Version 3

Elizabeth Duxbury
Danika Jensen

Oct. 14, 1994

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

D-4186 Rev B

PREFACE TO VERSION THREE

VICAR is an evolving software system constantly being updated with contributions from general users and flight projects. This makes it difficult to keep a document such as this VICAR User's Guide up-to-date. Part of the purpose in putting this latest version online is to enable revisions in VICAR to be more quickly reflected in this Guide.

Several significant changes have been made in this version of the Guide. An attempt has been made to keep the current format as similar as possible to past versions of the Guide, but for the online version, some changes have been made to take advantage of the cross-referencing capabilities of the NCSA Mosaic format. It is hoped that most of these changes will be transparent to anyone reading a hardcopy version. An attempt has been made once again to make the Index more useful in quickly finding critical information.

There have also been significant changes to VICAR since Version 3 was published. Perhaps most important are those changes due to the conversion of VICAR to enable it to run on UNIX systems. (These changes are detailed in Section 3.1.1 for the benefit of the experienced VMS-VICAR user.) The content of the VICAR applications libraries has changed with the addition of many new procs and the deletion of several obsolete or redundant ones. Finally, the Menu mode and the New User's Tutorial are undergoing significant changes, and have not yet been implemented for UNIX VICAR.

This version of the VICAR User's Guide applies to:

```
VICAR Version 12.1
     using TAE Version 5B/16B
     under UNIX Version SunOS 4.1.3 or Solaris 2.3
or   under VMS Version 5.4-2
```

ACKNOWLEDGEMENTS

This Guide would not have been completed if not for the assistance of many members of the Image Processing Laboratory. The authors extend special thanks to Ray Bambery, Nevin Bryant, Bob Deen, Sandy Dumarr, Greg Earle, Tom Greer, Dave Hodges, Steve Hwan, Sue LaVoie, Alan Mazer, Helen Mortensen, Steve Pohorsky, Niles Ritter, Allan Runkle, Cesar Vasquez, Lisa Wainio, Mitch White, and Gary Yagi for their assistance.

CONTENTS

Version 3 of the VICAR User's Guide is an update of previous versions, and the authors relied heavily on those earlier versions for its content. The authors of Version 2 of the Guide are Susan LaVoie, Doug Alexander, Charles Avis, Helen Mortensen, Carol Stanley, and Lisa Wainio. The writing and HTML formatting of Version 3 were done by Elizabeth.D.Duxbury@jpl.nasa.gov and Danika.Jensen@jpl.nasa.gov.Last modified May 21, 1996.

1 Introduction

Contents

1.1 Document Purpose

The VICAR User's Guide is designed to instruct the new user in the use of the VICAR (Video Image Communication and Retrieval) image processing system and to serve as a reference manual for the experienced user.

This document is a facility-independent guide to both the UNIX and VAX/VMS implementations of the VICAR system. Basic knowledge of UNIX and VAX/VMS is assumed.

The most important and useful user information has been gleaned from several sources, enhanced and incorporated into this

document. The VICAR User's Guide does not replace these sources, but provides the central focus of user documentation.

1.2 Document Organization

This document has been divided into eleven sections which introduce the VICAR system and describe its use.

1. Introduction, purpose and organization of the document; document conventions
2. User documentation and references
3. VICAR history, applications and acquisition
4. Basic VICAR concepts: entering and exiting VICAR; datasets; labels; procs
5. The VICAR environment: libraries; processing modes; subprocesses; aborting commands
6. Methods for obtaining Help; Tutor and Menu modes; explanation of how to interpret messages
7. Use of the VICAR executive: TAE Command Language; VICAR command structure; Proc Definition Files; media handling; session logging
8. Non-standard and non-VICAR topics: hardware and software assumptions that apply to a subset of the VICAR applications software; VICAR I/O formats
9. Advanced VICAR concepts: dataset and label structure
10. Appendices: VICAR application proc functional descriptions and classification by function; Intrinsic commands; command qualifiers; intrinsic global variables; Menu and Tutor mode user operations; special terminal keys; Proc Definition File examples; VICAR label examples; VICAR message interpretation; New User's Tutorial, VICAR Sample Session
11. Index

1.3 Conventions, Acronyms and Terms

1.3.1 Conventions

Several conventions are utilized throughout this document to ensure ease and consistency of interpretation.

1. **VICAR>** is the VICAR prompt.

2. Double quotes, " ", are used around special characters or technical terms to avoid confusion with everyday meanings (i.e., "*" and "help").
3. Vertical dots indicate that not all of the required data are shown.

Example: Use of vertical dots.

```
Process HELP=help-location
Parm  name1  Description1
Parm  name2  Description2
.
.
.
End-proc
```

4. Since VICAR makes no distinction between upper and lower case for user input, both are used in examples. (Note: On UNIX systems, filenames are still case sensitive, so "`out.img`" is a different file from "`OUT.IMG`".)

5. In all examples of user input in this document, VICAR responses are in bold-faced type and user-typed text is not.
6. Brackets, [ ], are used to represent optional entries.

Example:

```
VICAR> program [parameters]
```

7. Commands that can be abbreviated are shown with sqare brackets. The brackets indicate which letters do not need to be included as part of the command (i.e., `ALLO[C]` can be `ALLOC` or `ALLO`).

8. In order to make it possible for both new and experienced users to benefit from this document, the following words in boldfaced type are used to designate material that is for more advanced users.

When **EXPERT** appears at the beginning of a paragraph it warns the new user that this is difficult material and should be skipped.

Material "for wizards only" is flagged by **WIZARD** and should be avoided by all but the very brave.

9. Warnings to users regarding hazardous use of commands or potentially dangerous situations are *italicized* and are flagged by **BEWARE**.

10. USERID is used to represent the user's login id.

1.3.2 Acronyms and Terms

Acronyms and terms which are utilized in this document are listed below:

```
ASCII    American Standard Code for Information
Interchange
```

ANSI American National Standards Institute
BARC Block Adaptive Rate Compressor
BDV Bad Data Value
BTC Block Truncation Coding
CCSI Cerritos Computer Systems Incorporated
Printronix plotting routines
COSMIC COmputer Software Management and Information Center
DCL Digital Command Language
DDO Data Driven Object
DN Data Number
EBCDIC Extended Binary Coded Decimal Interchange Code
EDR Experimental Data Record
GUI Graphical User Interface
HIIPS Home Institution Image Processing Subsystem
IBIS Image Based Information System (a subset of VICAR used as a geographic information system)
IPL Image Processing Laboratory
LFW Low-Full-Pixels
MDF Menu Definition File
MIPS Multimission Image Processing Subsystem
NAG Numerical Algorithms Group
Mathematical subroutine package
NIMS Near Infrared Mapping Spectrometer
PDF Proc Definition File
PDS Planetary Data System
PIXEL Picture Element
PROC Process or procedure
RTS Real Time System
SCLK Spacecraft CLocK
SIS Software Interface Specification
TAE Transportable Applications Executive
TCL TAE Command Language
UDR Unprocessed Data Record
UNIX (originally UNICS) Uniplex Information and Computer
Systems
USH User SHell
VAX Virtual Address Extension
VICAR Video Image Communication and Retrieval
VMS Virtual Memory System
WPT Window Programming Tools

2 VICAR Documentation and References

The following list contains various documents which comprise a description of the VICAR system. This includes material describing VICAR from several aspects. Novice and experienced users, as well as programmers, will find their perspectives represented. In addition, information concerning the MIPS facility is provided. In-depth help is available for certain subsets of VICAR (AMOS, BROWSE, etc.) and programs used in conjection with it (dbView, SPICE, etc.).

Key resources for further information are listed below. This document and most of these items are available from:

IPL Documentarian
Image Processing Software Development Group
Jet Propulsion Laboratory
Mail Stop 168-414
4800 Oak Grove Drive
Pasadena, CA 91109

818-354-0506

MIPS

* *The MIPL Cookbook,* R. Bambery, 7/18/86, JPL Document D-4312 (This version is obsolete; its function will hopefully be replaced in the near future.)
* *Tutorial on the Use of the New MIPS System*, (includes a section on "Using VICAR under UNIX")

TAE

* *TAE Command Language (TCL) Programmer's Manual, Version 5.2*, Century Computing, 12/92
* *TAE Command Language (TCL) User's Manual, Version 5.2*, Century Computing, 12/92

VICAR

* *An Introduction to the VICAR Image Processing Executive,* D. Stanfill, M. Girard, 10/8/86, JPL Document D-4309
* *An Introduction to VICAR,* G. Bothwell, 5/89, JPL Document D-4379
* *MIPL Mathematics Libraries Used In The VICAR System,* S. Pohorsky, 4/87, JPL Document D-4377
* *MIPL VICAR Common Plotting System - Software User Guide,* L. Bynum, 2/10/87, JPL Document D-4375 (supported for VMS VICAR only)
* *MIPL VICAR Installation Guide,* R. Deen, 6/4/93, JPL Document D-4310, Version 4D/15F, (Part 1 - VMS, Part 2 - UNIX)
* *MIPS Delivery 12.0 System Release Description Document*, Mitchell White, 1/5/94, JPL Document D-11880
* *The VICAR File Format,* R. Deen, 9/25/92, JPL IOM IPSD:384-92-196
* *The VICAR Image Processing System* - general online information about VICAR
* *VICAR PDF Files With HTML Help* (Online help for VICAR PDFs.)
* *The VICAR Porting Guide*, R. Deen, 8/15/94, JPL Document D-9395
* *VICAR Run-time Library Reference Manual, Version 13J*, D. Stanfill, 3/3/89, JPL Document D-4311

AMOS

* *AMOS User's Guide,* G. Yagi, 4/17/89 (HTML formatting incomplete)

BROWSE

* *BROWSE Reference Card,* JPL Document D-6235
* *User Guide for Program Browse,* R. Stagner, 1/1/86, JPL Document D-2766

CATALOG

* *Accessing Information From the Voyager and Galileo Catalogs,* (Not Available Yet)

IBIS

* *Automated Mosaicking of Planetary Imagery: The VICAR/IBIS Approach,* E. Barragy and F. Evans, 6/15/88, JPL Document D-5529
* *IBIS Class Library - Guidelines for Creating IBIS-based file formats*, Niles D. Ritter, 6/2/94
* *IBIS Subroutine Library Programmers' Guide* , Niles D. Ritter, 6/2/94, JPL Document D-11944
* *Image Based Information System White Paper*, N. Bryant, et al, 1/9/85

MRPS

* *Multiple Recorder Production System User Guide Version 3.2,* Bambery, Capraro, Klein, Runkle, & Wainio, 4/23/93, JPL Document D-4383

SPAM

* *Spectral Analysis Manager Version 4, SPAM,* A. Mazer, et. al., 10/1/88

SYBASE

* *dbq User's Guide, Version 1.2,* , Rector &Jacobson, 6/10/95 (PostScript format)
* *dbView - A Database Access Tool, Version 1.5*, Rector, Jacobson &Young, 5/10/95 (PostScript format)

VIDS

* *VIDS Quick Reference Card,* VIDS Version 1B, JPL Document D-5970
* *VIDS User's Guide,* VIDS Version 3D, R. Deen, 9/1/89, JPL Document D-5970

VRDI

* *VRDI User's Reference Guide, Version 3*, Paul Bartholomew, 9/1/89, JPL Document D-5100 (.dvi format)

Other

* *Math 77 - A Library of Mathematical Subprograms for Fortran 77, Release 4.0*, JPL Supercomputing and Computer Mathematics Support Group, 5/92, JPL Document D-1341
* *MIPS Map Projection Software Users' Guide, Version 1.1*, Justin McNeill, Jr., 7/94, JPL Document D-11810

3 Overview of VICAR

Contents

3.1 Background

VICAR is a set of computer programs and procedures designed to facilitate the acquisition, processing and handling of digital image data. The VICAR image processing language was defined by JPL employees Stan Bressler, Howard Frieden and Fred Billingsley, and implemented in 1966 at the Jet Propulsion Laboratory to process image data produced by the planetary exploration program. The software package was originally designed for operation with the IBM 360/44 Programming System (44PS) and was later modified to run with the IBM OS/360 operating system. In the early 1980's VICAR underwent a major revision to run under the VMS operating system on the VAX series of Digital Equipment Corporation computers. Now, in the early 1990's, VICAR is once again undergoing a major revision and will soon be fully operational under both the SunOS and Solaris UNIX operating systems.

For its command-line interface, VICAR makes use of the Transportable Applications Executive (TAE) that was developed by the NASA Goddard Space Flight Center. This version of VICAR makes use of several enhancements to TAE, which should be included with your version of VICAR. Several VICAR programs may not run without these enhancements.

The VICAR executive is a body of software that forms the interface between the user, the library of application programs and procedures, and the computer's operating system. The VICAR executive does not replace the host computer's operating system; instead, it overlays the host resources. The objectives of the VICAR executive are to:

* standardize the user interface to application programs and procedures
* provide an interface that can be easily understood and used by experts and non-experts
* shield the user from the host operating system by making device-specific details transparent to the user (e.g., display devices)
* simplify programming for future expansions

VICAR's application library contains an extensive assortment of programs and procedures to perform a wide variety of functions (Appendices 10.1, 10.2 and 10.3). These programs are written in standard programming languages, FORTRAN, C and C++, with small segments in VAX assembler and array processor code (not portable code).

The user interfaces with the VICAR executive via a series of TAE Command Language (TCL) statements (Section 7.1). Through TCL, the user directs VICAR to invoke the programs, procedures and commands to perform the analytical and housekeeping activities necessary to complete a required task.

Programs interface with the VICAR executive via a standard set of subroutines that perform image, label and parameter processing plus control several external devices such as display devices and printers (Figure 3.1). The interface is known as the VICAR Run-time Library (see the *VICAR Run-time Library Reference Manual*).

Other libraries exist that provide access to databases, external devices such as display devices and printers, etc. The subroutines are designed to reduce some of the routine labor involved in writing an application program.

Because of its modular architecture and portability, the VICAR system will continue to grow. New capabilities are being added to take advantage of new technology, to meet the demands of the user, and to provide support for new application areas.

Figure 3.1 VICAR Program Interface

3.1.1 VICAR Changes Under UNIX

For the most part, the changes to VICAR that enable it to run on the UNIX operating system should be transparent to the user. However, there are a few things that have changed that the user should be aware of. These differences between the VAX/VMS and UNIX VICAR systems are covered in the following subsections:

* asynchronous processing - 5.2.2
* batch processing - 5.2.3
* case sensitive dataset names - 4.3.1
* case sensitive proc names - 7.3
* DCL / USH mode - 5.2.1
* definition of `ulogon` and `ulogoff` files - 5.4.1 and 5.4.2
* processes and subprocesses - 7.2.1.1
* session logging problems - 7.6
* tape handling - 7.5
* temporary files - 4.3.1.1

There is also an NCSA Mosaic page, *Using VICAR under UNIX*, detailing the differences between VAX/VMS and UNIX VICAR.

3.2 Applications of VICAR

VICAR's application has expanded over time and now supports image processing for many disciplines: planetary imaging, astronomy, earth resources, land use, biomedicine and forensics. Facilities using VICAR are world-wide and include: universities, the military, research institutions, aerospace corporations, commercial/industrial companies and the Galileo HIIPS (Home Institution Image Processing Subsystem) sites.

3.3 System Acquisition

VICAR is available to organizations through the Computer Software Management and Information Center (COSMIC):

```
COSMIC
Software Information Services
Computer Services Annex
The University of Georgia
Athens, Georgia 30602

706-542-3265
```

4 Basic VICAR Concepts

Contents

In this section, the user is introduced to some of the fundamental concepts of the VICAR image processing system. The new user will learn how to enter and exit VICAR and be introduced to VICAR datasets, labels and "procs". An understanding of these concepts is essential to effectively use VICAR and is assumed in subsequent sections.

4.1 Entering/Exiting VICAR

A VICAR session starts when a user invokes the VICAR executive by typing the command `vicar` at the prompt (% or $).

Syntax:

```
    % vicar    (UNIX)
or
    $ VICAR    (VMS)
```

The user is then prompted for a VICAR command (Section 7.1) with a VICAR prompt, **VICAR>**.

The session ends when a user exits VICAR by typing the command `exit` at the VICAR prompt.

Syntax:

```
    VICAR>exit
```

The user is then returned to the UNIX shell or DCL.

4.2 Getting Started

Once in VICAR, a user is able to execute VICAR commands (Section 7.1) using the following command line syntax. Each required and optional term is explained in Section 7.2.

Syntax:

```
    VICAR>name[-subcommand] [|qualifiers|]
[parameter_list] +
    VICAR>+[comments]
```

The most common and simplest form of a command line consists only of the proc or command name followed by a list of parameters.

Syntax:

```
VICAR>name[-subcommand] [parameter_list]
```

The following are examples to show how most procs and commands are invoked.

Example: Try these and see what happens.

```
VICAR>gen a 10 10
VICAR>list a
VICAR>usage    (should be available on UNIX with
V12.2)
VICAR>show
```

For those anxious users who would like to jump right in, we suggest invoking the Menu system (Section 6.3) or going through the New User Tutorial (NUT - Appendix 10.14).

Syntax: Invoking the Menu mode.

```
VICAR>menu
```

4.3 VICAR Datasets

A VICAR dataset is a file with fixed length records (Section 4.3.2). This dataset can be categorized as image data, which is a digital representation of a visual "picture", or as non-image data. The dataset structure for both types of data is the same, thus simplifying the VICAR interface.

4.3.1 Dataset Names

The VICAR user should name disk datasets with care.

BEWARE *Unlike the VAX/VMS operating system, the UNIX operating system is case sensitive. While VICAR commands are not case sensitive, the filenames still are. Therefore,* `planet.img` *and* `PLANET.IMG` *are two different files on a UNIX system.*

BEWARE *VICAR users should avoid using system assigned logical names (on VMS systems) or environment variables and aliases (on UNIX systems) for dataset names. Assigned logical names can be obtained by typing the* `SHOW LOGICAL` *command in DCL,*

Syntax:

```
$ SHOW LOGICAL
```

or `setenv` *and* `alias` *in UNIX.*

Syntax:

```
% setenv
% alias
```

BEWARE *When VICAR is required to write an output dataset, the VICAR executive I/O routines will check the directory for an existing dataset of the same name. If one exists, it will write the new dataset directly over the old dataset instead of creating a new version. For example, if a dataset named* `over.byt` *was created at 10:15, and another image with the same name was created at 10:18, there would be only a single entry in the current directory:*

```
UNIX:

    -rw-r--r--  1 edd         10400 Aug  5 10:18
over.byt
not
    -rw-r--r--  1 edd         10400 Aug  5 10:18
over.byt
    -rw-r--r--  1 edd           550 Aug  5 10:15
over.byt.old

VMS:

    OVER.BYT;1               21   5-AUG-1994
10:15:34.38
not
    OVER.BYT;2               21   5-AUG-1994
10:18:41.23
    OVER.BYT;1                2   5-AUG-1994
10:15:34.38
```

BEWARE *On VMS systems, when a file is overwritten, the space allocation is updated, but the creation date is not. On UNIX systems, both are updated.*

4.3.1.1 Temporary Datasets

Temporary files are special files that are deleted when a user logs off. Under the old VMS VICAR, temporary files were specified by leaving off the filename extensions and assigning instead a `.Z`*xx* extension, where *xx* was based on the process ID. While that approach still works under VMS, files that begin with a plus sign (+) will now be recognized as temporary files on both UNIX and VMS systems.

Under the old system, temporary files were distinguised only by their name, and could be generated anywhere. The new style is to collect them all in one directory. Prepending a plus sign (+) to the name tells the VICAR RTL to put the files in the temporary

directory. This directory is pointed at by the `$VTMP` environment variable in UNIX, and the `VTMP:` rooted logical name in VMS. `VTMP` is set up by `vicset2` for both systems. (It normally points at `/tmp/username` for UNIX and a scratch directory for VMS). On VMS machines, because `VTMP:` is a rooted directory, accessing the directory outside of VICAR is a little awkward; you must use `vtmp:[000000]`.

BEWARE *The automatic deletion of the `/tmp/username` directory contents has not yet been established in UNIX VICAR. Therefore, in order to delete those files when you exit VICAR, you will need to place a `ush /bin/rm /tmp/username/*` command in your `ulogoff.pdf`. (See Section 5.4.2 for details on how to do this.)*

Subdirectories of `VTMP` are permitted. Under UNIX, they are referenced by `+sub_dir/file`, while under VMS, they are referenced by `+[sub_dir]file`. The subdirectories are not automatically created; they must be created in advance using `mkdir $VTMP/sub_dir` under UNIX and `cre/dir vtmp:[sub_dir]` under VMS. Because of these differences, the use of subdirectories is not portable between systems.

Currently, all processes using the same login id share the same temporary directory. This may be changed in the future so concurrent independent jobs will have separate directories. In the meantime, a workaround can be used, which is to redefine `VTMP` to use a process-specific directory name.

4.3.2 Dataset Structure

A standard structure has been established for VICAR datasets and all VICAR programs operate with this structure by calling standard interface subroutines. (See the *VICAR Run-time Library Reference Manual* and *The VICAR File Format*.)

A VICAR dataset is a file of fixed-length records consisting of five parts:

* VICAR label (primary label)
* binary label header - optional
* binary label prefix - optional
* pixel data
* end-of-dataset label (EOL) - optional

Although the exact structure of a VICAR dataset varies depending on the presence of the three optional parts (binary label header, binary label prefix, and EOL), all VICAR datasets follow the same structure.

The VICAR label is an ASCII string containing information describing the size, origin, processing history and attributes of the dataset.

The binary label header and prefix are optional areas for storing information about a dataset in free form binary format.

The pixel data portion of the dataset is made up of samples (pixels) of specified data format (Section 4.3.3). The data dimensions are described in terms of "number of samples" (NS - record length), "number of lines" (NL - number of records of length NS), and "number of bands" (NB - number of NL x NS data planes).

The end-of-dataset label is an optional area for continuation of the VICAR label.

For a more comprehensive discussion of dataset structure, refer to Section 9.1.

4.3.3 Pixel Data Formats

Each sample of a dataset contains the same number of bytes, as defined by the FORMAT item in the VICAR label (Section 9.2). The allowed values for FORMAT and the characteristics of these pixel format types are defined in the following table.

Table 4.3.3

Format	Bytes per Sample	Bits per Sample	Description
BYTE	1	8	unsigned, binary integer (data range: 0 to 255)
HALF	2	16	signed, binary integer (data range: -32,768 to +32,767)
FULL	4	32	signed, binary integer
REAL	4	32	floating point
COMP	8	64	a pair of REAL values
DOUB	8	64	double precision floating point

Note: There may still be a few programs which use the old convention of WORD instead of HALF, and COMPLEX instead of COMP. VICAR will continue to support these programs until they are converted to use the proper formats.

4.4 VICAR Labels

A VICAR label contains dynamic information that describes the size, origin, processing history and attributes of the associated dataset. All VICAR application programs are designed to read

information from the VICAR labels of the input datasets and dynamically update them.

All VICAR datasets must be in "VICAR format" which means they are required to have a standard VICAR label in order to be processed. Data received from other facilities with "foreign" (non-VICAR) formats require a special purpose program, called a "logging" program, to read and convert the data into VICAR format (Section 8.3.3).

4.4.1 VICAR Label Structure

A VICAR label is an ASCII string composed of label items which are *keyword=value* pairs separated by spaces.

Syntax:

```
    keyword=value

     where:    keyword   is a text keyword that
identifies
                  the label item

           value      is the information portion of the
                  label item; may be of type string,
                  integer, real, or double, and may be
                  multi-valued
```

Examples: Possible *keyword=value* pairs.

NL=800
FORMAT='HALF'
SIZE=(1,1,800,800)

A VICAR label contains 3 different classes of *keyword=value* label items:

* system
* property
* history

The system portion consists of those items that are independent of the history of the dataset. These items include: size of the image, its organization, its pixel format, host type and items indicating the existence of the optional sections of the dataset.

The property portion of the label contains items that describe properties of the image in the image domain, such as the map projection, lookup table, and latitude/longitude information.

The history label portion consists of information relating to the processing history of the data. Each time a program processes a dataset, VICAR adds history items to the label. The history items include: processing task name, user's identification, processing date and time. Thus, a sequence of subsets are recorded chronologically in the dataset label.

For a detailed description of the VICAR label structure, refer to Section 9.2. For examples explaining how to list labels, see Appendix 10.12.

4.5 Procs

When commanding VICAR, the user either issues intrinsic commands or invokes procedures or processes. Intrinsic commands are predefined operations used to manage a session. A procedure is a collection of VICAR commands that may be executed as one function. A process is a program which gets activated by VICAR.

Procedures and processes are collectively referred to as "procs" because they are identical in invocation syntax. Therefore, throughout this document "proc" will be used whenever it is not necessary to distinguish between procedures and processes (Section 7.3).

5 The VICAR Environment

Contents

Within the following section, different aspects of the VICAR environment will be covered. The user will be introduced to the VICAR Library, processing modes, and subprocesses. The user will learn how to customize a VICAR session and abort a VICAR command. The novice user should note that several concepts presented in this section are explained in more detail later on in this document.

5.1 VICAR Libraries

Seven libraries contain the executive, applications and system routines within VICAR. Each library is known by its system-wide logical name or environment variable (pointing to a system sub-directory).

```
      LIBRARY      CONTENTS
     VMS          UNIX
    V2$LIB    $V2LIB        VICAR executive routines
    VIDS$LIB  $VIDSLIB  VICAR system routines
    TAE$UTIL  $TAEBIN/TAEPLAT    Various system
utilities
    R1LIB     $R1LIB        Application procs
    R2LIB     $R2LIB        Application procs
    R3LIB     $R3LIB        Application procs
    M2LIB $TAEMENU  Menus
```

Three libraries are available for application procs so that a facility may segregate its procs as it chooses.

VICAR establishes a default search hierarchy containing these seven libraries, as well as the user's current default directory. A change of the current default directory after entry into VICAR is not reflected in the search hierarchy. The user may bypass all such searches by explicitly prefixing the proc name with the library name and a colon (e.g., `R2LIB:STRETCH`).

The user may display the currently established search hierarchy at any time with the command `show` (Appendix 10.4). The default hierarchy is listed below with the search being from top to bottom.

