user should be sheltered from the networking procedures. However, to achieve a network environment of that type involves complete cooperation from all the nodes. The RCJM prototype was being developed in an environment where the nodes are managed independently and the approach taken was one which could be "implemented" within the required time line.

- A note was made that prototype RCJM only transfers text files.

- A note was made that prototype RCJM will be implemented under UNIX this summer.

A question of how different TAE subsystems/enhancements are going to "come together" and how distribution is going to occur was discussed. Suggestions were made and it was agreed that modularity was critical, so that an installation could pick and choose between optional subsystems and features based on his needs and system-configuration requirements.
ATTENDEE ADDRESS LISTS
<table>
<thead>
<tr>
<th>NAME</th>
<th>AGENCY</th>
<th>PHONE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>William Acevedo</td>
<td>USGS</td>
<td>(415) 694-6368</td>
</tr>
<tr>
<td>Lora Albanese</td>
<td>Century Computing</td>
<td>(301) 953-3330</td>
</tr>
<tr>
<td>Neil Allen</td>
<td>Colorado State University</td>
<td>(303) 491-8233</td>
</tr>
<tr>
<td>Troy Ames</td>
<td>GSFC</td>
<td>(301) 344-5673</td>
</tr>
<tr>
<td>Larry Babb</td>
<td>CSC</td>
<td>(415) 969-6626</td>
</tr>
<tr>
<td>Steve Borders</td>
<td>Global Imaging</td>
<td>(619) 481-5750</td>
</tr>
<tr>
<td>Kenneth Brunjes</td>
<td>DIA</td>
<td>(202) 373-3473</td>
</tr>
<tr>
<td>Grant Burton</td>
<td>Colorado State University</td>
<td>(303) 491-8340</td>
</tr>
<tr>
<td>Paul Butterfield</td>
<td>Century Computing, Inc.</td>
<td>(301) 953-3330</td>
</tr>
<tr>
<td>Tony Butzer</td>
<td>USGS</td>
<td>(605) 594-2271</td>
</tr>
<tr>
<td>Jody Caldwell</td>
<td>Sigma Data</td>
<td>(301) 344-6818</td>
</tr>
<tr>
<td>Laura H. Carey</td>
<td>NASA/GSFC</td>
<td>(301) 344-1166</td>
</tr>
<tr>
<td>Jim Chapman</td>
<td>GSFC</td>
<td>(301) 344-5715</td>
</tr>
<tr>
<td>S. Charalambides</td>
<td>Imperial College, London</td>
<td>( ) 807-8357</td>
</tr>
<tr>
<td>Heather M. Cheshire</td>
<td>NCSU</td>
<td>(919) 737-3430</td>
</tr>
<tr>
<td>Sher Contractor</td>
<td>NASA/GSFC</td>
<td>(301) 344-9417</td>
</tr>
<tr>
<td>James Cooper</td>
<td>University of Maryland</td>
<td>(301) 454-5186</td>
</tr>
<tr>
<td>Jay Costenbader</td>
<td>NASA/GSFC</td>
<td>(301) 344-1177</td>
</tr>
<tr>
<td>Brian Dealy</td>
<td>SAR</td>
<td>(301) 344-9531</td>
</tr>
<tr>
<td>Kay Edwards</td>
<td>USGS</td>
<td>( ) 265-7118</td>
</tr>
<tr>
<td>David R. Eike</td>
<td>Carlow Associates</td>
<td>(703) 698-6225</td>
</tr>
<tr>
<td>Curtis Emerson</td>
<td>GSFC/522.2</td>
<td>(301) 344-5149</td>
</tr>
<tr>
<td>Carmen Ana Emmanuelli</td>
<td>GSFC</td>
<td>(301) 344-9425</td>
</tr>
<tr>
<td>Joseph Erkes</td>
<td>GE/CRD</td>
<td>(518) 462-3204</td>
</tr>
<tr>
<td>Ai C. Fang</td>
<td>NASA/HQ</td>
<td>(202) 453-1502</td>
</tr>
<tr>
<td>Nancy Fitzgerald</td>
<td>Century Computing, Inc.</td>
<td>(301) 953-3330</td>
</tr>
<tr>
<td>A. J. Fleig</td>
<td>GSFC</td>
<td>(301) 344-9041</td>
</tr>
<tr>
<td>Ken Gacke</td>
<td>USGS</td>
<td>(605) 594-6581</td>
</tr>
<tr>
<td>Pat Gary</td>
<td>GSFC</td>
<td>(301) 344-8935</td>
</tr>
<tr>
<td>Raul Garza-Robles</td>
<td>GSFC</td>
<td>(301) 344-9513</td>
</tr>
</tbody>
</table>