Example: Show user's library hierarchy.

`VMS:` *Note the location of the user's current default directory is UD:[USERID].*

```
VICAR>SHOW

User Library ($USERLIB):
UD:[USERID]
```

```
    Application Libraries ($APLIB):
    liblst:-cpd
    liblst:-pdf
    r3lib:-cpd
    r3lib:-pdf

    System Library ($SYSLIB):
    TAEPDF
```

UNIX:

```
    VICAR>show

    User Library ($USERLIB):
    .

    Application Libraries ($APLIB):
    $V2LIB
    $VIDSLIB
    $R1LIB
    $R2LIB
    $TAEBIN/$TAEPLAT
    $R3LIB

    System Library ($SYSLIB):
    $TAEPDF
```

On VMS systems, `liblst` is a logical name which points to all the VICAR libraries.

Example: Show user's library hierarchy.

```
    VICAR>DCL SHOW LOGICAL LIBLST

    "LIBLST" = "V2$LIB" (LNM$PROCESS_TABLE)
             = "VIDS$LIB"
             = "R1LIB"
             = "R2LIB"
             = "$TAEUTIL"
```

Within each library, compiled PDFs (file type `.cpd`) are given preference over slower, uncompiled PDFs (file type `.pdf`). Therefore, a program which has a `.cpd` and a `.pdf` will always have its faster version chosen. See Section 7.3.6 for more information on compiled PDFs.

This default hierarchy is easily altered either with the use of the intrinsic command `setlib` (Appendix 10.4), or within the user's ULOGON procedure (Section 5.4.1).

Example: Place `starlib:` in front of the current library list and display.

```
VICAR>setlib (starlib:,*)
VICAR>show

User Library ($USERLIB):
.

Application Libraries ($APLIB):
starlib:
$V2LIB
$VIDSLIB
$R1LIB
$R2LIB
$TAEBIN/$TAEPLAT
$R3LIB

System Library ($SYSLIB):
TAEPDF
```

Example: Delete `starlib:` from the current library list and display.

```
VICAR>setlib-delete starlib:
```

```
VICAR>show

User Library ($USERLIB):
.

Application Libraries ($APLIB):
$V2LIB
$VIDSLIB
$R1LIB
$R2LIB
$TAEBIN/$TAEPLAT
$R3LIB

System Library ($SYSLIB):
TAEPDF
```

5.2 Processing Modes

The user has several types of processing modes available within VICAR. Each mode is described in detail within this section.

* Interactive/Synchronous processing mode
* Asynchronous processing mode
* Batch processing mode

5.2.1 Interactive or Synchronous Processing Mode

When a user communicates directly with VICAR by means of a terminal, and VICAR immediately acknowledges and executes the user's requests, the user is in the "interactive processing mode". Within the interactive session the user might use:

* The Tutor mode - to learn more about procs and the parameters associated with them (Section 6.2)

* The Menu mode - to select or execute application procs (Section 6.3)
* VICAR Command mode - to execute procs or intrinsic commands (Section 7.1)
* VICAR DCL or USH mode - to execute DCL or shell commands (see below)

5.2.1.1 VICAR DCL Mode - VMS Systems

On VMS systems, the user has the ability to execute DCL commands without leaving the VICAR environment. The user can execute a single DCL command, or actually enter the DCL mode while still within VICAR.

In the first case, the user is able to execute a one line DCL command by typing `DCL` and the command.

Example: Execute a DCL command from VICAR.

```
VICAR>DCL SHOW DEV MTA0
```

If the user wanted to do something in DCL that requires more than a single command line, a second method is available. If, for example, the user wanted to send a mail message or edit a file, `DCL` would be typed at the VICAR prompt. The user would then receive a new prompt, `_$`, indicating that the DCL mode of VICAR had been entered.

Syntax: Enter VICAR's DCL mode.

```
VICAR>DCL
_$
```

Within this mode the user is able to do almost everything that could be done in the normal DCL mode. Some DCL programs may not work in VICAR's DCL mode.

The normal VICAR mode can be re-entered by typing `EXIT`, `VICAR` or `Control-Z`.

Syntax: Return to VICAR from DCL mode.

```
_$ EXIT
VICAR>
```

5.2.1.2 VICAR USH Mode - UNIX Systems

Similarly, on UNIX systems, the user has the ability to execute UNIX shell commands without leaving the VICAR environment by typing `ush` at the command line. (USH stands for User SHell.) However, `ush` should be used much less frequently than `dcl` is, because in a windowing environment it makes more sense to open another window than to suspend VICAR in order to execute shell commands. The `ush` command should be used primarily in procs rather than interactively.

Example: Execute a USH command from VICAR.

```
VICAR>ush df
```

Syntax: Enter VICAR's USH mode.

```
VICAR>ush
%
```

The 'shell' that is invoked is the command interpreter 'sh' or 'csh' as defined by the symbol `SHELL` when you logged in. You may redefine the value of `SHELL` from 'sh' to 'csh' or vice versa before invoking TAE.

The normal VICAR mode can be re-entered by typing `Control-D`, or `exit` at the shell prompt.

Syntax: Return to VICAR from USH mode.

```
% exit
VICAR>
```

BEWARE *Changing the default device/directory via the shell command* `cd` *while in USH mode or terminating a USH command with a backslash (\) will cause TAE to malfunction.*

5.2.2 Asynchronous Processing

EXPERT The Asynchronous processing mode allows a user to execute a proc in a separate subprocess without interfering with the user's current interactive session. This mode may sometimes be preferable to the Batch mode because the separate subprocess runs simultaneously with, and at the same priority as, the interactive session. The user's interactive environment is duplicated in the asynchronous subprocess, thus relieving the user from having to redefine commands and globals. More than one asynchronous job may run at the same time. Once submitted, the job is executed immediately.

EXPERT Because asynchronous jobs are executing in a separate subprocess, they cannot directly communicate with the user's terminal. However, they may communicate indirectly by

requesting additional parameter inputs. See the *TAE Command Language User's Manual* and Section 7.2.5 for further information on these "dynamic parameters".

5.2.2.1 Asynchronous Processing - VMS Systems

EXPERT For asynchronous jobs run on a VMS system, the VICAR runstream information is written into a log file named *PROCNAME*.TML.

Example: Submit `TESTGEN.PDF` as an asynchronous job.

```
VICAR>TESTGEN|RUNTYPE=ASYNC|
[TAE-ASYNCJOB]Asynchronous job 'TESTGEN' initiated.
```

The user may check on the status of the job with the intrinsic command `SHOW-ASYNC`.

Example: Monitor the progress of `TESTGEN`

```
VICAR>SHOW-ASYNC

 NAME          PROC  STATE     SFI  SKEY

 TESTGEN       TESTGEN   ACTIVE    0    TESTGEN
```

BEWARE *The user should not exit a VICAR interactive session while asynchronous jobs are executing. If this happens, all asynchronous jobs will be aborted.*

BEWARE *The user is also advised to avoid using tape operations within the Asynchronous mode until a known bug can be corrected. Tape drives can be successfully allocated, mounted, written or read. However, the tapes cannot be successfully dismounted and/or remounted within the Asynchronous mode.*

5.2.2.2 "Background" Processing - UNIX Systems

VICAR running on UNIX systems does not have an "asynchronous mode"; given that a user can have several windows open at the same time, it is usually unnecessary. However, processing jobs in the "background" is similar in concept.

To do this, your proc will need to be submitted from the shell prompt.

Example: Submitting a VICAR proc from the shell prompt. (Note: this only works on SunOS, not Sun Solaris systems.)

```
% cat test.pdf
Procedure !test
Body
slogon
gen out.img 10 10
label-list out.img
ush ps
End-proc
% taetm test > test.log &
[1] 9147
```

The `slogon` statement in the proc after the `Body` command is necessary if the proc is going to be run from the UNIX shell prompt. However, it may not be present when running the proc from the VICAR command line. The > sign redirects the output from the proc to the file `test.log`, rather than to your screen.

(There is currently a bug in VICAR, such that the output is not written to the output file in the correct order. This should be fixed in future versions.) The ampersand (&) forces the job into the background. Typing `fg` at the prompt will bring the job back into the foreground.

5.2.3 Batch Processing Mode

The user may wish to execute a proc in batch. Batch processing is achieved by means of submitting a file, commonly referred to as a job, to a batch queue which will create the proper environment to execute a proc. Upon submitting the file, the user relinquishes control of the job to the operating system, thus freeing the terminal and allowing the user to continue with other tasks. The user may check on the status of the job with the intrinsic command `SHOW`.

BEWARE *The default directory for a batch job is the directory from which the submittal occurred. If submitted from a subdirectory, all file names should be fully qualified because this subdirectory may not exist on all disks referred to in the job.*

5.2.2.1 Batch Processing - VMS Systems

EXPERT There are several ways to invoke batch processing in VICAR on VMS systems. This section will address the VAX/VMS default batch queue, `SYS$BATCH`. If this queue is not on the user's system, consult the local system manager for information on what is available.

EXPERT One way to create a batch process is to use the intrinsic command `BATCH-SUBMIT` (Appendix 10.4). After Tutoring on a desired proc and performing a Tutor `SAVE` (Appendix 10.8) on the parameter values selected, the user can submit a job with the

BATCH-SUBMIT command. This can be done in a Tutor session on BATCH-SUBMIT or interactively.

Syntax:

```
VICAR>BATCH-SUBMIT PROC=proc_name +
VICAR>+ SAVEFILE=savefile_name QUEUE=queue_name +
VICAR>+ STDOUT=output_file
```

Example: After specifying parameters in Tutor on proc GEN, submit to batch queue.

```
VICAR> BATCH-SUBMIT PROC=GEN.PDF SAVE=GEN.PAR
Job 2038 submitted to queue SYS$BATCH
```

later:

```
Job GEN (queue SYS$BATCH, entry 2038) completed
```

EXPERT Another method for submittim a job is to use the command qualifier RUNTYPE (Section 7.2.1.3). Specifying a valid *queue_name* or NORUN will produce two different submitting techniques. Specifying SYS$BATCH as the *queue_name* causes the job to be automatically placed in that queue and executed.

Example: Submit proc TESTGEN to batch queue SYS$BATCH

```
VICAR>TESTGEN |RUNTYPE=(BATCH, SYS$BATCH)|
Job 2039 submitted to queue SYS$BATCH

VICAR>SHOW-BATCH SYS$BATCH
Batch queue MIPL1_SYS$BATCH, on MIPL1::
```

```
Jobname       Username  Entry     Status

TESTGEN        USERID   2039 Executing
```

later:

```
Job TESTGEN (queue SYS$BATCH, entry 2039) completed
```

EXPERT Specifying NORUN as the queue name disables the act of job submittal. Instead, a job file is created containing all commands needed to execute the proc in batch. This method allows the user to issue the DCL SUBMIT command with any or all of its associated qualifiers, rather than accepting the VICAR defaults.

Example: Create the job file for TESTGEN proc before submitting it.

```
VICAR> TESTGEN|RUNTYPE=(BATCH,NORUN)|
Created batch job file 'TESTGEN.JOB'.
```

EXPERT The .JOB file (TESTGEN.JOB for this example) would then be submitted using the DCL SUBMIT command, unless other provisions have been made by the system manager.

Example: Submit TESTGEN.JOB to SYS$BATCH queue.

```
  VICAR>DCL SUBMIT/NOPRINT/NOTIFY TESTGEN.JOB
  Job TESTGEN (queue SYS$BATCH, entry 2041) started on
SYS$BATCH
```

later:

```
Job TESTGEN (queue SYS$BATCH, entry 2041) completed
```

EXPERT Upon completion of a batch job, the user can access a file containing all of the processing information collected during the batch job execution. The log file is located in the directory from which the job was submitted, and it is called *PROCNAME*.LOG (`TESTGEN.LOG` for the above example).

EXPERT The user may also delete a batch job at any time.

Syntax:

```
VICAR>BATCH-DELETE QUEUE=queue_name JOBID=xxx
```

5.2.3.2 Batch Processing - UNIX Systems

Formal batch processing does not exist on UNIX systems at the current time. However, it is possible to submit a job to run at a later time, perhaps when the system load will be lower. This may be done by creating a shell script to initiate your proc. This shell script is submitted as a "batch" job by using the UNIX `at` command. (The UNIX `cron` command could also be used.)

Example: Use of the `at` command on a SunOS system. (Batch processing doesn't work on Solaris systems currently, but when it does, be aware that the `at` command has a slightly different syntax on those systems.)

```
% cat submit_test.sh
#!/bin/sh
```

```
taetm test
% at 23:00 submit_test.sh
job 11719 at Tue Aug  9 23:00:00 1994
```

(Remember to turn the execution permission for your shell script on using the UNIX command `chmod u+x submit_test.sh`). You will receive a mail message containing the output from your job when it is completed. As stated above in Section 5.2.2.2, there is currently a bug in the output files from these batch jobs.

5.3 VICAR Use of Subprocesses

This section is relevant for VMS systems only.

WIZARD The VICAR executive makes use of VMS subprocesses to establish environments and isolate functions. This discussion is intended to clarify how a user's session is being supported under the VMS operating system.

WIZARD A user logged in under VMS has a process associated with the session, e.g., process name `PRCNM`. Entering the command VICAR starts a VMS subprocess with a process name `PRCNM1`. This subprocess is the environment under which all VICAR operations are handled. The VICAR `EXIT` command returns the user to the parent VMS process and deletes the subprocess and all its descendent subprocesses.

WIZARD In the case of a batch job submittal (Section 5.2.3) from VICAR, a separate VMS process is initiated. The name of the submitted procedure PDF is taken as the process name of the batch job, e.g., `MYPDF`. When the job begins to execute, a subprocess (`MYPDF1`) is created as the environment from which the VICAR commands are executed. Both process and subprocess, of course, go away at job termination.

WIZARD Asynchronous processes (Section 5.2.2) are handled similarly. They differ from batch jobs in that a subprocess is initiated under which a second subprocess is created for the VICAR environment. The process name of the first subprocess is created by concatenating the eight character Process ID number of the VMS process with the first four characters of the PDF being run and appending a character zero. Therefore, the first subprocess may have a process name like `22058225IMAG0`. The second subprocess gets that name with a "`1`" appended to it.

WIZARD The following diagram illustrates the relationships of the VMS processes and subprocesses utilized by VICAR.

WIZARD Because of this interesting relationship of processes and subprocesses, a user must realize that observing the progress of a process with a DCL SHOW SYSTEM command, for instance, may be meaningless because most of the action is occurring within one or more subprocesses.

WIZARD Normally, actions performed in DCL mode of VICAR will take place in the subprocess. It is possible, however, to affect the parent process as well (see Section 7.2.1.1 for more discussion in this area). DCL commands which allow the /JOB qualifier affect both the parent and the subprocess when the qualifier is present.

Example: Define and use a logical name for a parent and subprocess.

```
VICAR>DCL DEFINE/JOB A UD:[USERID]
VICAR>ENABLE-SCRIPT A:GEOMIT.SCR
```

5.4 Session-customizing

Logon and logoff procedures are used by VICAR to create the proper environment. VICAR invokes its own logon and logoff system procedures upon entering and exiting the system. The session may be further customized by the execution of the user's logon and logoff procedures. The creation of the user procedures are explained in the following sections. The examples presented are very useful and might be used as a starting point.

5.4.1 Logon Procedures

When the user enters VICAR, the system logon procedure, `slogon`, is invoked. `slogon` is a facility-dependent logon procedure which is normally invisible to the interactive user and is typically created and maintained by the system manager. Once VICAR has been invoked, the operating system executes this logon procedure and a series of steps are executed in order to set up the VICAR environment. One of the last steps in the startup procedure is to examine the user's present directory for a `ulogon.pdf` and to execute that procedure if it exists.

The `ulogon` is a procedure typically written and maintained by the user in order to customize the initialization of the VICAR. The `ulogon` is not a required procedure but most users find it very useful. For example, the `ulogon` can be used to define the user's VICAR commands, specify the location of the directories where the user's application software resides or to configure the user's VICAR session.

Example: ulogon.pdf (A detailed, line-by-line, description can be found in Appendix 10.11).

```
Procedure
Refgbl $PROMPT
Refgbl $BECHO
Refgbl $ECHO
Body
ENABLE-LOG
DEFCMD SCR "Enable-script"
DEFCMD CHK "Syntax check"
DEFCMD NOCHK "Syntax nocheck"
DEFCMD QUE "ush lpq"
LET $ECHO= "YES"
LET $BECHO= ("YES","YES")
LET $PROMPT="GoGetum"
End-proc
```

The user may define a VMS logical name to point to a `ulogon`. Doing this causes the same `ulogon` to be executed regardless of the default directory. At the current time, this cannot be done on UNIX systems, and a user must have a copy of their `ulogon.pdf` in each directory from which they wish to run VICAR.

Example: Define a VMS logical name, in user's LOGIN.COM, pointing to user's `ULOGON`.

```
$DEFINE ULOGON MGN4:[USERID]ULOGON.PDF
```

Alternatively, the user could have a custom `ulogon` in each directory by not defining the logical name and maintaining separate `ulogon`s.

5.4.2 Logoff Procedures

The `slogoff` is a facility-dependent logoff procedure which is activated when the user exits from VICAR. One of the steps in the procedure is to examine the user's current directory for a `ulogoff.pdf` and to execute that procedure if it exists.

The `ulogoff` is typically written and maintained by the user in order to customize the exiting from the VICAR session. The `ulogoff` is not a required procedure but some users find it to be very useful for directory maintenance. For example, the `ulogoff` can be used to delete unnecessary files from the user's directories or to automatically print out the latest version of the `session.log`.

Example: ulogoff.pdf (A detailed, line-by-line, description can be found in Appendix 10.11)

```
Procedure
Body
DISABLE-LOG
ush /bin/rm last.par
ush /bin/rm session.tsl
ush lpr session.log
End-proc
```

As with the `ulogon` the user should define a VMS logical name to point to a `ulogoff`. Doing this causes the same `ulogoff` to be executed regardless of the default directory. (Again, this is not currently possible on UNIX systems.)

Example: Define a VMS logical name, in user's LOGIN.COM, pointing to user's `ulogoff`.

```
$DEFINE ULOGOFF SYS$LOGIN:ULOGOFF.PDF
```

Alternatively, the user could have a custom `ulogoff` in each directory by not defining the logical name and maintaining separate `ulogoff`s.

5.5 Aborting a VICAR Command

VICAR provides the user with the ability to interrupt a VICAR operation once execution has started. VICAR has defined the key sequence `Control-C` to activate "proc interrupt mode". Upon entering `Control-C`, the operation is suspended and the user is prompted by the "interrupt prompt" for appropriate commands.

Syntax:

```
VICAR-INTERRUPT>command
```

The user may enter one of the following commands:

* ABORT - This forces termination of the operation and returns the user to normal VICAR Command mode
* CONTINUE - This resumes the operation
* Other VICAR Intrinsic commands (Appendix 10.4).

Note: If a synchronous proc is invoked, the following error is issued:

```
[TAE-NOSYNC] Synchronous procs not available in
proc
interrupt mode.
```

While a proc is interrupted, it is valuable to be able to perform Intrinsic commands and then resume the proc. The commands will take effect immediately.

Example: Specify an Intrinsic command and resume a proc.

```
VICAR-INTERRUPT>let $echo="yes"
VICAR-INTERRUPT>continue
```

BEWARE `Control-C` is the only sequence defined for "Proc interrupt mode". Other control characters will have very different results. (See Section 7.1.1.5 for more information.)

6 VICAR User Aids

Contents

Within this section, a variety of user aids will be introduced. These tools will prove invaluable in all stages of the user's development. The aids being covered include Help, Tutor and Menu modes, syntax checking and message interpretation.

6.1 VICAR Help

The VICAR executive assists users of all experience levels. It is able to give on-line information and instructions to a user when trouble is encountered at any point in a session. The following types of "help" are available:

* Help information on a menu (Section 6.3)
* Help information on the operation of Tutor mode (Section 6.2)
* Help information on procs or intrinsic commands Section 6.1.1)
* Help information on the parameters of a proc Section 6.1.1)
* Help information on VICAR messages (Section 6.5)

If questions still remain after exhausting these options, further assistance can be obtained by posting questions to the USENET newsgroup `jpl.vicar` or sending them to `vicar@mipl7.jpl.nasa.gov` via email.

6.1.1 Help Command

General help information on VICAR can be obtained by typing `help` with no subcommand. General information is available on most commands used in the Command mode.

Syntax: General Help information.

```
VICAR> help
```

Help on a specific topic can be obtained by using one of the following `HELP` subcommands:

```
help-command  (default for help)
help-proc     (default for help)
help-parm
help-global
help-message  (Section 6.5.2)
help-hardcopy
```

`help-command` and `help-proc` are the defaults for `help`. VICAR locates the command or proc by first doing a search among an intrinsic command list and then, if not found, performing a hierarchical library search.

The `-command` subcommand displays information on the specified command. If a subcommand is specified, then the information displayed will be for the subcommand only.

Syntax:

VICAR>help-command command=*command*[-*subcommand*]

Example: Display help on the command `enable-script`.

VICAR>help-command enable-script

The `-proc` subcommand displays information on the specified proc or proc subcommand.

Syntax:

VICAR>help-proc proc=*proc*[-*subcommand*]

Example: Display help on the proc `label-list`.

VICAR>help-proc label-list

The `-parm` subcommand is used to request detailed information on a specified parameter of a proc or a proc subcommand.

Syntax:

```
VICAR>help-parm parm=parm proc=proc[-subcommand]
```

Example: Display help on the parameter `nl` of the program `gen`.

```
VICAR>help-parm nl gen
```

The `-global` subcommand is used to request a detailed description of the specified Global variable.

Syntax:

```
VICAR>help-global variable=global_variable_name
```

Example: Display help on the global variable `$aplib`

```
VICAR>help-global $aplib
```

The `-hardcopy` subcommand writes a disk file containing information from both the `help-proc` and from Tutor. This file can then be printed in order to obtain a hardcopy of the information. If an output filename is not specified the file will be named *procname*`.mem` and it will be located in the user's current directory.

Syntax:

```
VICAR>help-hardcopy proc=proc [output=filename]
```

Example: Create a file (`difpic.mem`) of help information for the program `difpic`.

```
VICAR>help-hardcopy difpic
```

An alternate method to obtain a hardcopy of the help information is to print the *`procname`*`.pdf` from the applications library.

6.2 VICAR Tutor Mode

6.2.1 General Tutor Information

Within VICAR there is a mode called "Tutor" which does exactly what its name implies. It tutors or assists the user in the use of procs and the parameters associated with them. Tutor not only permits the user to obtain more information concerning procs, but it actually permits the user to select parameters and then execute the proc via the `run` command.

Tutor mode can be accessed by any one of the following methods:

* Typing `tutor` *`proc_name`* in Command mode
* Selecting a proc in Menu mode
* Executing the Tutor command in Dynamic Parameter mode (Section 7.2.5)

* Hitting the key sequence `esc esc` at the proc prompt (Section 7.1.1.5) in an interactive proc and then typing `tutor` There are actually two modes of Tutor available.

TUTOR-SCREEN is a formatted terminal screen display (Section 6.2.2).

TUTOR-NOSCREEN is similar in style to the TAE Command mode but provides a different prompt, lists the parameters requested and allows a set of commands unique to a Tutor session (Section 6.2.3).

When tutoring is initiated, the mode entered by Tutor will be determined by the Global variable `$tutor`. It can be overridden by using one of the Tutor subcommands (`-screen`, `-noscreen`).

Syntax:

```
VICAR>tutor[-subcmd] proc_name[-subcmd] [proc-parameters]
```

Example: Tutor the program `label-list` in SCREEN mode.

```
VICAR>tutor-screen label-list
```

The *proc-parameters* field is optional and may contain a list of parameters. Parameters supplied on the Tutor command line become initial or default values in the Tutor session.

6.2.2 TUTOR-SCREEN Mode

TUTOR-SCREEN mode commands allow the user to:

* Request detailed information for parameters
* Update parameter values
* Scroll between pages of the Tutor display
* Scroll between components of a multi-valued parameter
* Request Tutor information on parameter qualifiers
* Save the parameter values
* Restore parameter values
* Execute the proc

Upon entering Tutor mode the user is presented with the first page of a possibly multi-paged display showing information on each parameter of the selected proc or command. For each parameter, Tutor displays the following: the parameter name, a brief description of the parameter, its current value (if one exists), and a note on qualifiers defined for the parameter (if there are any).

EXPERT When a multi-valued parameter has more values than will fit on one Tutor screen, only one full screen is shown. Tutor maintains a "window" on the elements of the parameters. Unless a particular element is referenced (e.g., *parm*`(`*i*`)=`), the display window includes the first element. This window may be moved by using the intrinsic command `show`.

The "+" adjacent to the page number in the upper right corner of the Tutor display indicates that there are more pages available within the display. If the character in that position is a period, there are no more pages.

Descriptions of Tutor mode user commands and special Tutor line editor keys can be found in Appendices 10.8 and 10.9, respectively.

6.2.3 TUTOR-NOSCREEN Mode

The NOSCREEN mode of Tutor exists to support the following situations: hardcopy terminals, unsupported terminals for which VICAR can only operate in scrolling mode, low-speed terminals where the time for screen update is prohibitive, or personal choice (some users prefer the NOSCREEN Tutor mode).

NOSCREEN Tutor mode commands allow the user to:

* List parameters and their values
* Display parameters along with brief descriptions
* Update parameter values
* Request detailed information for parameters
* Request tutoring on parameter qualifiers
* Activate the proc

There are operational differences between NOSCREEN and SCREEN modes. Features unique to NOSCREEN mode are:

* When TUTOR-NOSCREEN is initially entered, a list of parameter names is automatically displayed.
* The user is prompted for commands with a **VICAR-*proc_name*>** prompt.
* The paging commands (`page` and `<cr>`) are not available.
* When a parameter is given a value, there is no automatic echo of the value.
* The `list` command may be used to display parameter names and current values. This command may be supplied with a list of specific parameters to be displayed, or if no list is supplied, all parameters for the proc will be displayed.
* The `display` command may be used to display parameter names, values and a descriptive text for each parameter.

Descriptions of Tutor mode user commands and special Tutor line editor keys can be found in Appendices 10.8 and 10.9, respectively.

Example: TUTOR-NOSCREEN Session

```
    VICAR>tutor-noscreen difpic
    proc "difpic", library "$R2LIB"
    Parameters requested:  INP, OUT, SIZE, SL, SS, NL,
NS...
    VICAR-difpic>list
     INP= (no value specified)
     OUT= -- (null value)
     SIZE=(1,1,0,0)
     SL=1
     SS=1
     NL=0
     NS=0
     NB=0
     FORMAT= -- (null value)
     STAT="STAT"
     MOD= -- (null value)
    VICAR-difpic>inp=(a.bat,b.cat)
    VICAR-difpic>out=d.dat
    VICAR-difpic>size=(1,1,400,350)
    VICAR-difpic>list size
     SIZE=(1,1,400,350)
    VICAR-difpic>run
```

6.3 VICAR Menu Mode

6.3.1 General Menu Information

(At the current time, only a bare skeleton of the menu system is available in UNIX VICAR. Most of the information below is for the VMS system, but should apply to the UNIX system in the future.)

Menu mode is the VICAR alternative to Command mode. In Menu mode, a user invokes applications by locating and selecting them through a series of menus. Each menu is a terminal display containing categories, each describing either a more detailed menu or a proc to be executed.

There are two methods to access the VICAR Menu mode. Depending on system configuration, the user may enter the Menu mode when invoking VICAR. In this instance, the user is given a "ROOT" menu and the Menu prompt **?**. Alternatively, the user can manually enter the Menu mode.

Syntax:

```
VICAR>menu [name=menu_name]
```

Where: menu_name is the name of the desired menu.

If this is the first entry into Menu mode for the current session, the default menu is the ROOT directory defined by the host system. If it is not the first entrance, then the default will be the most recent menu the user accessed. The menu is located using the hierarchical library search unless the user explicitly specifies the library name of the menu.

Example: A typical menu screen.

```
VICAR>menu name=root
```

The above figure is an example of the format of a menu terminal display. The categories are numbered entries arranged vertically on the screen. The prompt-line options are at the bottom of the screen arranged horizontally above the menu prompt.

At the menu prompt, **?**, the user selects a category or enters a Menu option. If the category selected is a menu, the new menu is displayed. If the category selected is a proc, VICAR enters Tutor mode in order to prompt the user for the proc's parameters. After execution of the proc, the display of the last menu can be obtained by pressing the `<CR>` key.

If a Menu option is selected, VICAR executes the command. Menu options are limited to those found in Appendix 10.7. These allow only basic management functions which include: accessing, moving back up the menu hierarchy, making a transition to Command mode, and exiting VICAR.

EXPERT A user may switch back and forth between Menu mode and Command mode. When entering the Command mode from a menu (using `command`), the current menu and the path to the current menu are remembered. This context is automatically restored when Menu mode is subsequently re-entered.

6.3.2 The MENUTREE

(At the current time the Menutree is only available on VMS systems.)

WIZARD The ability to move forward or backward through the levels of a menu is essential to operating within VICAR. Unfortunately the user also runs the risk of conceptually "losing their place" in the system because of this flexibility. There is a program, `menutree`, available within VICAR which permits the

user to generate a graphic representation of a selected menu system.

Syntax:

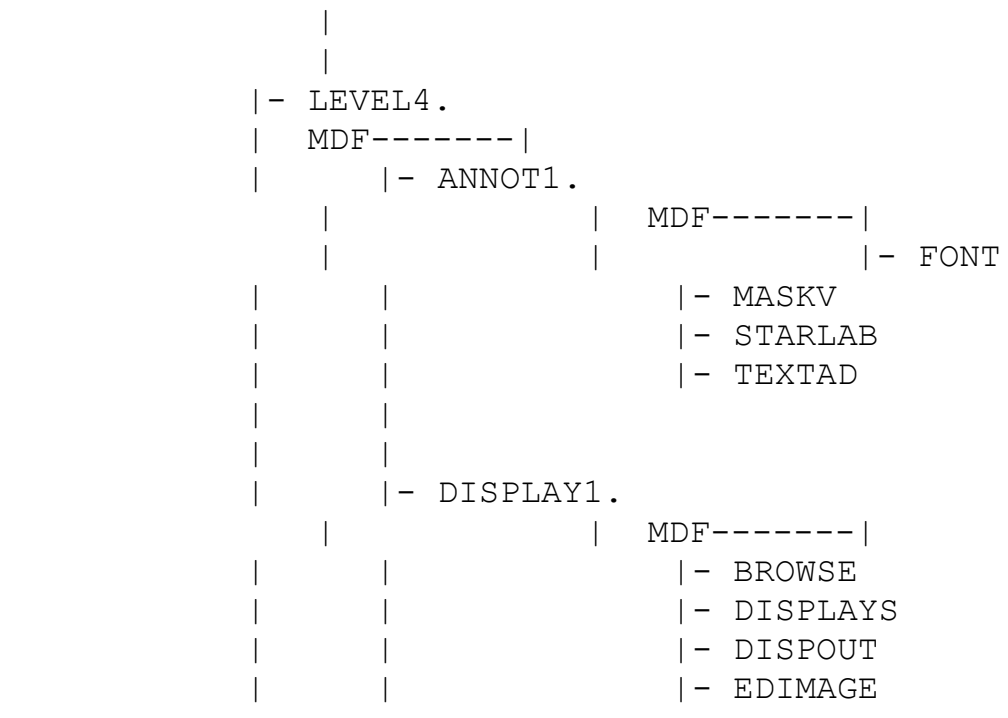
```
    VICAR>menutree [menu=menu_name]
[output=output_option]

Where:   menu specifies the starting menu name.  If
omitted,
         the default value is the top of the current
         menutree (see Global variable $menus).
     output   directs the listing of the menutree:
         = file       -  create menutree.txt
         = printer    -  send to line printer
         = terminal   -  send to terminal (default)
```

Example: Menutree run on the menu presented in Section 6.3.1.

```
    VICAR>MENUTREE MENU=TAE$MENU:ROOT OUT=T

    ****************  VICAR HELP  INFORMATION
****************
    TAE$MENU:
    ROOT-------|
            |> TAE$MENU:INTRO.PDF
            |> NUT
            |- LEVEL3.
            |  MDF-------|
               |            |> TAE$MENU:SAMPLE1.PDF
               |            |> DCL @TAE$MENU:GEOREF.COM
               |            |> DCL
@TAE$MENU:TIEPOINT.COM
               |            |> DCL
@TAE$MENU:PROJECTION.COM
               |            |> DCL
@TAE$MENU:VARIATE.COM
```



.
.
.

WIZARD If Menutree cannot be located in `TAE$UTIL`, ask your system administrator for the proper location.

6.3.3 Creating a Menu

WIZARD A menu corresponds to a Menu Definition File (MDF), which is a text file created or changed by using the host system's text editor. MDFs contain the title of the menu, text describing each menu entry, name of the proc or menu files associated with each entry and help information for the menu. Further information on menu creation can be found in the *TAE Command Language Programmer's Manual*.

6.4 Syntax Checking

A user has the ability, within VICAR, to check the syntax of a procedure (Section 7.3) that has been written. This is particularly useful for procedures that will be run in the Batch mode. The syntax checker sets a switch which prevents VICAR commands from executing. However, all normal processing up to the actual procedure execution does take place, so any syntax or parameter errors that are visible to VICAR will be detected and reported to the user.

Once the syntax checker has been invoked, anything that is typed will be verified (e.g., VICAR commands or procs).

Syntax:

```
VICAR>syntax check
```

BEWARE *The syntax checker only verifies the wording to make sure all commands are valid VICAR commands. It does not check logic errors, existence of referenced files or availability of resources.*

BEWARE *All parameters required by a proc need to be supplied in order to successfully run the checker. Syntax check will fail when variables are assigned at time of execution (e.g., the proc* `camparan`*).*

The syntax checker will remain invoked until the user turns it off wth the following command.

Syntax:

```
VICAR>syntax nocheck
```

BEWARE *The command* `let $switch=...` *should not be used in SYNTAX mode, since the syntax check flag is in* `$switch`*. By setting* `$switch` *to an absolute value, it is possible to accidently turn off the syntax check mode. To change other options within* `$switch`*, use the* `flag-add` *or* `flag-del` *commands. Further online Help information is available on flags or see* *Appendix 10.4**.*

Syntax:

```
VICAR>help flag
```

6.5 Message Interpretation

6.5.1 General Message Information

VICAR has the capability of providing online information and instruction to a user when trouble is encountered. The user is presented with brief messages when the error occurs and can request additional clarification and guidance.

Brief error messages are one-line comments preceded, in brackets, by a "message key".

Syntax:

```
    [SYSTEM-KEY] Message

where:    [SYSTEM-KEY]  is the "message key"
     SYSTEM          indicates which system issued the
message:
```

```
                    VIC2 - VICAR Run-time Library
                    TAE  - the TAE supervisor
     KEY          is the specific identifier for the
                  message given
     MESSAGE           is the text containing a comment
from
                  VICAR
```

Example: End of volume message.

```
    [VIC2-ENDOFVOL] End of volume (double tape mark)
reached
```

In many cases, the brief error message adequately informs the user of the problem. However, in some cases the user might still be at loss as to what VICAR is trying to explain. In that case, there are two methods for obtaining further online assistance on the message interpretation:

* by using the VICAR command `help-message`
* by using the VICAR command `?`

In addition to the online error facilities, the user can make use of several other sources containing common message explanations, including:

* Appendix 10.13 of this document
* the *VICAR Run-time Library Reference Manual*
* the disk file `$TAEHELP/msg/taefac.msg` on UNIX machines or `VICSYS:[VICAR121.TAE52_VAX-VMS.HELP.TM] TAEFAC.MSG` on VAXes.

6.5.2 HELP-MESSAGE

The command `help-message` is used when the user wishes VICAR to expand upon the received message. VICAR responds with an explanation of the error message and a suggested course of action for the user.

Syntax:

```
VICAR>help-message key=message-key
```

Example: Get more help on the end of volume message.

```
VICAR>help-message key=vic2-endofvol

Explanation:

The end of volume mark (double tape mark or double end
of file) was hit when trying to open a file on an input
tape.

User action:

Scan tape to determine the actual number of files on it,
and make sure that the proc does not try to access a file
beyond that number.
```

6.5.3 "?"

An alternate method to obtain the same Help information is to type the VICAR command "`?`". "`?`" is special in that no other parameters are allowed and it always causes VICAR to display help information on the last message received.

Syntax:

VICAR>?

7 The VICAR Image Processing Executive

Contents

Within this section, the major aspects of the VICAR executive are covered. Complete detail has not been attempted, but all information likely to be commonly required by users has been included. The major concepts being described include command rules and syntax, parameter entry, procedure and script file construction, session logging, and tape handling.

The new user may be confused the first time through this section, since a lot of important information is presented at the same time. Some of the earlier examples might be particularly confusing because they use concepts that are introduced a little later. The new VICAR user should try to understand the basic concepts the first time through and not worry about understanding everything. By the end of this section, the concepts should all start to make sense.

7.1 Introduction to the TAE Command Language (TCL)

Communication between the user and VICAR is accomplished through the TAE Command Language (TCL). TCL allows the user to access procs, supply parameters to them, and initiate their operation using a single command line. TCL provides the following tools to achieve this:

* local and global variable assignment

* expression evaluation
* use of built-in functions
* macro-level substitution for variables and parameters
* conditional execution and looping

7.1.1 TCL Command Line Rules

While VICAR GUIs are currently being developed, at the present time all user communication to VICAR is accomplished through the use of the TCL Command Line. VICAR functions and procs are invoked by using established rules and syntax. The standard rules governing command lines are covered in this section, while syntax is detailed in Section 7.2.1.

7.1.1.1 Abbreviations

The user is allowed to use abbreviations for the following cases:

* Intrinsic commands (Appendix 10.4)
* Tutor, Menu and Help commands and subcommands
* Names of parameters used in command lines

Proc names, library names and file names may not be abbreviated.

The extent of allowed truncation is governed by one rule: the abbreviation must uniquely specify one command or parameter name out of all those allowed at the time.

Example: Abbreviation of parameters.

```
VICAR>F2 A B FUN="IN1*10." 'HA
```

will work as well as the full command line:

```
VICAR>F2 A B FUNCTION="IN1*10." 'HALF
```

7.1.1.2 Line Continuation

A command line may be continued to a maximum of 2048 characters on subsequent lines by ending it with a "+" character. Lines may not be continued within command names, comment fields or parameter values or names.

Example: Continuation of a command line.

```
VICAR>FARENC URANUS.RED (A,B,G) +
VICAR>+ 'AUTO GEOM=3 AREA=(50,50,700,700)
```

BEWARE *DCL and USH commands submitted at the VICAR prompt follow the above rules (i.e., using the "+" character). All those submitted from the VICAR DCL mode or USH mode use the standard VMS DCL rule of continuing lines with a "-" character, or the UNIX rule of using a "\" character.*

7.1.1.3 Labels

EXPERT Any command line within VICAR procedures may be labelled. A label is created by prefixing a line with a string of characters (which make up the label) and the ">" character. This label may then be referenced by the `GOTO` command elsewhere in the procedure.

Example: Label use in procedures (the label is "`BATCH`").

```
Procedure
Refgbl $RUNTYPE
Body
IF ($RUNTYPE="BATCH") GOTO BATCH
WRITE "ARE WE HAVING FUN YET?"
```

```
BATCH>TESTPROC (A,B,C)
End-proc
```

BEWARE *Contrary to normal Fortran programming standards, only forward branching is permissible.*

7.1.1.4 Special Characters

The special characters allowed by TCL and their meanings are listed in Table 3-2 of the *TAE Command Language Programmer's Manual*. The most useful of those symbols are listed here. Only the "&" character retains its meaning when placed within a quoted string (a character string bracketed by " " characters).

```
CHARACTER      MEANING

 !     comment field indicator
 &     variable substitution flag
 @     variable dereferencing flag
 " "   start and end of quoted string
 |     start or end of qualifier field
 ,     separator used in lists
 SPACE     general separator
 ?     query for help on last message
 '     flag for keyword value
 +     line continuation indicator
```

7.1.1.5 Special VT100 Terminal Keys

The use of certain special keys has effects that are unique to TCL. The most important of these keys are `ESCAPE`, `Control-C`, `Control-Y` and `Control-O`. Information on the keys used by the Command Line Editor and Tutor mode are found in Section 7.1.1.6 and Appendix 10.9. The following information describes the VICAR Command mode.

Typing the ESCAPE key twice (ESC ESC) serves two functions within VICAR. When entered on any continuation line, it will cancel the entire command. When entered at a prompt from some interactive procedures, the procedure will be interrupted. Options available at this point are ABORT, TUTOR, and EXIT. ABORT will terminate the procedure, TUTOR will put VICAR in Tutor mode, and EXIT will return the user to the procedure prompt.

A Control-C entry will abort a VICAR operation (Section 5.5) and prompt the user for a further command. After appropriate response to the VICAR-INTERRUPT> prompt, the user is left in VICAR.

On VMS systems, Control-Y will abort an operation and, in addition, will end the VICAR session, lose VICAR dummy names and some allocated devices, and return the user to the DCL prompt. On UNIX systems, it will suspend the VICAR session and return the user to the shell prompt. Typing fg will re-initiate the VICAR session.

A Control-O entry will discontinue all further output to the terminal screen from the current proc. The proc continues to run to completion and continues to output to the session log file. A second Control-O entry will again route output to the terminal screen.

7.1.1.6 Command Line Editor

A command line editor is implemented in VICAR for VT100-compatible video terminals. This editor allows the user to recall previous commands and modify the current (or recalled) command. Commands which may be recalled include those issued from within an interactive program (IDX being an exception). The special keys utilized by the editor are described in Appendix 10.10.

On VMS systems, the method of line modification (either insertion or overstrike) is controlled by that selected under DCL (the SET TERM command). A Control-A will toggle between the options. (The insertion mode is the only one available on UNIX systems.)

The user may scroll backward and forward among the recallable commands with the UP ARROW and DOWN ARROW keys. UP gives earlier commands; DOWN gives later.

BEWARE *Restrictions:*

* *Only the twenty most recently entered commands are recallable.*
* *This feature is not available for commands entered in VICAR's DCL or USH modes.*
* *Only the first line of multi-line commands may be recalled.*

7.1.2 Variables

TCL allows the definition of named values called variables. These may be assigned values, be the object of substitution or be used in expressions. Variables may be of two distinct scopes: Local or Global. All variables must be defined before use. The intrinsic command DISPLAY may be used to examine the value of variables. (See Section 10.4 for an explanation of the DISPLAY command.)

7.1.2.1 Local Variables

A Local variable may be referenced only within the procedure or session in which it was declared. These variables must be defined with the LOCAL command before use in an interactive session or in a procedure. The definition of a Local variable in a procedure may be made before or after the BODY statement. The following characteristics are defined for the LOCAL command:

NAME The variable name, maximum of 8 characters.

TYPE The variable type, limited to the values: INTEGER, REAL, STRING and FILE.

COUNT The number of elements in the variable or the range of valid COUNTs. A count exceeding 1 creates a multi-valued variable; a count of 0 implies the variable is nullable (i.e., not required or may receive the null value (--)). Default is 1. Max count is 600.

Examples: Possible COUNT declarations.

COUNT=1 must have one value associated with the variable

COUNT=1:10 may have 1 to 10 values entered, which will be called *variable_name*(1)...*variable_name*(10)

COUNT=0:1 may have at most 1 value, but need not be entered (i.e., COUNT=0 is valid)

INITIAL The initial value for the variable, consistent with TYPE, VALID and COUNT.

VALID The allowed values or range of values of the variable, consistent with TYPE.

Examples: Possible VALID declarations.

VALID=4:10 values 4,5,6,7,8,9 or 10 are allowed

```
    VALID=(1:4,8:10)    values 1,2,3,4,8,9 or 10 are
allowed
    VALID=(J,S,U,N)     the listed values are allowed
```

Example: Declaration of Local Variables

```
    Local J TYPE=INTEGER COUNT=0:1 VALID=(5,6,7)
    Local (L,K) REAL COUNT=1 INITIAL=1.5
```

Example: Procedure `COUNTEM` using a Local variable.

```
 Procedure
 Body
 LOCAL  I  INTEGER  INIT=1
 WRITE "Watch me count to 10"
 LOOP
      WRITE "&I"
      LET  I=I+1
      IF (I>10) BREAK
 END-LOOP
 WRITE "How was that?"
 end-proc
```

Example: Running COUNTEM procedure.

```
VICAR>COUNTEM
Watch me count to 10
1
2
3
4
5
6
7
8
9
10
How was that?
```

Local variables are deleted when the operation that created them ends or they are explicitly deleted with the intrinsic commands DELETE or DELETE-LOCALS. The DELETE command will delete a single variable, whereas DELETE-LOCALS will delete all the currently defined local variables.

Syntax:

```
VICAR>DELETE NAME=variable_name
VICAR>DELETE-LOCALS
```

7.1.2.2 Global Variables

A Global variable may be accessed from any level of a session for which it was defined. If accessed in a procedure, a REFGBL statement is required unless it is implicitly defined by VICAR or used at the interactive level. Global variables are defined by PARM commands within special Global procedures (Section 7.3.5) or by the DEFGBL command in regular procedures (Section 7.3.1). A PARM command is similar in structure to the LOCAL command and a DEFGBL command has exactly the same structure. The characteristics defined by PARM commands are outlined in Section 7.2.2.

Example: A Global procedure

```
Global          !proc name is CDEF.PDF
Parm      CAT  STRING  COUNT=1  DEFAULT=GARFIELD +
          VALID=(Garfield, Heathcliff, Sassafras)
End-proc
```

Execution of the above procedure.

```
VICAR>cdef cat=sass
VICAR>display cat
 cat="Sassafras"
```

Example: Use of the DEFGBL command.

```
procedure     ! procedure game
body
defgbl play   type=string   count=1:2 +
         valid=("base","volley","ball")
refgbl play
let play=("base","ball")
write "We like to play &play"
end-proc
```

Execution of the procedure game:

```
VICAR>game
We like to play (base,ball)
```

Like Local variables, Global variables are deleted when the operation that created them ends or they are explicitly deleted with the intrinsic commands DELETE or DELETE-GLOBALS. The DELETE command will delete single Global variables, while the DELETE-GLOBALS will delete all those defined in a Global PDF.

Syntax:

```
VICAR>DELETE NAME=variable_name
VICAR>DELETE-GLOBALS PROC=proc_name
```

Note: Globals may not be of type KEYWORD (Section 7.2.3.2).

7.1.2.3 Assignment

All variables may be assigned values by the use of the LET command. TCL will not allow values which are inconsistent with the defined characteristics. In addition, the null value "--" or the empty string value " " may be assigned, if COUNT includes "0".

Example: Change the value of the Global variable $ECHO.

```
Procedure
Refgbl $ECHO
Body
Let $ECHO="YES"
COPY myfile.dat newfile.dat
End-proc
```

In all cases, if the variable definition allows multiple values, each value may be individually accessed or assigned by appending the element number in parentheses to the variable name. In addition, the element number may be an expression (Section 7.1.3).

Example: Possible assignments for a multi-valued parameter.

```
LET OUTOF(3) = "BOUNDS"
LET OUTOF(P+4) = "LUCK"
```

7.1.2.4 Intrinsic Variables

VICAR maintains implicitly defined local and global variables. In certain cases, the user may even be able to change the value of one of these intrinsic variables.

EXPERT There are four Intrinsic Local variables available to the user. All Intrinsic Locals may be accessed without defining within a procedure; only _STDOUT is not available in batch.

_ONFAIL Multi-valued variable; at most two values. Controls error handling (Section 7.1.5).
_PROC Contains the name of the currently running proc.
_SUBCMD Contains the name of the currently active subcommand.
_STDOUT Multi-valued variable; at most two values.
`_STDOUT(1)` contains the file or device name to which the standard output is directed. This is normally set to "`SYS$OUTPUT`", but may be directed to a file or the print queue (e.g., "`SYS$PRINT`").
`_STDOUT(2)` may contain either "`APPEND`" or "`CREATE`", defining whether to add to an existing file or create a new one. The default is `_STDOUT=("SYS$OUTPUT"," ")`.
EXPERT Intrinsic globals may be accessed without defining on the interactive level, but within a procedure a `REFGBL` command may be required. Descriptions of Intrinsic Global variables are contained in Appendix 10.6.

Example: Access Intrinsic variables `$RUNTYPE` and `_PROC`.

```
Procedure
Refgbl $RUNTYPE
Body
IF ($RUNTYPE="INTERACTIVE") GOTO INT
LOCAL PN STRING
LET PN = _PROC
WRITE "Proc &PN is invoking TESTPROC"
INT>TESTPROC (A,B,C)
End-proc
```

7.1.2.5 Substitution and Dereferencing

EXPERT TCL gives the user the flexibility of creating procedures which can both substitute and dereference variables. Individual values of a multi-valued variable may not be used for substituting or dereferencing.

Substitution

EXPERT Prefixing a variable name with the "&" character tells VICAR to replace the "*&variable_name*" string with the character string that represents the value of the variable. This replacement occurs before command interpretation.

Example: Substitution.

```
LET J = 14
LET I = "PAWS.DAT"
COPY &I F&J SIZE=(100,100,50,50)
```

is interpreted as

```
COPY PAWS.DAT F14 SIZE=(100,100,50,50)
```

BEWARE When substituting variables imbedded in a string or word, the variable name itself may require quotes to distinguish it from the rest of the string. This however, becomes a problem when substitution is to be performed inside an otherwise already quoted string, such as a `WRITE` statement. In this case, concatenation into another variable may be necessary.

Dereferencing

EXPERT Prefixing the variable name with the "@" character tells VICAR to replace the "*@variable_name*" string with the value of the variable. This is like a pointer operation and occurs at run time, not before command interpretation. This is allowed only in value fields (right of the equal sign). The replacing value must be consistent with the definition of the variable.

Example: Dereferencing in a procedure.

```
Procedure! proc name is GPRC.PDF
Parm F INTEGER COUNT=0:1 DEFAULT=--
Body
LIST OUT SPACE=@F
End-proc
```

Execution of the procedure GPRC:

```
VICAR>GPRC F=10     ! acts like: LIST OUT SPACE=10
VICAR>GPRC          ! acts like: LIST OUT
```

EXPERT The "@" is telling VICAR: "when the value of SPACE is needed, use the value of F". A "--" for F implies the same for SPACE, which acts like SPACE was not entered.

EXPERT Differences between Substitution and Dereferencing are detailed in the *TAE Command Language Programmer's Manual*. Generally speaking, though, "@" always works properly for value fields and "&" may not under certain circumstances.

Example: Dereferencing in a procedure.

```
VICAR>GEN OUT=DATASET&"NAME".IMG
```

7.1.3 Expressions

EXPERT TCL allows various kinds of expressions in order to calculate a value from variables, functions and operators. Expressions are only allowed in the commands LET, FOR, IF and ELSE-IF, and are not allowed in proc invocation commands. Expressions using variables of more than one type are not allowed.

EXPERT The various operators available to the user are listed below with the supported expression types.

Numeric Expression

```
+    addition
-    subtraction
*    multiplication
/    division
```

Example: Assignment statement.

```
LET X = Y*2
```

String Expression

```
//   string concatenation
```

Example: String concatenation.

```
LET TRUE = "GARFIELD "//"&verb"//" LASAGNE"
```

Relational Expression

```
>    greater than
<    less than
=    equal to
>=   greater than or equal to
<=   less than or equal to
<>   not equal to
```

Example: A relational expression.

```
    IF (WAIST >= 100) GOTO FAST
```

Logical Expression

```
          AND  logical AND
          OR   logical OR
          NOT  logical negation
```

Example: A conditional statement.

```
    IF (day="Monday" and floor="cold") GOTO SLEEPIN
```

7.1.4 Built-In Functions

EXPERT TCL supplies user-callable functions that can be used in commands and expressions, anywhere a variable name could be used. At run time, the value of the function replaces the function call on the command line.