-1-
<table>
<thead>
<tr>
<th>NAME</th>
<th>AGENCY</th>
<th>PHONE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dick Gemoets</td>
<td>CSC</td>
<td>(415) 969-6626</td>
</tr>
<tr>
<td>John Gibson</td>
<td>University of Georgia</td>
<td>(404) 542-3265</td>
</tr>
<tr>
<td>Michael Gough</td>
<td>SAR</td>
<td>(301) 344-9515</td>
</tr>
<tr>
<td>Lesley Grove</td>
<td>Imperial College</td>
<td>(01) 589-5111</td>
</tr>
<tr>
<td>M. Guberek</td>
<td>Global Imaging</td>
<td>(619) 481-5750</td>
</tr>
<tr>
<td>Vince Guidice</td>
<td>NASA/GSFC</td>
<td>(301) 344-1176</td>
</tr>
<tr>
<td>Ed Guiness</td>
<td>Washington University</td>
<td>(314) 889-5493</td>
</tr>
<tr>
<td>John Gundy</td>
<td>DMAHIC</td>
<td>( ) 227-2295</td>
</tr>
<tr>
<td>Michael Gunning</td>
<td>NPGS</td>
<td>(408) 646-3061</td>
</tr>
<tr>
<td>Milt Halem</td>
<td>GSFC</td>
<td>(301) 344-8834</td>
</tr>
<tr>
<td>Jeff Hall</td>
<td>JPL</td>
<td>(818) 354-4316</td>
</tr>
<tr>
<td>Lisa Hamet</td>
<td>NASA/GSFC</td>
<td>(301) 344-7877</td>
</tr>
<tr>
<td>James Hammad</td>
<td>DMAHTC</td>
<td>( ) 227-3320</td>
</tr>
<tr>
<td>Elfrieda Harris</td>
<td>SAR</td>
<td>(301) 344-6034</td>
</tr>
<tr>
<td>Lee Helmle</td>
<td>Informatics/NASA Ames</td>
<td>(415) 694-6235</td>
</tr>
<tr>
<td>George Huffman</td>
<td>University of Maryland</td>
<td>(301) 454-2467</td>
</tr>
<tr>
<td>Kevin J. Hussey</td>
<td>JPL</td>
<td>(818) 792-4016</td>
</tr>
<tr>
<td>Nick Ide</td>
<td>Century Computing, Inc.</td>
<td>(301) 953-3330</td>
</tr>
<tr>
<td>Fred Irani</td>
<td>SAR</td>
<td>(301) 344-9520</td>
</tr>
<tr>
<td>Chris Jay</td>
<td>E-Systems/Melpar</td>
<td>(703) 560-5000</td>
</tr>
<tr>
<td>Ken Johnson</td>
<td>EROS</td>
<td>(605) 694-2271</td>
</tr>
<tr>
<td>Ted Johnson</td>
<td>GSFC</td>
<td>(301) 344-6818</td>
</tr>
<tr>
<td>Harry Jones</td>
<td>NASA/Ames</td>
<td>(8) 464-6616</td>
</tr>
<tr>
<td>Steve Kempler</td>
<td>NASA/GSFC</td>
<td>(301) 344-5005</td>
</tr>
<tr>
<td>John Kogut</td>
<td>RDS</td>
<td>( ) 390-6100</td>
</tr>
<tr>
<td>David Kramer</td>