EXPERT Available functions include:

```
    Function      Returned Value

    $ASFI(jobname)  Returns the current value of $SFI
for the
              asynchronous job.
    $ASKEY(jobname) Returns the current value of $SKEY
for the
              asynchronous job.
    $COUNT(name)    The number of elements for the
variable
              named; -1 if variable has no value, 0 if
              variable has null value (--).
    $FIX(real)        Truncated real value converted
to integer.
    $FLOAT(integer) Integer value converted to real.
```

```
    $GLOBAL(name)   1 if variable is a Global, 0 if
not.
    $PANEL(panelname) Returns the status of existence
for the
              named panel.
    $STRLEN(string) Returns length of specified string.
```

Example: Use of the $COUNT function.

```
    LOCAL I INTEGER COUNT=0:4
    LET I = (1,2,4)
    LOCAL J INTEGER
    LET J = $COUNT(I)  !variable J is given the value 3
```

7.1.5 Error Handling

EXPERT The Intrinsic Global variable $SFI is the indicator of success or failure of a command. It is set to negative upon failures, or remains positive for successes. This variable may be referenced by procs for error handling. Alternatively, the Intrinsic Local variable _ONFAIL can do the error handling for the user. _ONFAIL will check the $SFI indicator and take the action specified by the user. _ONFAIL does not work for errors in intrinsic subcommands.

EXPERT _ONFAIL(1) is set by the user to the command which shall be executed in the event of command failure. _ONFAIL(2) is set by the user to the command which shall be executed upon ABORT after CONTROL-C by the user.

EXPERT The legal values of _ONFAIL are defined below.

```
    "RETURN"    The proc gets terminated and error
indicators
            are passed back up through the chain of
            invoking procs.
```

```
    "BREAK"     Control passes to the first command
outside
           the loop containing the failed command.

    "CONTINUE"      Error has no effect, proc
execution continues.

    "GOTO label"  Control passes to command indicated
by
           "label".

    "NEXT"      Control passes to first command of
current
           loop.

    "STOP"      The proc and its invoking procs
terminate.
```

EXPERT The default value is `_ONFAIL=("RETURN","RETURN")`.

Example: Equivalent `_ONFAIL` statements.

```
LET _ONFAIL="CONTINUE"
LET _ONFAIL(1)="CONTINUE"
```

Example: `_ONFAIL` use in a loop.

```
LET _ONFAIL="GOTO ERROR"
LOOP
 COPY IN/&F X
 MASKV X Y.&F
 NEXT
 ERROR>WRITE "WARNING: FILE &F FAILED"
END-LOOP
```

If any of the steps in the loop fails, a warning message is written and the loop continues to the next iteration.

7.2 VICAR Command Line Structure

The VICAR command is a string containing information that controls VICAR's execution. Executing a command line requires proper syntax. This section will describe the command line structure in detail.

7.2.1 Command Line Syntax

A full command line consists of five terms:

* an intrinsic command or a proc name
* a subcommand
* command qualifiers
* a parameter list
* a comment field

Syntax:

```
VICAR>name[-subcommand] [|qualifiers|] +
VICAR>+ [parameter_list] [!comments]
```

7.2.1.1 Commands and Proc Names

The first component of the command line may be a proc name or one of the VICAR intrinsic commands. See Appendicies 10.3 and 10.4, respectively, for lists of standard VICAR procs and intrinsic commands.

A proc name invokes a stored Proc Definition File (PDF - Section 7.3) from a hierarchical search of the VICAR libraries (Section 5.1).

Intrinsic commands cause the system to perform standard, predefined operations. These allow the user to interact with the VICAR executive rather than with application procs. For instance, the intrinsic command `ENABLE-SCRIPT` instructs VICAR to execute commands stored in a file.

Example: Execute intrinsic command `usage` after running the proc `gen`. (Note: The `usage` command has not yet been ported to UNIX but will be included in future releases. Also, the following example shows what the output will look like for SunOS UNIX; the format will be slightly different for Sun/Solaris and VMS operating systems.)

```
  VICAR>gen rate.img 10 10
  Beginning VICAR task gen
  GEN Version 6
  GEN task completed
  VICAR>usage

  Statistic                Session     Last Proc (gen)
  ---------                -------     ---------
  Proc Start Time                      Mon Sep 12
15:54:02 1994
  Block input operations   5           0
  Block output operations  0           0
  Page Faults(no ph I/O)   24          1
  Page Faults(phys I/O)    30          0
  System CPU Time          00:00:00.20 00:00:00.00
  User CPU Time            00:00:00.41 00:00:00.01
  Connect Time             00:00:16.81 00:00:00.32
```

Intrinsic commands are run by VICAR in the parent process, not the subprocess. It is possible, therefore, to get the two processes confused, as in the following examples.

Example: Confuse the parent process and yourself.

VMS:

```
    VICAR>DCL ASSIGN UD:[USERID] X  !Assign done in
subprocess.
    VICAR>DCL DIR X:*.SCR       !Dir done in
subprocess.

    Directory UD:[USERID]
     RING.SCR;2    1/3  13-JAN-1987   10:00
     Total of 1 file, 1/3 blocks.

    VICAR>ENA-SCR X:RING.SCR    !Ena-scr done in parent
process
                  !which didn't know about assign.
    [TAE-OPNRD] Unable to open script file. VMS/RMS
code
    99524 stv=2312.
```

UNIX: In UNIX, the problem is even worse, since each `ush` command is run in a separate subprocess. (Assume the user is currently in the /home/abc directory.)

```
    VICAR>ush ls /scr/abc
    ring.scr
    VICAR>ush setenv OUTDIR /scr/abc  !Setenv done in a
                         !subprocess.
    VICAR>ush ls $OUTDIR              !Lists current
directory,
    a.img   a.out   b.img   c.img     !since OUTDIR
unknown in
                         !this subprocess.
    VICAR>ena-scr $OUTDIR/ring.scr    !OUTDIR unknown
in the
                         !parent process.
    [TAE-OPNRD] Unable to open script file. No such
file
    or directory.
```

Example: Keeping things clear (VMS example).

```
    VICAR>DCL ASSIGN/JOB UD:[USERID] X    !Assign done
for all
                        !processes.
    VICAR>DCL DIR X:*.SCR        !Dir done in
subprocess.

    Directory UD:[USERID]
     RING.SCR;2    1/3  13-JAN-1987   10:00
     Total of 1 file, 1/3 blocks.

    VICAR>ENA-SCR  X:RING.SCR    !Ena-scr done in parent
process
                   !which now knows about assign.
```

Under UNIX there is no equivalent of assign/job. Since each program is run in a separate sub-process, the environment variable set with `ush` will not effect the next command. The only way to set an environment variable for a VICAR session, is to do it *before* entering VICAR.

7.2.1.2 Subcommands

Some commands and procs may perform several different but related functions, each of which requires a different set of parameters. These functions are called "subcommands". A subcommand is specified as a suffix to the proc or command name, separated with a hyphen and no spaces.

Example: Use of a subcommand.

```
    VICAR>LABEL-DELETE INP=A.DAT KEYS=STRETCH
```

A command may have a default subcommand. Subcommands of intrinsic commands are listed along with the commands in Appendix 10.4. Subcommands of procs can be found by tutoring the proc.

7.2.1.3 Command Qualifiers

A VICAR command may have qualifiers which are VICAR-defined parameters (not program specific) that specify the environment for command execution. Command qualifiers are optional and may appear in any VICAR proc invocation command. If qualifiers are present, they must be listed between vertical bars, "|", immediately after the proc name, and must be supplied in standard parameter list format (Section 7.2.1.4). As in any parameter list, qualifier values may be specified by position (Section 7.2.3.3) or associated with a qualifier name (Section 7.2.3.4). Qualifier names may be abbreviated using the same rules as for parameters (Section 7.2.2).

Example: Run the program F2 in Batch. Note: Batch mode is currently only available on VMS VICAR.

```
VICAR>F2 |RUNTYPE=BATCH| A.DAT B.DAT +
VICAR>+ (1,1,100,100) FUNCTION="2*IN1" 'HALF
```

Appendix 10.5 lists and briefly describes the command qualifiers. For detailed descriptions, see the *TAE Command Language User's Manual*.

7.2.1.4 Parameter List

A parameter list is a list of zero or more (maximum of 600) values, called parameters, (Section 7.2.2), to be provided to a proc or command.

Syntax: Parameter list.

```
parm-1=value-1,parm-2=value-2,...,parm-n=value-n

 where:   parm-1        is the first parameter name
      value-1       is a value to be assigned to the
                 first parameter
      etc.
```

The parameters within a parameter list may be separated by a comma and/or any number of spaces.

7.2.1.5 Comments

Comments are introduced by the character "!" and may appear anywhere on a line. Anything that appears after "!" is a comment. Comments are ignored by VICAR. Use the comment character to insert a blank line in a series of proc statements or to document a proc.

Examples: The use of comments.

```
VICAR>!
VICAR>! Test filtering procedure
VICAR>GEN T.DAT 100 100 +    !Generate test image
VICAR>+ linc=1 sinc=2
```

BEWARE *If the line continuation character, "+", is used, it must precede the comment character, "!".*

7.2.2 Parameters

Proc parameters are a form of data used to control the specific way a proc operates. Each parameter's attributes are defined in its Proc Definition File (PDF - Section 7.3) by `PARM` commands. Parameter attributes specify how the parameter value is handled when the proc is invoked. The characteristics specified by the definition are:

* name
* position on the command line
* type
* count (minimum and/or maximum number of values)
* default value
* valid values or range of the values
* if the parameter is a dataset parameter, whether the dataset is an input, output or input/output dataset
* name of the proc where the parameter qualifiers are defined

A parameter that has no default value is a mandatory parameter.

The number and type of parameters used on the command line are proc-dependent. They may be separated by blanks or commas. Parameter names and values may be abbreviated provided the parameter can be unambiguously identified among all the parameters for the proc.

Parameters may be one of five types:

INTEGER or REAL - A numeric integer or decimal value consisting of the characters "+", "-", ".", "0" through "9", and/or "e" (the power indicator). Spaces are not permitted within a numeric value. The null value "--" is allowed. This type is only for values being passed into procs.

STRING - A character string of maximum length 250. If any characters listed below appear in the string, the string must be surrounded by double quotes, " ". The empty string " " and the null value "--" are allowed. This type is only for values being passed into procs.

KEYWORD - A character string of maximum length 8. None of the characters listed below are allowed For more information on keywords, see Section 7.2.3.2. This type is only for values being passed into procs.

NAME - A character string which is the name of a declared TCL variable. Values output by procs must be of this type.

The characters recognized by TCL which are the subject of the above STRING and KEYWORD restrictions include:

```
space     !     ,     )     (     +
'     >     <     |     @     =
tab
```

7.2.3 Parameter Value Specification

A parameter may be assigned a value in several ways. The values may be defaulted, restored from previously saved values or explicitly specified.

Parameters with defined default values will automatically take those values if they are not given values any other way.

A set of parameters may be saved and later restored, in the same or a later session, using the SAVE and RESTORE command qualifiers or the SAVE and RESTORE Tutor commands.

Example: Run the program `TFILT` with its saved parameters.

```
VICAR>TFILT |RESTORE=TFILT.PAR|
```

Parameter values may be assigned explicitly in any of three formats:

* *parameter_name=value*
* '*value* (if *parameter* is of type `KEYWORD`)
* position

Parameters defined to have multiple values may be explicitly specified by parenthesized lists using commas or blanks as separators.

Example: Specification of multi-valued parameters.

```
VICAR>F2 inp=(a,b,c) out=d.dat size=(1,1,100,100) +
VICAR>+ func="2*IN1"
```

BEWARE *The values within multi-valued parameters are position-dependent. VICAR procs expect the sequence to be that defined in the PDF for the proc.*

7.2.3.1 *Parameter_name=value* Format

Parameters may be in any order when specified this way. If no value is given, then the default value is assigned to the parameter.

Example: A command line in *parameter_name=value* format.

```
VICAR>F2 inp=a.dat out=b.dat size=(1,1,100,100) +
VICAR>+ format=half function="2*IN1"
```

7.2.3.2 Keyword Format

Keyword format is a special case of *parameter_name=value* format for KEYWORD type parameters only. All of the above rules still apply, but a single quote, ', replaces the *parameter_name*= part of the specification. The ' is necessary to distinguish this format from positional format.

Example: Specification of keywords.

```
VICAR>F2 inp=a.dat out=b.dat size=(1,1,100,100) +
VICAR>+ 'half function="2*IN1"
```

7.2.3.3 Positional Format

The user should be aware that the order in which the proc receives the parameters is not necessarily that in which they were specified. The Proc Definition File (PDF) for a specified proc (Section 7.3) defines the order of the parameters by the sequence of the PARM statements in the PDF. VICAR allows the *parameter_name*= part of the parameter specification to be omitted if the parameters are specified on the command line in exactly the same order as they occur in the PDF. This order is the same as the order in which the parameters appear in the Tutor display.

Because VICAR procs generally have many parameters, whose order would be difficult to remember, a convention has been adopted requiring that all procs have the same three initial

parameters (except for programs such as GEN, AL and LIST). Their order in the program PDF is:

```
INP  a defined number of strings specifying input
     file or input device names in the PDF

OUT  a defined number of strings specifying output
     files or output device names in the PDF

SIZE      four integers specifying:

     starting line
     starting sample
     number of lines
     number of samples
```

Example: Standard convention for INP, OUT, SIZE.

```
VICAR>F2 a.dat b.dat (1,1,100,100) format=half +
VICAR>+function="2*IN1"
```

BEWARE *Once any parameter value is specified using an explicit parameter name, all remaining parameters on the command line must be specified using explicit parameter names. In a positional parameter list, two successive commas indicate that the corresponding parameter's default value is to be used. There cannot be two successive commas after a value has been provided using an explicit parameter name.*

Example: Correct use of explicit parameter names.

```
VICAR>F2 inp=a.dat out=b.dat size=(1,1,100,100) +
VICAR>+format=half function="2*IN1"
```

not

```
    VICAR>F2 inp=a.dat b.dat (1,1,100,100) format=half
+
    VICAR>+function="2*IN1"
```

7.2.3.4 Parameter Qualifiers

Parameter qualifiers are like sub-parameters (parameters to the parameters). Upon proc invocation, the values of qualifiers to a parameter are specified after the parameter value and set between vertical bars, "|".

Syntax:

```
parameter=value|qual1=val1,qual2=val2,...qualn=valn|
```

Example: Specification of parameter qualifiers.

```
    VICAR>IMGCOPY
from=file1|bands=(n1,n2),format="BSQ"| +
    VICAR>+ to=file2
```

The rules for specifying parameter qualifiers are identical to the rules for parameters. Therefore, parameter qualifier values may be entered by position.

Example: Specification of positional parameter qualifiers.

```
VICAR>IMGCOPY file1|(n1,n2), "BSQ"| file2
```

7.2.4 Missing or Invalid Parameter Values

Parameter errors occur when a proc is invoked in the following cases.

* Required parameters (those with no defined default) were not supplied
* The specified values were not consistent with the parameter definition
* An undefined (or misspelled) parameter name was entered

VICAR issues an error message in these cases and reprompts the user.

Example: Sample parameter error message.

```
VICAR>LIST CLIFF.ROT (100,100,10,10) 'SPAXE
[TAE-KEYWORD] Undefined keyword "SPAXE".
Re-enter the command line or type TUTOR to recover
specified values.
```

The user may respond with "Tutor" and enter Tutor mode to respecify the missing parameters or obtain help on the parameter in question (Section 6.5).

7.2.5 Dynamic Parameters

An executing proc may prompt the user for parameter entry. These parameters are called Dynamic parameters. Programs generate such requests with special subroutines described in the *VICAR*

Run-time Library Reference Manual and the *TAE Command Language Programmer's Manual*. Procedures, both Asynchronous and Synchronous may also request parameters through the use of the `GETPAR` command.

Example: Dynamic parameter use in the procedure `CON`.

```
Procedure ! procedure CON
Local  CONST  INTEGER
Body
GETPAR CONST
GEN A.DAT 10 10 IVAL=&CONST LINC=0 SINC=0
WRITE "Created A.DAT with all DNs = &CONST"
End-proc
```

Example: Execution of the procedure CON.

```
VICAR>CON
ENTER CONST
->12
Beginning VICAR task GEN
GEN Version 6
GEN task completed
Created A.DAT with all DNs = 12
```

7.3 Proc Definition Files (PDFs)

As discussed in Section 4.5, a proc may be either a procedure or a process. PDFs are text files defining parameters for the proc and have a `.pdf` file extension. For procedures, the PDF also contains VICAR commands to be executed. For processes, the PDF has an executable program file associated with it to be activated. In this section, the user will become familiar with procedure definition

files, process definition files, Global PDFs, the execution of PDFs, help files for PDFs, and compiled PDFs.

BEWARE *On UNIX systems, proc names must be in lower case letters. Procs may be called using either lower case or upper case characters, but the file itself must have a lower case name, or it will not be found by the VICAR executive.*

7.3.1 Procedure Definition Files (PDFs)

A VICAR Procedure Definition File is the method that most users will select to join parameter definitions and a series of VICAR commands together and selectively control their execution. A procedure may be simple or complex and yet all users, even the most novice user, will be able to write one.

The procedure PDF is comprised of two main parts: the declaration section and the body. The declaration section contains the parameter declarations, the Local variable declarations and the Global variable references. The body contains the VICAR commands to be executed.

Syntax: Basic format of a procedure PDF.

```
    Procedure   [HELP=help_information_location]
    Defcmd  !
    Refgbl  !This is the declaration section. All
entries
    Parm    !within this section are optional and they
can
    .        !appear in any order so long as they
appear
    .        !between the PROCEDURE and BODY commands.
    Body
```

```
    .         !This is the body of the PDF.  VICAR
commands
    .         !should be entered in the order they will
be
    .         !executed.
    End-proc
```

Note: PROCEDURE, BODY and END-PROC are all required elements of a procedure.

The easiest method to learn how to write procedure PDFs is to look at examples. The following example is a very simple PDF which creates a VICAR label for a dataset, copies that dataset, then performs a linear stretch on the copy. Note that within this PDF there is no declaration section. The declaration section is not mandatory and is omitted in this case.

Example: A procedure to label and stretch an image.

```
     Procedure               !1
     Body                    !2
     label-create a.raw a.img nl=100 ns=200    !3
     copy a.img b.img             !4
     stretch b.img b.str linear=(54,199)  !5
     End-proc                !6
```

Line 1 defines that this proc is a procedure.
Line 2 defines that the body of the procedure follows this
line.
Line 3 creates a label for a raw image with 100 lines and 200
samples.
Line 4 executes the VICAR program COPY, creating a second,
identical image.
Line 5 executes the VICAR program STRETCH, performing a linear

```
        stretch from 54 to 199 on the image.
Line 6   defines that the end of the proc has been
reached.
```

Additional sample PDFs can be found in Appendix 10.11.

7.3.2 Process Definition Files (PDFs)

A process is a program of executable code and all programs require a process PDF. A process PDF may specify:

* the process parameters
* Local variable declarations
* the Globals referenced by the process
* the process subcommands
* the location of the proc's executable program
* the location of the proc's help text

Syntax: A process PDF, conforming to recommended standards.

```
     Process [EXECUTE=executable_file
HELP=help_location]
     Parm NAME1  description1
     Parm NAME2  description2
          .
          .
          .
     End-proc
```

Example: Process PDF for VICAR program `COPY`.

```
    process help=*
    PARM INP  TYPE=STRING
    PARM OUT  TYPE=STRING
```

```
    PARM SIZE TYPE=INTEGER COUNT=0:4 DEFAULT=--
    PARM BANDS TYPE=INTEGER COUNT=0:2 DEFAULT=--
    PARM SL INTEGER DEFAULT=1
    PARM SS INTEGER DEFAULT=1
    PARM SB INTEGER DEFAULT=1
    PARM NL INTEGER DEFAULT=0
    PARM NS INTEGER DEFAULT=0
    PARM NB INTEGER DEFAULT=0
    PARM FORMAT KEYWORD
VALID=(BYTE,HALF,FULL,REAL,DOUB,COMP) +
          DEFAULT=BYTE
    PARM INSIZE INTEGER COUNT=0:3 DEFAULT=--
    PARM ORG KEYWORD VALID=(BSQ,BIL,BIP) DEFAULT=BSQ
    PARM BLOCK TYPE=KEYWORD VALID=(BLOCK,NOBLOCK)
DEFAULT=BLOCK
    PARM BINARY TYPE=KEYWORD VALID=(BINARY,NOBINARY) +
          DEFAULT=NOBINARY
    END-PROC
    .help
    Copy is a simple program which can be used to copy
all or
    .
     (See Section 7.3.4 for further
      information on Help files.)
    .
```

Additional sample PDFs can be found in Appendix 10.11.

7.3.3 Executing a PDF

If the user is new, or if the PDF is trying something new or tricky, the user is advised to run the PDF through the syntax checker (Section 6.4) before actually executing the proc on the host system.

Once a PDF has successfully passed through the syntax checker, the user is ready to execute the proc. The user must next select the mode in which to execute the PDF (Interactive, Asynchronous or Batch). The use of the latter two modes has already been explained in Section 5.2. If the Interactive mode has been chosen, all the user needs to do is enter the proc name and any necessary parameters.

Example: Equivalent methods of proc invocation.

```
VICAR>TAPE2DSK
```

or

```
VICAR>TAPE2DSK.PDF
```

Whenever a user enters a command, a hierarchical library search (Section 5.1) is initiated for a `.PDF` file of that name. If the PDF is not located in any of the libraries, the user will be notified that an illegal command has been entered.

7.3.4 Help Files

All procs which conform to the recommended standards have a Help file associated with them. Help files contain the information that the user will see when Tutor or Help information is requested on a specific proc.

EXPERT The typical user will probably not bother to write the Help file portion of a procedure PDF since the user designed the PDF and will not require any assistance with the proc. However, there are cases when a Help file is useful. For example, when a procedure is very complicated, or when more than one person will be using a proc (as in the case of the application programs).

EXPERT Help information on VICAR procs can be written in one of two ways: as a separate text file with the name `PROCNAME.HLP`, or as text appended to the end of the PDF. In the second case, the

PDF must include on the PROCEDURE or PROCESS command line `HELP=*`. In either case, the format of the help information is the same.

Syntax:

```
    .TITLE         Title to appear on each Tutor and
Help
               screen.

    .HELP      Information on the proc itself,
               typically including:
     Purpose        Purpose of the program.
     Execution Description of command line used to
               execute the program.
     Features  Special features the user might like to
               know about.
     Examples  Examples showing how the program can be
               used.
     Restrictions  Any special restriction the user
should
               be aware of.
     Written by    Programmer who originally wrote the
               program.
     Programmer    Programmer currently responsible for
the
               program.
     Revision  Version number and date.

    .LEVEL1        Header to tell VICAR that first
level
               help on the proc's variables follows.
    .VARIABLE name1    Help information on name1 to be
               displayed on the Tutor screen.
    .VARIABLE name2    Help information on name2 to be
               displayed on the Tutor screen.

    .LEVEL2        Header to tell VICAR that second
level
               or detailed help on the variables
```

```
                follows.
    .VARIABLE name1    Detailed information on name1.
This text
                is displayed on the "Help PARM" Tutor
                command or on the VICAR command:
                  HELP-PARM parm=name1 proc=proc_name
    .VARIABLE name2    Detailed information on name2.

    .END       End of help text.
```

See Appendix 10.11 for examples of Help files.

EXPERT It is possible to get hardcopy printouts of the proc's Help section. For formatted Help, use the `HELP-HARDCOPY` command to create a disk file which may then be printed. For unformatted Help, print the proc's `.PDF` file.

7.3.5 Global PDFs

EXPERT A Global proc defines Global variables. It may contain references to other Globals and/or declarations of Local variables. The Global PDF starts with "Global" and has no body.

Syntax:

```
Global [HELP=help_information_location]
definitions of Global variables
declarations of referenced Globals
declarations of Local variables
End-proc
```

Example: Sample Global PDF.

```
Global
```

```
Parm DATATYPE    String Default=" "
Parm RUNSEQ        Integer   Default=0
Parm COLTBL        String    Default="default"
Parm SL       Integer   Default=1  (null value)
Parm SS       Integer   Default=1
Parm NL       Integer   Default=1340
Parm NS       Integer   Default=3360
Parm RES      Real  Default=8.
Parm REGION        String    Default="global"
End-proc
```

EXPERT Note: Globals may not be of type KEYWORD (Section 7.2.3.2).

7.3.6 Compiled PDFs

WIZARD A compiled PDF is created when the VICAR command COMPILE is run on any process, procedure or Global PDF. During compilation, the declaration statements are scanned for proper syntax and symbol tables are generated. Because these steps do not have to be repeated during the PDF execution, compiled PDFs (with a .CPD extension) run faster than normal PDFs (.PDF).

WIZARD In order to compile a PDF, the following rules must be observed when writing the proc:

* internal procs are not allowed prior to the body
* DEFGBL statements are not allowed
* the RUNTYPE and NAME fields of PROCESS and PROCEDURE statements must be default
* the DEFAULT field of a parameter statement cannot be dereferenced
* parameter qualifiers may not be declared

WIZARD Procs are compiled by using the COMPILE command.

Syntax:

```
    VICAR>COMPILE INPROC=input_pdf
[OUTPROC=compiled_pdf]
```

Example: Compile the proc `TAXCALC.PDF`.

```
    VICAR>COMPILE inproc=TAXCALC.PDF
outproc=TAXCALC.CPD
```

WIZARD The execution of a compiled PDF is identical to the execution of a normal PDF. If the PDF resides within one of the VICAR application libraries,VICAR automatically looks to see if a .CPD version of the requested proc exists. If no compiled proc is available then the .PDF is executed.

Example: Execute a program which has a compiled PDF.

```
    VICAR>BROWSE
```

WIZARD In this example, because the compiled PDF exists, VICAR automatically executes `BROWSE.CPD`. However, if the compiled PDF resides within a personal directory, the compiled PDF must be explicitly specified unless the library has been customized to select compiled PDFs first.

Example: Explicitly invoke the compiled version of TAXCALC.

```
    VICAR>TAXCALC.CPD (W2.DAT,INTEREST.DAT) 1040.DONE
```

WIZARD Additional information on compiled PDFs can be found in the *TAE Command Language User's Manual* and the *TAE Command Language Programmer's Manual*.

7.4 Script Files

EXPERT A Script file is a text file consisting of VICAR commands used primarily for demonstration and test purposes. None of the special commands required in a procedure (e.g., BODY) are included in a script file. On the other hand, no parameters are allowed either.

Example: Script file LOG.SCR

```
F2 TEST.RED    OUTTEST   FUNC="100.*ALOG10(IN1)"
STRETCH OUTTEST STRTEST LINE=(0,150)
```

EXPERT A SCRIPT file can be invoked with the intrinsic command ENABLE-SCRIPT.

Syntax:

```
    VICAR>ENABLE-SCRIPT FILE=filename [COUNT=number]

    Where:  FILE    is the name of the SCRIPT file to
be
             executed.  The file must be of the type
             .SCR in order to be properly executed.
         COUNT   The number of times to execute the
script
             file.  The default is to execute the file
             once.  COUNT=ZERO causes the file to be
```

```
repeated endlessly until aborted by a host
interrupt sequence.
```

EXPERT The ENABLE-SCRIPT command causes VICAR to use the specified script file for its "interactive" input. There is no hierarchy search implemented for script files so the user should either be in the directory containing the script file prior to entering VICAR, or the user should fully specify the directory and filename in the invocation. In addition, the datasets used by the script file must be in the same directory as the script file or the directory should be fully specified. Once invoked, TAE then sequences through the script file, echoing each command as it is executed.

Example: Script file use in an interactive session.

```
VICAR>ENABLE-SCRIPT LOG.SCR
F2 TEST.RED OUTTEST FUNC="100.*ALOG10(IN1)"
Beginning VICAR task F2
F2 version 2-04-94
F2 using byte table lookup
FUNCTION EVALUATED 256 TIMES

STRETCH OUTTEST STRTEST LINE=(0,150)
Beginning VICAR task STRETCH
STRETCH version 2-11-93

*** LINEAR CONTRAST STRETCH MODE ***
Linear Stretch:      0 to      0 and    150 to
255
```

7.5 Tape Handling

The following section covers tape handling for VICAR under VMS only. There is no VICAR tape support under UNIX. It is suggested that the standard operating system utilities such as tar and dd be used for accessing tapes. VMS VICAR support of tape

handling may not continue in the future, and many applications which have accepted tape inputs in the past will no longer do so.

When running under VMS, operations involving magnetic tapes are handled within VICAR with the commands ALLOC, MOUNT, REWIND, DISMOUNT and DEALLOC (see Section 10.4). Both foreign and ANSI labelled tapes are supported, as are blocked tapes (those with multiple logical records per physical record). All applications which accept tape inputs require that the tapes be previously MOUNTed.

The ALLOC command assigns a symbolic name to the allocated device. All other tape commands use that symbolic name to access the device. In addition, the MOUNT command uses many of the same qualifier names as the DCL MOUNT command. However, ANSI-labelled tapes are identified by the presence of the LABEL parameter.

The syntax for referencing a particular file on a tape uses the "/" character. The symbolic name of the tape is suffixed by a "/*file_number*" string.

Example: Accessing file 10 on tape IN.

```
VICAR>COPY IN/10 X
```

It is not necessary to rewind the tape for VICAR to access the proper file number. VICAR always keeps track of the current file number and handles file number regressions properly.

Example: Tape processing under VMS.

```
    VICAR>ALLOC MT: IN      !allocates first available
                  !MT: drive and assigns
                  !symbolic name IN
    VICAR>MOUNT IN +        !tape mytape is mounted
foreign,
    VICAR>+ comm="mount mytape"  !unblocked and for read
    VICAR>COPY IN/4 A       !copy file 4 into A
    VICAR>COPY IN/2 B       !copy file 2 into B
    VICAR>REWIND IN         !tape is rewound
    VICAR>DISMOUNT IN       !tape is dismounted
    VICAR>DEALLOC IN        !device is deallocated
```

BEWARE *Device allocations may be performed at any time in a VICAR job or session. However, in practice, jobs should do all allocations at the beginning of the job stream, and do them as a group. This is because, in general, the job is started when the resources become available, but there is no guarantee that they will stay available.*

Users may exit VICAR and reenter with confidence that any tape drives will remain allocated, mounted and at the same position unless a Control-Y was used to exit. Exiting by Control-Y will lose dummy names associated with devices, asynchronous jobs and some allocated devices. Tape drives that had been allocated and mounted will remain mounted, but VICAR will know nothing about them.

Tape drives allocated within VICAR may be accessed from VICAR's DCL mode. Such access must, of course, be accomplished with DCL tape handling commands.

BEWARE *Tape drives allocated within VICAR are not available to the user outside VICAR (and vice versa). This is because the VMS process and the VICAR may not share resources.*

The general tape handling utility, TAPES, is not a standard VICAR proc and may not be invoked from VICAR. It may be invoked from VICAR DCL mode or from the user's VMS process. Because of the non-standard nature of TAPES, no VICAR user aids are available for it. However, there is on-line help for TAPES in DCL.

Syntax:

```
$ HELP TAPES
```

Hardcopy help for TAPES is obtainable using the following sequence of DCL commands.

Syntax:

```
$ HELP/OUT=TAPES.HLP TAPES
$ HELP/OUT=TAPES1.HLP TAPES*
$ PRINT TAPES.HLP, TAPES1.HLP
```

Note: "/OUT" must immediately follow HELP and not TAPES or TAPES*.

7.6 Session Logging

VICAR gives the user the ability to create a log of their interactive session activity. This option is enabled and disabled with the ENABLE-LOG and DISABLE-LOG commands. When enabled, this option creates or appends to two files called session.log and session.tsl. The .log file reflects the commands and responses as they appear on the standard output, while the .tsl file supplies additional useful information (e.g., a full list of proc parameters).

A new pair of session files is created at the initial ENABLE-LOG command after entry into VICAR. Further ENABLE-LOGs (after DISABLE-LOGs, of course) within a given session will append information to the existing files. Once the DISABLE-LOG option is selected, the files are closed and may be accessed, reopened, or printed.

At the current time, there are still some problems with session logging on the UNIX system. In batch log files, items often appear out of order, and FORTRAN programs do not appear at all. In interactive sessions, the session.log file is usually correct, but the session.tsl file can occasionally be out of order.

7.7 Sample Interactive Session

The following example shows a Synchronous or Interactive VICAR session. The commands are all entered from Command mode and perform several basic tasks. The reader may follow this session exactly or embellish it as desired for practice.

```
$ VICAR


          Welcome to VICAR

                System Release 12.1
                Executive Version 5B/16B


  --- Type NUT for the New User Tutorial ---

  --- Type MENU for a menu of available applications --
-
```

```
VICAR>gen a 10 10 ival=2 sinc=0 linc=0
Beginning VICAR task gen
GEN Version 6
GEN task completed
VICAR>list a
Beginning VICAR task list

   BYTE     samples are interpreted as   BYTE   data
 Task:GEN   User:edd   Date_Time:Thu Aug 11 15:05:44
1994
     Samp     1       3       5       7       9
   Line
      1       2   2   2   2   2   2   2   2   2   2
      2       2   2   2   2   2   2   2   2   2   2
      3       2   2   2   2   2   2   2   2   2   2
      4       2   2   2   2   2   2   2   2   2   2
      5       2   2   2   2   2   2   2   2   2   2
      6       2   2   2   2   2   2   2   2   2   2
      7       2   2   2   2   2   2   2   2   2   2
      8       2   2   2   2   2   2   2   2   2   2
      9       2   2   2   2   2   2   2   2   2   2
     10       2   2   2   2   2   2   2   2   2   2
VICAR>f2 a b.dat func="in1+10"
Beginning VICAR task f2
Beginning VICAR task f2
F2 version 2-04-94
F2 using byte table lookup
FUNCTION EVALUATED 256 TIMES
VICAR>list b.dat
Beginning VICAR task  LIST

   BYTE     samples are interpreted as   BYTE   data
 Task:GEN   User:edd   Date_Time:Thu Aug 11 15:05:44
1994
 Task:F2    User:edd   Date_Time:Thu Aug 11 15:06:16
1994
     Samp     1       3       5       7       9
   Line
      1      12  12  12  12  12  12  12  12  12  12
      2      12  12  12  12  12  12  12  12  12  12
      3      12  12  12  12  12  12  12  12  12  12
      4      12  12  12  12  12  12  12  12  12  12
      5      12  12  12  12  12  12  12  12  12  12
      6      12  12  12  12  12  12  12  12  12  12
```

```
      7       12  12  12  12  12  12  12  12  12  12
      8       12  12  12  12  12  12  12  12  12  12
      9       12  12  12  12  12  12  12  12  12  12
     10       12  12  12  12  12  12  12  12  12  12
VICAR>gen b 10 10 ival=4 sinc=0 linc=0
Beginning VICAR task gen
GEN Version 6
GEN task completed
VICAR>list b
Beginning VICAR task list

   BYTE     samples are interpreted as   BYTE   data
 Task:GEN   User:edd   Date_Time:Thu Aug 11 15:12:38
1994
     Samp     1       3       5       7       9
   Line
      1       4   4   4   4   4   4   4   4   4   4
      2       4   4   4   4   4   4   4   4   4   4
      3       4   4   4   4   4   4   4   4   4   4
      4       4   4   4   4   4   4   4   4   4   4
      5       4   4   4   4   4   4   4   4   4   4
      6       4   4   4   4   4   4   4   4   4   4
      7       4   4   4   4   4   4   4   4   4   4
      8       4   4   4   4   4   4   4   4   4   4
      9       4   4   4   4   4   4   4   4   4   4
     10       4   4   4   4   4   4   4   4   4   4
VICAR>f2 (a,b) b.dat func="in1*in2+10" 'half
Beginning VICAR task f2
Beginning VICAR task f2
F2 version 2-04-94
F2 using hash table lookup
FUNCTION EVALUATED 2 TIMES
VICAR>list b.dat (1,1,10,10)
Beginning VICAR task list

   HALF     samples are interpreted as HALFWORD data
 Task:GEN   User:edd   Date_Time:Thu Aug 11 15:05:44
1994
 Task:F2    User:edd   Date_Time:Thu Aug 11 15:23:45
1994
   Samp   1    2    3    4    5    6    7    8    9
10
   Line
```

```
     1    18   18   18   18   18   18   18   18   18
18
     2    18   18   18   18   18   18   18   18   18
18
     3    18   18   18   18   18   18   18   18   18
18
     4    18   18   18   18   18   18   18   18   18
18
     5    18   18   18   18   18   18   18   18   18
18
     6    18   18   18   18   18   18   18   18   18
18
     7    18   18   18   18   18   18   18   18   18
18
     8    18   18   18   18   18   18   18   18   18
18
     9    18   18   18   18   18   18   18   18   18
18
    10    18   18   18   18   18   18   18   18   18
18
VICAR>copy a c
Beginning VICAR task copy
 COPY VERSION 12-JUL-1993
VICAR>list c
Beginning VICAR task list

   BYTE     samples are interpreted as   BYTE   data
 Task:GEN    User:edd    Date_Time:Thu Aug 11 15:05:44
1994
 Task:COPY   User:edd    Date_Time:Thu Aug 11 15:27:50
1994
     Samp     1        3        5        7        9
   Line
      1       2   2   2   2   2   2   2   2   2   2
      2       2   2   2   2   2   2   2   2   2   2
      3       2   2   2   2   2   2   2   2   2   2
      4       2   2   2   2   2   2   2   2   2   2
      5       2   2   2   2   2   2   2   2   2   2
      6       2   2   2   2   2   2   2   2   2   2
      7       2   2   2   2   2   2   2   2   2   2
      8       2   2   2   2   2   2   2   2   2   2
      9       2   2   2   2   2   2   2   2   2   2
     10       2   2   2   2   2   2   2   2   2   2
VICAR>exit
```

%

8 Non-Standard Items

Contents

The VICAR software set has been designed with transportability in mind, and the bulk of the software is truly transportable to other machines, including those running UNIX. However, in order to benefit by and utilize certain hardware and proprietary software packages apvailable at the MIPS site, a subset of VICAR is facility-specific (some is, in fact, machine-specific) and thus non-standard. This subset will not work unless this hardware or software is resident (see the system manager for configuration details).

Transportation of data to or from a VICAR facility is supported in either VICAR or non-VICAR format. A set of application programs are available which convert from VICAR to specific non-VICAR formats (e.g., Voyager EDR format), and from specific non-VICAR formats (e.g., FITS format) to VICAR format. The purpose of this section is to describe these subsets of VICAR, the non-standard items.

8.1 Proprietary Software

Proprietary software packages utilized by VICAR applications procs are:

* the VAX Datatrieve data management system from Digital Equipment Corporation
* the Floating Point Systems (FPS) array processor library
* the PRINTRONIX plotting routines from Cerritos Computer Systems Incorporated
* The Sybase data management system from Sybase for UNIX
* the Image Display Manager (IDM)

Specific procs which call or link to this proprietary software are listed below and are described in more detail in Appendix 10.3. Asterisks,"*", indicate that the proc will utilize proprietary software if it is available, but, if not, will use less efficient but non-proprietary software. Pluses, "+", indicate that the proc will utilize proprietary software only in some of its operating modes. If the proc name has no asterisk or plus, it will only be available on systems with the specified software.

Procs which call Datatrieve:

```
AMOS      BADLABELS BROWSE+       CATARCH
CATCD          CATFILM       CATIMAGE  CATLABEL
CATLIST   CATMRPSO  CATNIMS   CATPRODUCTS
CATRADIO  CATREPORT      CATSEARCH      CATSEDR+
CATSLIST  CATSPICE  CDGEN          DTRTAE
EDRGEN+        EDRVAL+       GALSOS         GEDREAD+
GLLMASK        GLLREQUEST    GPWAUDIO  LISTSEDR
LISTSEDX  MAP2           MERGE          NIMSCMM
NIMSMERGE      NIMSMERGEX    PPLOAD         PWAUDIO
PWSMERGE  SEDRGEN   SRCH           SRCHGEN
SSIMERGE  STOREKDB  UVF        VEDR+
VGRMASK   VISIS
```

Procs that link to the FPS array processor library:

```
APFLAG         BROWSE+   EIGEN          FFT1*
FFT2*          FILTER*        MGNCORR*  PICMATCH*
R         RCBR*          RESTORW*  TFILT*
```

```
WIENER          XFORM*
```

Procs that link to IDM:

```
ANNOT           EXPER           MIDR_ARCH RDM
RDMINIT         VIEW
```

Procs which link to Sybase:

```
GLLTELEMPROC
```

Procs which link to the CCSI routines:

```
STATPLOT
```

8.2 Facility-specific Hardware

Peripherals at the MIPS site include display processors, film recorders, videodisk, array processor, terminals, personal computers and plotters. Procs which are dependent upon specific hardware are listed below and are described in more detail in Appendix 10.3. Asterisks,"*", indicate that the proc will utilize proprietary hardware if it is available, but, if not, will use less efficient but non-proprietary hardware. If the proc name has no asterisk, it will be available only on systems with the specified hardware.

```
Hardware Required  Procs

Array Processor - VAX only
```

Floating Point Systems AP120B

APFLAG, FFT1*, FFT2*, FILTER*, FILTER2*, PICMATCH, R, TFILT*, WIENER, XFORM*

Display Processors - VAX only

Adage 3000, DeAnza 8500, I2S IVAS and Ramtek RM-9465

AMOS, BROWSE, CCDNOISE*, CCDRECIP*, CCDSLOPE*, DISPOUT, IDX, IFFT, IMP, INTERLOC*, LEONARDO, LOGMOS, MGNFIT*, NAV, NAV2, OTF1*, PHOTFIT2*, PICREG, PLOT3D*, PLOTINT*, PLOTTING*, PLTGRAF*, PLTSYM*, POWER*, QD, QPLOT*, RG*, SARGON, SPAM, STERGEN,TIEPLOT*, UVFMAP*

DeAnza 8500 only

EDIMAGE, FOOTPRINT

Film Recorders

Matrix Quality Color Recorder, MDA Colorfire, MDA Laserfire, KODAK XL7700

BARNE, PHOTO, WILMA

Printers/Plotters - VAX only

Calcomp 1044, Printronix P300 and P600, Regis- and Tektronix-compatible terminals

CCDNOISE*, CCDRECIP*, CCDSLOPE*, LEONARDO, MGNFIT*, OTF1*, PHOTFIT2*, PLOT3D*, PLOTINT*, PLOTTING*, PLTGRAF*, PLTSYM*, POWER*, QPLOT*, RG*, TIEPLOT*, UVFMAP*

Printronix P300 and P600

STATPLOT

PWS Audio Generator (a JPL-built device) - VAX only

GPWAUDIO, PWAUDIO, PWECHO

8.3 VICAR I/O Formats

Transportation of data to or from a VICAR facility is performed using tape media, cd-rom or network transfer. VMS VICAR supports tapes with and without ANSI labels. The tape densities that are supported are 800, 1600 and 6250 on 9 track tape. 8mm and 4mm data tapes are also used. The data transfer is accomplished with applications procs. The tape handling commands are ALLOC, DEALLOC, MOUNT, DISMOUNT and REWIND. For UNIX, a simple tar command will transfer data to either 8mm or 4mm data tapes. The network can be used to transfer data as long as the VICAR site has an anonymous FTP.

8.3.1 Input/Output of VICAR Datasets

VICAR is able to output data in a format that most other sites can handle. Under VMS, the capability exists to output datasets with or without VICAR labels, either blocked (n logical records per physical record) or unblocked (1 logical record per physical record). The procs LABEL-REMOVE or COPY are generally used for this purpose.

8.3.2 Output of Non-VICAR Datasets

Using the VICAR I/O subroutines (see the VICAR Run-time Library Reference Manual), it is possible to generate output datasets in any format desired. The programs which currently do

this and their output formats are described below. For program descriptions, see Appendix 10.3.

```
PROGRAM       FORMAT

BIDRSIM       Multimission SAR Processing Lab Basic
Image Data
         Record
EDRGEN        Voyager Experiment Data Record
PSCRIPT       Converts VICAR images into a Postscript
file
TAPES         optionally, EBCDIC
VCOPOUT       ASCII Text
VMAC     Transfers data between PICT or TIFF format
         images and IMG files
VTIFF         Convert between VICAR labeled images and
TIFF
         format files
WRISOUT       U.S. Forest Service WRIS exchange format
```

8.3.3 Input of Non-VICAR Datasets

Unlabeled tape datasets, blocked or unblocked, can be converted to VICAR format using the proc CONVIM or LABEL-CREATE. However, special purpose programs, called logging programs, are often written to extract special information from the data, construct special VICAR labels or unpack the data in unique ways. The programs which currently do this and their input formats are listed below. For program descriptions, see Appendix 10.3.

```
PROGRAM       FORMAT

AFEWCLOG Air Force Electronic Warfare Center terrain
data
AVHRRLOG Advanced Very High Resolution Radiometer
```

BIDRLOG Multimission SAR Processing Lab Basic Image Data Record
CONR4FIL CONVIM'd IBM floating-point files
CONVIM General blocked or unblocked tape files
DEMLOGA USGS Digital Elevation Model
DTTLOG A National Cartographic Information Center 1x1 degree Digital Elevation Model
INTCON IBM IBIS interface (tabular) files
SDRLOG Voyager System Data Record
SWAPPER CONVIM'd IBM 16-bit integer files
TAPES EBCDIC
VDEMLOG Digital Elevation Model
VEDR Voyager Experiment Data Record
VERTSLOG Landsat 1-3 ERTS
VFITS2 FITS
VGLAS0 Goddard Laboratory for Atmospheric Sciences
VGLLOG Galileo calibration files
VGOES Geostationary Operational Environmental Satellite
VGRLOG Voyager Experiment Data Record
VMDPIN Goddard MDP ground control points
VNEPRF Naval Environmental Prediction Research Facility
VOLOG Viking Orbiter Experiment Data Record
VQUIC ASCII text
VWRIS Wild Land Resource Inventory System

9 Advanced VICAR Concepts

Contents

EXPERT This section expands on two basic concepts introduced in Section 4: datasets and labels. More detailed information on dataset and label structure can be found in *The VICAR File Format*.

9.1 Dataset Structure

EXPERT The standard structure for VICAR datasets is a file of fixed-length records which consists of the parts listed below. Figure 9.1 is a visual representation of the dataset structure.

* VICAR label (Section 9.2)
* binary label header - optional
* binary label prefix - optional
* pixel data
* end-of-dataset label (EOL; Section 4.4) - optional

EXPERT The binary label is an optional area for storing information about a dataset in free-form binary format. The data in the binary label are not defined by the executive, but by the individual applications. There are two parts to the binary label, the binary header and the binary prefix. The binary header is useful for storing information which may pertain to the entire dataset. The binary prefix consists of a fixed number of bytes from each pixel data line and is useful for storing line-dependent information (e.g.,

marking "bad" data). The binary label is hidden from programs unless they specifically request access to it. If a program does request access, the binary label is then treated as a part of the data. If access is not requested, the binary label will not be present in the output dataset.

EXPERT The pixel data portion of a dataset may contain between 1 and 2.147E9 (2 to the power of 31, minus 1) lines, each of the same length. The maximum length is dependent upon the storage device. Each line is a sequence of pixels whose format is represented by 1, 2, 4 or 8 bytes (Table 4.3.3) and is fixed for a given dataset. In the "picture" interpretation, each line of a dataset represents one raster scan line. The sequence of lines represents the sequence of raster scan lines beginning with line 1 and proceeding down. Multi-dimensional pixel data can be organized in one of three ways: band-sequential (BSQ), band-interleaved by line (BIL) or band-interleaved by pixel (BIP). Additional information on multi-dimensional pixel data organization can be found in the *VICAR Run-time Library Reference Manual*.

9.2 VICAR Label Structure

EXPERT The VICAR label is an ASCII string containing free-field items of the form `keyword=value` (Section 4.4.1) separated by spaces.

EXPERT The VICAR label contains three classes of information. Dataset description ("system") items describe the size, organization, data format, and existence of the optional sections of the dataset. "Property" items describe properties of the image in the image domain, such as the map projection, lookup table, and latitude/longitude information. "History" items describe the history of the pixel data in the dataset, the procs that have processed the

pixel data, sometimes their parameters, the user identification, and the processing date and time.

EXPERT System items in the VICAR label include, in order:

```
LBLSIZE       the size of the label in bytes
FORMAT        the data format of the pixels in the
image
          (byte, half, full, real, doub, or comp; word
          long, and complex may also be found but are
          obsolete)
TYPE      the dataset type (image, param, graph1, graph2
          graph3, tabular)
BUFSIZ         (obsolete, but still required; set it
equal to
          RECSIZE in new files) the internal blocksize
          VICAR will use during I/O
DIM       the number of dimensions in the file (always
          equals 3)
EOL       end-of-dataset label (if there is an EOL,
EOL=1)
RECSIZE       the size in bytes of each record in the
VICAR
          file
ORG       data organization:
          BSQ  band sequential
          BIL  band interleaved by line
          BIP  band interleaved by pixel
NL        number of lines (number of records)
NS        number of samples (record length)
NB        number of image bands (number of data planes)
N1        equal to NS or NB depending on pixel data
          organization (ORG - Table 9.2)
N2        equal to NL, NS or NB depending on pixel data
          organization (ORG - Table 9.2)
N3        equal to NB or NL depending on pixel data
          organization (ORG - Table 9.2)
N4        not yet used; defaults to 0
NBB       number of binary prefix bytes
NLB       number of binary header records
HOST      the type of computer used to generate the
```

```
          image (alliant, cray, decstatn, hp-700,
          mac-aux, mac-mpw, sgi, sun-3, sun-4, tek,
          vax-vms)
INTFMT         format used ot represent integer pixels
          (byte, half, and full) in the file. (low, for
          vax-vms and decstatn; high for all other
          except cray, which isn't implemented yet)
REALFMT        format used to represent floating-point
          pixels (real, doub, and comp) in the file.
          (rieee for decstatn; vax for vax-vms; ieee
          for all others except cray which isn't
          implemented yet)
BHOST          type of computer used to generate the
          binary label. (same values as HOST)
BINTFMT        format used to represent integers in the
          binary label (same values as INTFMT)
BREALFMT  format used to represent floating-point
          data in the binary label (same values as
          REALFMT)
BLTYPE         type of binary label
```

DIM	ORG		
	BSQ	BIL	BIP
N1	NS	NS	NB
N2	NL	NB	NS
N3	NB	NL	NL

Table 9.2 VICAR Data Organization

EXPERT Property labels are located between the system and the history labels. They begin with the first occurrence of the keyword PROPERTY and end with the first occurrence of the keyword TASK. A dataset may exclude property labels entirely. Each property begins with a PROPERTY keyword, which is the name of the property set. This is followed by the label items that make up the

property. The valid property names, and the keywords that make up each property, are defined in a name registry maintained by the VICAR system programmer.

EXPERT History items in the VICAR label include:

```
TASK      a proc that has processed the dataset
USER      user identification (login id.)
DAT_TIM       processing date and time
optional items     added by the application proc listed
under task
```

Example: History items added by VICAR

```
     TASK='RESSAR77'
     USER='USERID'
     DAT_TIM='Wed Nov 12 19:06:24 1986'
     PIX_CNT=22259
     PARMS='AUTO-STRETCH:    0 to    0 and   138 to
255'
```

EXPERT The contents of a VICAR label may be listed either as formatted ASCII or as an ASCII dump (Appendix 10.12).

10 APPENDIX

Contents

10.1 VICAR Proc Function Definitions

This section briefly defines the classes into which VICAR procs may be classified. See Section 10.2 for lists of which procs fall into which classes.

Annotation
Superimpose text on or around an image

Arithmetic Functions
Arithmetic operations.

Atmospheric Features Analysis
Collect feature tiepoints and statistics

Blemish, Noise & Artifact Removal

Locate, remove, classify, or modify blemishes, noise or artifacts

Brightness Corrections
Match brightness or remove gradients

Calibration
Generate and/or remove intrinsic transfer functions

Catalog & Database Management
Manipulate or modify databases

Color
Analyze, enhance or reconstruct color imagery

Contrast Enhancement
Enhance the contrast of images

Data Compression
Compress or decompress data

Data Format Manipulation
Change data from one format to another (e.g., halfword to byte)

Data Transfer (logging)
Transfer VICAR data from one storage medium to another (e.g. tape to disk, disk to disk)

Display
Interactive display of images

Film Recording
Generate a film product from a VICAR file

Filtering
Perform image filtering

Foreign Data Transfer
Transform or log foreign data to or from the VICAR format

Fourier Transforms
Transform image to/from, or operate on an image in the Fourier Transfer domain

Galileo Specific Programs
Programs written specifically for the Galileo mission.