<td>DMAHTC</td>
<td>( ) 287-2018</td>
</tr>
<tr>
<td>Nand Lal</td>
<td>NASA/GSFC</td>
<td>(301) 344-5668</td>
</tr>
<tr>
<td>Craig Leff</td>
<td>National Air and Space Museum</td>
<td>( ) 357-1424</td>
</tr>
<tr>
<td>Patricia K. Liggett</td>
<td>JPL</td>
<td>(818) 354-8486</td>
</tr>
<tr>
<td>Bill Likens</td>
<td>NASA Ames</td>
<td>(FTS) 464-5596</td>
</tr>
<tr>
<td>Barbara E. Lowrey</td>
<td>NASA/GSFC</td>
<td>(301) 344-9513</td>
</tr>
<tr>
<td>Yun-Chi Lu</td>
<td>NASA/GSFC</td>
<td>(301) 344-9526</td>
</tr>
<tr>
<td>Marilyn Mack</td>
<td>NASA/GSFC</td>
<td>(301) 344-7980</td>
</tr>
<tr>
<td>NAME</td>
<td>AGENCY</td>
<td>PHONE NUMBER</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>John T. McBeth</td>
<td>Century Computing, Inc.</td>
<td>(301) 953-3330</td>
</tr>
<tr>
<td>Caldwell McCoy</td>
<td>NASA/HQ</td>
<td>(202) 453-1495</td>
</tr>
<tr>
<td>Phil Miller</td>
<td>Century Computing, Inc.</td>
<td>(301) 953-3330</td>
</tr>
<tr>
<td>Dharitri Misra</td>
<td>Century Computing, Inc.</td>
<td>(301) 953-3330</td>
</tr>
<tr>
<td>Karen Moe</td>
<td>NASA/GSFC</td>
<td>(301) 344-5292</td>
</tr>
<tr>
<td>Walt Moleski</td>
<td>NASA/GSFC</td>
<td>(301) 344-5673</td>
</tr>
<tr>
<td>K. Narayanan</td>
<td>SAR</td>
<td>(301) 344-7372</td>
</tr>
<tr>
<td>David Nichols</td>
<td>JPL</td>
<td>(818) 354-4994</td>
</tr>
<tr>
<td>Larry Novak</td>
<td>NASA/GSFC</td>
<td>(301) 344-1464</td>
</tr>
<tr>
<td>Lyn Oleson</td>
<td>USGS/EROS Data Center</td>
<td>(605) 594-6555</td>
</tr>
<tr>
<td>Jan Owings</td>
<td>NASA/GSFC</td>
<td>(301) 344-6450</td>
</tr>
<tr>
<td>Tom Parris</td>
<td>ERIM</td>
<td>(313) 994-1200</td>
</tr>
<tr>
<td>Philip B. Pease</td>
<td>NASA/GSFC</td>
<td>(301) 344-6693</td>
</tr>
<tr>
<td>Dolly Perkins</td>
<td>NASA/GSFC</td>
<td>(301) 344-5069</td>
</tr>
<tr>
<td>Bruce Quirk</td>
<td>USGS</td>
<td>(605) 594-6114</td>
</tr>
<tr>
<td>David Roberts</td>
<td>UCAR</td>
<td>(303) 443-5349</td>
</tr>
<tr>
<td>Jon W. Robinson</td>
<td>SAR/GSFC</td>
<td>(301) 344-9846</td>
</tr>
<tr>
<td>Marshall Ross</td>