Geometric Tranformations
Change the spacial characteristics of an image (e.g., the shape)

Graphics
Superimpose graphics within an image

Histogram Generation
Generate histograms

IBIS Data Transfer
Transform or log IBIS data to or from the VICAR format

IBIS Programs
Programs which utilize the IBIS data format.

Labels
Display, create, remove or modify VICAR labels

LANDSAT Specific Programs
Programs written specifically to handle LANDSAT data.

Listings
Examine data values within a file (e.g., look at DN numbers)

Magellan Specific Programs
Programs written specifically for the Magellan mission.

Magnification & Reduction
Alter the image size

Map Projections
Alter the image projection

Mars Observer Specific Programs
Programs written specifically to process data from the Mars Observor Camera

Mask Generation
Reformat data and generate a mask around the data, suitable for film recording

Modification
Modify or operate on image subareas

Mosaic Generation
Generate a mosaic

Multispectral Analysis
Identify regions in multiple data planes

Pattern Recognition & Location
Identify or locate objects within an image

Photometry
Apply or determine photometric functions

Planetary Data Transfer
Transform or log planetary data to or from the VICAR format (e.g., Voyager, Viking)

Planetary Navigation
Determine camera viewing geometry

Plotter Displays
Generate plots

Registration
Find correspondence between 2 or more images

Reseau Location & Removal
Locate and remove reseau and fiducial marks from Voyager and Viking Orbiter images

Rotation
Change image orientation by rotating

Statistics
Collect, extract and/or display statistical information about an image

Stereo
Create or process stereoscopic image pairs

Synthetic Data Generation
Generate synthetic or simulated data

Tape Utilities
Perform magnetic tape functions

Tiepointing
Generate, modify or transform tiepoints

Utilities

Miscellaneous utilities (e.g., parameter passing, sending messages to system operator, current date and time etc.)

Viking Specific Programs
Programs written specifically to support the Viking mission.

Voyager Specific Programs
Programs written specifically to support the Voyager mission.

10.2 Classification of VICAR Procs by Function

A major strength of the VICAR system is the flexibility of its software set. This feature, however, makes clear classification of procs by function a difficult task. Many procs have a primary function, but may perform other secondary tasks that may also be useful. For example, the proc AMOS performs planetary navigation, but its main function is atmospheric motion analysis. Therefore, in the lists that follow, proc names in uppercase denote the primary function, and those in lowercase denote a secondary function.

Annotation

```
EDIMAGE        FONT      MASKV          STARLAB
TEXTAD
```

Arithmetic Functions

```
AVERAGE        DIFPIC         F2         F2_3D
F2COMP         fastmos        GF         LAVE
MF        NF2       PICSUM         qsar
RATIO          sargon         sargonb        XFORM
ZINTERP
```

Atmospheric Features Analysis

```
AMOS      UVECTOR       UVF       UVFMAP
UVFSTATS
```

Blemish, Noise & Artifact Removal

```
ADESPIKE  BLEMFIX       BLEMGEN       BLEMPIC
BLEMVORB  CCDNOISE  CONCOMP1  DROPOUT
DS4       GLLBLEMCOR    INSERT        lave
MERGE          OSBLEMLOC REPAIR        RESSAR75
RESSAR77  sargon        sargonb       SIMPLIFY
VGRBLEM        VGRFILLIN
```

Brightness Corrections

```
GEOMZ          GETZVAL       GRADREM       GRAFIMG
MZGEOM         PHOTFUNC  PIXGRAD
```

Calibration

```
ABLE86         CCDNOISE  CCDRECIP  CCDSLOPE
DC        FICOR77       FICOR86       FICORGEN
FIXLOC         FIXVGR        GALGEN        GALSOS
GETLOC         GLLCALNAME    GLLPFCF       GRIDGEN
GRIDLOCB  INTERLOC  LTGEN         MJSPFCF
MOMGEN         otf1      PICSUM        SOS
SPLITCAL  VGLLOG        XLOCUS
```

Catalog & Database Management

```
AGGRG          AGGRG2        CAMPARAM  CATARCH
CATCD          CATFILM       CATIMAGE  CATLABEL
CATLIST        CATMRPS0  CATNIMS       CATPRODUCTS
```

```
CATRADIO  CATREPORT CATSEDR       CATSLIST
CATSPICE  CNT       COUNT         DTRTAE
EDRVAL          LISTSEDR  MERGE         MOCINDEX
NXT       PHOTOM          PPLOAD        RESET
SRCH      SRCHGEN         STARCAT       STARCAT2
VOLSTSDR  VOSDRIBIS
```

Color

```
COLOR           COLORT          CSEND           DNSYMBOL
EDIMAGE         EIGEN           GIACONDA  GLLCOLOR
IDX       LEONARDO  LOOKUP          MAPGEN
PAINT           PAINTG3         RGB       RGB2PSEUDO
TRISTIM         TRUCOLOR  VLOOKUP
```

Contrast Enhancement

```
ASTRTCHR  color           FIT       HSTRETCH
idx       LOOKUP          MAPGEN          SSTRETCH
STRETCH         STRETVAR
```

Data Compression

```
BTC_COMP  BTC_DECOMP    CBIDRGEN  DEBARC
HUFFCODE  ICTCOMP       ICTDECOMP MGNSIZE
RUNCODE         STRPMTCH  VGRPWS
```

Data Format Manipulation

```
C         CCOMP           fit
```

Data Transfer (logging)

```
AVHRRLOG  COPY      DEMLOGA         DTTLOGA
label           LOGMOS          PIDRLOG         TAPES
```

```
VDEMLOG       VERTSLOG VGLLOG        VGRLOG
VOLOG         vsar
```

Display

```
BROWSE        DISPLAYS DISPOUT       edimage
FOOTPRINTIDX       IFFT      IGIS
IMP       leonardo QD        SHOWDISP
STEREOPICVIDS      XVICDISP
```

Film Recording

```
BARNE         PHOTO         QKLK      WILMA
```

Filtering

```
APODIZE       BICUBIC       BOXFLT2       CONCOMP1
FFT1AP        FIL2      FILTER        FILTER2
FILTERAP FILTEREM GIACONDA GLLCOLOR
MEDIAN        SBOXFLT       SHADY         SHADY2
SIMPLIFY TFILT         TFILT0        TRUCOLOR
WIENER
```

Foreign Data Transfer

```
AFEWCLOG AVHRRLOG CONR4FIL CONVIM
DEMLOGA       DTTLOGA       GRAF2DGN SWAPPER
TOXYZ         VDEMLOG       VERTSLOG VFITS2
VGOES         VSAR      XSECT
```

Fourier Transforms

```
APODIZE       FFT1      FFT11         FFT1PIX
FFT2      FFT22         FFTCLAS       FFTFIT
FFTFLIP       FFTMAG        FFTMAGIC FFTPIC
```

```
FT2       FTPACK        IFFT      MEM
OTF1      POWER         PSF       RESTORW
WIENER
```

Galileo Specific Programs

```
ABLE86        AMOS      BLEMGEN       BLEMPIC
CATCD         CATLABEL  CATMRPS       CATMRPS0
CATRADIO  CATREPORT CCDNOISE  CCDRECIP
CCDSLOPE  DEBARC        FICOR86       GALGEN
GALSOS        GEDREAD       GEDRGEN       GEDRLIST
GEDRVAL       GLLBLEMCOR    GLLCALNAME    GLLCAMPAR
GLLCOLOR  GLLFILLIN GLLGCOR       GLLMASK
GLLPFCF       GLLPSF        GLLREQUEST    GPWAUDIO
GPWS      GPWSFORMAT    GPWSMASK  GPWSSIM
GREDRGEN  GRIDGEN       ICTCOMP       ICTDECOMP
MANUAL3       MANUAL4       PICSUM        PWAUDIO
PWECHO        PWSMEK        SCTOMARK  SHUTOFF
SPLITCAL  SSIMEK        VGEDR         VGLLOG
VGSDR         XLOCUS
```

Geometric Tranformations

```
ARDREC        C130RECT  FLOT      GEOM
GEOMA         GEOMREC       KEPLER        LGEOM
map2      maptrans  MGEOM         MZGEOM
POLARECT  POLYGEOM  POLYREG       PTP
RING      ROTATE        size
```

Graphics

```
ADL       BOXGEN        CONTOUR       GRID
IDX       MAPGRID       MARK      OVERLAY
POLYSCRB  SHADY         STERMAP       UVECTOR
UVFMAP
```

Histogram Generation

```
fit       HIICCUP          HIST        HIST2D
HISTGEN       idx       maskv          multovly
SHIST         vgrmask
```

IBIS Data Transfer

```
INTCON        VCOPOUT       VMDPIN        VQUIC
VWRIS         WRISOUT
```

IBIS Programs

```
AGGRG         AGGRG2        BIDDRINDXCATFILE
COLOR         CONTOUR       COPYFILE  CORNER
CTRMATCH  DGN2GRAF  EDIBIS        G3APPEND
G3GEN         G3LIST        GETZVAL       GR2GEN
GR2IM         GR2POLY       GR2REP        GRAF2DGN
GRAFIMG       GRLIST        GRUTIL        HEX2INT
HEXGEN        HICCUP        IBIS          IBISGCP
IBIS2TCL  IBISLSQ       IBISNAV       IBISNBR
IBISREGR  IBISSTAT  IBISUPDATE    IGENER
INTCON        LINEMTCH  MAPGEN        MF
MSSIBIS       MULTOVLY  OMC       OMCOR
PAINT         PAINTG3       PCOPIN        PCOPOUT
PHOTFIT2  PHOTOM        PICMATCH  PLANETCNV
PLOT3D        PLOTINT       PLTGRAF       PLTSYM
POLYCLIP  POLYCON       POLYGEN       POLYGEOM
POLYMAP       POLYMASK  POLYPMAP  POLYPNT
POLYREG       POLYSCRB  QREP      QUERYLAB
RASTOGRAFRASTOVEC  REPORT        RIV2HEX
ROADHEX       ROWOP         RUNCODE       SDSIBIS
SORT      TIEPARM       TIEPLOT       TOIBIS
TRANSCOL  VCOPOUT       VGRBLEM       VGRIBIS
VMDPIN        VOSRIBIS  VQUIC         VWRIS
WRISOUT       XYZPIC        ZINTERP       ZIPCOL
```

Labels

```
BADLABELS CLABEL        GETLAB        GETPWS
LABEL         LAB2TCL           LABLIST       LABSWTCH
LABVFY        MAPLABPROG    QUERYLAB  SLABEL
STARLAB       VCOPY         VSAR
```

LANDSAT Specific Programs

```
DS4       THERMMOS  TIPSMOS       VERTSLOG
```

Listings

```
BWSEND        CSEND         DISPARMS  DNSYMBOL
G3LIST        GR2REP        GRLIST        idx
LIST      PRINTPIX  PRNTIM        QREP
REPORT        XVPLIST
```

Magellan Specific Programs

```
BANRMOV       BIDRINDX  BIDRLOG       BIDRSIM
BIDRLOG       C_BIDR        CBIDRGEN  CONTENTS
CUMDIR        FBIDR_DIRF    FINDTIE       FRAME
GAIN      GMASKMGN  LISTLOCS  LOGMOS
MASKMGN       MGNDIRNAME    MGNFIT        MGNSIZE
MGNZPAD       MIDR      MIDR_ARCH     MIDR_INSERT
MIDRPOS       OVERLAP       PDIR_ARCH PFASTMOS
PMASKMGN  RCBR      RDM       RFMINIT
SFASTMOS  SIPHON        tapesiphon    TIESIM
V2BIDR        VIEW
```

Magnification & Reduction

```
FULLSIZE  idx       SIZE
```

Map Projections

```
FOOTPRINT LOOKUP        MAP       MAP2
MAP3      MAPGEN        MAPGRID   MAPMATCH
MAPTRAN   MAPTRAN3  MAPTRANS  MAPTRANS2
MASKMOS       PERSLAB       POLARECT  POLYMAP
POLYPMAP  PTP       RING      STERMAP
THERMMOS  TOTOPO        UVECTOR       UVF
UVFMAP        UVFSTATS  UVMAP         VLOOKUP
VWRIS         XFORMAP
```

Mars Observer Specific Programs

```
BTC_COMP  MOCINDEX  MOCSUMMARY
```

Mask Generation

```
FMASK         GLLMASK       GMASKMGN  GPWSMASK
HSTRETCH  MASKMGN       MASKMOS       MASKV
MOCMASK       NIMSMASK  PMASKMGN  POLYMASK
VGRMASK
```

Modification

```
EDIMAGE       FMASK         LEONARDO  MAPGEN
PAINT         PAINTG3       POLYMASK  QSAR
SARGON        SARGONB       STRPMTCH  XYZPIC
ZCIRCLE       ZFILL
```

Mosaic Generation

```
APPEND        AUTOMATCH DPOLYCUTS FASTMOS
IBISGCP       INSECT        KEYMOS1       LOGMOS
MANMATCH  MANUAL        MAPMATCH  MESURMOS
MESURMOSCAHV  MGNFIT        MOSPLOT       MSS
NEWMOS        OMCOR         RAPIDMOS  SFASTMOS
THERMMOS  TIPSMOS
```

Multispectral Analysis

```
CLUSAN        DENDSTAT EDSTATS       EIGEN
EIGENVEC FASTCLAS IMP      MSS
MSSIBIS       NIMSMASK rgb       SIMPLIFY
SPAM     STATS         USTATS
```

Pattern Recognition & Location

```
CORNER        FASTCLAS GRIDLOCB INTERLOC
PRNTIM        STARCAT       ZCIRCLE
```

Photometry

```
PHOTFIT2 PHOTFUNC PHOTOM        PHOTTEST
```

Planetary Data Transfer

```
GET_SFDU EDRGEN        PIC2VIC       SIPHON
tapesiphon    VEDR     VGEDR         VGLLOG
VGRLOG        VGSDR         VMAC     VOLOG
```

Planetary Navigation

```
amos     FARENC        GETSPICE map2
MGNCORR       NAV      NAV2     NEARENC
```

Plotter Displays

```
IMP      otf1     PLOT3D        PLOTINT
PLOTTING PLTDGN        PLTGRAF       PLTSYM
POWER         QPLOT         rg       STATPLT
TIEPLOT
```

Registration

```
GEOM      GEOMA          INSERT        LGEOM
MGEOM          MZGEOM         PICMATCH  PICREG
R         RG        TOPTRANS
```

Reseau Location & Removal

```
FIXLOC         RES77          RESLGEN       RESLOC
RESLOCVO  RESSAR75  RESSAR77
```

Rotation

```
FLOT      ROTATE         xlocus
```

Statistics

```
CLUSAN         CLUSTEST  DENDSTAT  DIFPIC
EDSTATS        EIGEN          F2        FASTCLAS
HIST      HISTGEN        IBISLSQ       IBISREGR
IBISSTAT  IMP       LAVE      LIST
MAXMIN         MF         MSSVIEW       PIXGRAD
PIXSTAT        SDSIBIS        SLIST          SPAM
STATPLT        STATS          USTATS        UVFSTATS
```

Stereo

```
STEREO         STEREOPIC STERGEN       STERMAP
```

Synthetic Data Generation

```
CELLGEN        FRACGEN        GAUSNOIS  GEN
```

```
GENTHIS       PHOTTEST  POLYNOIS  RADAGEN
SPOT     TESTGEN
```

Tape Utilities

```
ADDON         ALLOC         CONVIM        COPY
DEALLOC       DISMOUNT  MOUNT         MTCLEAR
REWIND        SFDULOG       TAPES
```

Tiepointing

```
convpr        convrp        corner        getloc
locus         MANMATCH  MAPMATCH  OMCOR2
PICMATCH  picreg        R
rg        STERGEN       STERMAP       tieconm
tieparm       TIEPLOT       TOTOPO        TPTCOORD
TPTEDT        TPTEDT2       TRACKER       TRACKER2
TRACKER3  xvplist
```

Utilities

```
AL        APFLAG        CAMPARAM  DATETIME
DTRTAE        FORM      GETLAB        IBIS2TCL
LAB2TCL       MENUTREE  REQUEST       SYNTAX
TRANSLOG  USERNAME  V2VERSION WILDCARD
```

Viking Specific Programs

```
BLEMVORB  DROPOUT       OMCOR         PLANETCNV
RESLOCVO  RESSAR75  SOS       VOLOG
VOLSTSDR  VOSDRIBIS
```

Voyager Specific Programs

```
AMOS      BLEMFIX       BROWSE        CAMPARAM
```

```
CATIMAGE  CATLIST       CATSEDR       CATSLIST
EDRGEN        EDRVAL        FARENC        FICOR77
FICORGEN  FIXLOC        FIXVGR        FOOTPRINT
GRIDGEN       IMLIST        KEPLER        LISTSEDR
MERGE         MJSPFCF       NAV       NAV2
NEARENC       OSBLEMLOC PHOTFIT2  PHOTFUNC
PHOTOM        PPLOAD        PTP       PWAUDIO
PWECHO        QKLK      RADIANCE  RES77
RESLGEN       RESLOC        RESSAR77  RING
SEDRGEN       SRCH      UVECTOR       UVF
UVFMAP        UVFSTATS  VEDR      VGRBLEM
VGRFILLINVGRIBIS        VGRLOG        VGRMASK
VGRPWS
```

10.3 Standard VICAR Procs and their Functions

Certain VICAR procs make use of facility-specific or propietary hardware devices or software packages. The lack of these resources may cause a given proc to fail. Procs with this characteristic are denoted in this section by symbols in the NOTE column. "Required" means that the proc will not function without the indicated resource. "May be required" means that the proc is probably a procedure which will call one program if the resource is available and another if it is not.

Also included in the NOTE column is an indication of the whether or not the proc has been ported to UNIX yet, or whether a proc is obsolete. (Obsolete programs may still be available, but are no longer supported.)

The symbols in the column "NOTE" are:

```
A    -    Array processor required
A*   -    Array processor may be required
```

C - CCSI routines required
D - Datatrieve required
D* - Datatrieve may be required
F - Film recorder required
G - Audio generator required
I - Interactive display required
I* - Interactive display may be required
L - DEC LA50 printer required
N - Intergraph hardware required
O - Obsolete or unsupported software.
P - Procedure
R - Rainbow PC graphics required
T - Device-independent plotting performed
U - Have not yet been ported to UNIX, but will be in the future.
V* - VLSI filter device may be required
X - Will not be ported to UNIX.

VICAR PROC NOTE FUNCTION

ABLE86 OX Extracts data from the Galileo SSI flight and ground calibration labels.

ADDON OX Advances a tape to the end of the last file or to a position after any given number of files.

ADESPIKE Detects pixel spikes and removes them by averaging adjacent pixels. Single missing lines are also filled in.

ADL O Adds a line to an image by adding a constant value to pixels specified by endpoints.

AFEWCLOG OX Reads the Air Force Electronic Warfare Center digital terrain database and produces VICAR images.

AGGRG O IBIS program aggregating columns of numbers using any designated column of the file as a

control column.
AGGRG2 O IBIS program collapsing columns of numbers in
an interface file into smaller columns using a designated column as the control.
AL U Allocates disk image files.
ALLOC X Allocates a tape drive to the user.
ALT_DAILY X Processes a nominal set of 7 consecutive orbits
of ALT-EDR data from one input tape into 7 BADRS.
ALTSIM X Generates the orbital intermediate files for
MIPL-MGN processing of altimetric/radiometric data.
AMOS DIU Interactive program supporting Atmospheric
MOtion Studies on Voyager and Galileo images.
ANNOT U Accesses the britton-lee database and returns
the database output to a file specified by the parameter ANN_FILE.
APFLAG AU Returns into local variables the array processor
availability, format and size of an input image.
APODIZE O Modifies picture borders so that the Fourier
transform will be free from spikes through the zero frequency axis, allowing filters to operate
without ringing on the edge of an image.
APPEND O Concatenates up to 30 images vertically.
ARCDR_GEN X Makes the OHFnnnnn.LBL, EPFnnnnn.LBL,
ADFnnnnn.LBL, RDFnnnnn.LBL and ACONTENT.TAB files for the ARCDR CD-ROMs.
ARCHIVE X Creates the ancillary file of all archival
products from the MIPL/MAGELLAN systematic stream.
ARDREC OX Corrects aircraft line scanner imagery for
relief displacement using digital elevation

(DEM) data.

ASTRTCHR Performs automatic linear stretches on fullword floating point or integer images.

AUTOMATCH A program which automatically acquires tiepoints from sets of images for the purpose of producing mosaics.

AVERAGE Averages up to 48 separate images into one image.

AVHRRLOG OX Logs NOAA AVHRR data from tape into VICAR format.

BADLABELS Determines the "Bad Data Values" for an input GLL SSI UDR. Optionally, it will also calculate the entropy of the image.

BADR_ARCH X Archives BADRs onto magnetic tape and optical disk.

BADRLOG X An ALT-DAILY internal process which takes the BADR data records created by RCBR and forms BADR orbital files which will be run into BIN to make the GADR.

BADSHOW U Prints or displays the bad data values in an image.

BADRSIM X Generates the basic altimeter data record files for MIPL-MGN processing of altimetric data. Will place ramp-generated data or user specified imagery into the output files.

BANRMOV OX A stream used to remove bands from F- or C1-MIDRs.

BARNE FX Submits film recorder requests to the MRPS (Multiple Recorder Production System). Will be replaced by BARNER in the near future. (See the MRPS User Guide for more information.)

BICUBIC Enlarges images by integer zoom factors using
cubic convolutional reconstruction filters.
BIDRINDX X Creates an IBIS file containing information
from an F or C BIDR. The IBIS file can be used to navigate around in the BIDR without having to read the file.
BIDRLOG X Transforms a single bidr file into a VICAR
format image file.
BIDRSIM X Creates files/tapes in the Magellan BIDR format.
BIN X Places data into sinusoidally projected image
files from VICAR orbital files.
BLEMFIX IU Interactive program for constructing Voyager
blemish files.
BLEMGEN Generates Galileo Blemish Definition Files
from the output of GALGEN. Blemish file to be used by GALSOS.
BLEMPIC U Creates images of the permanent blemishes, low
full-well pixels and total blemishes recorded in the BLEMGEN blemish file. Initially intended for Galileo processing.
BLEMVORB Removes blemishes, wrinkles, and warts from raw VIKING Orbiter images by replacing rectilinear areas with an average of nearby points.
BOXFLT2 Applies either a highpass or lowpass filter
to a dataset.
BOXGEN OX Generates a step wedge series of discrete blocks arranged horizontally in a big "box". Used in legends and keys.
BRDR_ARCH X Archives BRDRs onto magnetic tape and optical
disk.
BRDRLOG X Processes the Magellan radiometer data from
an input file containing one orbit's worth of data into output orbital files.

BROWSE A*DIU Interactive program allowing the user to col-
lect display images from the Voyager Image Catalog, based on user-supplied criteria.
BTC COMP ? MIPS VICAR application for performing 4x4 BTC
compression on images using the algorithm developed by Delp and Mitchell. Used on the Mars Pathfinder MicroRover imaging system.
BTC DECOMP ? MIPS VICAR based application for decompressing
4x4 BTC compressed images.
BWSEND ORX Generates an ASCII file representing a grey
scale image or pseudo color image in a compressed format to be downloaded to a Rainbow PC with a graphics board.
C U Converts images between all recognized format
types by using simple linear transformations.
C_BIDR X Used for daily processing and archiving of
the C-BIDR data stream.
C130RECT OX Removes panorama distortion, adjusts aspect
ratio and corrects tilt of images from scanners
sampling at equal angular increments.
CAMPARAM Returns into local variables information from
a Voyager image label.
CATARCH X A program procedure which enters the archival
tape name, and the archival file number into the image catalog.
CATCD Updates Galileo SSI image catalog with infor-
mation on systematic processing products that are stored on CDROM devices. Supplements the program CATPRODUCTS.
CATFILE OX IBIS program concatenating up to 10 interface
files.
CATFILM X Sorts through the Image Catalog and forms

a collection of records which match the input sort parameters.

CATIMAGE DX Enters data from processed Voyager images into the Voyager Image Catalog.

CATLABEL Updates the VICAR label of a Galileo SSI UDR by retrieving the most current information from the Galileo IMAGE Catalog and SPICE kernels.

CATLIST DX Allows user to sort through the Voyager Image Catalog with up to 12 user-supplied sort criteria.

CATMRPS X Galileo procedure facilitating the submission of processing requests to MRPS.

CATMRPS0 X Stores data related to GLL Barne submissions in the PHOTO_PROD and MRPS_FILE domains of the Processing Request Catalog.

CATNIMS U Reads the NIMS Systematic Processing request and creates corresponding records in the PROCESSING, MOSAIC, and MOSEDR domains in the NIMS Catalog.

CATNIMS2

CATPRODUCTS Allows the analyst to perform an update of one of the Galileo SSI, NIMS, and PWS Catalog domains containing product information.

CATRADIO U Stores data about Galileo SSI radiometric cali-bration files in the SSI_RADIOMETRIC domain of the Galileo (Datatrieve) Catalog.

CATRADIO2

CATREPORT U Provides a means to search the Galileo SSI, PWS, and NIMS Catalogs and produce either a report with a pre-defined format or an adhoc report.

CATSEARCH U Aids in the automation of systematic processing and for the production of EDRs and SSI REDRs.

CATSEDR X Reads a Voyager SEDR tape, and optionally stores
the data into the image catalog.
CATSLIST DX Allows user to sort through Voyager Image Catalog with up to 7 user-supplied sort criteria.
CATSPICE U Reads SSI and PWS SPICE kernels and stores the
data in the SSI Overview Domain and PWS Predict
Domain of the Galileo Catalog.
CATSUP
CBIDRGEN X Performs a 3x3 compression on MAGELLAN F-BIDR
SAR data, while perserving the format as defined
in RPDS-101, revision 1.
CCDNOISE T Calculates the system gain constant and noise
floor of the Galileo CCD Light Transfer File generated by MOMGEN.
CCDRECIP T Calculates the sensitivity and shutter offset
for the Galileo CCD camera system from the output from MOMGEN.
CCDSLOPE T Calculates slope and offset from the Galileo
CCD Light Transfer File generated by MOMGEN.
CCOMP Converts images between the complex pixel format and two real format images (amplitude and phase, or real and imaginery) and vice versa.
CD_ARC X Creates tje DMAS directory file for the CD-ROM.
CDGEN Generates PDS label files for the following
files:
CDGEOM U Takes the output from a command that accesses
the experimenters notebook and creates the files
geom.tab and geom.lbl.
CELLGEN OX Creates an image composed of square, equal-

sized regions of encoded DN values increasing in value from left to right and top to bottom.

CFORM Converts images between any of the recognized data formats and performs linear transformations on images.

CHKSPACE Returns the number of free block (512 bytes) available to this user on a specific disk drive.

CHKSPACE UNIX

CHKSPACE VMS

CLABEL Reads the contour information generated by CONTOUR, after CONTOUR's output graphics file has been expanded by POLYSCRB. Then generates a FONT parameter file, which may be used to add labels to the contour file.

CLBLFONT U Writes text onto any VICAR2 byte image with a choice of seven available FONT styles.

CLEANLABEL Removed duplicated label items from the history label of any VICAR image.

CLUSAN Creates a statistics file using the simulated annealing optimization technique to find the best cluster partition for MSS data.

CLUSTEST Determines the statistical significance of clusters in a stats file.

CNT Returns into a local variable the number of files in a SRCH list.

COLOR OX IBIS program coloring "painted" regions with 5 or fewer colors so that adjacent regions have different colors (DN levels).

COLORT Transforms images between color domains (RGB, Tristim, CIE, USC, Spherical, HSR, HSI or Cube Root).

COMMENT O Allows a PDF file to display a comment as it executes.

COMMON PROC DONE

COMMON SAVE PAR
CONCOMP1 O Removes high frequency noise components from
an image. Typically used on classified or stratified images produced by FASTCLAS or USTATS.
CONLAB Speeds up the contour label program written by
Mike Girard.
CONR4FIL OX Converts IBM floating-point (REAL*4) data files
to VAX format.
CONTENTS X Creates the contents.tab and the contents.lbl
files which reside in the root directory of the CD-ROM and the browse.lbl and hist.lbl files which are in the MIDR subdirectories.
CONTOUR O IBIS program creating a Graphics-1 file of
isolines.
CONVIM X Converts images on tapes of various formats
into VICAR format (e.g., IBM to VAX). Will be replaced by DCONVIM.
CONVISOS OX Converts between image and object space coordinates for all flight projects.
CONVPR OX Transforms a PICREG tiepoint file into the
format needed by R.
CONVRP X Converts R, RG2 or TIECONM tiepoint files into
the format used by PICREG.
COPY Copies all or part of an image to another file.
COPYOD ? Copies all or part of an image into another
file. Works on labeled or unlabeled files on tape or disk.
COPYFILE OU IBIS program which copies selected columns of
an input interface file into selected columns of an output interface file.
CORNER IBIS program finding good tiepoint locations

by looking for corners or features in the image.

COUNT PU Types the number of files in a list created by SRCH.

CSEND ORX Allows color images to be downloaded onto a Rainbow PC graphics board in 16 color steps.

CTRMATCH OX IBIS program matching an image produced by PAINT with information contained in a Graphics-1 file.

CUMDIR X Magellan program to create CD-ROM CUMDIR files.

DATETIME Types the current date and time.

DC Creates "dark current" image from several input images.

DEALLOC X Deallocates a tape drive assigned to the user.

DEBARC Decompresses a Galileo image output from the Block Adaptive Rate Compressor.

DEBARC2 Same as DEBARC, but for diagnostic purposes only.

DEMLOGA OX Logs USGS DEM (Digital Elevation Model) data into VICAR format.

DENDSTAT OU Produces a dendrogram (tree-like diagram) for classification statistics files produced by STATS and USTATS.

DGN2GRAF OX Converts either 2-D or 3-D INTERGRAPH IGDS design files to IBIS Graphics-1 format.

DIFPIC Differences two images, outputs a difference images and reports statistics.

DISMOUNT X Dismounts a tape from a mounted tape drive assigned to the user.

DISPARMS OU Displays the parameters in a VICAR parameter

file.
DISPLAYS IX Allocates or deallocates a display device.
DISPOUT IOU Produces VICAR formatted datasets from the
video and/or graphics planes of frame buffers supported by the MIPL Virtual Frame Buffer Interface.
DNSYMBOL OX Converts DN values into symbols and shading.
DPOLYCUTS OX Generates cutlines for mosaicking.
DROPOUT Finds and fixes pixels and missing lines in
VIKING Orbiter images by interpolating over the
missing data.
DS4 OX Removes 6-line striping from LANDSAT imagery
by performing localized histogram matching of the 6 sensors.
DTRTAE DX Allows user to run Datatrieve commands from
within VICAR procedure.
DTTLOGA OX Logs NCIC 1X1 degree DEM (Digital Elevation
Model) data into VICAR format.
DVECTOR Draws any number of vectors on a picture of any
size.
EDIBIS O IBIS interactive program for editing interface
files.
EDIMAGE I Allows interactive image editing and annotating.
EDRGEN U Generates Voyager EDR tapes from VICAR image
files.
EDRVAL DU Validates Voyager EDR tapes.
EDSTATS U Allows editing of classification statistics
files
created by STATS and USTATS.
EIGEN P Procedure computing the principal component

transformation matrix of up to 32 input channels. Also contains the color decorrelation stretch algorithm.

EIGENVEC Computes the principal component transformation matrix for up to 32 input channels.

EXPEDIT U Takes the output of the program EXPER and puts it in a form that is readable by the program GEOM.

EXPER U Accesses the experimenters notebook and returns the output of the database query it calls to a file specified by the user.

F2 Performs general arithmetic operations on images.

F2_3D X Similar to F2 but allows for arithmetic operations between three dimensional input files and can produce a three dimensional output. Will be replaced by F2.

F2COMP OU Handles arithmetic operations on images in complex number format.

FARENC Determines target body center from an image containing a planetary limb.

FASTCLAS Classifies multispectral images using parallelepiped algorithm and Bayesian Maximum Likelihood method.

FASTMOS Mosaics up to 48 images based on user-specified input parameters.

FBIDR_DIRF X Creates DMAS directory file for archiving Magellan FBIDR's.

FFT1 A*PU Procedure computing forward and inverse complex 1-D Fourier transforms.

FFT11 Computes the forward or inverse complex Fourier Transform on a line-by-line basis. It is analogous to the program FFT1AP except that it does not use the array processor and is more flexible in its input format requirements: it

will allow any data format and most FFT sizes.

FFT1AP AX Used in conjunction with FFT1PIX, FFT1AP provides a versatile one-dimensional filtering technique.

FFT1PIX Displays an amplitude and/or phase picture of the input Fourier transform. Also permits user to modify the Fourier transform.

FFT2 A*P Procedure computing forward and inverse complex 2-D Fourier transforms.

FFT22 Performs direct and inverse 2-D fast Fourier transforms. Doesn't require the array processor.

FFT2AP AX Same function as FFT22 but uses the array processor.

FFTCLAS OU Analyzes halfword FFT images and produces 5 attribute values for each FFT examined.

FFTFIT O Modifies complex Fourier Transforms (FT) by either multiplying the FT by an input image or by making the amplitude of the FT proportional to an input image.

FFTFLIP O Rearranges the internal structure of a Fourier Transform such that DC is moved from the upper-left corner to the center of the image and optionally performs a transposition of upper-left to lower-right diagonal.

FFTMAG O Expands a picture's size by a factor of 2**N by enlarging the Fourier Transform of the picture using the sampling method.

FFTMAGIC O Obtains the amplitude from the phase of a Fourier Transform or vice versa.

FFTPIC Extracts data from a Fourier transform for image display.

FICOR77 Removes Voyager vidicon radiometric distor-
tions.
FICOR86 X Removes summation-mode Galileo CCD radiometric
distortions.
FICORGEN X Generates a scale correction file for use by
Voyager radiometric programs FICOR77 and
FIXVGR.
FIL2 Computes filter weights from various functions
for input to FILTER.
FILSUM U Displays the DACS file summaries for both MIPS
real-time streams.
FILTER A*PV* Filters images using user-specified weights.
FILTER2 A*PV* Procedure which calculates filter weights from
various functions and then applies weights in
the spatial domain. Calls FIL2 and FILTER.
FILTERAP X Performs two-dimensional convolution filtering,
using the FPS array processor.
FILTEREM Performs 2-D convolution filtering. Can be
used to perform high-pass or low-pass
filtering.
FINDSPACE U Selects, from a specified list of disks (or
directories), the one that has the most
available disk space.
FINDTIE U Finds and records tiepoint information for
Magellan F-BIDRs.
FIT Performs automatic linear stretches on halfword
images, based on a histogram. May be used to
convert from halfword to byte format.
FIXLOC Lists and edits images containing coordinate
information generated by RESLOC.
FIXVGR Multiplies Voyager halfword images by a con-

stant to produce more radiometrically accurate data. May be run after FICOR77.

FLOT "Flips" or rotates images by 90 degrees.

FMASK OX Replaces any pixel in an image with the specified DN, if the corresponding pixel in the second input "mask" file has a DN=0.

FONT Superimposes text on an image allowing the selection of font type, size and intensity.

FOOTPRINT DIPU Interactive program displaying a Voyager image "footprint" on a specified map with graphical indication of the limb, terminator, and lighting angles. Requires the Voyager Image Catalog.

FORM Returns into local variables data format, NL and NS of a file.

FRACGEN O Generates images which are similar to elevation images of mountainous regions using the fractional Brownian Motion process.

FRAME X Creates the frame.tab, frame.lbl, and 56 subframe label files that will go in a MIDR subdirectory of a CD-ROM.

FT2 AX Computes forward and inverse complex 2-D Fourier transforms.

FTPACK X Converts data formats from FT2 "packed" to FFT22 "unpacked" and vice versa.

FULLSIZE OX Sizes an image down by integral factors. Supports byte, half and fullword formats.

G3APPEND OX IBIS program taking Graphics-3 files and appending them together to make a single Graphics-3 output file.

G3GEN OX IBIS program generating a Graphics-3 file from user parameters.

G3LIST OX IBIS program producing a listing of a Graphics-3 file.

GADR_ARCH X Archives the terrain output from the GADR procedure to tape and optical disk.

GAIN X Writes the Magellan rset gain report.
GALGEN Generates Galileo CCD radiometric calibration
files and dark current files for input to
GALSOS.
GALSOS U Decalibrates Galileo CCD images with calibra-
tion files produced by GALGEN and BLEMGEN.
GALSOSI U Generates "raw" images from radiometrically
corrected flight images.
GAUSNOIS OU Generates a random DN (noise) image using a
Gaussian probability distribution.
GEDREAD U Copies the EDRs on a Galileo SSI or PWS EDR
tape to disk, producing files in the VICAR
file format.
GEDRGEN U Generates multiple copies of Galileo SSI or PWS
EDR tapes from a sequence of input EDR disk
files.
GEDRLIST Lists the Galileo SSI, PWS, or NIMS binary and
line headers in a readable form.
GEDRVAL U Validates Galileo SSI or PWS EDR tapes.
GEN Generates an artificial image with specified
size and values.
GENTHIS Generates a small image given a list of DN
values for each pixel in the image.
GEOM PU Procedure automatically selecting MGEOM or
LGEOM to perform geometric transformations.
GEOMA Performs geometric transformations on images
using quadrilaterals.
GEOMREC OU Transforms slant range measurements to ground
distance measurements.
GEOMZ OU Performs vertical (brightness or DN) correc-
tions on images specified by a set of points in
a parameter dataset.

GET_SFDU X Transfers telemetry data, known as SFDUs, from
AMMOS using TDS (Telemetry Delivery Subsystem) and TOT (Telemetry Output Tool) Software. In addition, this program has the capability of reading in files containing SFDUs.

GETGEOM U Returns in restab the geoma parameters required
to geometrically correct an image.

GETLAB Retrieves a label item from any part of the
VICAR label and assigns it to a local variable.
(Same as LAB2TCL - which has better help file)

GETLOC U Extracts subarea from a tiepoint file con-
taining theodolite measurements of grid images in order to calibrate a vidicon or CCD camera.

GETPLACON X Returns planetary constants given either the
planet name or the planet ID# as in the sedr/spice.

GETPWS X Unpacks a PWS EDR file into bytes and removes
the binary headers and labels.

GETSPICE X Program returns the SEDR or SPICE navigation
data for an image from any flight project.

GETZVAL OU IBIS program extracting brightness values in
an image from points specified in an interface file.

GF OX Allows the user to create FORTRAN IV-like expressions to perform general mathematic operations on one or more graphics-1 file coordinates.

GIACONDA U Transforms images taken through several filters
into a color image in which designated spectra are accurately reproduced.

GLLBLEMCOR U Performs blemish corrections for a SSI REDR
image. Also identifies the Low-Full-Pixels and

stores that information in the binary header record of the output image.

GLLCALNAME X Returns the file name of the proper calibration file given a set of defining parameters.

GLLCAMPAR X Reads information from the label of a Galileo image and returns it to a TAE procedure.

GLLCOLOR Transforms Galileo images taken through several filters into a color image in which an attempt is made to reproduce accurately certain designated spectra.

GLLFILLIN Fills in missing lines, partial lines, or trucated lines for Galileo SSI images.

GLLGCOR Returns the geometrically correct location of a pixel based on the uncorrected pixel location. Specific to the Galileo camera system (SSI).

GLLMASK U Prepares Galileo iamges for film recorder playback by surrounding the image with a mask.

GLLPFCF X Creates radiometric calibration files for summation-mode Galileo images.

GLLPSF X Generates a 2-D point spread function in fft format suitable to be converted into a 2-D optical transfer function by vicar program FFT22.

GLLREQUEST X Constucts a Datatrieve command file consisting of Processing Request information.

GMASKMGN X A set of mask routines used to produce small image masks for the photoproducts of the Magellan project.

GNAV IOX Determines the camera viewing geometry of an image by locating one or more distinct features in the image.

GPWAUDIO U This program reads the GLL PWS file (UDR or

EDR), and outputs the data, via DR11-W (logical name = PWSDEV), a Direct Memory Access interface, to the MIPL Audio generator.

GPWS PU Procedure performing Galileo PWS Systematic Processing

GPWSFORMAT U Adjusts Galileo PWS image size for the FFT and converts information in the input binary label to the output history label.

GPWSMASK U Formats the Galileo PWS data for hardcopy display.

GPWSSIM X Generates one RIM of Galileo PWS data to simulate the EDR disk file. Used to generate data for Galileo software testing.

GR2GEN OX IBIS program generating a Graphics-2 file from user parameters.

GR2IM OX IBIS program converting parts of a Graphics-2 file into VICAR images.

GR2POLY OX IBIS program converting user-selected parts of a Graphics-2 file into Graphics-1 format.

GR2REP OX IBIS program producing a listing of a Graphics-2 file.

GRADREM OX Removes large, systematic and non-linear artificial brightness gradients along scan lines of wide-angle aircraft scanners.

GRAF2DGN OX Converts IBIS formatted data to either 2-D or 3-D Integraph IGDS design files.

GRAFIMG OX IBIS program converting image data into a gridded 3-D Graphic-1 file with z-values corresponding to image DN values.

GRDR_ARCH X Performs batch processing of the GRDR data stream.

GRDR_DAILY X Initializes the GRDR which consists of 4 files.
GREDRGEN U Generates multiple copies of Galileo SSI REDR
tapes from a sequence of input REDR disk files.
GREDRVAL U Validates Galileo SSI tapes.
GRID Superimposes a user-defined reference grid on a byte image.
GRIDGEN Generates a "perfect" grid for use with anal-
ysis of Voyager and Galileo geometric grid images.
GRIDLOCB U Locates grid intersections to sub-pixel accuracy.
GRLIST OX IBIS program producing a line and sample list-
ing of a 2-D or 3-D Graphics-1 file.
GRUTIL OX IBIS program appending or converting 3D and 2D
Graphics-1 files.
GXFRAME X Creates the detached framelat label files that
go in the image subdirectories of a GxDR CD-ROM.
HEX2INT OX IBIS program converting Graphics-1 file "hex"
edges into a graphics file with "hex cc" coordinates.
HEXGEN OX IBIS program generating image and attribute
files of hexagon grids from user parameters.
HICCUP Computes the histogram, mean value, and stan-
dard deviation of an image and outputs thee values to an IBIS-2 format histogram file.
HIST Prints DN-frequency histogram of all or por-
tions of an image along with image statistics.
HIST2D U Generates a 2-D histogram file from a VICAR
file of three dimensions.
HISTGEN Generates a histogram dataset of an image.
HSTRETCH OU Produces a binary mask or may be used to

modify the DN values of an image (similar to a table stretch).

HUFFCODE OX Compresses or decompresses VICAR images using Huffman coding.

IBIS IBIS is an IBIS-2 program, designed to provide basic file utilties for IBIS-2 and IBIS-1 tabular and graphics files.

IBIS2TCL OU IBIS program returning tabular file values into local variables for use in procedures.

IBISGCP U Collects Ground Control points to use in mosaicking.

IBISLSQ OU IBIS program performing least-squares fit on columns of a tabular file, placing them in a column of the tabular file.

IBISNAV U Extracts information form the SPICE or SEDR file, augments it with camera constants and then reformats the data as a VICAR IBIS interface file.

IBISNBR OX IBIS program finding the nearest neighbor point for every point in a column of a tabular file.

IBISREGR OU IBIS program performing a series of multiple linear regression analyses on tabular files, searching for a best fit.

IBISSTAT OU IBIS program performing statistical analyses (e.g., multiple regression, factor analysis, correlation) on tabular files.

IBISUPDATE U Updates the SPICE/SEDR file with navigation information from an IBIS SEDR file.

ICTCOMP X Simulates Galileo SSI/PWS telemetry data which has been ICT or losslessly compressed. The resulting compressed image may be decompressed using ICTDECOMP.

ICTDECOMP X Decompresses a Galileo SSI or PWS image previously compressed via an ICT or lossless

compression.
IDX IX Displays VICAR images interactively.
IFFT IOU Allows user to interactively select and modify
areas within an FFT.
IGENER OX IBIS program creating an interface file from
user parameters. Will be replaced by IBIS.
IGIS IOX Displays VICAR images interactively allowing
geologist to determine: strike/dip, slope,
generate terrain files and trace contacts.
IMG2ASCII OU Creates an ASCII file of values from a VICAR
image.
IMLIST X Prints the picture summary of a raw Voyager
image file.
IMP IU Plots multispectral image data interactively
on a display device.
INSECT Mosaics or merges two images of unequal sizes
into a composite image.
INSERT Performs image copying and duplicates lines
when a larger image size is requested. It is
useful for correcting misregistration due to
line dropouts.
INSERT3D Inserts a band (or depth) into a 3-D file.
INTCON OX IBIS program converting IBM VICAR interface
files into VAX VICAR interface files
INTERLOC I*U Roughly locates grid intersections in byte
data.
ISISLAB X Prints the PDS label of an ISIS cube to the
terminal or VICAR log file.
KEPLER X Performs correvolving and Keplerian motion
compensations on images of planetary rings
systems.
KEYMOS1 OX Mosaics up to 10 images.

LAB2TCL OU Retrieves a label item from any part of the
VICAR label and assigns it to a local variable.
(Has better help file than GETLAB).
LABEL Lists, creates, modifies or removes a VICAR
label.
LABLIST Prints the project label of an image followed
by either a simple listing of history label program/task names or a complete listing of all history labels.
LABSWTCH U Switches the history labels of two VICAR images.
LABVFY OU Verifies that an image label contains a speci-
fied string.
LAVE OU Calculates the average DN or standard deviation
for each line or column as specified within an image.
LEONARDO ITU Allows the interactive definition, modification
and examination of "special colors" for the program GIACONDA.
LGEOM Performs geometric transformations.
LINEMTCH OU IBIS program performing one-dimensional line-
to-line correlations, placing the output in an interface file.
LIST Lists the DN values of specified regions within
an image.
LISTBITS Prints the bits in an image area as strings of ones and zeroes.
LISTLOCS X Lists the seam locations in a MAGELLAN MIDR
file.
LISTSEDR X Computes useful geometric parameters from the
Voyager SEDR.
LISTSEDX OX Computes useful geometric parameters contained

within the Datatrieve SEDR, and prints a report based upon user input sort criteria.

LOCUS U Compares sets of input coordinates and displays the differences.

LOGMOS X Logs and mosaics Magellan BIDR files to create MIDR files.

LOOKUP X Maps input DN values to red, green, blue output files using user-supplied pseudo color lookup table, such as from IDX.

LOS2DEM OX Runs line of sight (LOS) between two points on digital elevation model (DEM) to: find coordinates and elevation of points where LOS meets the DEM and finds the minimum, maximum and mean elevation along projection of LOS onto DEM.

LSTOXYZ U Converts the tiepoint locations stored in a Mark file and written by program TRACKER either to: X,Y,Z,Error coordinates in the planet reference frame or Lat,Long,Range-radius,error coordinates on the planet and writes them into another Mark file.

LTGEN U Initializes the Light Transfer File or the Reciprocity File used by MOMGEN, to measure the radiometric properties of a camera system.

MANMATCH U Acquires manual tiepoints from sets of images for the purpose of producing mosaics.

MANUAL OX Produces a mosaic from images which are geometrically corrected or uncorrected.

MANUAL2 OX Creates an updated VGR SEDR which can then be used by MAP2 to create an accurate mosaic.

MANUAL3 OU Creates an updated Galileo SEDR which can then

be used by MAP3 to create an accurate mosaic.
MANUAL4 OU Creates an updated Galileo SEDR which can then
be used by MAP3 to create an accurate mosaic.
MAP U Performs the calculations necessary for LGEOM
to perform a rubber-sheet stretch on a picture to convert it from a perspective projection, in which latitudes and longitudes are complicated functions of line and sample which depend on the position and orientation of the camera, to a standard cartographic projection in which longitude and latitude are simpler functions of line and sample which are independent of camera viewing geometry.
MAP2 PU Generates map projections by rubber-sheet transformation.
MAP3 U Performs the calculations needed to perform a rubber-sheet stretch on a picture to convert it from a perspective to a standard cartographic projections.
MAPGEN OX IBIS program taking an image divided into polygon regions, and assigning a grey value to each region according to information in an interface file, or by user-defined input parameters.
MAPGRID Superimposes a 25 x 25 pixel reference grid on an image.
MAPLABPROG U Adds the MAP2 task-label to an image label.
MAPMATCH U Acquires manual tiepoints between many input iamges and a reference image or mosaic for the purpose of producing mosaics.
MAPTRAN X Converts map projected images from one projection to another.

MAPTRAN3 OX Performs a brute-force transformation from one map projection to another by reprojecting every pixel.

MAPTRANS PU Transforms an image from one map-projection to another.

MAPTRANS2 U Computes map projection transformation parameters.

MARK U Scribes rectangles about specified pixel location.

MASKMGN X A set of mask routines used to produce image masks for the photoproducts of the Magellan project.

MASKMOS Converts map projected images into masks for input to NEWMOS.

MASKV U Formats any image for film recording by placing an information mask around the image. Labels and histograms optional.

MAXMIN Prints and returns into local variables the minimum and maximum data values in an image.

MEDIAN Performs nonlinear spatial filtering (highpass or lowpass) based upon the median DN of a rectangular window.

MEM U Performs a non-linear restoration (deconvolution) of a degraded image by the Maximum Entropy Method.

MENUTREE U Shows the branching of a specified menu.

MERGE DU Creates a new Voyager image from all versions of a specified image and updates the EDR header and the Voyager Image Catalog.

MESURCAHV Converts MESUR lander or rover images from distorted (CAHVOR) to linear (CAHV) coordinates.

MESURMOS Assembles multiple MESUR PATHFINDER lander images obtained with the same camera (either

left or right camera) into a mosaic. (This is a more simplified model than that used by the program MESURMOSCAHV.)

MESURMOSCAHV Assembles multiple MESUR PATHFINDER lander images obtained with the same camera (either left or right camera) into a mosaic using the CAHVOR model for the input images and the CAHV model for the completed mosaic.

MF OU IBIS program allowing use of Fortran-like expressions to perform general arithmetic functions on interface files.

MGEOM U Performs geometric transformations on images. Faster and more accurate than LGEOM for rotations of up to 85 degrees.

MGNCORR X Corrects for errors in deadreckoning between two 24-hour navigation solutions.

MGNDIRNAME X Creates Magellan directory names (as specified by relevent SIS's) from input parameters.

MGNFIT X Performs Magellan SAR mosaicking.

MGNSIZE X Compresses the size of a C-MIDR by a factor of 3 in both line and sample.

MGNZPAD X Pads zeroes to the front of orbit numbers to correspond to the file name suffixes of Magellan orbital data files.

MIDR X Performs batch processing of the various -MIDR data streams.

MIDR_ARCH X Performs batch processing of -MIDR data streams. Archive MIDR frames to optical disk, disassembles frames for copy to magnetic tape, and generates photoproducts.

MIDR_INSERT X Used for inserting Cycle 2 F-BIDRs into Cycle 1 MIDRs.

MIDRPOS OX Reads in the center coordinate and type of a MIDR and outputs the corner coordinates.

MINFILT Selects the minimum DN within a rectangular convolution window for images in either BYTE or HALF format.

MJSPFCF X Creates radiometric calibration files for the Voyager cameras.

MOCINDEX Processes Mars Observer Camera decompressed image CD-ROM volume index tables to produce a master index table to catalog the masked MOC image files that are film recorded via MRPS.

MOCMASK Generates masked images (VICAR image files) to be film recorded in the MOC systematic processing.

MOCSUMMARY Produces the Roll Contents Summary frame for the Mars Observer photographic products.

MOMGEN U Extracts statistical information from specified areas within the input image and stores the data in a record of the Light Transfer File or the Reciprocity File created by LTGEN.

MORPH OX Produces intermediate images between two input images.

MOSPLOT U Performs functions to assist the user in the generation of a mosaic.

MOUNT X Mounts a tape on a tape drive.