<td>SAR</td>
<td>(301) 344-7372</td>
</tr>
<tr>
<td>Scott Ross</td>
<td>SAR</td>
<td>(301) 344-7372</td>
</tr>
<tr>
<td>Scott Sandell</td>
<td>DIPIX</td>
<td>(301) 596-0505</td>
</tr>
<tr>
<td>Donald Sawyer</td>
<td>M/A-COM-NSSDC</td>
<td>( ) -8148</td>
</tr>
<tr>
<td>Joe Seamone</td>
<td>SAR</td>
<td>(301) 344-7372</td>
</tr>
<tr>
<td>Joseph M. Seay</td>
<td>DMAHTC</td>
<td>( ) 227-2294</td>
</tr>
<tr>
<td>David L. Sherwood</td>
<td>DIA</td>
<td>( ) 373-2420</td>
</tr>
<tr>
<td>Allen Silver</td>
<td>NASA/GSFC</td>
<td>(301) 344-8531</td>
</tr>
<tr>
<td>Dennis Small</td>
<td>NASA/GSFC</td>
<td>(301) 462-6488</td>
</tr>
<tr>
<td>William Smythe</td>
<td>JPL</td>
<td>(818) 792-2636</td>
</tr>
<tr>
<td>Maria So</td>
<td>NASA/GSFC</td>
<td>(301) 344-6818</td>
</tr>
<tr>
<td>Mark Stevens</td>
<td>NASA/GSFC</td>
<td>(301) 344-7877</td>
</tr>
<tr>
<td>Peter Stoll</td>
<td>DIA</td>
<td>(202) 373-3481</td>
</tr>
<tr>
<td>Brian Swafford</td>
<td>M/A-COM-NSSDC</td>
<td>( ) -6818</td>
</tr>
<tr>
<td>Marti Szczur</td>
<td>NASA/GSFC</td>
<td>(301) 344-9425</td>
</tr>
<tr>
<td>Valerie Thomas</td>
<td>NASA/GSFC</td>
<td>(301) 344-9489</td>
</tr>
<tr>
<td>NAME</td>
<td>AGENCY</td>
<td>PHONE NUMBER</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>R. J. Thompson</td>
<td>EROS</td>
<td>(FTS) 784-7555</td>
</tr>
<tr>
<td>Donald M. Thompson</td>
<td>DIA</td>
<td>(202)</td>
</tr>
<tr>
<td>Lloyd Treinish</td>
<td>NASA/GSFC</td>
<td>(301) 344-9489</td>
</tr>
<tr>
<td>Arnie Voketaitis</td>
<td>NASA/GSFC</td>
<td>(301) 344-9417</td>
</tr>
<tr>
<td>Tami Weaver</td>
<td>SAR</td>
<td>(301) 344-9291</td>
</tr>
<tr>
<td>Philip Wheeler</td>
<td>Century Computing, Inc.</td>
<td>(301) 953-3330</td>
</tr>
<tr>
<td>Ed Wilson</td>
<td>SAR</td>
<td>(301) 344-9536</td>
</tr>
<tr>
<td>Karl Wolf</td>
<td>Century Computing, Inc.</td>
<td>(301) 953-3330</td>
</tr>
<tr>
<td>Bob Woodward</td>
<td>GSFC</td>
<td>(301) 344-9176</td>
</tr>
</tbody>
</table>
Attendees to the Fifth TAE Users' Conference
June 4, 5, and 6, 1985