MOUNTDISK U Mounts optical disks for users.

MSS Concatenates up to 30 images horizontally.

MSSIBIS OU IBIS program converting MSS formatted data into an interface (tabular) file.

MSSVIEW OU Creates an image that contains a scatterplot of 2 or 3 bands in the center of an MSS formatted image

MTCLEAR OX Unloads a tape from a tape drive.

MULTOVLY OX IBIS program performing image overlay to produce a table of DN-combination counts vs. DN-combinations (a histogram of DN-combinations) in sorted order in an IBIS interface file format.

MZGEOM OU Performs simultaneous geometric and brightness corrections.

NAV IU Determines interactively the camera viewing geometry of an image by locating one or more distinct features in the image (e.g., planet limb).

NAV2 IU Determines interactively the camera viewing geometry of an image by registering it to a reference image with a known viewing geometry.

NEARENC X Determines non-interactively the camera viewing geometry of an image by registering it to a reference image with a known viewing geometry.

NETGEN OX Supports the primary experimental program NETWORK. Creates as output an image file whose purpose is to act as a "synapse matrix" for the network.

NETWORK OX An experimental program that simulates a neural network of forward-feed, back-propagation, Hopfield, or other hybrid architectures.

NEWMOS U Mosaics up to thirty images.

NF2 OX Allows general arithmetic operations to be performed.

NIMSCMM U Creates a NIMS Merged Mosaic (MM) "cube file" from one or more EDR files, in a non-interactive (batch) mode.

NIMSFLOAT U Converts all or part of a NIMS scaled radiance cube to floating point, using the scaling constants in the label.

NIMSMASK U Generates a 1250x1750 pixel mask for NIMS multispectral data.

NIMSMERGE U Receives a set of UDR files, merges them, and
outputs the merged data in EDR format.
NIMSMERGEX OX An extended version of NIMSMERGE.
NREP OX An experimental program that simulates a
neutral NREP of forward-feed, back-propagation,
Hopfield, or other hybrid architectures.
NUT U New User Tutorial.
NUTPROMPT U Aids in drawing the user's screen in the New
User Tutorial.
NXT Returns into local variables information about
the next file in a SRCH file.
OMC OU IBIS program converting camera pointing matrices
and spacecraft vectors between Earth coordinates
and planet coordinates.
OMCOR U IBIS program correcting the OM matrices for a
set of image frames in a planetary mosaic. For
VIKING Orbiter data ONLY.
OMCOR2 U Performs a global function minimization of
tiepoint residuals by determining the OM
matrices which cause the tiepoints to disagree
between image pairs by a minimum.
OSBLEMLOC U Computes object space blemish locations for
Voyager cameras.
OTF1 TU Performs 1-dimensional FFTs to compute Optical
Transfer Function from degraded edges in images,
or from a line spread function or tabular real
function.
OVERLAP U Creates a Magellan RSET Overlap Report from
two BIDRs.
OVERLAY U Superimposes a user-defined latitude-longitude

grid over an image space, object space or carto-
graphically projected image.

PAINT OU IBIS program which takes an image divided into regions and "paints" (assigns specific DN values to) each region.

PAINTG3 OX IBIS program which takes a Graphics-3 image divided into regions and "paints" (assigns DN values to) each region.

PCOPIN OX IBIS program generating an interface file from a Graphics-1 file. Replaced by IBIS.

PCOPOUT OX IBIS program generating a Graphics-1 file from an interface file. Replaced by IBIS.

PERSLAB X Places a perspective map projection label onto an object space image.

PFASTMOS X Designed to carry out the mosaiking and fragmenting functions for the Magellan P-MIDR products.

PHOTFIT2 TU IBIS program determining the coefficients of various photometric functions.

PHOTFUNC U Compensates an image for variations in apparent brightness due to illumination, viewing geometry and target properties.

PHOTO X Produces a Polaroid or 35mm hardcopy of VICAR images on the QCR device.

PHOTOM U IBIS program producing a photometric catalog from which the photometric function of a planet can be evaluated using PHOTFIT2.

PHOTTEST U Generates synthetic photometric data to test PHOTFIT2.

PIC2VIC X Performs a straight transfer of an input sub-
image to an output VICAR image.
PICMATCH AU IBIS image correlation routine for matching
images or ground control points datasets to an image.
PICREG I Allows interactive registration of 2 images,
generates the tiepoint and/or geom file (used with TIECONM, GEOMA, LGEOM or MGEOM).
PICSUM Adds together multiple image files and flags
saturated pixels for Galileo CCD calibration.
PIDR_ARCH X Procedure for archiving the Magellan PIDR products.
PIDRLOG X Logs one pidr file at a time into a VICAR image file in the correct oblique sinusoidal projection.
PIXGRAD OU Calculates the magnitude and gradient of the
brightness in an image.
PIXSTAT Generates images of statistical quantities
derived from input image.
PLANETCNV OU IBIS program converting points in an interface
file between line/sample in an image and latitude/longitude on a planet surface. For VIKING Orbiter images only.
PLOT3D OTU IBIS program plotting 3-D Graphics-1 files in
perspective view.
PLOTINT OTU IBIS program plotting columns of an interface
file.
PLOTTING TX Selects (allocates) a plotting device for use
with the device-independent plotting system.
PLTDGN OX Plots INTERGRAPH IGDS design files using the
device independent Calcomp plotting system.
PLTGRAF OTX IBIS program plotting a Graphics-1 file inside

a labeled box.

PLTSYM OTX IBIS program plotting CALCOMP special characters on the plotting device via a 3-D graphics file.

PMASKMGN X Produces a P-MIDR image mask for the photoproduct of the Magellan project.

POLARECT Performs rectangular to polar coordinates transformation (or vice versa) on an image with respect to an origin.

POLYCLIP OU IBIS program clipping a specified window of graphics elements from one Graphics-1 file, placing the output in another Graphics-1 file.

POLYCON OX IBIS program converting Graphics-1 file to Graphics-2 file and vice versa or copies Graphics-2 to Graphics-2.

POLYGEN OX IBIS program creating a Graphics-1 file.

POLYGEOM OU IBIS program performing geometric transformations to correct for distortion. It may increase or decrease the size of a point polygon dataset or return it to its original state.

POLYMAP OU IBIS program converting Graphics-1 files from one Earth-oriented map projection to another.

POLYMASK OX IBIS program cutting the portion of an image delineated by a set of polygons contained in a Graphics-1 file, setting the data on the inside or outside of the polygons to a specified DN level.

POLYNOIS U Produces an image containing discrete level noise.

POLYPMAP U IBIS program performing planet-wide map projections on Graphics-1 files, between latitude-longitude space and line-sample space.

POLYPNT OX IBIS program transforming standard polygon
files into an image file of polygon borders each painted a unique DN level.
POLYREG OU IBIS program rigidly transforming a polygon
dataset to correct for differences in size, rotation, aspect or skew.
POLYSCRB OU IBIS program transforming Graphics-1 files into
an image file containing polygon borders drawn on a uniform background.
POWER TU Computes a 1-dimensional power spectrum of a
specified area in an image using FFT.
PPLOAD DX Reads a special database file containing
predict PRA/PWS data into the Voyager Image Catalog.
PRINTPIX U Produces a pseudo-gray-level image for a
printer from an input image.
PRNTIM LOX Prints images in graphics mode on a DEC LA-50
printer using patterns having varying shades of
gray.
PROCGEN X Generates VICAR TAE procedure for image processing.
PSCRIPT Converts a standard VICAR image into a text
file of POSTSCRIPT commands, downloadable to a laserprinter.
PSF U Formats a Point Spread Function image for FFT22
or FT2.
PTP U Projects a Voyager planetary or a satellite
image to a different perspective.
PUTPWS X Repacks an unpacked pws file into edr format.
PUTSPICE X Updates the SPICE or SEDR for multiple
projects.
PWAUDIO GPU Enters PWS image files into the PWS audio

generator based on a range of FDS counts.

PWECHO GU Generates an audio tone on the PWS audio generator from one Voyager or Galileo PWS file.

PWSMEK X Prints the data for a Galileo PWS Mini-E Kernel.

PWSMERGE U Merges a maximum of 20 PWS UDR files into a single UDR file, which is further processed to generate the EDR file.

QD IXO Provides a "quick" display of a 512 x 512 (or less) image on a display monitor. There is no interactive mode and it exits immediately upon completion of display.

QKLK PX A procedure producing up to 12 "masked" Voyager images for a "quick look" playback on the DICOMED film recorder.

QPLOT PTU A procedure plotting DN values along a straight line through an image as specified by the user.

QPLOT2 U Plots the DN values along a specified straight line through an image. Also create spectral plots for multi-channel data.

QREP OX IBIS program which formats and lists tabular interface files and Graphics-1 files. Replaced by IBIS.

QSAR Adds specified values to the DN numbers of rectangular sections of an image.

QUERYLAB X IBIS program inserting user-defined columns of names and labels into first record of interface file.

R AX Performs automated registration of two images that are similar geometrically except for offset.

R90 OX Rotates a large picture 90 degrees clockwise.

RADAGEN OU Transforms a terrain image, in which DN values represent altitude, into a synthetic radar image.

RADIANCE OX Given a Voyager input image, locates the target body and calculates the mean DN value.

RAPIDMOS U Mosaics up to 40 byte images into one image. Similar to FASTMOS but with reduced functionality.

RASTOGRAF OU Converts from a line image in raster format to an IBIS Graphics I file consisting of line segment information.

RASTOVEC OX IBIS procedure transforming image polygons into vector representation of the polygon boundaries.

RATIO PU Procedure ratioing two images (e.g., ratio, log ratio, difference or log difference).

RATIO0 U Operates as part of the procedure RATIO and should not be run outside of its procedure.

RCBR X Processes up to 7 orbits of Magellan altimeter data and generates, for each orbit, a BADR altimeter data file.

RDM X A procedure PDF that calls the Magellan program RDM.

RDMINIT X Sets up symbols and logical names for use by the Magellan program RDM.

REPAIR OU Locates bad lines and repairs them using interpolation.

REPORT OU IBIS program generating tabular listings of information within interface files.

REQUEST X Allows the user to send a request or a message
to the system operator.
RES77 X Stores or retrieves Voyager reseau coordinate
information from the Voyager Master Reseau
Location File.
RESET Resets the next file pointer in the list of
files output from the program SRCH.
RESLGEN X Generates a Voyager Master Reseau Location
File.
RESLOC U Locates reseau marks on Voyager
images.(optionally creates a geometrc
transformation file)
RESLOCVO Locates reseau marks on VIKING Orbiter images.
RESSAR75 U Removes reseaux, fiducial marks and blemishes
from VIKING Orbiter images.
RESSAR77 U Removes reseaux, fiducial marks and blemishes
from Voyager images.
RESTORW PU Procedure restoring an image using the Wiener
noise additive model and and input Point Spread
Function image.
RG TX Converts tiepoints generated by R into GEOMA
parameters and produces a plot of the tiepoint
shift in the output tiepoint transformation.
RGB X Merges three input files into one RGB file for
use with the MDA film recorder.
RGB2PSEUDO OU Takes a 24-bit RGB triplet and converts it to 8
bit pseudo-color.
RING U Projects a planet's ring plane onto a
radius-azimuth grid for Voyager images (e.g.,
to straighten the rings).
RINGORBS U Creates an Orbital Elements File containing
data for each of the known planetary ring

systems.
RIV2HEX OX IBIS program generating a Graphics-1 file
containing hexagon edges used to represent
rivers from the original Graphics-1 dataset.
ROADHEX OX IBIS program generating an interface file of a
list of hexagon edges, used to represent roads,
from a Graphics-1 dataset.
ROLL X Will right-shift each line of the input to a
new position in the same line in the output,
according to 'scale' and 'offset' parameters.
ROTATE P Procedure rotating an image about a specified
point by some angle.
ROTATE2 Computes the geometric transformation parameters
for rotating a picture by any amount about a
specified point. Usually called from ROTATE.
ROWOP OU IBIS program to delete or select rows from an
interface file.
RUNCODE OX IBIS program compressing or decompressing VICAR
images using run-length encoding.
SARGON I Performs arithmatic operations interactively on
user-defined areas (polygons) within an image
SARGONB Performs arithmatic operations on user-defined
areas (polygons) within an image
SBOXFLT OPU Performs 2-D highpass box filter with filter
weights equal to 1.
SCTOMARK U Reformats output files from STARCAT for input
to MARK.
SDSIBIS OX IBIS program converting statistical datasets
generated by classification programs into
interface file format and vice versa.
SEDRGEN OX Creates or updates a Voyager SEDR record.

SFASTMOS X Carries out the mosaicking and fragmenting
functions for the Magellan mission.
SHADY Superimposes user-defined contour lines and/or
shading on an image.
SHADY2 Shades an image as if image was illuminated by a
source at a given azimuth and elevation.
SHIST OPX Produces a simple histogram on a terminal
screen.
SHOWDISP X Prints information about display devices.
SHUTOFF X Converts Galileo shutter-offset parameter
datasets to new image format.
SIMPLIFY OX Removes high frequency noise components from
digital images especially from classified or
stratified data.
SINPROJ U Reprojects a vicar image file of a sinusoidal
projection of a given projection longitude into
an image file with a different projection
longitude; can also produce an output file with
the same projection longitude but different
size.
SIPHON X Transfers F-BIDR data from the Multimission
SAR Processing Lab (MSPL) to the Multimission
Image Processing Lab (MIPL).
SIZE U Expands or reduces the size of an image, or
changes the aspect ratio of an image.
SLABEL OPX Lists the VICAR label of an image.
SLIST OPU Lists the DN values of pixels within a 10x10
section of an image.
SORT OX IBIS program performing multiple column sort on
an interface file in ascending or descending
order.

SOS U Removes camera shading from VIKING Orbiter
images.
SPAM I SPectral Analysis Manager performing a wide
variety of spectral analysis functions
interactively.
SPECPLOT U Calculates the mean and standard deviation of a
user-specified pixel region of each band of a
multispectral or bultiband VICAR file.
SPICE U Lists all the information available on any
image from any flight project.
SPICEBDF Creates a binary dump file from a P_constants
kernel, SCLK kernel, and leapseconds kernel.
SPICESERVER Retrieves the NAIF spice information returned
by subroutine GETSPICE from a remote site
rather than from the local site; also stores
the NAIF spice information stored by subroutine
PUTSPICE at a remote site rather than at a
local site.
SPLITCAL X Converts Galileo SSI slope-offset radiometry
file into 2 files containing: slope terms and
the dark current values.
SPOT Generates byte images containing spots of
various profiles and sizes.
SRCH DX Performs an arbitrary search of the MIPL
Voyager Image Catalog and generates an output
list based on requested search criteria.
SRCHGEN DX Allows user to submit Datatrieve commands and
construct customized output files from query of
a Datatrieve domain.
SSIMEK X Prints the data for a Galileo SSI Mini-E
Kernel.
SSIMERGE U Merges the data from several versions of a
Galileo SSI image.
SSTRETCH OPX Performs simple linear stretch between

user-specified limits.

STARCAT U Locates and integrates features above specified DN threshold values into a catalog entry.

STARCAT2 U Locates and integrates features consisting of contiguous pixels above a given threshold in an image and places the results into a catalog.

STARLAB U Places identifier labels beneath each object located by the program STARCAT.

STATPLT COPU Procedure generating a plot of the classification statistics file containing the Centroids and Bayesian confidence regions for each class.

STATS U Computes statistics of training areas for classifications using multispectral data.

STEREO OX Produces stereoscopic image pairs from an image corresponding to the "right eye" image and a "depth" image for each point in order to produce a "left eye" image.

STEREOPIC OX Displays a stereo image on the Tektronix display terminal.

STERGEN IX Generates a correlation tiepoint file from a stereo pair of images, based on interactive user input.

STERMAP X Displays the tiepoint information produced by STERGEN as either a map of detrended vectors, an elevation map, map of quality or a listing of tiepoint data.

STOREKDB X Creates an entry for a kernel in Kernel Database.

STRATIG OX Intergraph software/hardware dependent program called by Intergraph User Command XSECT.

STRETCH Performs 7 types of contrast enhancements on an image (adaptive law, complement, contrast, cuberoot, linear, power law and table).

STRETVAR Performs a linear contrast enhancement such that
the high and low stretch limits vary as a
function of line number.
STRPMTCH OX Extracts an area from a raw image, producing a
compressed image of that area.
SURVEYOR IOX An interactive, pull-down menu based program
used to create terrain flyby animations.
SWAPPER OX Converts IBM halfword data to VAX halfword
format.
SYNTAX X Enables syntax-checking of programs run under
VICAR. Equivalent to FLAG-SET SYNTAX.
SYSNIMS X Performs systematic processing of NIMS data.
TAPES X Copies or scans tape files. Program runs under
VMS. See DCL HELP Tapes.
TAPESIPHON X Imitates the program SIPHON when transferring
F-BIDRs from SDPS (or other sources) by tape.
TEXTAD X Superimposes rectangles and/or text on an image
TFILT A*P Performs convolution filtering (divide, high-
pass, lowpass, or scene dependent filter) by
averaging pixels in the neighborhood above a
threshold value.
TFILTAP AX An Array processor version of program TFILT0.
TFILT0 Performs various boxfilter type convolutional
filters.
THERMMOS OPX Mosaics 4 quadrants of an EROS Data Center TIPS
processed LANDSAT 4 or 5 Thematic Mapper image
(thermal bands only).
TIECONM OU Prepares a gridded dataset for POLYGEOM, GEOMA,
LGEOM, MGEOM or GEOMZ transformations using
paired sets of tiepoints, applying the finite

element method (triangulation) for surface fitting.

TIEPARM OX IBIS program converting tiepoints from an interface file to a parameter file (for TIECONM) and vice versa.

TIEPLOT OTU IBIS program plotting tiepoints in an interface file by drawing vectors indicating direction and amount of shift between old and new pixels.

TIESIM OX Creates an artificial Magellan tiepoint file, as would be produced by the program MGNFIT.

TIPSMOS OPX Mosaics 4 quadrants of an EROS Data Center TIPS processed LANDSAT 4 or 5 Thematic Mapper image. (bands 1, 2, 3, 4, 5 and 7).

TOIBIS OU Converts a file with an arbitrary line count, sample count, and format into an IBIS graphics or interface file.

TOPOTRANS OPU Procedure registering a radar image to a digital elevation image.

TOTOPO U Converts the tiepoint locations stored in a Mark file and written by program TRACKER and XYZ coordinates created by program LSTOXYZ to one of: 1) a topomap, 2) an orthonormal view of the left image, 3) an orthonormal view of the right images, or 4) an error map.

TOXYZ OX Converts VICAR digital elevation files to Intergraph digital terrain modeling XYZ files.

TPTCOORD X Converts a MARK tiepoint file from line, sample format to latitude, longitude format.

TPTEDT X Permits editing of a tiepoint file in MARK format.

TPTEDT2 Batch editor for automated removal of erroneous

vectors (tiepoints) from a mark file produced by
TRACKER or TRACKER2. The equivalent interactive
program is TPTEDT.

TRACKER U Locates tiepoints on a grid between two images
and writes the tiepoint locations into a mark file.

TRACKER2 X Locates tiepoints on a grid between two images
and writes the tiepoint locations into a mark file; follows features already marked in a mark
file through a sequence of images, creating a new mark file for each pair in the chain.

TRACKER3 Acquires points on a regular grid; reads tie-point locations from an input tiepoint file and
correlates these locations only. Chaining of tiepoints is permitted.

TRAN Converts an image from one organization to another. Possible organizations are BSQ, BIL, BIP, and MSS.

TRANSCOL OU IBIS program converting long columns of data
into smaller columns based on user-specified parameters or vice versa.

TRANSLOG P Returns into a local variable the translation of
an input logical name defined under VMS or VICAR.

TRICOEF Used to compute and store for later retrieval
the conformal and authalic coefficients permit-
ting computation of conformal (angle preserving)
and authalic (equal area) projections of triaxial ellipsoids.

TRISTIM Computes tristimulus values and chromaticity
coordinates for a spectrum in order to perform color reconstruction.

TRUCOLOR X Transforms images taken through several filters
into a color image in which an attempt is made to reproduce accurately certain designated spectra ("special colors").
UPPERCASE X Converts a string to uppercase and returns
it to a tcl variable UPR_CASE>
USERNAME PU Returns into a local variable the current userid.
USTATS U Performs unsupervised clustering on multispectral data.
UVECTOR X Generates a graphical display of vectors used
for atmospheric motion analysis on Voyager images. Uses output from UVF. Replaced by UVMAP.
UVF DX Generates a uniform vector field of cloud velocities from randomly spaced input points. Uses output from AMOS. Replaced by UVMAP.
UVFMAP TX Maps a set of graphical contours, defining
regions of meteorological significance over a Voyager image. Uses output from UVF or UVFSTATS. Replaced by UVMAP.
UVFSTATS X Extracts information from the output of UVF and
generates a scalar dataset of selected quantities. Replaced by UVMAP.
UVMAP U Generates images containin information pertinent
to atmospheric analysis.
V2BIDR OX Places Soviet Venera formatted files into Magellan F-SBIDR format.
V2VERSION OPX Types the current VICAR/TAE version number.
VCOPOUT OX IBIS program transforms an IBIS interface file
into a user-specified output file for transfer to other data centers.
VCOPY OPX Lists the VICAR labels of all files on a tape.
VDEMLOG OX Logs in NCIC data from a DEM (Digital Elevation

Model) formatted tape into VICAR formatted files.

VEDR D*U Reads a Voyager EDR tape and copies the selected images onto disk in VAX VICAR format. Optionally needs image catalog.

VERTSLOG OX Logs ERTS data in the format used by LANDSAT's 1 through 3, correcting geometric distortions found in ERTS data.

VFITS2 OX Converts FITS formatted astronomical data into VICAR image files. Replaced by FITSIN, FITSOUT.

VGEDR X Reads the Galileo data tape created by VGSDR and converts it to VICAR format on magnetic tape.

VGLLOG X Logs Galileo halfword calibration images into VICAR byte images.

VGOES OX Converts GOES (Geostationary Operational Environmental Satellite) imagery from the University of Wisconsin Save-tape format to VICAR format.

VGRBLEM X Generates an IBIS Graphics-1 file containing locations of the Voyager camera blemishes.

VGRFILLIN U Fills partial or entire missing lines in raw Voyager image files using line interpolation methods.

VGRIBIS X Extracts information from a Voyager SEDR file, augments it with camera constants and then reformats the data as a VICAR IBIS interface file.

VGRLOG X Logs or scans Voyager EDR files and converts them to VICAR format.

VGRMASK DX Formats processed Voyager images into the Voyager Systematic format with an information mask for film recorder display.

VGRPWS X Decompresses 4-bit Voyager PWS data to 8-bit
data using a lookup table.
VGSDR X Converts Galileo System Data Records into VICAR
format. Output can be read by VGEDR.
VIDS I Displays VICAR images interactively.
VIEW X Can be used to examine any area of interest in
any of the following Magellan products: MIDRs
GADRs, GRDRs, BIDRs, PIDRs, BADRs, and BRDRs.
VISIS U Provides transformations from NIMS systematic
VICAR files to a NIMS ISIS Cube file and from
a NIMS ISIS Cube file to VICAR files.
VLOOKUP Generates output images from input images
using data number mappings defined in a
multi-channel lookup table. The user can
specify the table by giving its location (if
it is contained in a file) or by naming a
standard pseudocolor transformation.
VMAC OX Transfers data between PICT or TIFF format
images and IMG files.
VMDPIN OX IBIS program converting Goddard MDP ground
control point file into Graphics-2 format.
VOLOG U Logs or scans VIKING Orbiter EDR files and
converts them to VICAR format.
VOLSTSDR U Lists VIKING Orbiter SEDR data for a specified
FSC time.
VOSDRIBIS IOX IBIS program extracting certain items of
information from each VIKING Orbiter record in
the SEDR image catalog, placing them in a
tabular file.
VQUIC OU IBIS program converting VAX ASCII Edit file
into an interface file.
VSAR X Converts tape or disk images from the labeled
to the unlabeled format or vice versa.

VTIFF Converts between VICAR labeled images and TIFF
format files, using either scanline (strip) organization, or the newer Revision 6.0 TIFF tiled-image format. Currently grayscale, image-lookup table pairs and RGB triplets are supported. In addition, multiple auxilliary images may be placed in the same file, such as "thumbnail" preview images.

VWRIS OX IBIS program converting US Forest Service WRIS
(Wild Land Resource Information System - RID*POLY) polygon map files in Universal Data Exchange Format into IBIS vector and centroid graphics files.

WIENER AU Restores a Fourier transform image using the
Wiener Noise additive model.

WILDCARD U Returns all filenames found using a wildcard
file specification into a multivalued local variable.

WILMA FX VMS program allowing tracking and modification
of film recorder requests.

WORMDIR X Creates the directory specification for WORM archival.

WRISOUT OX IBIS program converting IBIS vector and
annotation data into U.S. Forest Service WRIS/RID*poly files.

XFORM A*PU Performs a matrix-defined linear transformation.

XFORMAP X Performs a linear transformation using an array processor.

XFORMEM U Performs a linear transformation on the input
data.

XLOCUS OU Applies LOCUS rotation and offset transformation matrix to the Galileo Theodolite
data.

XSECT NOX An Intergraph user command menu producing

stratigraphic and subsurface profiles using 3-D elevation design files with strike/dip and contact information on Intergraph hardware.

XVICDISP This is the first version of a display program based around the Imaging Widget. It is a simple first cut, and will change as development progresses on the GUI.

XVPLIST OX Lists the tiepoints in a dataset written by the XVP routines in programs such as RG2, TIECONM or TIEPARM.

XYZPIC OX IBIS program taking a Graphics-1 file and creating a 0 DN image, except for pixels listed in the Graphics-1 file.

ZCIRCLE OU Removes or replaces data from inside or outside a circle or elliptical pattern in an image.

ZFILL Fills voids within an image using the mean of all non-void DN pixels within the specified window.

ZINTERP OU IBIS program interpolating elevation values (z values) from random control points into a rectangular gridded image.

ZIPCOL OU IBIS program merging data into specified columns of an existing interface file from another interface file.

Index

(The numbers listed after each entry are sections, not page numbers.)

Defined 1.3.1