Goddard -

Troy Ames
Edward Burton
Laura Carey
Jim Chapman
Sher Contractor
Jay, Costenbader
Curtis Emerson
Carmen Ana Emmanuelli
Al Fleig
Pat Gary
Raul Garza-Robles
Vince Guidice
Milt Halem
Lisa Hamet
Ted Johnson
Steve Kempler
Nand Lal
Barbara Lowrey
Yun-chi Lu
Marilyn Mack
Karen Moe
Walt Moleski
Larry Novak
Jan Owings
Philip Pease
Dolly Perkins
Allan Silver
Dennis Small
Maria So
Mark Stephens
Marti Szczur
Valerie Thomas
Lloyd Treinish
Arnie Voketaitis

Headquarters -

Ai Fang
Caldwell McCoy
<table>
<thead>
<tr>
<th>Organization</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defense Intelligence Agency</td>
<td>Kenneth Brunjes</td>
</tr>
<tr>
<td>DB-4D3</td>
<td>David Sherwood</td>
</tr>
<tr>
<td>Washington, D.C. 20301-6111</td>
<td>Peter Stoll</td>
</tr>
<tr>
<td></td>
<td>Donald Thompson</td>
</tr>
<tr>
<td>School of Forestry</td>
<td>Heather Cheshire</td>
</tr>
<tr>
<td>Biltmore Hall</td>
<td></td>
</tr>
<tr>
<td>Box 8002</td>
<td></td>
</tr>
<tr>
<td>North Carolina State University</td>
<td></td>
</tr>
<tr>
<td>Raleigh, NC 27650</td>
<td></td>
</tr>
<tr>
<td>Department of Meteorology</td>
<td>James Cooper</td>
</tr>
<tr>
<td>2207 Space Sciences Building</td>
<td>George Huffman</td>
</tr>
<tr>
<td>University of Maryland</td>
<td></td>
</tr>
<tr>
<td>College Park, MD 20742</td>
<td></td>
</tr>
<tr>
<td>U. S. Geological Survey</td>
<td>Kay Edwards</td>
</tr>
<tr>
<td>2255 North Gemini</td>
<td></td>
</tr>
<tr>
<td>Flagstaff, AZ 86001</td>
<td></td>
</tr>
<tr>
<td>General Electric</td>
<td>Joseph Erkes</td>
</tr>
<tr>
<td>37/261</td>
<td></td>
</tr>
<tr>
<td>1 River Road</td>
<td></td>
</tr>
<tr>
<td>Schenectady, N.Y. 12345</td>
<td></td>
</tr>
<tr>
<td>COSMIC</td>
<td>John Gibson</td>
</tr>
<tr>
<td>Suite 112, Barrow Hall</td>
<td></td>
</tr>
<tr>
<td>University of Georgia</td>
<td></td>
</tr>
<tr>
<td>Athens, GA 30602</td>
<td></td>
</tr>
<tr>
<td>Imperial College of London</td>
<td>S. Charalambides</td>
</tr>
<tr>
<td>Atmospheric Physics Group</td>
<td>Lesley Grove</td>
</tr>
<tr>
<td>Blackett Laborator</td>
<td></td>
</tr>
<tr>
<td>South Kensington</td>
<td></td>
</tr>
<tr>
<td>London SW7 2BZ</td>
<td></td>
</tr>
<tr>
<td>United Kingdom</td>
<td></td>
</tr>
</tbody>
</table>
Research & Data Systems
10300 Greenbelt Road
Lanham, MD 20706

ERIM
Box 8618
Ann Arbor, Michigan 48107

UCAR
1001 Pine Street
Boulder, CO 80302

Dipex, Inc.
10220 Old Columbia Rd.
Columbia, MD 21046

E-Systems/Melpar Division
7700 Arlington Blvd.
Falls Church, VA 22046

National Air and Space Museum
Smithsonian Institution
Washington, D.C. 20560

Sigma Data Computing Corp.
5515 Security Lane
Rockville, MD 20852

Science Applications Research
4400 Forbes Blvd.
Lanham, MD 20706

Carlow Associates

GSC
**Abstract**

The Transportable Applications Executive (TAE) was conceived in 1979. It was proposed to be a general purpose software executive that could be applied in various systems. The success of this concept and of TAE was demonstrated at the Fifth Annual TAE Users' Conference. More than 120 people attended, representing six universities, five government agencies, 10 private industries, Goddard Space Flight Center, and NASA Headquarters. The attendees included both TAE users and potential users. Presentation topics ranged from "TAE and the Space Station," "Unidata -- The Next Generation Meteorological Data System in Universities," and "System 9000: A TAE-based Interactive Digital Image Processing Workstation," to "Porting TAE to the Apollo Micro Computer".

**Key Words (Selected by Author(s))**
- portable software
- executive
- TAE applications

**Distribution Statement**
- Unclassified
- Unlimited

**Security Classif. (of this report)**
- Unclassified

**Security Classif. (of this page)**
- Unclassified

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